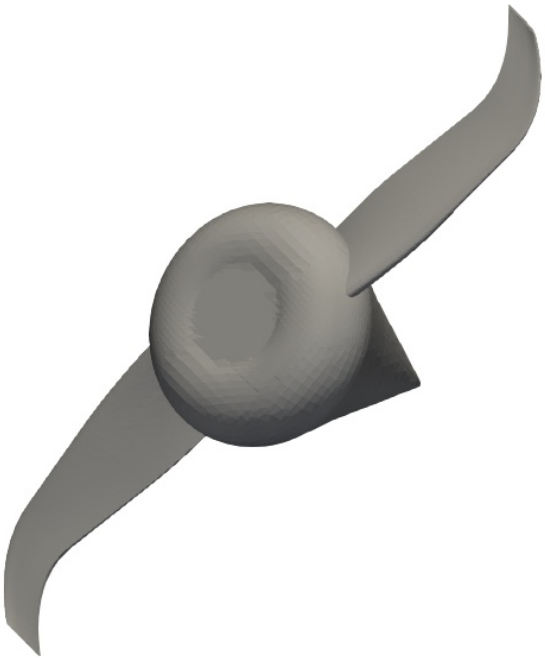
CHALMERS
UNIVERSITY OF TECHNOLOGY



Optimization Method Analysis for Tidal Turbine Blade Design

Master's thesis in Physics

SAMUEL EDLUND

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis in Physics, MSc 2025

Optimization Method Analysis for Tidal Turbine Blade Design

SAMUEL EDLUND

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Optimization Method Analysis for Tidal Turbine Blade Design
SAMUEL EDLUND

Supervisor: Björn Bergqvist, Minesto AB
Examiner: Håkan Nilsson, Department of Mechanics and Maritime Sciences

Degree project report 2025
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Minesto's turbine model.

Typeset in LaTeX
Gothenburg, Sweden 2025

Optimization Method Analysis for Tidal Turbine Blade Design
SAMUEL EDLUND
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

As demand for green energy increases, Minesto AB produces marine technology for renewable tidal energy solutions. Underwater turbines are used to harness the energy of ocean currents caused by tidal movements. To maximize the electricity generation, the design of the turbine is crucial. The purpose of this thesis project is to study the optimization problem related to a turbine model developed by Minesto AB. The turbine performance for different blade geometries was estimated by calculating the power and thrust coefficients (C_P and C_T) based on results from computational fluid dynamics (CFD) simulations in the software OpenFOAM. The former coefficient relates the generated to the total available power, and the latter the thrust force to the water flow momentum, and the primary focus was to maximize C_P and the ratio C_P/C_T. There were 18 parameters to be optimized, related to the geometry of the turbine, including chord lengths, and angles defining the bend and rotation of the blade. The optimization methods tested and analysed were Nelder-Mead, NSGA-II, simulated annealing, and Bayesian optimization with Gaussian processes. The methods were implemented primarily in Python with different settings and operators, and combinations of the methods were also tested. Initially, the optimization was focused on exploring the parameter space, and the maximum power coefficient was found to be $C_P = 0.3902$. The final goal was to minimize the number of CFD simulations needed to find the optimum, by identifying and implementing the most effective algorithm. The project found that, for a constant parameter space, Bayesian optimization is the most efficient individual method, and reliably produces results close to the overall maximum C_P within about 260 simulations, or about 30 h. However, combining it with NSGA-II and Nelder-Mead can be advantageous.

Keywords: turbine, tidal energy, optimization, multi-objective optimization, computational fluid dynamics, genetic algorithm, Bayesian optimization

Acknowledgements

Firstly, I would like to thank my supervisor at Minesto, Björn Bergqvist, for constructing a solid simulation setup, and for all his help, advice and ideas during the project. His work made it possible for me to carry out the project at Minesto, which was a wonderful and valuable experience. Thank you also to Martin Edlund who accepted my thesis request, and everyone else at Minesto, for making me feel welcome and at home while working there, and for showing an interest in my thesis.

Next, I would like to extend my gratitude to my examiner Professor Håkan Nilsson, for providing tools to help me with the CFD simulation software, and for his feedback during the project. His advice was especially helpful in the planning and report writing processes.

Lastly, thank you to my family and friends for all your support and encouragement.

Samuel Edlund, Gothenburg, June 2025

Glossary

Below follow lists of definitions for abbreviations and symbols used throughout the report, in alphabetical order. Where applicable, SI units are provided in square brackets.

List of Abbreviations

Term	Definition
ARD	Auto Relevance Determination
BLX	Blended Crossover
CFD	Computational Fluid Dynamics
DEX	Differential Evolution Crossover
GAMG	Geometric-Algebraic Multi-Grid
GP	Gaussian Process
MLP	Multi-Layer Perceptron
MOO	Multi-Objective Optimization
NM	Nelder-Mead
NSGA	Non-dominated Sorting Genetic Algorithm
OpenFOAM	Open Field Operation And Manipulation
RBF	Radial Basis Function
SA	Simulated Annealing
SBX	Simulated Binary Crossover
TSR	Tip-Speed Ratio
UCB	Upper Confidence Bound

Symbols

Fluid Dynamics

$\vec{b}$	Body force per unit mass [Nkg^{-1}]
c_V	Specific heat capacity [$\mathrm{Jkg}^{-1}\mathrm{K}^{-1}$]
e	Spec. thermal energy [Jkg^{-1}]
K	Spec. kinetic energy [Jkg^{-1}]
k	Spec. turbulent kinetic energy [Jkg^{-1}]
p	Pressure [Pa]
$\vec{q}$	Heat per unit area vector [Wm^{-2}]
r	Heat source per unit mass [Wkg^{-1}]
S	Surface [m^2]
T	Temperature [K]
t	Time [s]
$\vec{u}$	Velocity vector [ms^{-1}]
V	Volume [m^3]
ε	Turbulent dissipation rate [$\mathrm{m}^2\mathrm{s}^{-3}$]
κ	Thermal conductivity [$\mathrm{Wm}^{-1}\mathrm{K}^{-1}$]
ν_{eff}	Effective viscosity [$\mathrm{m}^2\mathrm{s}^{-1}$]
ν_{t}	Turbulent viscosity [$\mathrm{m}^2\mathrm{s}^{-1}$]
ρ	Mass density [kgm^{-3}]
$\boldsymbol{\sigma}$	Stress tensor [Pa]
ϕ_f	Volumetric surface flux [ms^{-1}]
ψ	General variable
ω	Spec. turbulent dissipation rate [s^{-1}]

Operators

D_t	Material derivative
Δ	Laplacian operator
∂_t	Partial time derivative
$\vec{\nabla}$	Gradient (nabla) operator

Optimization

f	Objective function
$\mathcal{GP}$	Gaussian process (GP)
K	Kernel (covariance) function (GP)
K^{ARD}	Kernel with ARD (GP)
P_{accept}	Acceptance probability (SA)
p	Probability distribution (GP)
S_i	Decision vector state (SA)
T_i	Artificial temperature (SA)
U	Uniform probability distribution
X	Parameter space
$\boldsymbol{x}$	Parameter decision vector
x_i	Individual parameter value
x_i^{offs}	Offspring parameter value (NSGA)
x_i^{parent}	Parent parameter value (NSGA)
Y	Objective space
$\boldsymbol{y}$	Objective vector
y	Observed objective value (GP)
μ	Mean function (GP)
σ	Hyperparameter (GP)
ϕ	Objective function value (GP)

Turbine Theory

A	Turbine rotor area [m^2]
C_P	Power coefficient
C_T	Thrust coefficient
$\dot{m}$	Mass flow [kgs^{-1}]
P_{rotor}	Rotor power [Js^{-1}]
R	Turbine rotor radius [m]
T	Thrust force [N]
$\mathcal{U}_{\mathrm{d}}$	Rotor disc plane velocity [ms^{-1}]
$\mathcal{U}_{\mathrm{w}}$	Far wake velocity [ms^{-1}]
$\mathcal{U}_{\infty}$	Free stream velocity [ms^{-1}]
ρ	Mass density [kgm^{-3}]
τ	Torque [Nm]
ω	Angular velocity [s^{-1}]

Table of Contents

1.
Introduction

Minesto is a developer of marine energy technology in the renewable energy sector, based in Gothenburg [1]. The company's main focus concerns electricity generation from tidal streams and ocean currents. Instead of a stationary power plant, Minesto produces a unique tidal energy solution [2], which consists of an underwater kite turbine system called *Deep Green* [3]. The kite is connected to the seabed with a tether, and moves autonomously using a built-in control system in a figure-of-eight path. This allows for a water flow through the attached turbine higher than the actual stream velocity. A generator converts the mechanical energy of the turbine to electricity, which can be transferred to the electricity grid through a cable in the tether.

Minesto is currently working on the production and improvements of new tidal energy technologies. One project concerns a modified version of the kite system, which is designed to function better at shallower depths and slower ocean currents. This system relies on an entirely different kind of turbine, and the company is therefore interested in studying the design and optimization of this turbine.

The electricity output of the system is directly dependent on the energy efficiency of the turbine, which relates to the conversion from the kinetic energy of the water flow to the mechanical energy of the turbine [4], [5]. The efficiency is affected by the geometry and structure of the turbine, meaning that the design of the turbine is an important factor in the final electricity output [6]–[14].

Minesto's design requirements are focused on the power and thrust coefficients (C_P and C_T). The power coefficient is a factor of the efficiency, and refers to the ratio of the power produced to the total power available [15]. The thrust coefficient is defined by the relation between the thrust force from the turbine and the incoming momentum of the water flow [6]. In order to optimize the performance of the turbine, it is therefore necessary to maximize the power coefficient [6], [7], and simultaneously maximize the ratio between the coefficients: C_P/C_T.

The characteristics of the water flow through the turbine can be estimated using computational fluid dynamics (CFD) simulations [16]. This is one method to study how the geometric properties of the turbine affect the flow, and in turn the output [6], [8]–[11]. Designing the optimal turbine is an optimization problem where the equations describing the system depend on parameters determining the geometry of the turbine. Minesto currently employs a solution that comprises computational methods such as Nelder-Mead and NSGA-II, together with CFD simulations using the software OpenFOAM. This method provides a reliable structure for determining the relevant geometric design parameters describing the turbine.

1.1 Purpose and objective

The purpose of this master's thesis project is to investigate the performance and efficiency of different optimization methods and algorithms when applied to Minesto's turbine model and CFD simulation setup. Different optimization methods and implementations require different quantities of data, and thus different computation times (i.e. number of CFD simulations), to achieve a reliable result. It is therefore of interest to document how different computational methods and implementations affect the number of required CFD simulations to find the optimal turbine design with respect to the power and thrust coefficients. The goal is to determine the parameters resulting in the optimal design, and to implement an optimization method that minimizes the time required to find the solution.

1.2 Delimitations

The project is specifically concerned with the optimization process of a single turbine model (with geometry and parameters described in Section 2.1.3), and CFD simulation setup (described in Section 3.2), both developed by Minesto. The simulation structure involves a constant water flow and a constant tip-speed ratio (TSR). The results to be optimized are the power coefficient and power to thrust coefficient ratio, based on CFD simulations, and the project will not deal with modifying the CFD simulation setup.

The turbine is only studied through 3D model software and computer simulations. How the design can or should be constructed in reality, and questions regarding the construction, such as strength of materials, are not taken into account.

1.3 Report structure

Following the Introduction, Chapter 2 presents relevant theory regarding turbines, fluid dynamics, and optimization. In Chapter 3, the main components of the project procedure, the simulation and optimization methods, are explained. Thereafter, the results are presented and discussed in Chapter 4, and the conclusions can be found in Chapter 5.

2.
Theory

This chapter aims to present a relevant theoretical background to the concepts dealt with in this project. This includes general information about the turbine involved, and the properties considered for optimization. A background to the subject of optimization and the specific methods implemented is also provided.

2.1 Turbine theory

Turbines are devices that convert the energy of a moving fluid, such as water or wind, to mechanical power [17]. They are therefore commonly used for electricity generation. The rotor power extracted from a flow can be expressed using the difference in kinetic energy of the fluid, as

$$P_{\text{rotor}} = \frac{1}{2}\dot{m}(u_\infty^2 - u_{\text{w}}^2) \ , \tag{2.1}$$

where $\dot{m}$ is the mass flow through the rotor plane, u_∞ is the free stream velocity (before the turbine), and u_{w} is the far wake velocity (after the turbine). The mass flow is a product of the fluid density ρ, the rotor area $A = \pi R^2$, and the fluid velocity through the rotor disc plane u_{d}. The power can also be expressed using the torque τ and angular momentum ω of the blade, which means that

$$P_{\text{rotor}} = \tau\omega = \frac{1}{2}\rho\pi R^2 u_{\text{d}}(u_\infty^2 - u_{\text{w}}^2) \ . \tag{2.2}$$

The variables and the system are visualized in Figure 2.1

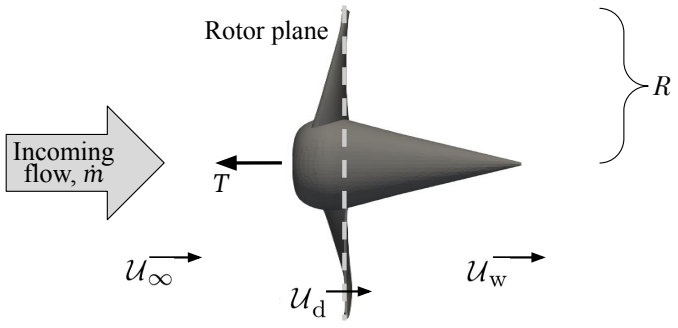


Figure 2.1: Two-dimensional schematic overview of the system. The rotor plane is marked with a dashed line, and T is the thrust force from the turbine.

2.1.1 Power and thrust coefficients

The power and thrust coefficients are the two properties used in this project to quantify the performance of the turbine. Continuing from Eq. 2.2, the power coefficient C_P is defined [17] such that

$$C_P = \frac{\tau\omega}{\frac{1}{2}\rho R^2 \pi \mathcal{U}_\infty^3} \,. \tag{2.3}$$

This is the ratio of the produced rotor power to the total available power [15]. To increase the electricity generation, the power coefficient should therefore be maximized.

The thrust coefficient C_T is defined as

$$C_T = \frac{T}{\frac{1}{2}\rho R^2 \pi \mathcal{U}_\infty^2} \,, \tag{2.4}$$

where T is the thrust force from the turbine [6]. This relates the thrust force to the total momentum of the fluid flow.

The ratio of these coefficients, C_P/C_T, describes a particularly interesting relation, which is desired to be maximized to achieve a high power generation in relation to the produced thrust force. The project focused on the optimization process of maximizing both C_P and C_P/C_T, with a high C_P being the main concern.

2.1.2 Tip-speed ratio

The tip-speed ratio (TSR) is another relevant parameter, which is used in the project model to define the rotational speed of the turbine. The TSR is the ratio of the tangential tip-speed of the blade to the wind velocity [11], defined as

$$\text{TSR} = \frac{\omega R}{\mathcal{U}_\infty} \,. \tag{2.5}$$

2.1.3 Geometric parameters

This project focused on a set of geometric parameters belonging to four aerofoil segments of the turbine blade, located at the beginning, half, three-quarters, and end of the wing length, see Figure 2.3, as well as the three intermediate wing sections. A total of 18 inputs determining the differences in the wing geometry between the simulations were considered, while the main wing structure remained constant and was seen an inherent part of the model. The optimized parameters are referred to as the lift coefficients (4, one per aerofoil segment), chords (4), twists (4), dihedrals (3, one per wing section), and sweeps (3). These terms are explained below, and visualized in Figures 2.2–2.3.

The lift coefficient is a measure of the lift force generated by the aerofoil, and is related to the angle of attack. Therefore, varying this parameter, specifically the ideal lift coefficient, affects the mean camber line, and the overall curvature and shape of the aerofoil.

The chord line intersects both edges of the aerofoil, and the chord is the straight distance between the edges.

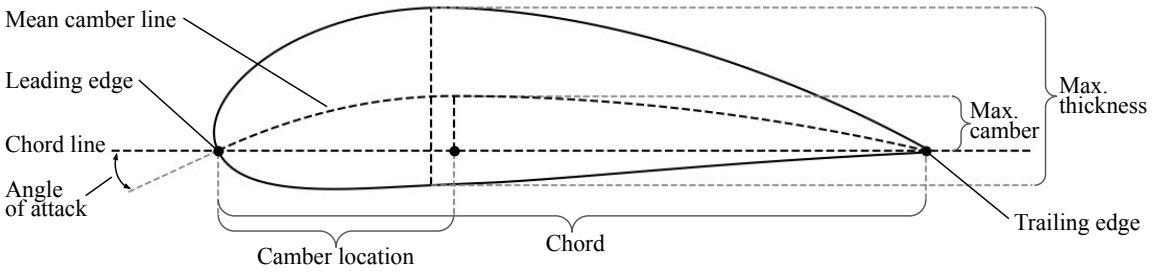


Figure 2.2: Example of aerofoil with relevant terms displayed.

The twists, dihedrals, and sweeps all refer to angles defining how each section of the blade is positioned and rotated in relation to the other sections. These angles are illustrated in Figure 2.3.

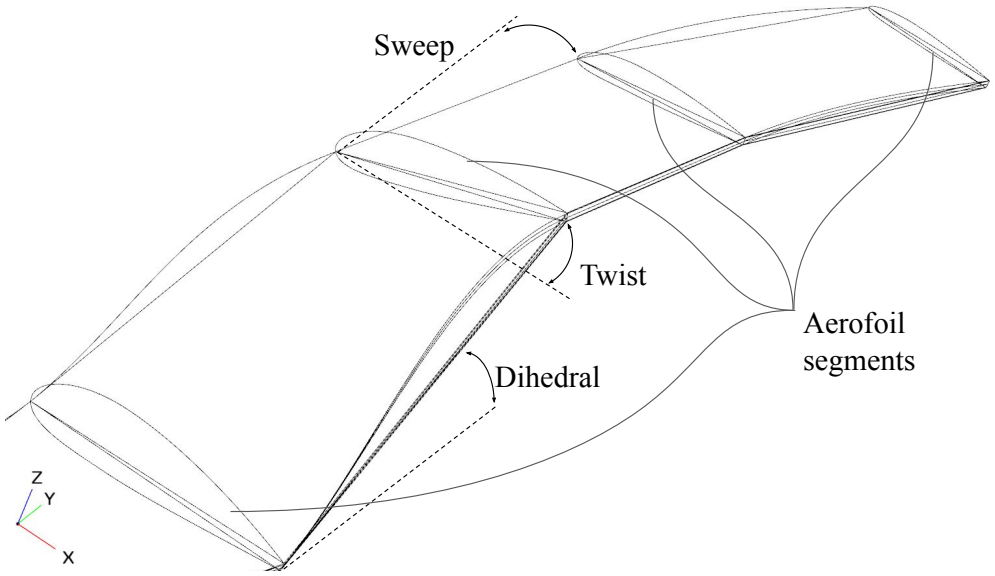


Figure 2.3: Schematic visualization of the terms dihedral, sweep, and twist. The dihedrals are angles in the YZ-plane of the figure, the sweeps in the XY-plane, and the twists in the XZ-plane. Note that every section has separate values for all angles, even though only one example of each angle is shown. The four aerofoil sections are also marked.

Variations in geometry based on these parameters cause differences in how exactly the fluid flows through the turbine, and thus result in different forces and torques. This is why studying such parameters is important for the design of the turbine and performance optimization.

Other parameters related to the turbine blade geometry in the model were not studied and kept constant. This for example includes the length of the blade (not to be confused with the rotor radius, which is affected by the bend of the blade and thus not constant), and the exact formula for the aerofoil curvature. Furthermore, the model had the maximum thickness for each aerofoil segment, as a fraction of the chord, constant at 0.25, 0.20, 0.15, and 0.10, respectively, and the camber locations were 0.30 of the individual chords. Camber location here refers to the point where the mean camber line is maximized. This line marks the middle between the upper and lower surfaces of the wing. These properties of the turbine model were not modified in this project.

2.2 Fluid dynamics

This section presents a theoretical background to the physics of fluid dynamics. The computational simulation process relies on these concepts to estimate how the fluid interacts with the turbine in reality. This section is based on [18].

2.2.1 Equations of conservation

The equations of conservation for mass, momentum, and energy can be derived by studying the rates of change for the relevant physical quantities. This results in sets of volume and surface integrals, which can be evaluated over volume regions through which the fluid is flowing.

In this section, the fluid mass is described by integrating its density ρ over the volume V, which is constrained by a surface S. The inflow velocity is here denoted $\vec{u}$, and other relevant physical quantities are typically expressed as specific quantities (i.e. per unit mass). The volumetric flux ϕ_f through a surface segment is described by $d\phi_f = d\vec{S}\cdot\vec{u}$.

Gauss's theorem is applied to rewrite integrals over surfaces as volume integrals.

The conservation of mass can be explained by that the rate of increase of the mass inside a fixed volume must equal the rate of mass inflow through the volume surface. This can be formulated as

$$\begin{aligned}\int_V dV\,\partial_t\rho = -\int_S d\vec{S}\cdot(\rho\vec{u}) = -\int_V dV\,\vec{\nabla}\cdot(\rho\vec{u}) \implies \\ \implies \partial_t\rho + \vec{\nabla}\cdot(\rho\vec{u}) = 0\,.\end{aligned} \tag{2.6}$$

The conservation of momentum is derived by studying the momentum of a fixed mass of particles. In particular its rate of change, which is represented by the material derivative D_t, with $D_t\vec{\psi} = \partial_t\vec{\psi} + \vec{u}\cdot\vec{\nabla}\vec{\psi}$. This must equal the sum of the involved forces, including all internal body forces per unit mass $\vec{b}$, e.g. gravity, and the external net force on the surface, which is represented by the stress tensor $\boldsymbol{\sigma}$. The result is

$$\begin{aligned}\int_V dV\,\rho D_t\vec{u} = \int_V dV\,\rho\vec{b} + \int_S d\vec{S}\cdot\boldsymbol{\sigma} = \int_V dV\left(\rho\vec{b} + \vec{\nabla}\cdot\boldsymbol{\sigma}\right) \implies \\ \implies D_t\vec{u} = \vec{b} + \frac{1}{\rho}\vec{\nabla}\cdot\boldsymbol{\sigma}\,.\end{aligned} \tag{2.7}$$

The energy is constant over time in an isolated system, according to the law of conservation of energy. The energy equation can be derived by formulating the material rates of change for the specific kinetic and thermal energies, $K = \vec{u}^2/2$ and e, in the fixed mass of particles in the volume. The rates of change are affected by both influx and internal sources of power and heat. Internal power sources $\rho\vec{b}\cdot\vec{u}$ represent changes in potential energy caused by body forces, while power influxes $(d\vec{S}\cdot\boldsymbol{\sigma})\cdot\vec{u}$ represent strain energy changes. The heat is affected by internal heat sources with strength per unit mass r, and surface heat flux per unit area $\vec{q}$ (direction outward). This can be written as

$$\begin{aligned}\int_V dV\,\rho D_t\left(e+K\right) = \int_V dV\,\rho\left(\vec{b}\cdot\vec{u} + r\right) + \int_S\left((d\vec{S}\cdot\boldsymbol{\sigma})\cdot\vec{u} - d\vec{S}\cdot\vec{q}\right) = \\ = \int_V dV\left(\rho\vec{b}\cdot\vec{u} + \rho r + \vec{\nabla}\cdot(\boldsymbol{\sigma}\cdot\vec{u} - \vec{q})\right) \implies \\ \implies D_t\left(e+K\right) = \vec{b}\cdot\vec{u} + r + \frac{1}{\rho}\vec{\nabla}\cdot(\boldsymbol{\sigma}\cdot\vec{u} - \vec{q})\,.\end{aligned} \tag{2.8}$$

2.2.2 Temperature and heat capacity

Temperature and heat capacity are two other important properties in fluid dynamics. The temperature T affects the energy conservation through its relation to $\vec{q}$ modelled by Fourier's law,

$$\vec{q} = -\kappa\vec{\nabla}T \ , \tag{2.9}$$

where κ is thermal conductivity. The specific heat capacity c_V relates the temperature to the thermal energy, defined by

$$c_V = \left(\frac{\partial e}{\partial T}\right)_V \ . \tag{2.10}$$

2.3 Optimization

Optimization of a system or design refers to the process of making it produce the best possible result, potentially conditional on given constraints [19]. This is often formulated as the maximization or minimization of an algebraic function describing the system. In this project, the primary concern was to maximize the power coefficient, based on a function $f(\boldsymbol{x})$ estimating the turbine performance via CFD simulations:

$$\begin{aligned} &\max f(\boldsymbol{x}) \ , \text{ with} \\ &\boldsymbol{x} = \Big[x_1, \quad x_2, \quad ..., \quad x_{18}\Big] \ , \end{aligned} \tag{2.11}$$

where $\{x_i\}$ were the 18 geometric parameters defined in Section 2.1.3. Some algorithms implemented and analysed in the project therefore focused only on this goal.

The design of a turbine can however also be formulated as a multi-objective optimization (MOO) problem [8], [9], a process that involves the simultaneous optimization of multiple, possibly conflicting, criteria [20]. This is relevant for the project, since both C_P and C_T are of interest.

2.3.1 Multi-objective optimization

When there are several performance measures of interest, solving the optimization problem may require trade-offs; optimizing all objectives may be impossible, and there is generally no single solution [20]. This type of problem can be formulated formally as

$$\begin{aligned} &\max / \min \boldsymbol{y} \ , \text{ with} \\ &\boldsymbol{y} = f(\boldsymbol{x}) = \Big[f_1(\boldsymbol{x}), \quad f_2(\boldsymbol{x}), \quad ..., \quad f_n(\boldsymbol{x})\Big] \in Y \ , \\ &\boldsymbol{x} = \Big[x_1, \quad x_2, \quad ..., \quad x_m\Big] \in X \ , \end{aligned} \tag{2.12}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$, X, and Y are the decision vector, objective vector, parameter space, and objective space, respectively. The objective function f maps the m parameters to n objectives.

In this project, the decision vector consisted of $\boldsymbol{x}$ as defined above in Section 2.3, and the objective vector consisted of functions estimating C_P and C_P/C_T, chosen so that both were to be maximized. The parameter space was defined by lower and upper boundaries defined for each parameter.

Comparing different decision vectors is done based on their objective vectors using the term dominance [20]. Mathematically, in a maximization problem, considering two decision vectors $\boldsymbol{a}, \boldsymbol{b} \in X$, then $\boldsymbol{a}$ dominates $\boldsymbol{b}$, if and only if

$$\begin{aligned} &\forall\, i \in \{1, 2, ..., n\} : f_i(\boldsymbol{a}) \geq f_i(\boldsymbol{b}) \text{ , and} \\ &\exists\, j \in \{1, 2, ..., n\} : f_j(\boldsymbol{a}) > f_j(\boldsymbol{b}) \text{ .} \end{aligned} \tag{2.13}$$

All non-dominated decision vectors, i.e. those for which no component in the objective vector can be increased without decreasing another, belong to the solution set known as the Pareto-optimal set, or the Pareto front.

2.3.2 Optimization methods

Solving an optimization problem relies on determining the output result for a sufficient set of inputs. For large parameter spaces, all decision vectors generally cannot be evaluated. Exactly which decision vectors to study needs to be decided, and this process varies between different optimization algorithms.

The methods studied in this project are explained in Sections 3.3.1-3.3.4. For maximization of a single objective, implementations of Nelder-Mead, simulated annealing, and Gaussian processes were tested. The genetic algorithm NSGA-II was used for multi-objective optimization of both C_P and C_P/C_T.

3.
Methods

In this chapter, an outline is provided over the procedural structure of the project, which is illustrated by a flow chart in Figure 3.1. This primarily consisted of the fluid flow simulations, based on which the turbine performance was determined, and the optimization process, where the results of multiple simulations were analysed and processed to reach conclusions regarding the parameters determining the turbine design. Section 3.1 provides an overview of the general simulation structure, whereas Section 3.2 outlines the project specific setup. The tested optimization algorithms and how they were implemented are explained in Sections 3.3–3.4.

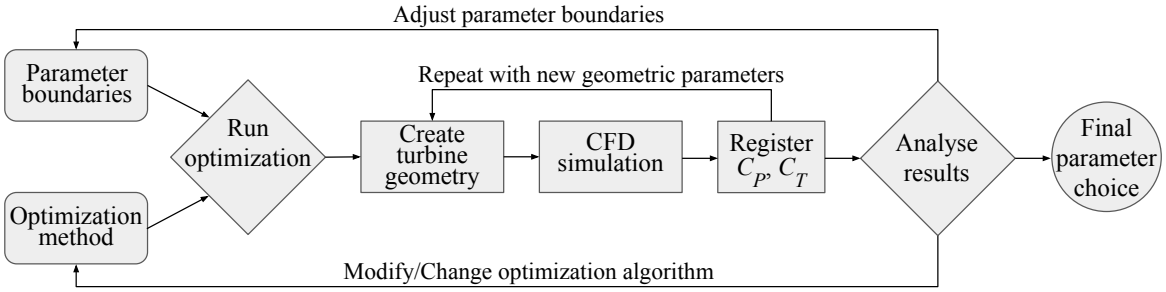


Figure 3.1: Flow chart of the optimization setup.

3.1 Fluid flow simulation framework

This section contains an overview of the simulation methods, including the computer software and mathematical techniques, utilized to estimate the performance of the turbine. The simulations were performed using OpenFOAM, an open-source CFD software, which allows for many flexible options and high robustness and stability in the estimation of fluid flows. The following Sections 3.1.1–3.1.4 are primarily based on [18].

3.1.1 Computational fluid dynamics

The concept of computational fluid dynamics refers to the estimation of fluid flow characteristics based on calculations and numerical analysis. CFD relies on the equations described in Section 2.2.1, as well as physical models of for example viscosity, thermodynamics, and turbulence. Analytical solutions require simple systems and geometries, therefore numerical methods are usually more effective for solving the equations.

In these methods, the continuous flow is approximated by discrete physical variables. Time is divided into intervals, and the spatial domain is represented by cells of a mesh, where each cell is associated with discrete values replacing the fields of the physical properties, e.g. velocity, pressure, and temperature. This results in sets of linear equa-

tions instead of differential equations. The OpenFOAM setup used in this project relies on discretization utilizing the finite volume method.

3.1.2 Finite volume method

The finite volume method is used to model a system by dividing its solution domain into a mesh of multiple connected small volume regions. These are referred to as control volumes, and the fluid flows in and out of their bounding surfaces. The equations of conservation (Eq. 2.6–2.8) can be applied to each volume segment, to balance the inflows and outflows.

The volume mesh can comprise cells of any irregular polyhedral shape, and each cell is defined by its faces, edges, and vertices, as shown in Figure 3.2, which provides an example of two connected three-dimensional cells. The number of faces of each cell and edges of each face have no upper limits.

All faces not on the outer boundary of the solution domain are shared between both neighbouring cells. After generating the mesh, each cell is given an index $i = 1, 2, ..., N$, where N is the total number of cells.

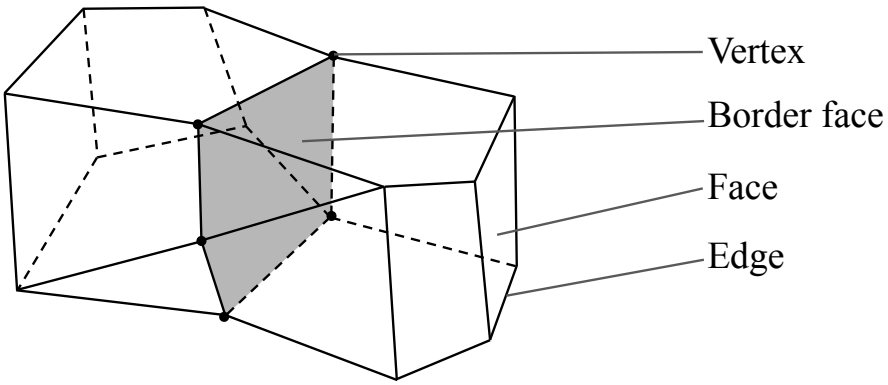


Figure 3.2: Two connected cells, where the grey face constitutes a shared border.

3.1.3 Equation discretization

Each cell in the mesh is associated with discrete values for all the relevant physical quantities, e.g. temperature, pressure, and velocity, here generally denoted ψ. The fields are thus seen instead as arrays of values, ψ_i. Discretization generates a linear equation for each cell and quantity, with matrix and source coefficients a_{ij} and b_i respectively, corresponding to each cell pair ij and cell i. This results in matrix equations on the form

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} \implies \boldsymbol{A}\,\boldsymbol{\psi} = \boldsymbol{b}\,, \tag{3.1}$$

where $\boldsymbol{\psi}$ here is the notation for the array. Note that for arrays of vector field variables (e.g. velocity), there is one equation in each dimension (e.g. $\boldsymbol{u_x}$). The finite volume method generates separate matrix equations for each physical variable, which are solved iteratively, meaning that each solution variable may depend on current values of other variables in the source vector.

Each cell only contributes to the equations corresponding to itself and its nearest neighbours, meaning that only the matrix elements in each row associated with these cells are non-zero. Most elements in the matrices are thus zero, which is a computational advantage, since these coefficients need not be stored in the processing memory.

The matrix and source elements are constructed by calculating each individual term, such as gradients and fluxes, from the relevant equations and boundary conditions. For this process, the finite volume method in OpenFOAM applies numerical discretization schemes selected by the user in the simulation setup.

3.1.4 Equation solving

In the finite volume method, the matrix equations are generally solved by employing iterative, approximative methods. Starting from initial values for all variables, each ψ_i is solved for, which is repeated with increasing accuracy for each iteration, until convergence is reached. Iterative methods are preferred, since exact numerical solutions are too cost expensive for large meshes, as they rely on Gaussian elimination with a computational cost $\propto N^3$.

Since the exact solutions are unknown, the convergence can be defined based on the residual, i.e. the change in solutions between sequential iterations. The tolerance for the residual is usually set to ≈ 0, meaning that convergence is assumed when the change in ψ_i is sufficiently small. OpenFOAM provides built-in versions of solvers and algorithms for these processes.

3.2 Simulation and domain setup

OpenFOAM simulations follow a structure defined and communicated to the software through a file directory system. The following Sections 3.2.1–3.2.5 discuss necessary parts of the specific settings used in the initialization of the OpenFOAM simulations in the project, and contain explanations based on [21]. The simulation process was designed for a balance between minimum runtime and accurate results, and was kept constant during the whole course of the project.

After defining the OpenFOAM structure, the CFD simulation process and subsequent calculation of the coefficients C_P and C_T were primarily treated as a black-box function, where the input, or decision vector, was an array of the geometric parameters, and the output was C_P (and in some cases also C_P/C_T). This was adapted into a callable function.

3.2.1 Turbine model geometry and mesh

Each simulation was preceded by the creation of a model of the turbine blade geometry to evaluate. The first part of the black-box function collected the geometric data input and then controlled the creation of the model. This was performed with OpenVSP, a parametric geometry software tool used for creating 3D models of wings and generating files compatible with OpenFOAM simulations.

The mesh was generated automatically by the software, and consisted each time of around 400 000 cells and 500 000 vertices. A 2D view of the mesh around the turbine can be seen in Figure 3.3.

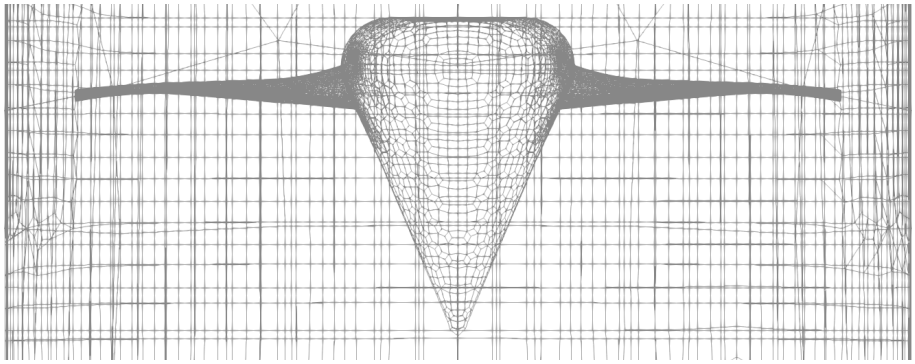


Figure 3.3: 2D view of the turbine mesh.

3.2.2 Discretization schemes

In the setup of the simulation, numerical schemes were specified for the discretization of the time derivative, gradients, divergence terms, Laplacian terms, surface-normal gradients, and distance to wall calculations. See Appendix A.1 for how the schemes were defined in the OpenFOAM setup. The following paragraphs explain the reasoning behind the scheme choices.

This project model was based on a fixed TSR = 5 for the turbine and a time-independent fluid flow. The steady-state scheme was therefore chosen for the time derivative terms, $\partial_t(\cdots)$, thus setting them to zero.

The gradient terms, $\vec{\nabla}(\cdots)$, were calculated with a least-squares approximation scheme using both cell centres and vertices, with the added requirement that values on the cell faces cannot exceed bounds of neighbouring cells. These options are well suited for complex meshes.

The divergence terms included three advection terms, $\vec{\nabla} \cdot (\phi_f(\cdots))$, and diffusion, $\vec{\nabla} \cdot \nu_{\text{eff}} (\vec{\nabla}\vec{u})^{\text{T}}$, where ν_{eff} is effective viscosity. Each term had an individually defined scheme, however Gauss integration was used in all cases. The velocity advection terms were discretized using a bounded, second-order scheme interpolating values with an upwind bias, i.e. biased towards the cell in the direction against the flow. For the advection of specific turbulent kinetic energy and dissipation rate, a bounded, pure upwind scheme was used. Although less accurate, upwind schemes were preferred for advection due to their stability. In the case of diffusion, linear interpolation was used.

Laplacian terms, $\Delta(\cdots)$, were discretized using Gauss integration and linear interpolation, and limiting gradients not to be too sharp, to ensure stability.

Interpolation of values from cell centres to face centres were controlled by a second-order linear scheme.

Components of gradients normal to face surfaces need an individual scheme, here a gradient limiting methods was used due to a high non-orthogonality in the mesh, to prevent non-physical values.

For turbulence modelling, the distance to the nearest wall for each cell was calculated using a scheme based on equations for waves propagating from the walls through the mesh.

3.2.3 Turbulence modelling

To model the turbulence in the flow, and the viscous stress in the fluid, the Reynolds-averaged simulation option was specified, with a k-ω shear stress transport model. This

is primarily based on the equations of conservation for specific turbulent kinetic energy k and dissipation rate ω.

3.2.4 Boundary and initial conditions

Boundary and initial conditions were set for the velocity field $\vec{u}$, pressure field p, specific turbulent kinetic energy k, turbulent dissipation rate ε, specific turbulent dissipation rate ω, and turbulent kinematic viscosity ν_t. Conditions were set for each part of the domain: the internal field domain, and all boundary fields, including the far-field (open extension of the domain), inlet, outlet, and turbine surface.

The initial velocity was defined as 1.5 m/s directly towards the turbine. A summary of the relevant conditions is visible in Table 3.1, using OpenFOAM notation. A dictionary excerpt of the implementation is shown in Appendix A.2. The necessary terms are explained below.

Table 3.1: Summary of initial and boundary conditions.

Field	Internal field	Far-field	Inlet	Outlet	Turbine surface
$\vec{u}$	`uniform (0 0 0)`	`freestream uniform (1.5 0 0)`	`fixedValue uniform (1.5 0 0)`	`inletOutlet uniform (0 0 0)`	`fixedValue uniform (0 0 0)`
p	`uniform 0`	`freestreamPressure 0`	`zeroGradient`	`fixedValue uniform 0`	`zeroGradient`
k	`uniform 0.06`	`freestream $internalField`	`fixedValue $internalField`	`zeroGradient`	`kqRWallFunction $internalField`
ε	`uniform 0.0495`	`freestream $internalField`	`fixedValue $internalField`	`inletOutlet $internalField`	`epsilonWallFunction $internalField`
ω	`uniform 0.0495`	`freestream $internalField`	`fixedValue $internalField`	`inletOutlet $internalField`	`omegaWallFunction $internalField`
ν_t	`uniform 0`	`freestream $internalField`	`calculated uniform 0`	`calculated uniform 0`	`nutkWallFunction uniform 0`

For the internal domain, the initial conditions were applied to all internal cells with the `uniform` condition. Non-zero values were used to define the fields for k, ε, and ω.

To ensure a natural flow in and from the extended far-field region, the `freestream` condition was used. Here, the velocity was specified to $u_x = 1.5\,\mathrm{m/s}$, and the remaining variables used the previously defined internal values, apart from p where a pressure specific option was used.

For the inlet boundary, `fixedValue` was used to apply the values as constants to all faces of the boundary. The velocity was again set to $u_x = 1.5\,\mathrm{m/s}$ to specify that the inflow matched the far-field flow. The surface-normal pressure gradient was set to zero with `zeroGradient`, to ensure dependence on the internal flow. The ν_t values were initialized to zero, and then specified to be solved for, with `calculated`.

To prevent reverse flow and allow a natural outflow, the `inletOutlet` condition was used for the outlet $\vec{u}$, ε, and ω.

The turbine surface boundary required specific wall functions to model the flow behaviour using the turbulence model.

3.2.5 Solvers and algorithms

Settings for the equation solving methods and algorithms were defined for the fluid velocity $\vec{u}$, pressure p, flux ϕ_f, specific turbulent kinetic energy k, and specific turbulent dissipation

rate ω. The Gauss-Seidel and the geometric-algebraic multi-grid (GAMG) methods were used as solvers.

The former is a simple, iterative method with a computational cost $\propto N$. Here, the system of equations is rewritten into expressions for all ψ_i. Then, in each iteration, ψ_1 is updated first based on the previous ψ_i values, and this result is used for the remaining equations, and so on. This method is effective when convergence is reached quickly, in around ≤ 10 iterations.

The latter is a multi-grid method that starts from the original mesh, and then successively forms coarser meshes, until the matrix equation is small enough to be efficiently solved exactly. The method solves for corrections in ψ_i that are then applied to the finer meshes. Finally, a few iterations of the Gauss-Seidel method can be used for the solution. GAMG creates the courser meshes by pairwise agglomeration, i.e. joining together the pairs of cells with the largest shared face area.

A more detailed view is shown in Appendix A.3.

3.3 Optimization algorithms and implementations

In this section, the optimization methods used in the project are presented. When the goal is to minimize the number of evaluations of the function during the optimization, in this case the number of CFD simulations, the selection of each new decision vector is crucial to the optimization process. The implemented algorithms employ different strategies to propose new decision vectors to evaluate, which are explained here.

3.3.1 Nelder-Mead

The Nelder-Mead (NM) method is a simplex search algorithm commonly used for multi-dimensional optimization problems without derivatives and constraints [22]. A simplex is a shape of $N+1$ vertices in N-dimensional space (e.g. a line segment in $\mathbb{R}^1$, or a triangle in $\mathbb{R}^2$).

The Nelder-Mead algorithm starts from a decision vector $\boldsymbol{x}_0$ with the evaluation of the objective function f, and repeats this with new inputs N times, to produce a simplex consisting of $N+1$ non-degenerate points $\boldsymbol{x}_0, ..., \boldsymbol{x}_N$ for which the function outputs $f(\boldsymbol{x}_0), ..., f(\boldsymbol{x}_N)$ are known. The goal is then to decrease the size of the simplex by iteratively evaluating more points, and comparing the new objective values with the results at the vertices.

In the project, this method was implemented both using the R library `neldermead` and the Python library `scipy`. The method was primarily used as a local search method, to get a set of function evaluations around a given decision vector as a starting point.

3.3.2 Non-dominated sorting genetic algorithm II

Genetic algorithms are based on properties of natural selection and genetics, and are commonly used for multi-objective optimization [23]. Each iteration, the algorithm analyses a set of decision vectors referred to as strings, where each parameter is treated as an artificial gene. This population is systematically modified into new generations of strings, or offspring, mimicking the process of evolution. The main three components in genetic algorithms are selection, crossover, and mutation.

Selection corresponds to the survival of the fittest, where the fitness in this case is the CFD simulation result. The goal is that the fittest strings should be favoured in producing the next generation [24]. Selection can, for example, be achieved through a tournament process, where states are drawn randomly from the parent population, and the one with the highest fitness enters the mating pool [23].

Crossover determines how the offsprings are produced based on the parent genes, which is a way to control the exploration capacity of the algorithm. The offsprings are placed in the new generation, and the pairwise mating process continues until the new population is complete. The different crossover operators tested in this project are listed in Table 3.2. From the Python `pymoo` library [25], simulated binary crossover (SBX) and differential evolution crossover (DEX) were used, while blended crossover (BLX-α) [26] was implemented separately.

Table 3.2: Crossover operators and definitions.

Operator	Definition
SBX	$\begin{cases} x_i^{\text{offs},1} = 0.5 \cdot \left((x_i^{\text{parent},1} + x_i^{\text{parent},2}) - \beta_q (x_i^{\text{parent},2} - x_i^{\text{parent},1}) \right) \\ x_i^{\text{offs},2} = 0.5 \cdot \left((x_i^{\text{parent},1} + x_i^{\text{parent},2}) + \beta_q (x_i^{\text{parent},2} - x_i^{\text{parent},1}) \right) \end{cases}$, with β_q calculated based on the parent genes, a variable $\sim U[0,1]$, and a crossover distribution index parameter
DEX	$x_i^{\text{offs}} = \begin{cases} x_i^{\text{parent},0} + F \cdot (x_i^{\text{parent},1} - x_i^{\text{parent},2}) & \text{if } M_i = 1, \\ x_i^{\text{parent},0} & \text{if } M_i = 0, \end{cases}$ where M_i is based on a binomial probability distribution, and F is a randomized weight factor
BLX-α	$x_i^{\text{offs}} \sim U[x_i^{\text{parent},1} - \alpha \cdot (x_i^{\text{parent},2} - x_i^{\text{parent},1}),\ x_i^{\text{parent},2} + \alpha \cdot (x_i^{\text{parent},2} - x_i^{\text{parent},1})]$, where α is a small constant, and $x_i^{\text{parent},2} > x_i^{\text{parent},1}$

Mutation is the generation of new, random parameter values in the offspring genes, and is controlled by a probability parameter [24]. This operation is necessary to reach convergence at the global optimum as opposed to a local one. The downside is that the introduced divergence will slow down the convergence.

Non-dominated sorting genetic algorithm (NSGA) was one of the first algorithms of this kind, and NSGA-II is an improved version [27]. Specifically, NSGA-II was developed to address flaws regarding e.g. the non-dominated sorting computational cost, and the lack of elitism. Because of its sorting procedure, the original NSGA has a computational complexity of $O(MN^3)$, where M, and N are the number of objectives, and population size, respectively. The procedure used by NSGA-II, where for each string in the population, the number of strings it dominates and is dominated by are calculated, reduces the complexity to $O(MN^2)$. Elitism refers to the practice of keeping the best solutions each new generation, regardless of those being among the parents of offsprings. This means that offsprings are discarded if they are dominated by the parent strings, which ensures that the fittest decision vectors stay in the population, and can speed up the performance.

NSGA-II was primarily implemented with `pymoo`, either with a randomized or a pre-defined initial population. Using the data collected during the project, the fittest decision vectors were identified and could be used as the setup for the algorithm.

The probability for crossover to occur was typically set in the interval $[0.7, 0.95]$, to ensure that new strings were generated each time, and increase the diversity in the population. The mutation rate was kept quite low, within the range $[0.2, 0.5]$, to still

allow for new values, but prevent too many simultaneous replacements. The population size was usually between 20 and 50, to ensure the creation of a few new generations each optimization run.

3.3.3 Simulated annealing

The simulated annealing algorithm (SA) is a random-search optimization method incorporating a temperature property, based on the cooling process of metals [28]. Similar to how atom movements slow down as the temperature of a metal decreases, the search process of the algorithm grows more static with each iteration.

The initial state is a randomized decision vector within the pre-defined parameter bounds, and is stored as the currently accepted state, S_i, with the corresponding function output $f(S_i)$. Each iteration includes the evaluation of a new decision vector, which is selected by applying a change (e.g. a step size parameter) to the current state. The new output $f(S_{i+1})$ is then compared to the current state and (in a maximization problem) accepted with the probability

$$P_{\text{accept}} = \begin{cases} 1 & \text{if } f(S_{i+1}) \geq f(S_i) \ , \\ \exp\left(\dfrac{f(S_{i+1}) - f(S_i)}{T_i}\right) = \exp\left(\dfrac{\Delta f_i}{T_i}\right) & \text{if } f(S_{i+1}) < f(S_i) \ , \end{cases} \tag{3.2}$$

known as the Metropolis criterion, where T_i is the temperature that decreases with each iteration. This means that a better state is always accepted, but that there is a probability of accepting a worse state, which is reduced as the iterations progress. The reason for this is to decrease the risk of getting stuck in local maxima. The result is that in the early iterations, the algorithm accepts almost all new states, but slowly becomes more selective of which states to accept.

The definition of the temperature and its decreasing is crucial to the convergence of the algorithm. In this project, the temperature was controlled with a focus on balancing the trade-off between exploration and exploitation. Different choices were tested for the initial temperature, $T_0 \in [0.1, 10]$, and the cooling was modelled according to

$$T_i = \frac{T_0}{i} \ . \tag{3.3}$$

The algorithm was implemented in Python using a loop structure to perform the necessary steps each iteration. To reduce the dependence of the initial state, a version of SA with multiple simultaneously stored states, or 'walkers', was also tested. Essentially equivalent to running multiple sequential simulations, this had the benefit of ensuring that each walker evaluated the same number of states and could be terminated at the same time. In this version, one state and probability per walker were evaluated each iteration.

In order to maintain control over the search, it was initially limited to a parameter space defined by boundaries based on previous knowledge and observations. Proposed parameter values outside this space were projected onto the boundary values. However, to improve the flexibility of the algorithm, a version where the boundaries were adjusted when new maxima were found was also tested. After finding a new maximum, each parameter in that state with a value on a boundary had its ranges moved to expand the parameter space in that direction, otherwise the range was reduced to further limit the search space.

3.3.4 Bayesian optimization with Gaussian processes

The Bayesian approach to optimization relies on Bayesian inference and statistical modelling of the objective function [29]. Constructing and modifying these models can itself be computationally costly, but has potential to result in significant efficiency rewards. Bayesian optimization is therefore commonly applied to black-box functions that are expensive to evaluate.

Bayesian inference is a method of estimating unknown properties of a system based on probability theory. Beliefs about all unknown parameters are represented by probability distributions, where the parameters are treated as random variables. Observed data is then used to iteratively adjust these beliefs and increase their accuracy.

The inference of the objective function value at a given location, $\phi = f(\boldsymbol{x})$, starts with defining a prior distribution, $p(\phi|\boldsymbol{x})$, which represents values for ϕ that are considered probable before performing any measurements, as a way to encode previous knowledge and assumptions about the system into the model. Next, a measurement of the objective function is modelled with the likelihood distribution, $p(y|\boldsymbol{x}, \phi)$, which uses the value of interest ϕ to explain the observed value y.

Using Bayes' theorem, the posterior distribution

$$p(\phi|\boldsymbol{x}, y) = \frac{p(\phi|\boldsymbol{x})\, p(y|\boldsymbol{x}, \phi)}{p(y|\boldsymbol{x})} \tag{3.4}$$

is determined, which gives the distribution of the inferred value given the observed data. To normalize the distribution, the denominator is defined as

$$p(y|\boldsymbol{x}) = \int d\phi\; p(\phi|\boldsymbol{x})\, p(y|\boldsymbol{x}, \phi)\;. \tag{3.5}$$

As more measurements are performed, the previous posterior is used as a prior, which multiplied by a new likelihood results in a new posterior, after renormalization.

The posterior distribution is incorporated into an acquisition function, that is used to determine the next sample. This decision is based on a balance of exploration (sampling where the uncertainty is high) and exploitation (where the model expects the best result). Optimizing the acquisition function, which typically is computationally inexpensive, leads to predictions of where the optimum of the objective function is located. This location can then be chosen as the next decision vector to evaluate.

Gaussian processes (GPs) are commonly used for constructing the prior, and computing the posterior based on observed data [29], [30]. They constitute a class of non-parametric regression models that are useful for modelling complex objective functions. As generalizations of the mathematical properties of multivariate Gaussian (normal) probability distributions, GPs treat the objective function as a set of random variables indexed by input points, forming a stochastic process. Instead of using fixed parameters (e.g. weights), the process is modelled by a mean function and in particular a covariance function (kernel), defined by hyperparameters, over the domain. A GP can then be described as

$$p(f) = \mathcal{GP}(f; \mu, K)\;, \tag{3.6}$$

where μ and K are the mean and kernel functions,

$$\mu(\boldsymbol{x}) = \mathbb{E}[\phi|\boldsymbol{x}]\;, \tag{3.7}$$

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \mathrm{cov}[\phi, \phi'\,|\boldsymbol{x}, \boldsymbol{x'}]\;, \tag{3.8}$$

where $\phi' = f(\boldsymbol{x'})$, which define the expected function value, and the structure of similarities between different data points, respectively.

Table 3.3 shows the different individual kernels tested in the project and their definitions [31]. Combinations (additions) of them were also studied. Hyperparameters optimized during training are denoted with σ. The isotropic kernel definitions are given, i.e. for shared hyperparameters between all input dimensions. Using individual hyperparameters for each dimension, known as auto relevance determination (ARD), was also tested.

Table 3.3: Definitions for the kernels tested in the project.

Kernel	**Definition**
Bias (constant)	$K_{\text{Bias}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_b^2$
Radial Basis Function (RBF)	$K_{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_s^2 \exp\left(- \frac{\|\|\boldsymbol{x} - \boldsymbol{x'}\|\|^2}{2\sigma_l^2} \right)$
Multi-Layer Perceptron (MLP)	$K_{\text{MLP}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_s^2 \frac{2}{\pi} \text{asin}\left(\frac{\sigma_w^2 \boldsymbol{x}^{\text{T}} \boldsymbol{x'} + \sigma_b^2}{\sqrt{\sigma_w^2 \boldsymbol{x}^{\text{T}} \boldsymbol{x} + \sigma_b^2 + 1} \sqrt{\sigma_w^2 \boldsymbol{x'}^{\text{T}} \boldsymbol{x'} + \sigma_b^2 + 1}} \right)$
Exponential	$K_{\text{Exp}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_s^2 \exp\left(- \frac{\|\|\boldsymbol{x} - \boldsymbol{x'}\|\|}{\sigma_l} \right)$
Matérn 3/2	$K_{\text{Matern}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_s^2 \left(1 + \sqrt{3} \frac{\|\|\boldsymbol{x} - \boldsymbol{x'}\|\|}{\sigma_l} \right) \exp\left(- \sqrt{3} \frac{\|\|\boldsymbol{x} - \boldsymbol{x'}\|\|}{\sigma_l} \right)$
Linear	$K_{\text{Lin}}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_s^2 \boldsymbol{x}^{\text{T}} \boldsymbol{x'}$

This optimization method was implemented primarily through the Python library `GPy`.Kernels were constructed with `GPy.kern`, and then used for defining the GPs with `GPy.models.GPRegression`. Training based on typically a minimum of 50 data points was performed with respect to C_P, C_P/C_T, or a sum of the two, for different models. The acquisition function used the model predictions to compute the upper confidence bound (UCB), $\mu + \beta\sigma$, meaning that the sampling was based on both the expected value, and the variance σ, with β being a controlling constant. Each iteration, the acquisition function was optimized either by using built-in `scipy` methods such as Nelder-Mead or Powell's method, using NSGA-II, or by evaluating an array of decision vectors and finding the argument of the maximum.

3.4 General optimization strategies

Aside from the individual implementations of each method, general strategies were developed regarding for example the definition of the parameter search space, and the storing and utilization of the gathered simulation data. Finally, a combination of multiple methods into a single algorithm was also tested, as a way to simultaneously apply the advantages of the different methods.

3.4.1 Parameter space definition and exploration

In order to simplify the optimization problem, the parameter space dimension was reduced so that 18 parameters were studied. The camber locations for all sections were kept constant at 0.3 of the chord length, to keep a realistic and simple shape of the aerofoil.

The first priority of the project was to focus on extensively exploring the parameter space, to learn what parameter values resulted in higher C_P and C_P/C_T, and to reach an initial estimate of the Pareto front, in particular the maximum C_P. As more data was collected, the boundaries were moved to focus on parameter values with better results. Such adjustments were made by manually defining the parameter space to search, e.g. based on studying the parameter ranges for a selection of the latest results, or by automatically loading all the data and determining the ranges based on the best results.

3.4.2 Data utilization

Loading data as part of the optimization run setup was not only used for defining the boundaries, depending on the method. For NSGA-II and Bayesian optimization, previous data could be used as the initial population, or for GP training. For the Nelder-Mead method, the starting point could be chosen as a decision vector corresponding to one of the best objective vectors located so far.

The selection of the data subset to use, e.g. as training data, was adjusted between the runs. Primarily, combinations of the best C_P and C_P/C_T results were used. Other strategies were also tried, such as using the decision vectors yielding the highest sum of C_P and C_P/C_T, or looking only at one of the coefficients.

Each iteration, the training data and new results were stored to identify if new optima were found. This was used for retraining or new local searches.

3.4.3 Combining methods

Combinations of the methods were also tested, in particular an algorithm that combined NSGA-II, GPs, and a local search.After initializing a starting population, these data points were used to produce a new generation with NSGA-II, and to train a GP to predict new decision vectors. Each iteration was then finished with evaluations of samples around the best located C_P and C_P/C_T values. This method was tested with different amounts of evaluations based on NSGA-II and GPs, and criteria for local searches.

3.4.4 Comparing methods and efficiency

When enough simulation data was gathered to draw conclusions regarding the optimal turbine performance, the focus became improving the optimization efficiency. The methods were compared in terms of simulation results, primarily focusing on C_P, within a limited number of simulations. For these tests, the parameter space was fixed to ensure a fair analysis, with boundaries based on the Pareto-set established from the testing.

4.
Results and Discussion

In this chapter, the results are presented and discussed. Firstly, the complete results of the simulations and the located optima are laid out. These were then used as a basis for the performance quality of the optimization methods. The different optimization methods are compared and analysed in terms of their simulation results, and their implementation settings and properties.

4.1 Pareto front

During the initial testing, 60 optimization runs were performed, with a total of 12 000 simulations. The average simulation runtime was approximately 6.9 min/simulation, which implies a significant challenge for this optimization problem. Figure 4.1 shows the best results of the C_P values and C_P/C_T ratios. In particular, the Pareto front is marked and presents 80 identified non-dominated solutions to the optimization problem, which shows an anti-correlation between the coefficients. The results suggest that $C_{P\max} = 0.3902$. A more detailed view of the parameter values can be seen in a parallel coordinate plot in Appendix B.1.

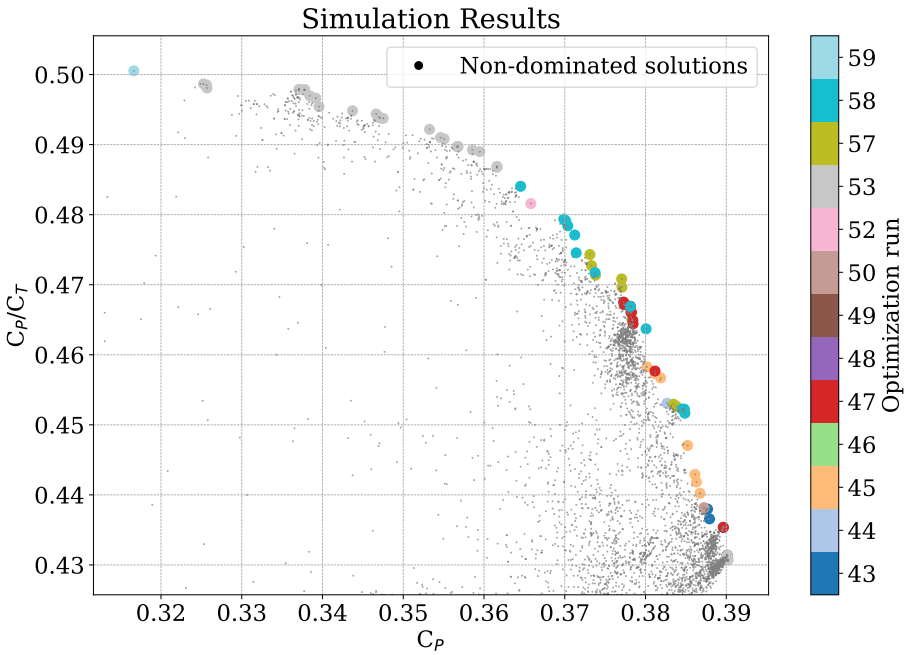


Figure 4.1: Simulation results including the Pareto-optimal set.

4.2 Parameter space exploration

In the process of exploring the parameter space and determining the Pareto front, each optimization method was tested multiple times with different settings, parameter bounds, and data utilization. The size and boundaries of the parameter space had a crucial impact on the results. Multiple optimization runs were needed to understand what parameter values resulted in better turbine performance, which was the initial focus of the project. The parameter space boundaries needed frequent modifying to reach new optima previously outside the space, while simultaneously keeping some restrictions, since a larger space slows down the search. Incorporating automatic adjustments of the boundaries in the algorithms was an important contribution to improving the results in this process.

The amount and quality of the available previous data were also factors in the optimization results for the methods utilizing loaded data. As more and more simulations were performed, this contributed to improving the results, meaning that the quality of these methods depended on the available results. For example, training GPs on higher C_P values allowed for predictions and locations of even higher values. These methods were used in the exploration, and led to locating $C_{P\max}$, but the need for large amounts of data means they are not ideal in terms of efficiency.

Figures 4.2–4.6 show how the results for different methods improved over time, as more simulation data, and information about the parameter boundaries were gathered. The exact boundary limits used for each optimization run can be found in Appendix B.2. The parameter space boundaries were updated before each run, and the methods were alternated rather than tested in order (and therefore numbered individually in the figures). This is why the earlier runs for each method resulted in similarly low values, while the parameter space for the later runs had been updated to improve the results.

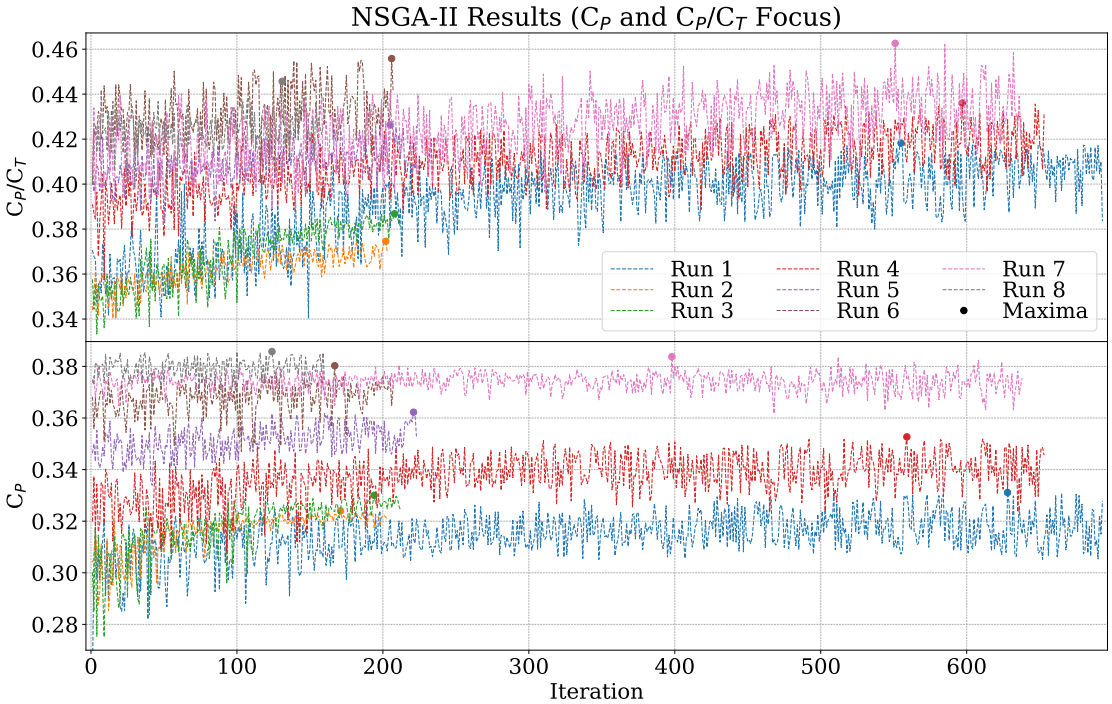


Figure 4.2: Results from NSGA-II optimization runs. The initial populations were either randomized or based on the previous best results. SBX was used with a crossover rate around 0.8, and the mutation rate was around 0.3.

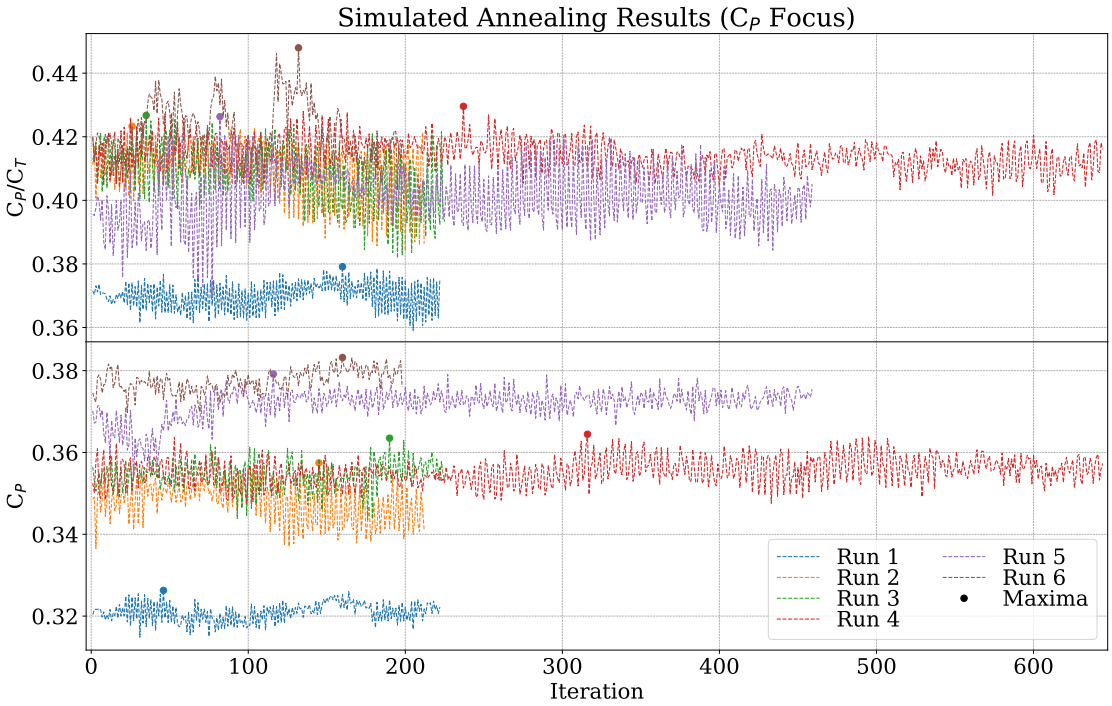


Figure 4.3: Results from SA optimization runs. The temperature and step size parameters were varied between the runs, and different numbers of walkers were used.

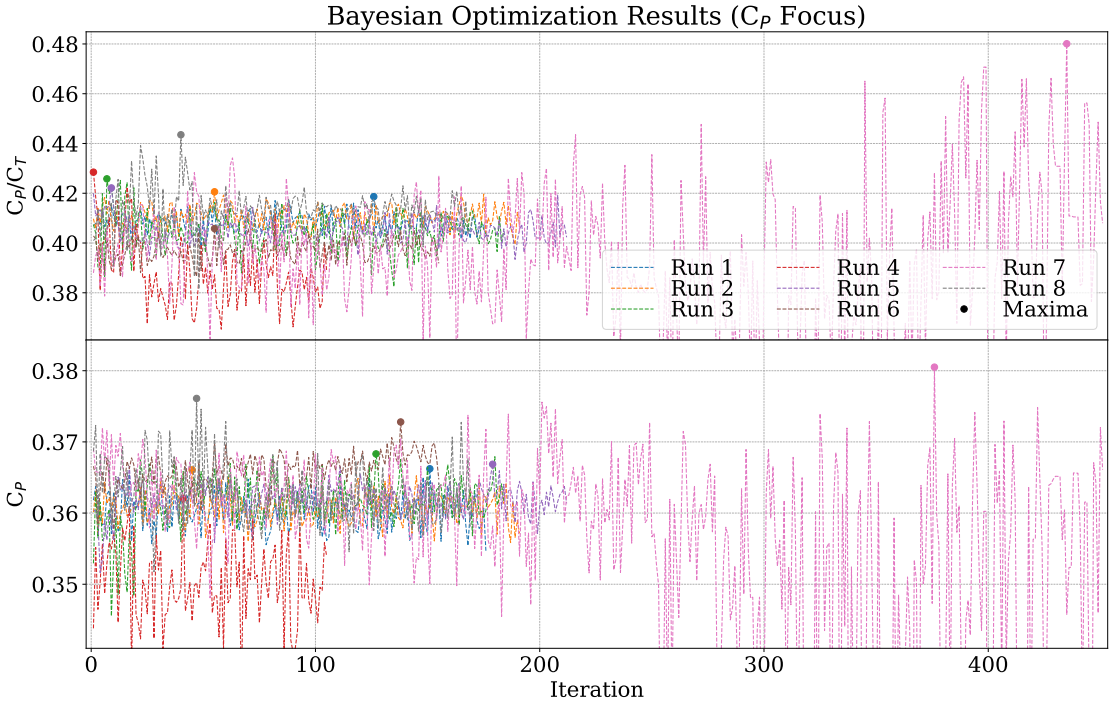


Figure 4.4: Results from Bayesian optimization runs with GPs. The kernels consisted primarily of RBF and MLP kernels with a bias. The UCB acquisition function used $\beta \in [2, 5]$.

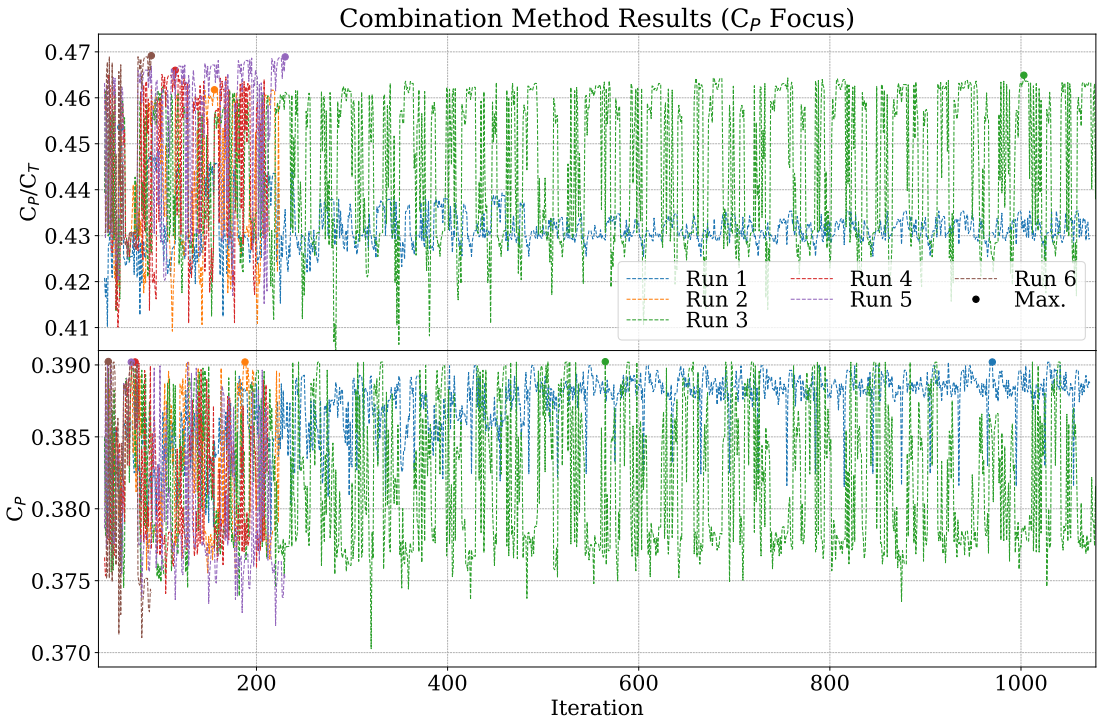


Figure 4.5: Results from combining NSGA-II, GPs, and NM. Each run consisted of similar proportions of evaluations based on NSGA-II, GP predictions alternating between the objectives, and a NM search starting from the best C_P result. The runs were initialized with the best previous results based on both objectives.

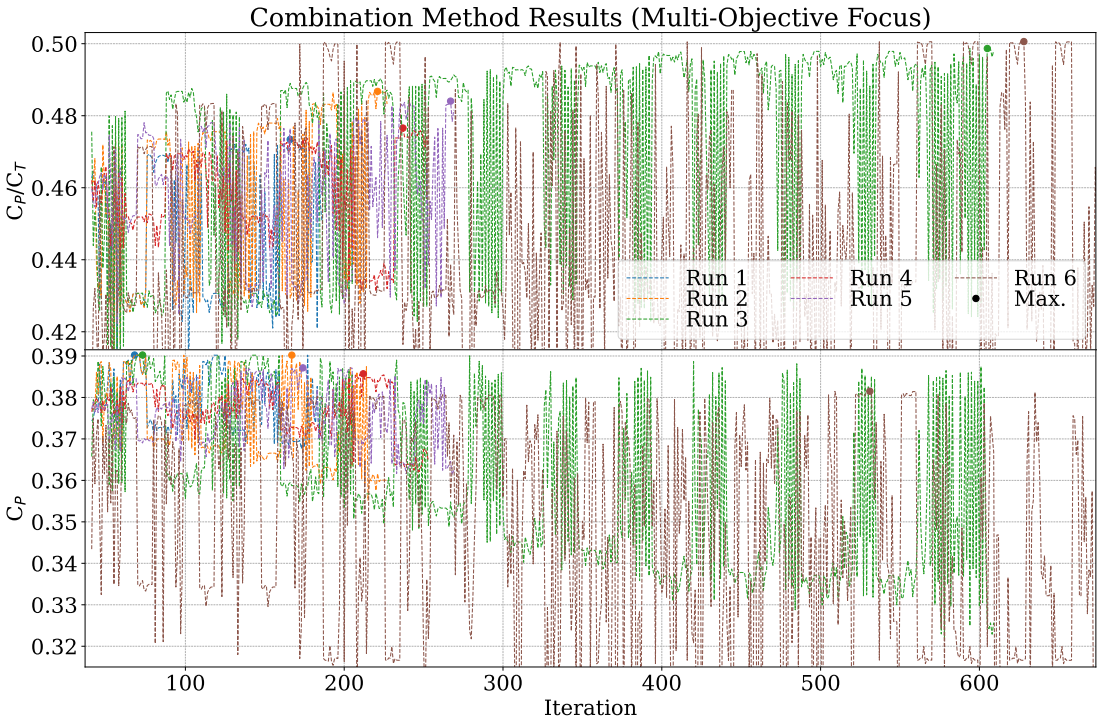


Figure 4.6: Results from the same method as above, with a similar setup, but the NM search was alternated between C_P and C_P/C_T.

Most runs were performed over a day, the longest for about five days. The high dimensionality of the problem combined with the long simulation runtime meant that each method generally required a long time to reach new optima. However, the shorter runs were preferred in the beginning to maximize the testing of different implementations. Due to the risk of getting stuck in local maxima, the runs were often restarted with new parameter bounds, based on studying the distribution of the best simulation results.

4.2.1 NSGA-II results

NSGA-II (Figure 4.2) had the advantage of multi-objective optimization, which likely is the reason it resulted in higher C_P/C_T averages than simulated annealing and Bayesian optimization. It is clear that the located maxima appear toward the end of each run, meaning that the results improved during the runs, and shows a correlation between the result and number of simulations. However, after around 200 iterations, the improvement in simulated C_P is almost non-existent, and the results seem to show signs of convergence, even though large fluctuations and variations still occur. The fluctuations generally decrease to some degree over time, but longer runs would be needed to study when the result converges.

4.2.2 Simulated annealing results

Simulated annealing (Figure 4.3) did not result in any significant improvements in the conducted runs. There were no clear signs of convergence, which can be explained by the chosen parameters, the inherent randomness of the method, and the runtime. With too large step size and temperature parameters, the method behaves similarly to a random search process, which is not expected to result in convergence.

The parameters were changed between the runs, and the two latest runs (5–6) showed some improvement in convergence tendency, using a step size determined by a uniformly sampled perturbation based on the parameter range, scaled by 0.2, and $T_0 = 0.1$. It could be interesting to conduct further testing, and optimize these parameters, which could lead to potential improvements. Due to the high dimensionality of the problem, it is likely that the method requires more iterations per run to get consistent results. Then the temperature could be allowed to cool down further and more slowly.

4.2.3 Bayesian optimization results

Bayesian optimization (Figure 4.4) was here used to focus on C_P exploration, meaning a bias towards the model variance in the acquisition function (large β). Based on these simulation results it is clear that the model predictions were very different each iteration, which points toward large areas of the parameter space being explored.

This is especially noticeable in the seventh run, where a method to automatically adjust the parameter ranges based on new located optima was used, which lead to significant fluctuations and improvements of both coefficients. This was in particular related to increasing the upper boundary for the first chord length, making the width of the first turbine blade section considerably wider. This discovery in turn led to even further improvements in future runs.

The kernels used during this part were different combinations of bias, RBF, and MLP kernels. There likely is a potential that more complex combinations could be used

to improve the results. More details regarding comparisons of kernels is presented in Section 4.3.

4.2.4 Results from combining methods

Due to the utilization of the collected simulation data, combining multiple methods (Figures 4.5–4.6) consistently produced high C_P and C_P/C_T values, and was able to increase C_P and C_P/C_T compared to the individual methods. This was however only possible after using the collected data from the previous methods for NSGA-II selection, GP training, and defining the parameter space.

The first run with this setup located multiple optima with, for the first time, $C_P > 0.39$, with $C_{P\max} = 0.3902$, which was the highest occurring value each run. The number of iterations to reach this value was decreased significantly each run, since the training data improved in quality. Due to the structure of the NSGA-II process, the offspring generated each run were very similar, which lead to similar results each run, and some evaluation repetition. The time to reach the higher values is therefore not so interesting, but rather that no larger C_P was found. This could be a consequence of bias in the training data, causing repeated explorations of roughly the same space. It is possible that increasing the exploration parameter β and the mutation probability could lead to improvements.

This method was tested with different GP training data, to achieve focus on either improving C_P, or C_P/C_T, or both. Based on the differences in the simulation results it is clear that this had the desired effect, and it also shows the anti-correlation between the two coefficients. The differences in fluctuations can likely be explained partially by differences in proportions between GP predictions based on the different objectives.

4.3 Efficiency analysis

The results in Figure 4.1, and $C_{P\max} = 0.3902$ in particular, were used as a basis for fixing the boundaries in this part, which can be seen below in Table 4.1. Methods were compared based on results within a limited runtime of about 260 simulations/run, or about 30 h.

Since NSGA-II and GPs require an initial dataset (as an initial population and training data, respectively), a collection of 40 values per parameter, evenly distributed on each parameter space interval, was randomly recombined to form a data set for these purposes. These samples had objective values within the range $C_P \in [0.348, 0.385]$ and $C_P/C_T \in [0.394, 0.475]$, and were re-used each run to save time.

Table 4.1: Parameter space boundaries during the efficiency comparisons.

Param.	Lift coefficient				Dihedral			Chord				Sweep			Twist			
Segm.	0	1	2	3	1	2	3	0	1	2	3	1	2	3	0	1	2	3
Min.	0.10	0.04	0.25	0.55	11.0	-2.5	2.0	3.40	2.00	1.00	1.90	-8.0	-35.0	40.0	-70.0	-10.0	-3.0	-11.0
Max.	0.90	0.30	0.55	0.90	13.5	-1.0	3.5	5.20	3.00	1.60	2.30	-5.0	-20.0	45.0	-50.0	-9.0	-1.0	-8.0

4.3.1 Limited NSGA-II with different crossover operators

The efficiency of NSGA-II was tested mainly with different crossover operators and crossover rates. A Nelder-Mead local search around the highest C_P result was also performed each time. The Pareto-optimal set of each run can be seen in Figure 4.7, and

further details in Appendix C.1. The best result was $C_P = 0.3881$, which is close to the previous maximum, and three results with $C_P \approx 0.382$ dominated members of, or would have been part of, the previously established Pareto-set. One other run reached $C_P > 0.3870$, while the remaining ones resulted in $C_P \in [0.3861, 0.3869]$. Since the randomized samples included $C_P = 0.385$, this is not a significant improvement.

It is likely that NSGA-II limited by the set conditions is not a strong method for this optimization problem, unless other crossover operators can improve the performance. Further testing could be done regarding other operators and probabilities to evaluate if the efficiency can be improved. Based on the results, SBX seems to be the strongest crossover operator, and more generations with fewer evaluations per generation should be preferred, but there were no large differences between the runs.

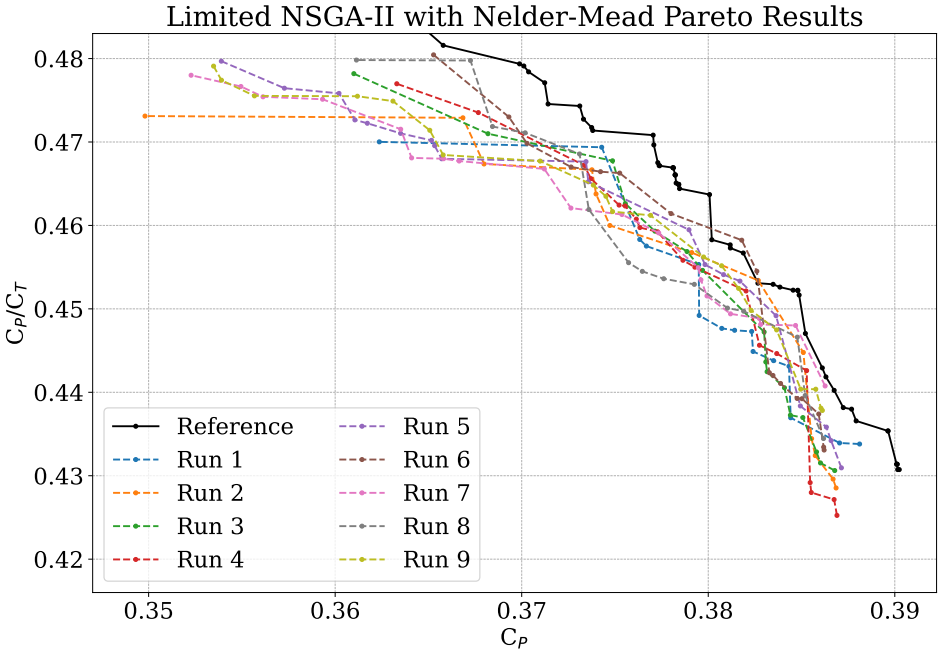


Figure 4.7: Pareto front results from NSGA-II runs where the best results formed the initial simplex for a Nelder-Mead search, with a limit of maximum 260 simulations/run, compared to the results of the initial exploration (Figure 4.1) as reference.

4.3.2 Limited Bayesian optimization with different kernels

To assess the efficiency of Bayesian optimization for this problem, multiple runs were performed with 4–6 different kernel constructions, with one function evaluation per kernel and iteration. The exploration parameter was set at $\beta = 5$ initially, and decreased each iteration to gradually shift focus to exploitation.

The located Pareto-optimal sets are shown in Figure 4.8, while the exact kernels used are listed in Appendix C.2. Values of $C_P > 0.3890$ could be found within 260 simulation with some consistency. All runs were able to improve on the values from the initial samples. The best result was $C_P = 0.3897$, which is very close to the overall maximum, with a difference of 0.1281%. Due to the limited number of evaluations, it is logical that no new maxima were found. One solution that dominated two members of

the previously established Pareto-set was found, with $C_P = 0.3814$ and $C_P/C_T = 0.4577$.

It is possible that the number, complexity, and combinations of different kernels used in each run had an impact on the results, but no significant differences were noted. Based on the results it seems like 4–5 unique combinations of 3–4 of the kernel functions listed in Table 3.3 is a decent strategy. Due to the inherent randomness of GPs, it is however difficult to assess exactly what combinations are truly optimal for this problem. This could perhaps be achieved with further testing.

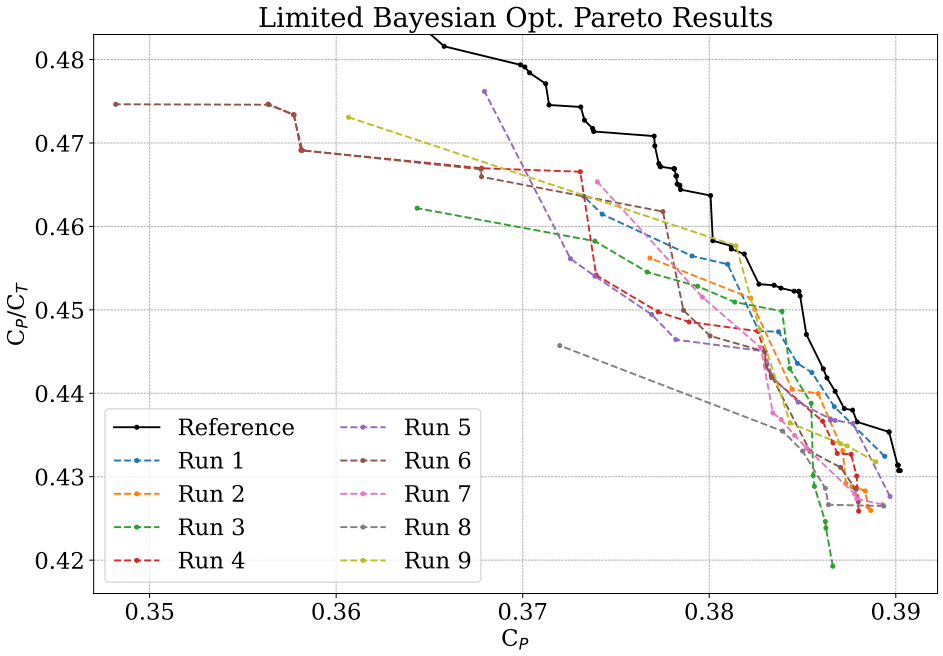


Figure 4.8: Pareto results from Bayesian optimization limited by 260 simulations/run, with the Pareto-set of the initial exploration (Figure 4.1) as reference.

4.3.3 Limited method combination

Combining NSGA-II, GPs, and NM was tried with different settings regarding population size, using the SBX operator, and the kernel combination that produced the highest C_P from the previous tests. The Pareto results can be seen in Figure 4.9, with the maximum $C_P = 0.3891$, and another significant result with $C_P = 0.3890$ and $C_P/C_T = 0.4380$, which dominates one member of the reference Pareto front. This method did not seem as reliable as using GPs only, suggesting that the GP predictions can be quite random and not only influenced by the training data in these cases. The results were better than using only NSGA-II, which shows the advantages of using Bayesian optimization in this problem.

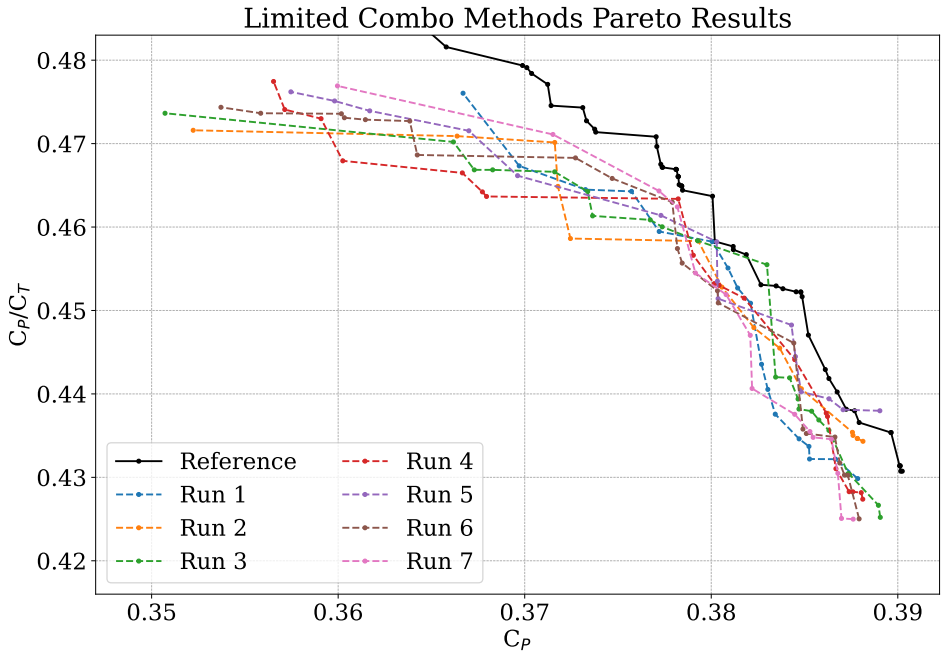
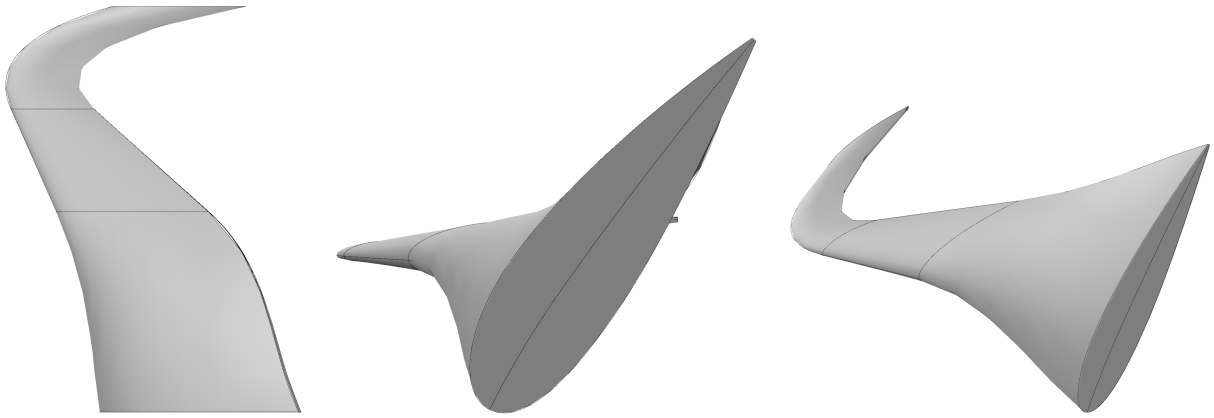


Figure 4.9: Pareto results from combining Bayesian optimization, NSGA-II and NM, limited by 260 simulations/run, with the Pareto-set of the initial exploration (Figure 4.1) as reference.

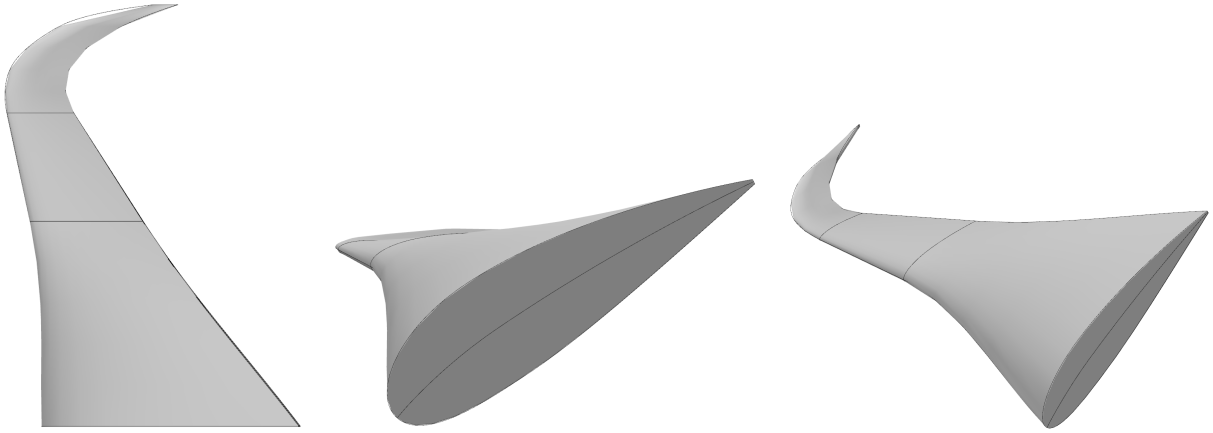
4.4 Geometric results

The turbine blade shapes corresponding to the optimal C_P and C_P/C_T results are shown in Figures 4.10–4.11. Generally, a large chord and twist angle at the base combined with a narrow, curved end resulted in the best performance.



(a) Top view. **(b)** Left view. **(c)** Left isometric view.

Figure 4.10: Turbine blade appearance for maximized C_P.



(a) Top view. **(b)** Left view. **(c)** Left isometric view.

Figure 4.11: Turbine blade appearance for maximized C_P/C_T.

5.
Conclusion

The optimization process of the studied turbine model is a complex and computationally costly problem. Because of the high dimensionality of the parameter space (18), many simulations are required for a successful result. In addition, the simulation speed (6.9 minutes/simulation in this case) leads to a significant runtime, however this is of course also affected by the computational capacity of the hardware used.

The optimal geometries were found after successively using, modifying and combining multiple methods with different properties. Bayesian optimization with a set of different GPs was the most successful individual method, at least for single-objective optimization, considering that a high C_P was the main focus. The ability to model the CFD simulation result, and relationships between the different parameters, based on known data and probability theory seems to be advantageous. While the training process does not guarantee accurate predictions, inaccurate predictions also contribute to more knowledge about the parameter space, which can improve later predictions. Despite the inherent stochastic properties, the model consistently produced at least one strong prediction within the decided limit.

NSGA-II showed some potential, but seemed to require a much longer runtime to reach similar results. Its direct capacity for multi-objective optimization means that it could be preferred depending on the specific demand. The versatile exploration capacity is also a useful property for gathering information about the system.

The parameter space is likely too large for methods like simulated annealing, which also is too random in its search capacity, and Nelder-Mead alone. Nelder-Mead is however a good method for fine-tuning the results achieved by Bayesian optimization or NSGA-II. It is possible that simulated annealing can be used for this purpose as well.

The most successful optimization strategy found in this project involves a combination of methods, in particular NSGA-II and GPs. Since the GPs needed at least about 200 data points for training before finding the optimum each run, NSGA-II can be used first to focus on exploration and collect a diverse training data set.

In terms of efficiency, Bayesian optimization is able to consistently find turbine designs performing on levels close to the overall optimal design result, within the shortest time frame of the tested methods.

The project found that the turbine blade shapes for optimal C_P and C_P/C_T are quite different, despite sharing some similarities, and that a design that maximizes one coefficient generally reduces the other. For maximum C_P, the optimal model design consists of a wide base with a large initial twist, where the width smoothly decreases and the twist is heavily reduced for the other segments. The end of the blade is narrow with a large sweep angle. For optimal C_P/C_T, the initial twist is smaller, and the blade is overall narrower, especially in the end.

Due to the time required by the optimization process, and the multitude of settings and options for each method, there is plenty of room for improving the process, and further

research regarding this turbine model. For example, there is a potential that optimizing the trade-off between exploration and exploitation can lead to increased efficiency. For Bayesian optimization, this could be done by increasing the initial sample size to ensure a wider representation of the parameter space, or by increasing the exploration parameter β. In NSGA-II, the population size, mutation and crossover rates, and individual crossover operator parameters can be modified further to improve this trade-off. In addition, more combinations of the methods can be explored, for example with varying proportions of decision vectors proposed by each method, and including multiple versions of the same method.

References

[1] Minesto AB. "About us," 2025. [Online]. Available: https://minesto.com/about-us/ (accessed on: 2025-01-20).

[2] SUBMERSIBLE PLANT, by M. Landberg. (Jul. 23, 2009). US2009185904A1 [Online]. Available: https://worldwide.espacenet.com/patent/search?q=pn%3DUS2009185904A1.

[3] Minesto AB. "Our technology," 2025. [Online]. Available: https://minesto.com/our-technology/ (accessed on: 2025-01-20).

[4] R. A. Budenholzer and F. Landis. "Turbine," Encyclopædia Britannica 2022. [Online]. Available: https://academic-eb-com.eu1.proxy.openathens.net/levels/collegiate/article/turbine/106035 (accessed on: 2025-01-24).

[5] M. Khan, G. Bhuyan, M. Iqbal, and J. Quaicoe, "Hydrokinetic energy conversion systems and assessment of horizontal and vertical axis turbines for river and tidal applications: A technology status review," *Applied Energy*, vol. 86, no. 10, pp. 1823–1835, 2009, ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2009.02.017.

[6] Y. Nishi, H. Koga, and Y. H. Wee, "Multi-objective optimization of an axial flow hydraulic turbine with a collection device to be installed in an open channel," *Renewable Energy*, vol. 209, pp. 644–660, 2023, ISSN: 0960-1481. DOI: 10.1016/j.renene.2023.03.130.

[7] L. Cavalari Labigalini, R. de Vasconcelos Salvo, R. Sene de Lima, R. Corrêa da Silva, and I. de Marchi Neto, "Hydrokinetic turbine design through performance prediction and hybrid metaheuristic multi-objective optimization," *Energy Conversion and Management*, vol. 238, 2021, ISSN: 0196-8904. DOI: 10.1016/j.enconman.2021.114169.

[8] L. Li, W. Zhang, Y. Li, C. Jiang, and Y. Wang, "Multi-objective optimization of turbine blade profiles based on multi-agent reinforcement learning," *Energy Conversion and Management*, vol. 297, 2023, ISSN: 0196-8904. DOI: 10.1016/j.enconman.2023.117637.

[9] M. M. Shamsuddeen, S.-B. Ma, N.-H. Park, K. M. Kim, and J.-H. Kim, "Design analysis and optimization of a hydraulic gate turbine for power production from ultra-low head sites," *Energy*, vol. 275, 2023, ISSN: 0360-5442. DOI: 10.1016/j.energy.2023.127371.

[10] S. Shaw and E. J. Cortes, "Advanced flow control innovations for optimizing wind and water turbine performance: Toward sustainable energy solutions," *Journal of Fluid Flow, Heat and Mass Transfer*, vol. 11, pp. 404–415, 2024, ISSN: 2368-6111. DOI: 10.11159/jffhmt.2024.040.

[11] S. Laín, O. López, B. Quintero, and D. Meneses, "Design Optimization of a Vertical Axis Water Turbine with CFD," in *Alternative Energies: Updates on Progress*, G. Ferreira, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 113–139, ISBN: 978-3-642-40680-5. DOI: 10.1007/978-3-642-40680-5_6.

[12] R. Lanzafame and M. Messina, "Fluid dynamics wind turbine design: Critical analysis, optimization and application of BEM theory," *Renewable Energy*, vol. 32, no. 14, pp. 2291–2305, 2007, ISSN: 0960-1481. DOI: 10.1016/j.renene.2006.12.010.

[13] M. Mosbahi, A. Ayadi, Y. Chouaibi, Z. Driss, and T. Tucciarelli, "Performance improvement of a novel combined water turbine," *Energy Conversion and Management*, vol. 205, 2020, ISSN: 0196-8904. DOI: 10.1016/j.enconman.2020.112473.

[14] R. A. Subekti et al., "Design and performance of very low head water turbines using a surface vorticity model algorithm," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 13, pp. 1140–1149, Jun. 2022. DOI: 10.11591/ijpeds.v13.i2.pp1140-1149.

[15] M. Douak, Z. Aouachria, R. Rabehi, and N. Allam, "Wind energy systems: Analysis of the self-starting physics of vertical axis wind turbine," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 1602–1610, 2018, ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.05.238.

[16] E. Paterson and F. Stern, "Computational fluid dynamics," *AccessScience*, 2020. DOI: 10.1036/1097-8542.757259.

[17] E. C. Navarrete, M. Trejo Perea, J. C. Jáuregui Correa, R. V. Carrillo Serrano, and G. J. R. Moreno, "Expert Control Systems Implemented in a Pitch Control of Wind Turbine: A Review," *IEEE Access*, vol. 7, pp. 13 241–13 259, 2019. DOI: 10.1109/ACCESS.2019.2892728.

[18] C. Greenshields and H. Weller, *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022.

[19] D. A. Pierre, "Optimization," *AccessScience*, 2020. DOI: 10.1036/1097-8542.474000.

[20] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999. DOI: 10.1109/4235.797969.

[21] C. Greenshields, *OpenFOAM v12 User Guide*. London, UK: The OpenFOAM Foundation, 2024. [Online]. Available: https://doc.cfd.direct/openfoam/user-guide-v12.

[22] S. Singer and J. Nelder, "Nelder-Mead algorithm," *Scholarpedia*, vol. 4, no. 7, p. 2928, 2009. DOI: 10.4249/scholarpedia.2928.

[23] D. E. Goldberg, "Genetic algorithm," *AccessScience*, 2020. DOI: 10.1036/1097-8542.757293.

[24] A. Meyer-Baese and V. Schmid, "Chapter 5 - Genetic Algorithms," in *Pattern Recognition and Signal Analysis in Medical Imaging*, Second Edition, Oxford: Academic Press, 2014, pp. 135–149, ISBN: 978-0-12-409545-8. DOI: 10.1016/B978-0-12-409545-8.00005-4.

[25] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[26] S. Achiche, L. Baron, and M. Balazinski, "Scheduling exploration/exploitation levels in genetically-generated fuzzy knowledge bases," vol. 1, Jul. 2004, pp. 401–406, ISBN: 0-7803-8376-1. DOI: 10.1109/NAFIPS.2004.1336316.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. DOI: 10.1109/4235.996017.

[28] P. Siarry, "Simulated annealing," in *Metaheuristics*, P. Siarry, Ed. Springer International Publishing, 2016, pp. 19–50, ISBN: 978-3-319-45403-0. DOI: 10.1007/978-3-319-45403-0_2.

[29] R. Garnett, *Bayesian Optimization*. Cambridge, United Kingdom: Cambridge University Press, 2023, ISBN: 978-1-108-42578-0. DOI: 10.1017/9781108348973.

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2006, Copyright 2006 Massachusetts Institute of Technology. All rights reserved. Typeset by the authors using LaTeX., ISBN: 026218253X. [Online]. Available: http://www.GaussianProcess.org/gpml.

[31] GPy. "Gpy.kern.src package" 2020. [Online]. Available: https://gpy.readthedocs.io/en/devel/GPy.kern.src.html#submodules (accessed on: 2025-04-09).

A.
OpenFOAM setup

A.1 Discretization scheme dictionary

The discretization schemes were specified in the following way for the different terms as a dictionary in the OpenFOAM setup.

```
ddtSchemes
{
    default          steadyState;
}
gradSchemes
{
    default         cellLimited pointCellsLeastSquares 1.0;
}
divSchemes
{
    default          none;
    div(phi,U)       bounded Gauss linearUpwindV grad(U);
    div(phi,k)       bounded Gauss upwind;
    div(phi,omega)   bounded Gauss upwind;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default          Gauss linear limited 1.0;
}
interpolationSchemes
{
    default          linear;
}
snGradSchemes
{
    default          limited 1.0;
}
wallDist
{
    method meshWave;
}
```

A.2 Initial and boundary condition dictionary

Here follows an excerpt from the dictionary where the initial and boundary conditions were specified. The velocity field was used as the example.

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    farfield
    {
```

```
        type            freestream;
        freestreamValue uniform (1.5 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (1.5 0 0);
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           uniform (0 0 0);
    }
    turbine
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}
```

A.3 Solver setting dictionary

Here follows an excerpt from the dictionary where the equation solvers were specified for each variable in the relevant file in the OpenFOAM setup.

```
solvers
{
    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-5;
        relTol          0.1;
        nSweeps         1;
    }
    p
    {
        solver          GAMG;
        tolerance       1e-5;
        relTol          0.1;
        smoother        GaussSeidel;
        nPreSweeps      0;
        nPostSweeps     2;
        cacheAgglomeration on;
        agglomerator    faceAreaPair;
        nCellsInCoarsestLevel 10;
        mergeLevels     1;
    }
    Phi
    {
        $p;
    }
    k
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-6;
        relTol          0.1;
        nSweeps         1;
    }
    omega
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-6;
        relTol          0.1;
```

```
42          nSweeps           1;
43      }
44  }
45  SIMPLE
46  {
47      nNonOrthogonalCorrectors 0;
48      consistent yes;
49
50      residualControl
51      {
52          p                   1e-3;
53          U                   1e-3;
54          "(k|omega|omega|f|v2)"  1e-3;
55      }
56  }
57  potentialFlow
58  {
59      nNonOrthogonalCorrectors 10;
60  }
61  relaxationFactors
62  {
63      fields
64      {
65          p               0.3;
66      }
67      equations
68      {
69          U               0.7;
70          k               0.5;
71          omega           0.5;
72      }
73  }
74  cache
75  {
76      grad(U);
77  }
```

B.
Parameter space boundary details

B.1 Parameters for the Pareto-set

Figure B.1 shows the Pareto-set with the corresponding decision vector parameter values in a parallel coordinate plot.

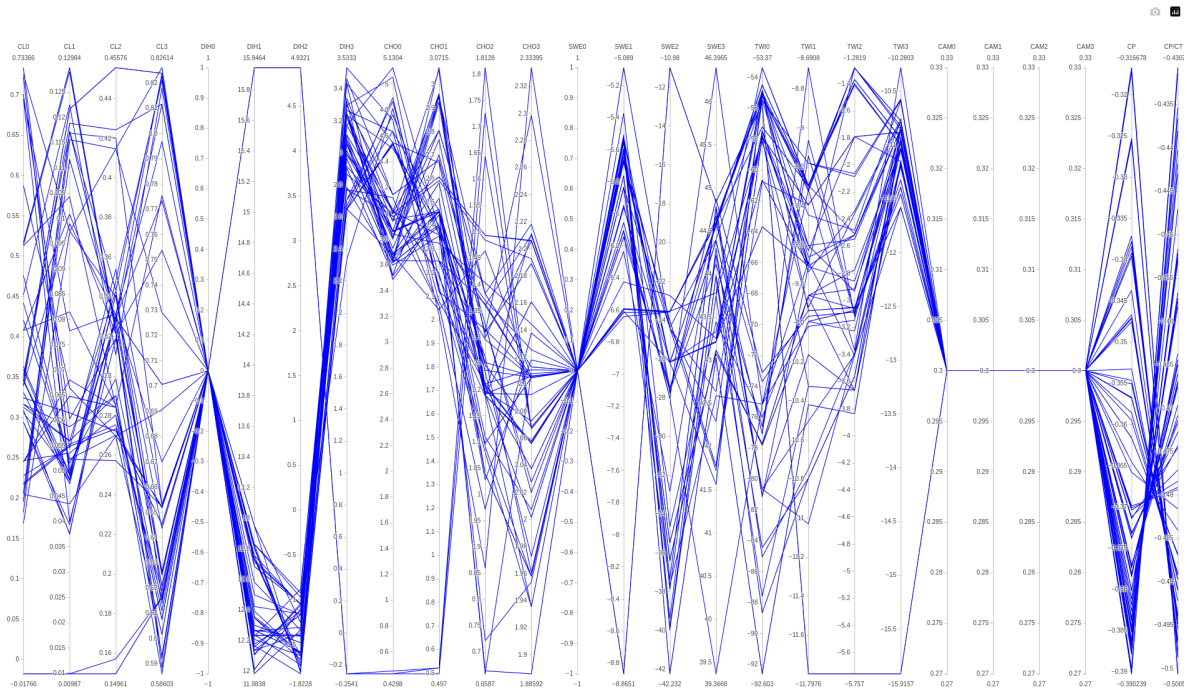


Figure B.1: Parallel coordinate plot of the Pareto-optimal set.

B.2 Exact parameter boundaries per run

Here follows tables listing the boundary values used to limit the parameter space for each optimization run presented in Section 4.2.

Table B.1: Boundary values for the NSGA-II optimization runs.

	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6		Run 7		Run 8	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
CL0	0.7	0.9	0.7	1.0	0.7	1.0	0.7	1.0	0.8	1.0	0.2	0.5	0.0	4.9	0.0	0.8
CL1	0.6	0.8	0.7	0.8	0.7	0.8	0.5	0.7	0.5	0.6	0.1	0.5	0.0	0.5	0.1	0.3
CL2	0.6	0.8	0.7	0.9	0.7	0.9	0.6	0.7	0.6	0.7	0.3	0.6	0.2	0.6	0.4	0.6
CL3	0.6	0.8	0.7	0.8	0.8	0.8	0.6	0.7	0.5	0.7	0.3	0.7	0.2	0.9	0.3	0.8
DIHEDRAL1	4.0	6.0	3.0	5.0	3.5	5.3	5.5	7.0	6.5	7.3	9.5	12.0	9.0	12.6	10.4	13.0
DIHEDRAL2	4.0	6.0	4.5	6.5	5.0	6.7	4.3	7.0	6.2	7.3	2.4	5.2	-3.4	4.6	-1.9	3.7
DIHEDRAL3	4.0	6.0	3.7	6.0	3.5	6.0	4.2	7.0	3.7	6.5	-2.0	1.3	-2.5	3.2	-0.8	3.0
CHORD0	1.0	2.0	1.0	1.5	1.0	1.5	0.6	2.0	1.5	2.0	1.0	4.5	0.7	4.5	3.0	4.4
CHORD1	1.0	2.0	1.0	2.5	1.0	2.2	0.5	1.5	1.2	1.5	1.2	2.1	1.0	2.3	1.7	2.1
CHORD2	1.0	2.0	1.0	2.0	1.5	2.5	1.5	2.5	1.0	1.9	0.8	1.5	0.6	1.6	0.8	1.6
CHORD3	1.0	2.0	1.0	2.0	1.1	2.7	0.5	1.4	1.1	1.4	1.7	2.2	1.7	2.4	1.7	2.2
SWEEP1	-10.0	19.4	-7.5	12.0	-6.1	13.7	-15.0	-3.1	-11.0	-4.1	-10.0	-6.0	-9.2	-3.8	-6.9	-5.4
SWEEP2	-10.0	19.9	-4.1	22.0	-5.6	24.0	-6.0	5.8	-9.8	0.9	-16.8	-5.3	-31.4	-3.9	-25.1	-13.1
SWEEP3	-10.0	20.0	-9.7	-0.0	-16.9	-6.0	12.2	28.0	20.1	33.0	32.8	45.5	32.7	47.5	39.7	45.1
TWIST0	-35.0	-30.1	-34.5	-30.1	-34.8	-30.3	-37.0	-32.0	-38.0	-35.1	-51.9	-38.2	-63.4	-41.4	-55.9	-45.1
TWIST1	-15.0	-12.0	-14.5	-13.0	-14.9	-12.5	-14.5	-13.0	-13.7	-13.1	-13.1	-8.2	-11.6	-7.1	-9.9	-9.3
TWIST2	-7.0	-5.0	-6.0	-5.0	-6.0	-5.0	-6.0	-4.2	-5.0	-3.5	-3.5	-2.0	-3.4	-1.2	-2.4	-1.1
TWIST3	-4.0	-3.0	-4.8	-3.5	-5.0	-3.5	-4.0	-3.2	-4.0	-3.2	-13.5	-6.2	-11.8	-5.2	-10.8	-6.5
CAMLOC0-3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Table B.2: Boundary values for the simulated annealing optimization runs. Where the boundaries were dynamic, the lowest and highest values that occurred are listed.

	Run 1		Run 2		Run 3		Run 4		Run 5 (dyn.)		Run 6 (dyn.)	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
CL0	0.8	0.8	0.8	0.9	0.7	1.0	0.8	1.0	0.0	0.7	0.1	0.7
CL1	0.7	0.7	0.5	0.6	0.5	0.6	0.5	0.6	0.2	0.6	0.1	0.5
CL2	0.8	0.8	0.6	0.7	0.6	0.7	0.6	0.7	0.3	0.7	0.4	0.6
CL3	0.8	0.8	0.5	0.6	0.5	0.7	0.5	0.7	0.3	0.9	0.5	0.8
DIHEDRAL1	3.7	5.2	6.5	7.2	6.3	7.6	6.5	7.4	8.8	11.4	10.4	12.6
DIHEDRAL2	6.2	6.7	6.2	7.3	6.3	7.9	6.3	7.4	-1.2	7.1	-1.9	3.9
DIHEDRAL3	3.7	5.1	3.7	5.5	3.2	6.9	3.7	5.8	-2.6	3.1	-1.3	3.0
CHORD0	0.3	1.2	1.4	2.0	1.4	2.1	1.6	2.0	0.7	1.4	0.8	4.5
CHORD1	1.4	1.8	1.2	1.5	1.2	1.5	1.2	1.5	1.0	1.7	1.1	2.2
CHORD2	1.6	2.5	1.1	1.9	0.9	1.7	1.1	1.6	0.6	1.6	0.6	1.6
CHORD3	1.5	1.9	1.1	1.4	1.0	1.4	1.1	1.4	1.2	2.1	1.7	2.2
SWEEP1	7.1	11.0	-10.1	-4.0	-9.0	-2.1	-9.0	-5.0	-8.3	-4.0	-7.7	-5.3
SWEEP2	12.9	24.2	-10.0	1.0	-13.7	-3.6	-12.0	-4.3	-10.0	-2.6	-25.5	-7.5
SWEEP3	-15.7	-13.7	20.0	33.0	30.0	37.2	30.0	36.0	33.2	41.8	35.4	44.9
TWIST0	-35.0	-32.1	-38.0	-35.0	-38.0	-35.0	-38.0	-35.0	-50.0	-39.6	-54.0	-45.4
TWIST1	-14.4	-13.0	-13.5	-13.1	-13.8	-13.1	-13.7	-13.1	-12.5	-8.8	-10.8	-9.5
TWIST2	-5.9	-5.3	-5.0	-4.1	-5.0	-3.2	-5.0	-3.5	-3.0	-1.3	-2.7	-1.4
TWIST3	-3.6	-2.9	-4.0	-3.2	-4.0	-3.0	-4.0	-3.2	-9.6	-4.6	-12.0	-6.3
CAMLOC0-3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Table B.3: Boundary values for the Bayesian optimization runs. Where the boundaries were dynamic, the lowest and highest values that occurred are listed.

	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6		Run 7 (dyn.)		Run 8 (dyn.)	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
CL0	0.8	1.0	0.8	1.0	0.6	1.1	0.6	1.2	0.8	1.0	0.5	0.7	0.0	7.3	0.0	0.7
CL1	0.5	0.6	0.5	0.6	0.4	0.7	0.4	0.7	0.5	0.6	0.4	0.6	0.0	0.8	0.1	0.6
CL2	0.6	0.7	0.6	0.7	0.6	0.7	0.5	0.8	0.6	0.7	0.5	0.6	0.0	1.2	0.2	0.6
CL3	0.5	0.6	0.5	0.6	0.4	0.7	0.4	0.7	0.5	0.6	0.6	0.7	0.0	1.2	0.2	1.0
DIHEDRAL1	7.0	7.6	7.0	7.6	6.7	8.2	6.5	8.5	6.9	7.7	7.7	8.4	3.7	21.5	7.9	11.1
DIHEDRAL2	6.4	7.3	6.4	7.3	5.5	8.2	5.2	8.6	6.4	7.5	5.6	7.0	-9.0	10.8	-1.2	5.2
DIHEDRAL3	3.9	6.3	3.9	6.3	1.5	8.6	0.4	9.7	3.4	6.7	-0.3	3.7 -14.1	17.5	-2.6	3.0	3.0
CHORD0	1.7	2.1	1.7	2.1	1.3	2.5	1.1	2.7	1.6	2.2	1.0	1.6	0.6	12.2	0.7	1.4
CHORD1	1.4	1.5	1.4	1.5	1.2	1.7	1.1	1.8	1.3	1.6	1.3	1.6	0.6	4.8	1.0	1.7
CHORD2	1.0	1.4	1.0	1.5	0.5	1.9	0.3	2.2	0.9	1.6	0.5	1.2	0.6	2.5	0.6	1.6
CHORD3	1.1	1.3	1.1	1.3	0.9	1.5	0.8	1.6	1.1	1.3	1.4	1.5	0.7	4.1	1.3	2.0
SWEEP1	-8.0	-5.0	-7.9	-5.0	-11.0	-2.0	-12.5	-0.6	-8.5	-4.4	-10.9	-6.2	-14.3	5.8	-8.6	-3.9
SWEEP2	-9.0	-5.0	-9.0	-5.1	-13.0	-1.2	-15.0	0.8	-9.8	-4.2	-5.2	1.2	-50.9	4.2	-11.9	-2.3
SWEEP3	33.0	37.0	33.0	37.0	29.1	40.9	27.0	42.9	32.3	37.8	30.9	35.5	29.5	61.1	33.2	41.8
TWIST0	-38.0	-35.0	-38.0	-35.5	-41.0	-32.5	-42.4	-30.5	-38.6	-34.4	-42.3	-39.3	-76.6	-28.2	-49.1	-39.8
TWIST1	-13.4	-13.1	-13.7	-13.2	-14.3	-12.5	-14.6	-12.2	-13.8	-13.0	-12.9	-12.1	-14.3	-4.5	-11.5	-8.8
TWIST2	-4.4	-3.5	-4.5	-3.5	-5.3	-2.5	-5.8	-2.0	-4.5	-3.3	-2.8	-2.3	-4.9	2.7	-3.0	-1.3
TWIST3	-4.2	-3.2	-4.0	-3.2	-5.2	-2.3	-5.2	-1.7	-4.4	-3.0	-5.5	-4.9 -21.2	-2.5	-9.6	-5.3	-6.3
CAMLOC0-3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Table B.4: Maximum and minimum parameter values during the C_P focused runs with the combo method.

	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
CL0	0.0	0.9	0.1	0.9	0.1	1.0	0.2	0.9	0.1	0.9	0.1	0.9
CL1	0.1	0.3	0.0	0.2	0.0	0.3	0.0	0.3	0.0	0.1	0.0	0.1
CL2	0.4	0.6	0.3	0.5	0.3	0.6	0.3	0.5	0.3	0.5	0.3	0.5
CL3	0.5	0.9	0.6	1.0	0.6	1.0	0.6	0.9	0.6	0.9	0.6	0.9
DIHEDRAL1	11.9	13.4	12.0	13.4	12.0	13.6	12.0	13.4	12.0	13.4	12.0	13.4
DIHEDRAL2	-1.9	-0.8	-2.5	-1.0	-2.5	-1.0	-2.1	-1.0	-2.4	-1.0	-2.3	-1.0
DIHEDRAL3	1.6	3.3	2.1	3.3	2.0	3.4	2.1	3.4	2.1	3.4	2.4	3.4
CHORD0	2.7	5.1	3.5	5.0	3.5	5.1	3.5	5.0	3.5	5.0	3.7	5.2
CHORD1	1.7	2.4	2.0	2.6	2.0	2.6	2.0	2.6	2.0	2.6	2.0	2.6
CHORD2	0.8	1.6	1.0	1.6	1.0	1.6	1.0	1.6	1.0	1.6	1.1	1.6
CHORD3	1.7	2.2	1.9	2.3	1.9	2.3	2.0	2.3	1.9	2.3	1.9	2.2
SWEEP1	-6.9	-5.4	-7.8	-5.0	-8.0	-4.9	-7.4	-5.1	-7.8	-5.1	-6.9	-4.9
SWEEP2	-25.1	-15.4	-26.2	-21.7	-26.2	-20.1	-27.5	-20.6	-27.5	-21.2	-27.8	-22.1
SWEEP3	39.4	45.1	38.8	47.1	38.8	47.2	41.2	46.9	39.4	46.8	39.8	47.0
TWIST0	-56.1	-45.4	-56.2	-51.3	-56.2	-50.1	-57.7	-51.7	-73.7	-52.9	-77.4	-53.5
TWIST1	-10.5	-9.3	-9.7	-9.1	-9.7	-9.1	-9.7	-9.1	-9.9	-9.1	-9.8	-9.3
TWIST2	-2.5	-1.0	-2.6	-0.8	-2.6	-0.8	-2.6	-0.8	-2.6	-0.8	-2.5	-0.8
TWIST3	-11.1	-6.6	-11.1	-8.9	-11.1	-7.9	-11.1	-8.0	-11.1	-8.3	-11.0	-8.9
CAMBERLOC0-3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

Table B.5: Maximum and minimum parameter values during the C_P/C_T focused runs with the combo method.

	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6	
Parameter	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
CL0	0.1	0.9	0.2	0.9	0.2	0.9	0.1	0.9	0.0	1.0	0.0	1.3
CL1	0.0	0.3	0.0	0.3	0.0	0.2	0.0	0.2	0.0	0.2	0.0	0.6
CL2	0.3	0.5	0.3	0.5	0.3	0.5	0.3	0.5	0.3	0.5	0.0	1.2
CL3	0.6	0.9	0.6	0.9	0.6	1.0	0.6	0.8	0.6	0.8	0.0	1.3
DIHEDRAL1	12.1	13.4	12.0	13.4	11.7	13.4	11.4	13.3	11.8	13.6	5.1	23.1
DIHEDRAL2	-2.4	-0.9	-2.3	-1.0	-1.9	-1.0	-1.8	-1.0	-1.9	-0.8	-1.9	7.6
DIHEDRAL3	2.1	3.4	2.3	3.5	2.0	3.5	2.4	3.6	2.1	3.5	-7.9	7.7
CHORD0	3.5	5.0	3.6	5.0	3.5	5.2	3.4	4.8	3.4	5.1	3.4	5.5
CHORD1	2.0	2.6	2.0	2.8	2.0	3.1	2.0	2.7	1.8	2.6	1.8	3.9
CHORD2	1.0	1.4	1.1	1.6	1.1	1.6	1.0	1.6	0.6	1.6	0.6	2.9
CHORD3	2.0	2.3	2.0	2.3	1.9	2.3	1.9	2.3	1.9	2.3	1.9	2.8
SWEEP1	-7.2	-5.4	-6.9	-5.4	-6.9	-4.9	-7.1	-5.2	-7.3	-5.0	-16.5	-4.9
SWEEP2	-31.0	-22.0	-35.7	-21.8	-42.2	-21.7	-29.7	-22.0	-28.2	-22.1	-34.2	-22.1
SWEEP3	40.3	47.1	41.2	48.0	39.1	46.8	40.5	47.7	38.8	46.4	21.3	54.7
TWIST0	-81.4	-52.8	-81.8	-52.3	-93.9	-50.9	-81.6	-52.2	-75.6	-49.7	-67.0	-31.5
TWIST1	-9.9	-9.2	-9.9	-9.1	-11.6	-8.9	-10.3	-8.2	-10.1	-8.4	-15.8	-6.3
TWIST2	-2.6	-0.9	-2.8	-0.9	-3.8	-0.7	-3.5	-1.7	-3.4	-1.7	-7.7	-0.3
TWIST3	-11.2	-8.4	-11.1	-9.8	-11.7	-9.5	-12.1	-9.6	-11.8	-9.2	-20.2	-0.6
CAMBERLOC0-3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3

C.
Result details

C.1 NSGA-II setting details

Here follows details regarding the optimization runs presented in Section 4.7.

Table C.1: Details for NSGA-II runs succeeded by a Nelder-Mead search based on an initial simplex consisting of the best results. The initial population consisted of 40 samples of values evenly distributed on the parameter interval.

Run	Crossover operator (rate)	Mutation rate	# evaluations per generation	# generations	# Nelder-Mead evaluations	Max. C_P
1	SBX (0.8)	0.3	20	8	50	0.3881
2	SBX (0.8)	0.3	40	4	55	0.3869
3	SBX (0.8)	0.3	30	6	35	0.3868
4	SBX (0.8)	0.3	50	4	15	0.3869
5	BLX-α=0.3 (0.9)	0.3	20	9	35	0.3871
6	BLX-α=0.3 (0.9)	0.3	30	6	35	0.3862
7	DEX (0.9)	0.3	20	9	35	0.3863
8	DEX (0.95)	0.4	20	9	35	0.3862
9	DEX (0.85)	0.4	20	9	35	0.3861

C.2 Kernel combination details

Here follows details regarding the optimization runs presented in Section 4.8.

Table C.2: Results and details for different kernels. In all cases a bias term was also added.

Run	# kernels	Kernel constructions	Max. C_P	Iteration for max.
1	4	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}} + K_{\text{Lin}}^{\text{ARD}}$	0.3894	226
2	4	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$, $K_{\text{RBF}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}$ $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}} + K_{\text{Lin}}$	0.3887	154
3	4	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$ $(\times 2)$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$ $(\times 2)$	0.3866	180
4	6	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$ $(\times 2)$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$ $(\times 2)$	0.3880	165
5	4	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$	0.3897	251
6	4	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$ $(\times 2)$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$ $(\times 2)$	0.3880	231
7	5	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$ $(\times 2)$	0.3893	237
8	5	$K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$ $(\times 2)$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$ $(\times 3)$	0.3894	216
9	5	$K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Matern}}$, $K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}}^{\text{ARD}} + K_{\text{MLP}}^{\text{ARD}} + K_{\text{Exp}}^{\text{ARD}} + K_{\text{Matern}}^{\text{ARD}}$, $K_{\text{RBF}} + K_{\text{MLP}} + K_{\text{Exp}} + K_{\text{Matern}}$	0.3889	236

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY