Multi-Point Fluid-Dynamic Shape Optimization for Turbomachinery

L. S. Wilkens

TU Delft Delft University of Technology

MULTI-POINT FLUID-DYNAMIC SHAPE OPTIMIZATION FOR TURBOMACHINERY

by

L. S. Wilkens

in partial fulfillment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at the Delft University of Technology,
to be defended publicly on Thursday September 29, 2022 at 13:00.

Student number:	4158210		
Supervised by:	Dr. ir. M. Pini		TU Delft - Propulsion & Power
	Dr. ir. N. Anand		Vito | TU Delft - Propulsion & Power
Thesis committee:	Dr. ir. R. Dwight	Chair	TU Delft - Aerodynamics
	Dr. ir. M. Pini	Supervisor	TU Delft - Propulsion & Power
	Dr. ir. A. H. van Zuijlen	External	TU Delft - Aerodynamics
	Dr. ir. N. Anand	Examiner	Vito | TU Delft - Propulsion & Power

An electronic version of this thesis is available at https://repository.tudelft.nl/.

TU Delft Delft University of Technology

ACKNOWLEDGEMENTS

This marks the end of my time in the master programme at the Aerospace engineering faculty at the TU Delft. Before you lies a work in which I present a software tool for the fluid-dynamic optimization of turbomachine blades. My personal goal was to apply knowledge on artificial intelligence and Bayesian optimization gained during my internship on a practical application within the master track. A respectable period later, this has resulted in copious lines of code and a broadened perspective. In all, the project proved a valuable experience in learning, exploration and failure, which has given me awareness, renewed energy and eagerness going forward.

This project would not have been realized without my supervisors. I want to thank Matteo Pini for providing the opportunity of performing my thesis with him and this department, ensuring the relevancy of this project and providing the freedom and structure to complete this challenging work. Nitish specially for the numerous sessions of feedback, guidance and motivation. His availability and determination was invaluable during the development of the software and the shaping of this work.

I want to thank my parents for the unconditional love and support throughout, enabling the opportunities I have enjoyed as a student. Leonie for the unwavering support, bearing with me through the challenging times, and your sincere joy in the good ones. I appreciate all who have dedicated time by proofreading or through other means improving this work. I'm grateful for my friends, encouraging me to the end and for being part of this adventure.

L. S. Wilkens
Delft, September 2022

ABSTRACT

This work sets out the creation, verification and demonstration of an automated end-to-end tool for the multi-point fluid-dynamic optimization of turbomachinery. Taking into account multiple operating conditions during optimization aims to benefit the overall performance of turbomachines which are characterized by off-design operation and to produce more robust designs with respect to deviations in operating conditions. The proposed tool builds upon the in-house turbomachinery parametrization software ParaBlade, which provides a unified approach to the parametrization of turbomachine blades.

The code is written in accordance with the Object-Oriented Programming model in order to ease maintenance. It will be published as an open-source software tool, driving adoption of the multi-point optimization tool by the online community. The structure of the code allows for future straightforward extension into many-point and many-zone problems and by other optimization algorithms and analysis modules, enabling multi-disciplinary optimization.

The feasibility and effectiveness of the tool is demonstrated by a two-point fluid-dynamic optimization of a two-dimensional Aachen turbine stator cascade, performed on the High-Performance Computing cluster of the TU Delft. Automated optimization workflows using global optimizers on large design spaces require a grid re-generation approach, for which an integrated meshing module is created and verified through a grid convergence study. By utilizing the stochastic NSGA2 optimization algorithm, the numerically noisy problem is successfully optimized. The entropy generation of the blade is reduced at both nominal and off-design operating condition by an average of 7.57%, while satisfying a constraint on the flow turning when passing through the stator cascade. Additionally, a numerical noise analysis shows that disregarding the most noisy design variables increases the probability of obtaining an improved design.

This work aims to reduce the development time of turbomachinery optimization by providing an automated end-to-end optimization tool making use of global optimization algorithms. The saved development time promotes the rate of innovation of turbomachinery, supporting the transition to global net-zero emissions.

Contents

List of Figures

List of Tables

NOMENCLATURE

Acronyms

2D	Two-dimensional
3D	Three-dimensional
AD	Algorithmic Differentiation
ADJ	Adjoint
CAD	Computer-Aided Design
CFD	Computational Fluid-Dynamics
CFL	Courant-Friedrichs-Lewy
CPU	Central Processing Unit
DoE	Design of Experiments
EVG	Expected Volume-Gain infill criterion
FD	Finite-Differences
FFD	Free-Form Deformation
GCI	Grid Convergence Index
GEK	Gradient-Enhanced Kriging
Gmsh	Graphical user interface mesh program
LE	Leading Edge
LES	Large Eddy Simulation
LHS	Latin Hypercube Sampling
MO	Multi-Objective
NPV	Net Present Value
NURBS	Non-Uniform Rational B-Spline
PR	Polynomial Regression
RANS	Reynolds-Averaged Navier Stokes
SA	Spalart-Allamaras
SU2	Stanford University Unstructured
TE	Trailing Edge
UML	Unified Modeling Language
URANS	Unsteady Reynolds-Averaged Navier Stokes

Optimization Algorithms

ALPSO	Augmented-Lagrangian Particle-Swarm Optimization
ANN	Artificial Neural Network
DE	Differential Evolution
DIRECT	DIViding RECTangles
EA	Evolutionary Algorithm
EGO	Efficient Global Optimization
IPOPT	Interior-Point OPTimization
MOGA	Multi-Objective Genetic Algorithm
NSGA2	Non-Dominated Sorting Genetic Algorithm II
OMOPSO	Optimized Multi-Objective Particle-Swarm Optimization
PSO	Particle-Swarm Optimization
SD	Steepest-Descent
SLSQP	Sequential Least-Squares Quadratic Programming
SNOPT	Sparse Non-linear OPTimizer

Greek

$\boldsymbol{\alpha}$	Design vector
$\boldsymbol{\delta}$	Kronecker delta vector
$\boldsymbol{\psi}$	Adjoint vector
$\mathbb{\alpha}$	Scaled design vector
α	Design variable
β	Total-to-static expansion ratio
β_{out}	Flow angle at outlet boundary
Δ	Difference
δ	Step
ζ	Fluid-dynamic loss coefficient
ϵ	Error (relative)
θ	Blade metal angle or boundary layer thickness
ξ	Stagger angle
λ	Lagrange multiplier
μ	Mean
μ'_{Δ}	Harmonic mean of differences
ρ	Density
σ	Standard deviation
Ω	Fluid control volume

Symbols

$\boldsymbol{\mathcal{J}}$	(Multi-point) Objective function vector
$\boldsymbol{g}$	Inequality constraint vector
$\boldsymbol{h}$	Equality constraint vector
$\boldsymbol{R}$	Residual of flow system of equations
$\boldsymbol{U}$	Flow system of equations
$\boldsymbol{u}, \boldsymbol{v}$	Parametric values for blade matching
$\boldsymbol{X}$	Matrix of coordinates
$\dot{m}$	Mass flow rate
$\mathcal{J}$	Objective function
$\mathbb{J}$	Scaled objective function
$\mathcal{G}$	Flow Solver
$\mathcal{L}$	Lagrangian or trade-off loss metric
$\mathcal{M}$	Mesh generator
$\mathcal{S}$	Surface parametrizer
∂	Partial difference
d	Distance
h	Inflation layer height
I	Turbulence intensity
M	Mach number
P	Pressure
r	Inflation ratio of boundary layer mesh
s	Entropy
T	Temperature
v_0	Spouting velocity
w	Weight attributed to operating condition
$\mathbf{C}$	Coordinates of reference blade surface
$\mathbf{P}$	Surface control point coordinates
$\mathbf{Q}$	Prescribed blade coordinates

Number sets

$\mathbb{R}$	Real numbers

Subscripts

$()_*$	Variables at their final value
$()_0$	Stagnation
$()_D$	Fluid Domain
$()_{gen}$	Generation
$()_{is}$	Isentropic
$()_{MO}$	Multi-Objective
$()_n$	Operating condition point number
$()_{ref}$	Reference
$()_{surf}$	Surface
$()_{tur}$	Turbulence
$()_T$	Total
$()_t$	Target
$()_{i,\,j}$	Numbers

Superscripts

$()^*$	Variables at their optimal or final value
$()^+$	Along surface of a wall
$()^0$	Variables at their initial value
$()^L$	Lower bound on design variable
$()^U$	Upper bound on design variable
$()^k$	Iteration number

Prefixes

k	kilo	10^3
M	mega	10^6
m	milli	10^{-3}

1

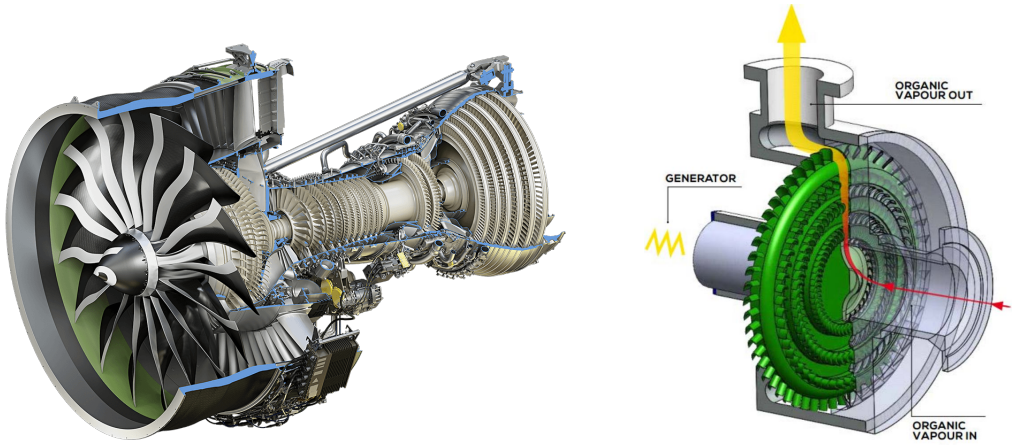
INTRODUCTION

Over the last decades, numerical optimization has become indispensable for many fields in the industry. Increasingly complex and multidisciplinary problems of turbomachine design in energy power systems and aircraft benefit considerably from automated design strategies which include multiple objectives and disciplines. However, challenges remain in the reduction of to-market time of innovative technologies providing environmentally-friendly solutions. Solutions are needed for the transformation of energy systems as required for the net-zero emission targets by 2050 as set out by the International Energy Agency (IEA) [Cozzi et al. 2021]. These targets are instrumental to curbing global warming to 1.5 °C and reversing the years of industrial emissions. Exceeding this temperature increase will likely constrain the opportunities for adaptation to climate risks, as projected by the recent assessment of the working group of the Intergovernmental Panel on Climate Change (IPCC) [Masson-Delmotte et al. 2021]. The IEA project that 50% of reductions will originate from innovations currently in the prototype phase and a total of 70% will be accounted for by wind and solar sources. In addition, the aircraft industry is set to comply with the research agenda of The Advisory Council for Aeronautics Research in Europe. Aiming to enforce specific reductions considering multiple criteria such as emissions and noise, of carbon emissions per passenger per kilometre, nitrogen oxides and perceived noise by 75%, 90% and 65% respectively, with respect to the year 2000 [European Commission and Directorate-General for Mobility and Transport and Directorate-General for Research and Innovation 2011]. In the theoretical studies and practical applications of new configurations realizing these energy transition goals and regulations, numerical optimization of the fluid-dynamic performance of turbomachinery in energy systems and aircraft remains essential.

The optimization algorithm plays a central role in the efficacy of design optimization. With numerical optimization methods, engineers are less reliant on trial and error and are able to quickly iterate between designs during the design phases. This allows to make more informed decisions during later stages of design, reducing the risk of costly re-designs and benefiting final design performance.

Turbomachines can be described as “devices in which energy is transferred either to, or from, a continuously flowing fluid by the *dynamic action* of one or more moving blade rows” [Dixon and Hall 2014a]. Turbomachines consist of rows, or cascades, of blades alternating between stationary *stators* and rotating *rotors*. A famous example of a turbomachine is a turbofan engine used in aviation, see Figure 1.1a. Rotors change the stagnation enthalpy, and thereby also the pressure, of the fluid moving through. Stators ensure that the flow direction is appropriate for admission for the downstream rotor. Blades can either increase or decrease the fluid pressure by *absorbing* or *producing* power from the fluid respectively.

Over the past decades, enormous efforts have been allocated towards the advancement of turbomachine efficiency. Currently, the scope of numerical optimization in turbomachinery is expanding. Areas of interest being multi-cascade [Ma et al. 2021; Rubino et al. 2020; Vitale et al. 2020; Walther and Nadarajah 2015b; Wang and He 2010], thermo-physical [Pini et al. 2015; Rubino et al. 2018; Vitale et al. 2015 2017], multi-discipline such as aero-elastic [Anand et al. 2020], aero-mechanical [Aissa and Verstraete 2019; Backhaus et al. 2017], aero-acoustic [Wang and Zangeneh 2014] and unsteady flow [Anand et al. 2020; Da et al. 2020; Rubino et al. 2020]. Although not integrated into an automated design loop, Computational Fluid-Dynamics



(a) Turbofan engine [GE Aviation 2018].

(b) Radial outflow turbine [Exergy 2017].

Figure 1.1: Examples of turbomachines. Cutaway of the GE9x turbofan engine (a). Working scheme of a radial outflow turbine in an Organic Rankine Cycle system (b).

(CFD) fidelity advancements such as Large-Eddy Simulations [Hoarau et al. 2020] and Direct Numerical Simulations [Michelassi et al. 2015; Wheeler et al. 2016] are also becoming more prevalent.

As such, multi-point optimization is gaining interest with the expanding scope of design optimization. Multi-point analyses take multiple operating conditions into account, allowing for a more holistic performance measure in line with the operational lifetime of turbomachinery. Turbomachines in Organic Ranking Cycle (ORC) power systems (Figure 1.1b) are characterized by an inherent off-design operation due to fluctuating availability of thermal energy [Andrić et al. 2017; Casella et al. 2013; Chen et al. 2020]. Thus, the inclusion of multiple operating conditions during optimization is of direct relevance for the performance of ORC's. Besides operational off-designs, geometric off-designs associated with variable-geometry turbomachines form another source of application for multi-point analyses.

ORC's represent a notable technology for their role in the reduction of carbon emissions by producing electricity in a CO_2-free way, through the recovery of waste heat. They can efficiently convert waste heat at low and medium temperatures from renewable and industrial plants and process sources [Colonna et al. 2015], where a significant portion of the consumed energy is dissipated as waste heat. Besides electricity generation from industrial waste heat, ORC's can be applied in electricity generation from geothermal reservoirs, biomass combustion and distributed electricity generation [Casella et al. 2013]. ORC systems are preferred for their reliability, power range and wide component availability [Praß et al. 2017]. In addition, the same ORC system can be applied to different sources without major effort [Akorede et al. 2010].

The attractiveness of waste heat recovery ORC's is projected to increase in the foreseeable future, increasing the market for ORC applications. Payback periods of down to 2.7 years are demonstrated in biomass-based plant applications [Eyidogan et al. 2016], of which the Net Present Value (NPV) is very sensitive to the price of electricity. An increase of 10% in electricity price causes a rise of 26% in the NPV [Lemmens and Lecompte 2017]. With electricity prices rising by 11.7% between 2017–2021 for industrial consumers in the European Union, and little signs of stagnation [Eurostat 2022]. Adding on economic opportunity, ORC's provide a stable source of energy as compared to fossil fuel sources due to the fluctuating prices stemming from geopolitics and environmental issues [Klun et al. 2021]. Accordingly, reliable and decentralized power generation systems are identified by the IPCC as a feasible option for future adaptation against vulnerabilities to climate change [Masson-Delmotte et al. 2021].

Of high-temperature and high-expansion ratio ORC systems, the turbine stator accounts for around two-thirds of the total turbine fluid-dynamic losses [Anand et al. 2019; Rinaldi et al. 2016]. This operating condition is common for distributed, small-scale power generation applications of ORC's due to economic constraints. To limit the cost of the ORC, the complexity is reduced by restricting the number of stages, leading

to higher loaded blades and therefore large expansion ratios. The molecular complexity of ORC's decrease the speed of sound of the fluid and enthalpy drop over the cascade. Which leads to, respectively, an increase of the operating Mach number for constant pressure ratios, and an increase volume flow rate. Both causing ORC's to operate in supersonic flow regimes [Colonna et al. 2015]. As a result, off-design performance decays rapidly due to the significant shockwave losses downstream of the blade. This emphasizes the requirement for a satisfactory off-design performance.

Wind turbines and compressor blades are examples of other fluid-dynamic systems which often operate outside the nominal conditions, and are therefore amenable to multi-point fluid-dynamic optimization. Wind turbine blades operate in varying conditions due to wind speed fluctuations. Blades ideally operate at a constant angle of attack α, which is controlled by varying the blade pitch angle, leading to varying rotational speeds. However, the rotational speed has a lower bound due to avoiding of tower excitation, which necessitates an off-design blade α [Madsen et al. 2019]. The maximum rotational velocity must be kept limited at high wind conditions, incurring an off-design blade α aswell. Having a blade shape which performs well under a range of α due to varying wind speeds increases overall electricity production. Compressors in air conditioners must operate in all seasons, which relates to a fluctuating mass flow rate but constant pressure ratio. For this, low-solidity diffusers are utilized which allow for satisfactory performance at varying inlet mass flow rate conditions [Van den Braembussche 2008]. For constant rotational velocity, stall and choke issues occur at low and high mass flow rate conditions respectively. At low mass flow rate conditions, α becomes too large, while at high mass flow rate conditions, α becomes too negative. This presents as a multi-point optimization problem with competing corresponding operating condition objectives, demonstrated by assessing the effect of varying the leading edge (LE) radius of the blade. Enlarging the LE radius postpones stall to larger α which allows for lower mass flow rates, while it decreases the maximum mass flow rate at which chocking occurs due to a narrower flow passage between the blades.

The design process of turbomachines can roughly be grouped into a three-step strategy [Dixon and Hall 2014a; Lewis 1996]. Firstly, conceptual design using one-dimensional meanline models such as similarity parametrization to determine the machine type and target flow angles throughout the stages. Secondly, two-dimensional through-flow solvers in combination with empirical models determine the radial distribution of flow quantities. Lastly, three-dimensional fluid-dynamic design optimization is performed by radially stacking blade profiles. This work is situated between steps two and three – already having an initial guess for the blade coordinates but not concerning a full three-dimensional blade.

Two general classes of numerical optimization algorithms are commonly identified; gradient-based and gradient-free. Gradient-based algorithms move the design towards an optimal point by making use of information on how the objective function changes with respect to a change in its design variables, and generally converge to a local optimum. Consequently, using gradient-based algorithms produces a non-optimal final design when the local optimum in design space demonstrates a worse performance compared to the global optimum. Contrarily, gradient-free algorithms do not utilize the gradient, and pose a lower risk of getting trapped in a local optimum. In addition, gradient-free optimizers are less sensitive to stochastic responses of the objective function. However, generally requiring one or two orders of magnitude more function evaluations [Skinner and Zare-Behtash 2018]. Function evaluations require a significant amount of time in fluid-dynamic optimization, which makes gradient-based optimizers the choice for many optimization studies in literature. However, surrogate-assisted optimizers and other hybrid approaches which aim to combine the benefits of both optimizer classes, are gaining interest [Aissa et al. 2019; Châtel et al. 2020; He and Zheng 2017; Song et al. 2019].

Increased computing power allows for larger optimization problems to be solved, facilitating multi-point analyses. For optimization problems which pose constraints on certain output flow quantities, which is generally the case in turbomachinery, multi-point optimizations using gradient-based algorithms require a larger increase in number of flow evaluations for increasing number of operating conditions than the gradient-free counterpart. While the number of flow evaluations for increasing numbers of design variables favour gradient-based algorithms [Yu et al. 2018]. This indicates a necessity of a trade-off between the gradient-based and gradient-free optimization algorithm classes.

The geometry parametrization approach is an important aspect of design optimization of turbomachinery. A parametrization method is selected which appropriately represents the design space with a minimal number of design variables. Two main approaches exist; constructive and deformative [Agromayor et al. 2021]. Computer-Aided Design (CAD) is a well known constructive method which enables imposing constraints in a natural way and reduces the number of design variables with respect to deformative approaches [Anand et al. 2018; Samareh 2001], such as Free-Form Deformation (FFD) [Sederberg and Parry 1986]. To this end, the in-house open-source blade parametrization framework ParaBlade [Anand and Agromayor 2020] is developed. ParaBlade provides a systematic approach to CAD-based parametrization of turbomachine blades defined by a cloud of points, able to handle blade shapes of a wide range of geometries.

Open-Source software, in contrast to proprietary software, is software of which the source code is openly shared. With the goal of encouraging voluntary improvements to the code by the community. A few benefits of the open-source model include little to no initial cost for users, reliability due to distributed co-production by a larger audience of developers, longevity and flexibility due to the absence of user agreements. Drawbacks with respect to proprietary software include lessened user-friendliness for users [Heron et al. 2013; Morgan and Finnegan 2007]. Well-known examples of open source software include the GNU/Linux operating system and its derivatives, the Android mobile operating system and the Firefox and Chromium browsers. Business models around open-source are shown to be successful [Schireson and Thakker 2016]. Governments are adopting open-source policies aswell, the United States announced a policy in 2016 mandating that 20% of source code be released as open-source in order to join forces between the federal agencies and the public [Scott and Rung 2016]. The European Commission regards open-source software as one of nine key drivers for innovation in the software industry, alongside big data, cloud computing, cyber security and the internet of things [European Commission and Directorate-General for Communications Networks, Content and Technology et al. 2017].

Following the previous observations, a need is identified to develop an open-source optimization tool capable of the multi-point optimization of turbomachinery using a CAD-parametrization approach and global and surrogate-assisted optimization algorithms. The developed tool as described in this work builds upon the in-house open-source blade parametrization framework ParaBlade [Anand and Agromayor 2020].

Research goal and motivation

In consideration of the above, the research objective is defined as:

> " To develop an optimization library capable of multi-point fluid-dynamic turbomachinery global shape optimization, by implementing the software components and demonstrating a multi-point blade design problem."

From the research objective, the main research question is posed as follows:

What is the best approach for implementing the detailed shape design of turbomachinery blades for multi-point optimization using global optimizers?

The sub questions are the following:

1. Which key code modules require implementation in order to extend ParaBlade for performing multi-point optimization of turbomachinery blades using global optimizers?
2. Which optimization algorithms are applied most frequently for the multi-point shape optimization of turbomachinery?
3. Which verification methods can be applied to ensure accurate results of the implemented modules?
4. What is the improvement in percentage of objective function value for the two-point optimization of a two-dimensional axial turbine blade using the optimization algorithm NSGA2?
5. Which fluid-dynamic loss mechanisms affect the differences in objective function value of the optimized blade shape?

Originality of work

Although a rising interest in multi-point fluid-dynamic optimization, no open-source tool is available for the CAD-based optimization of multi-point, multi-objective and multi-zone turbomachinery problems in an end-to-end framework using local and global optimization methods. This work contributes to research in turbomachinery shape optimization by presenting a fluid-dynamic assessment of a turbomachine blade at on- and off-design operating conditions and demonstrating the feasibility and effectiveness of such a tool. A tool which aims to reduce the development costs of turbomachinery. More specifically, yielding turbomachine designs showing reduced sensitivity to variations in the operating conditions.

Report outline

This report serves to document the work performed of this research and discusses the results. This work is organized as follows: Chapter 2 introduces concepts and assesses literature behind the fluid-dynamic multi-point shape optimization of turbomachinery blades. Chapter 3 outlines the specific methods that are implemented throughout the research. Chapter 4 serves as a documentation on the code framework behind the single- and multi-point optimization capabilities. Chapter 5 describes the selection and formulation of the multi-point stator optimization problem and performs verification on the adopted numerical models in light of the test case in this work. Chapter 6 presents the results of the multi-point optimization and discusses the obtained insights and experienced difficulties. The conclusions and recommendations in chapter 7 finalize the report by highlighting the research findings with regard to the research questions and providing recommendations for future research avenues stemming from this work.

2

BACKGROUND

The main motivation of this work is to provide an automated end-to-end optimization tool for the shape optimization of turbomachinery containing a variety of optimization algorithms to accommodate the demand of differing single and multi-point problem applications. The purpose of this chapter is to introduce the concepts behind the fluid-dynamic multi-point shape optimization of turbomachinery blades.

To this end, section 2.1 elaborates on fluid-dynamic optimization problems in general. Section 2.2 focuses on shape optimization problems specific to turbomachines. Robust design optimization is described in section 2.3, with an emphasis on multi-point optimization. This chapter is concluded by section 2.4 containing a literature review on multi-point shape optimization studies with regard to turbomachinery applications.

The reader is advised on the notation of variables throughout this report; bold-face notation denotes vectors, or an array of variables. Normal face symbols denote scalar variables. Roman symbols are parameters and italicized symbols are variables. A vector x is a column vector and x^{T} a row vector. The superscript k indicates a quantity at iteration number k during optimization. Similarly, superscripts 0 and * indicates quantities at the initial and optimized states respectively.

2.1. FLUID-DYNAMIC DESIGN OPTIMIZATION

Optimization is the procedure of finding the most suitable solution to the problem at hand. More specifically, finding the best values for a set of design parameters in order to optimize some measure of quality or fitness that quantifies how good a design is with regards to the observed problem. Problems may have a single or multiple solutions or single or multiple competing objectives. This work focuses on a variant of the multi-objective problem, namely multi-point.

This section presents a background on numerical optimization algorithms, starting with the standard form of general nonlinear constrained problems for fluid-dynamic shape optimization, followed by an elaboration on gradient-based and gradient-free optimizers. For the extension of single point optimization, as described in this section, to multi-point, see section 2.3.1.

2.1.1. SPECIFICATION

A bound-constrained optimization problem generally has the following mathematical form [Vanderplaats and Sugimoto 1986]. Here adapted for the fluid-dynamic application in this work. The goal of the optimization is to find the set of n design variable values α_i of the design vector $\boldsymbol{\alpha}$ which minimizes the objective function $\mathcal{J}(\boldsymbol{\alpha})$.

Minimize:

$$\mathcal{J}\left(\boldsymbol{U}(\boldsymbol{\alpha}), \boldsymbol{X}_{\text{vol}}(\boldsymbol{\alpha})\right) \tag{2.1}$$

Subject to:

$$\boldsymbol{U}(\boldsymbol{\alpha}) = \mathcal{G}(\boldsymbol{U}(\boldsymbol{\alpha}), \boldsymbol{X}_{\text{vol}}(\boldsymbol{\alpha})) \tag{2.2}$$

$$\boldsymbol{X}_{\text{vol}}(\boldsymbol{\alpha}) = \mathcal{M}_g(\boldsymbol{\alpha}) \tag{2.3}$$

$$\boldsymbol{g}_j(\boldsymbol{\alpha}) \leq 0 \tag{2.4}$$

$$\boldsymbol{h}_k(\boldsymbol{\alpha}) = 0 \tag{2.5}$$

$$\alpha_i^{\text{L}} \leq \alpha_i \leq \alpha_i^{\text{U}} \tag{2.6}$$

Where:

$$\boldsymbol{\alpha} = \left[\alpha_1, \alpha_2, .., \alpha_n\right]^{\text{T}} \tag{2.7}$$

$\mathcal{J}$ is some quantitative measure of the performance of the system under investigation. In fluid-dynamic optimization, $\mathcal{J}$ is obtained from the output of a fluid-dynamic simulation. $\mathcal{J}$ is thus a function of the flow state variables $\boldsymbol{U}$ and the coordinates of the fluid domain, or volume, discretization $\boldsymbol{X}_{\text{vol}}$, both a function of $\boldsymbol{\alpha}$. Hence, $\mathcal{J}$ is indirectly a function of $\boldsymbol{\alpha}$. In this work, $\boldsymbol{\alpha}$ is comprised of merely geometric variables. Section 3.1.1 describes the approach to how $\boldsymbol{\alpha}$ characterizes the blade design, termed parametrization. Section 5.2.1 shows the specific values of $\boldsymbol{\alpha}$ of the blade optimized in this work, termed the baseline configuration. $\mathcal{G}$ represents a fixed-point flow solver, which solves the governing flow equations in an iterative manner. $\mathcal{M}_g$ represents the mesh generation procedure, outputting the coordinates of the nodes discretizing the flow domain $\boldsymbol{X}_{\text{vol}}$ from $\boldsymbol{\alpha}$. $\boldsymbol{g}_j$ and $\boldsymbol{h}_k$ represent the vector of inequality and equality constraint vectors respectively. Each variable in the design vector is bounded by an upper and lower bound α_i^{U} and α_i^{L} respectively. This ensures that designs evaluated during optimization do not vary too much, reducing the risk of simulation errors.

Figure 2.1 illustrates a constrained design problem with two design variables. Here, the center of the contours depict where $\mathcal{J}$ is minimal. The non-linear function c_1 and linear function c_2 pose inequality constraints on the design space, causing the optimal design vector $\boldsymbol{\alpha}^*$ of the constrained problem, to not be at the center of the contours but at the intersection of the constraints.

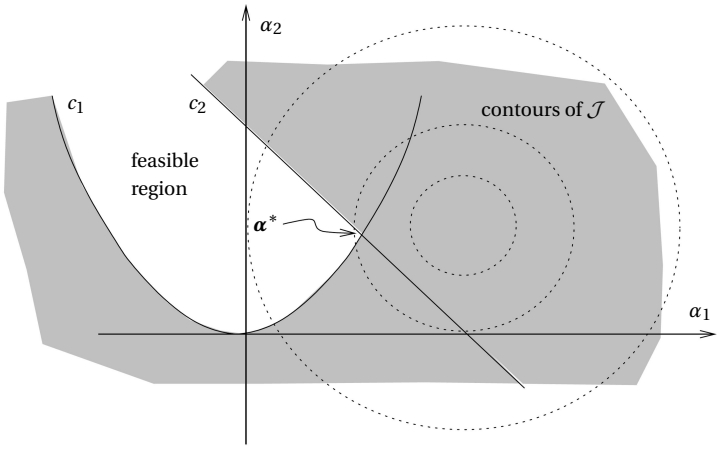


Figure 2.1: Geometrical representation of the design space defined by design variable α_{1-2} and constrained by inequality constraints c_{1-2}. Illustration adapted from Nocedal and Wright [2006].

2.1.2. OPTIMIZATION ALGORITHM

Once the optimization problem is formulated, the optimization algorithm attempts to find the configuration of $\boldsymbol{\alpha}$ within its bounds which minimizes $\mathcal{J}$, while satisfying the constraints. Most optimization algorithms utilize an iterative approach. There are many numerical optimization algorithms, and their performance is

highly problem dependant [Zingg et al. 2012], thus no algorithm can be considered best for all design problems.

The no free lunch theorem is an important notion within design optimization. It states: "for any algorithm, any elevated performance over one class of problems is offset by performance over another clas" [Wolpert and Macready 1997]. It refers to the reality that there may not be one algorithm optimal for all classes of problems. Problems which may vary in for instance simulation cost, number of response variables, response stochasticity, or *noise*, and modality. As a consequence, problem-specific trade-offs will have to be made. Though, this poses an opportunity for engineers to apply problem-specific understanding tailoring optimization algorithms.

The existence of multiple local optima in the design space, or multi-modality, should be taken into account in the selection of optimization algorithms in aerodynamic shape design [Leung and Zingg 2012]. A larger number of design variables tends to increase the modality of the problem to be optimized [Chernukhin and Zingg 2013], the same applies to increasing the geometric dimension of the problem from two to three-dimensions. In the referenced study, at least eight local optima were found for a blended wing optimization with 368 design variables, where the objective function value varies by approximately 5%.

Design optimization algorithms can be classified into gradient-free and gradient-based. A key aspect of optimization is the trade-off between exploration and exploitation of the design space. Gradient-free algorithms do not require the gradient of the objective function to determine the next $\boldsymbol{\alpha}$ to evaluate, but perform a random or systematic sweep over the design space, and seek for a global optimum, displaying exploration properties. Global optimizers are better suited to deal with noisy and multi-modal objective functions and in general demonstrate a greater computation scalability because evaluations can be performed independently [Skinner and Zare-Behtash 2018]. Their main benefit being the lower risk of getting trapped inside a local minimum. In contrast, gradient-based algorithms do make use of the gradient, and generally converge to a local optimum, tending towards exploitation of the design space.

In fluid-dynamic shape optimization, gradient-based algorithms generally require one to two orders of magnitude less function evaluations until convergence [Skinner and Zare-Behtash 2018; Zingg et al. 2012]. However, the difference generally reduces for increasing number of constraints, because each constraint requires a separate adjoint flow solution. The following summarizes the benefits of the gradient-based and gradient-free optimization algorithms:

Gradient-based:	+ Low number of evaluations.
	+ Problem exploitation.
	+ Convergence rate.
Gradient-free:	+ Problem exploration; global minimum.
	+ Robust to noisy problems.
	+ Discrete design variables.
	+ Parallel evaluation of candidate designs.

Gradient-based algorithms are applied frequently in numerical design optimization due to the generally favourable convergence rate and thus lower number of function evaluations. Lower number of function evaluations decreases the number of flow simulations, which especially in fluid-dynamic optimization, means a shortened development cycle, which is an important driver in optimization algorithm performance. The following subsections elaborate on the steps of general gradient-based and gradient-free optimization algorithms.

GRADIENT BASED

Gradient-based optimization algorithms rely on the function gradient with respect to $\boldsymbol{\alpha}$ in determining the next design point to evaluate. The general steps of gradient-based optimization are presented in Figure 2.2. Two main steps exist per iteration k, starting from the initial design vector $\boldsymbol{\alpha}^0$ at iteration $k = 0$. Firstly, the search direction S^k at iteration k is identified. A first-order method to determine S^k is the Steepest-Descent (SD) method, making use of the first derivative of $\mathcal{J}$. Second-order methods require the second derivative

of $\mathcal{J}$. Secondly, a one dimensional line search for the step length d^k is performed. For which generally a backtracking algorithm is used, aiming to find a sufficient decrease of its merit function. See the Wolfe conditions for an approach for specifying a sufficient decrease [Wolfe 1969]. The process is finished when a convergence criterion is satisfied. Such as a small enough change in objective function value or design vector magnitude or number of design iterations.

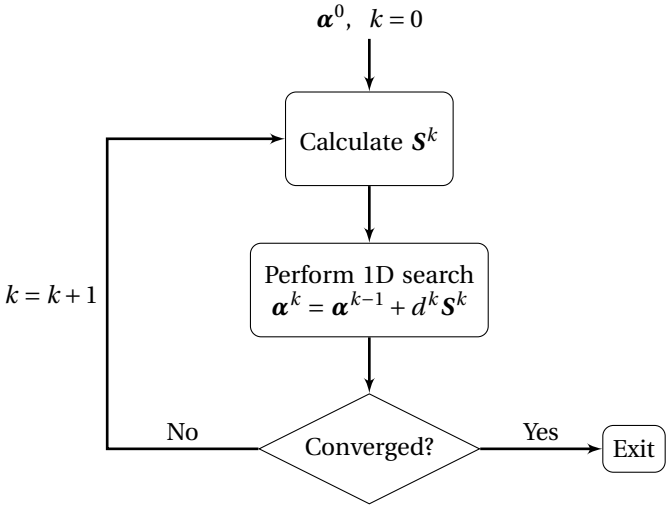


Figure 2.2: General unconstrained gradient-based design optimization [Vanderplaats 1999].

Second-order methods improve on convergence rate, but require the calculation of the function Hessian, which can be obtained with the Newton method as shown in section A.2. Obtaining the Hessian is expensive with numerical methods and generally not available using analytical methods. Thus in practice, quasi-Newton methods are applied which approximate the (inverse) Hessian from the first derivative of $\mathcal{J}$, as detailed in subsection A.2.1.

An important method of enforcing constraints is the Lagrangian multiplier approach. In this method, constraints are added in the expression for the objective function, multiplied by the Lagrange multipliers λ_j, as shown in Equation 2.8. λ_j is zero when constraint j is satisfied, or inactive, and non-zero when the constraint is violated, or active. When all constraints are satisfied, $\boldsymbol{\alpha}$, or the design, is called feasible.

$$\mathcal{L}(\boldsymbol{\alpha}) = \mathcal{J}(\boldsymbol{\alpha}) + \sum_{j=1}^{m} \lambda_j c_j(\boldsymbol{\alpha}) \tag{2.8}$$

In this dual optimization problem, a design vector $\boldsymbol{\alpha}^*$ is deemed optimal if the following three conditions are met, also known as the Karush–Kuhn–Tucker conditions. These aim to ensure that the solution of the constrained problem at convergence is feasible and locally optimal.

1. $\boldsymbol{\alpha}^*$ is feasible.
2. $\lambda_j = 0$ for inactive c_j.
3. $\frac{d\mathcal{L}}{d\boldsymbol{\alpha}} = 0$ for $\boldsymbol{\alpha}^*$.

GRADIENT FREE

Gradient-free methods do not need the gradient of the problem, thus do not require the problem response to be continuous and are less sensitive to noisy objective functions. Notable gradient-free algorithms are the evolutionary algorithm (EA) introduced by Holland [1992], the Particle Swarm Optimization (PSO) [Eberhart and Kennedy 1995; Venter and Sobieszczanski-Sobieski 2012], Simulated Annealing [Kirkpatrick et al. 1983; Tiow et al. 2002] and surrogate-assisted optimization. The latter increases the convergence rate over EA, see appendix B.1. Of gradient free algorithms, the EA is applied frequently in fluid-dynamic shape design [Aissa et al. 2019; Bonaiuti and Zangeneh 2009; Châtel et al. 2020; He and Zheng 2017; Ma et al. 2021; Mengistu and Ghaly 2008; Tüchler et al. 2018; Zingg et al. 2012].

Evolutionary algorithms optimize through mimicking the process of evolution. Where individuals with higher fitness have increased likelihood to survive. A population of individuals, each individual containing

a set of variables $\boldsymbol{\alpha}$, are evaluated and ranked according to a fitness metric. The best performing individuals reproduce among themselves to generate the subsequent population. Evolutionary algorithms in its basic form are comprised of the steps as shown in figure 2.3. Firstly, the initial population is generated randomly, and parametrized to convert the design vector *genotype* to design geometry *phenotype*. In each following iterations k until the optimality condition is met, the fitness of the individuals in the population is evaluated, selection is applied of the individuals which produce the offspring, and random perturbations are applied.

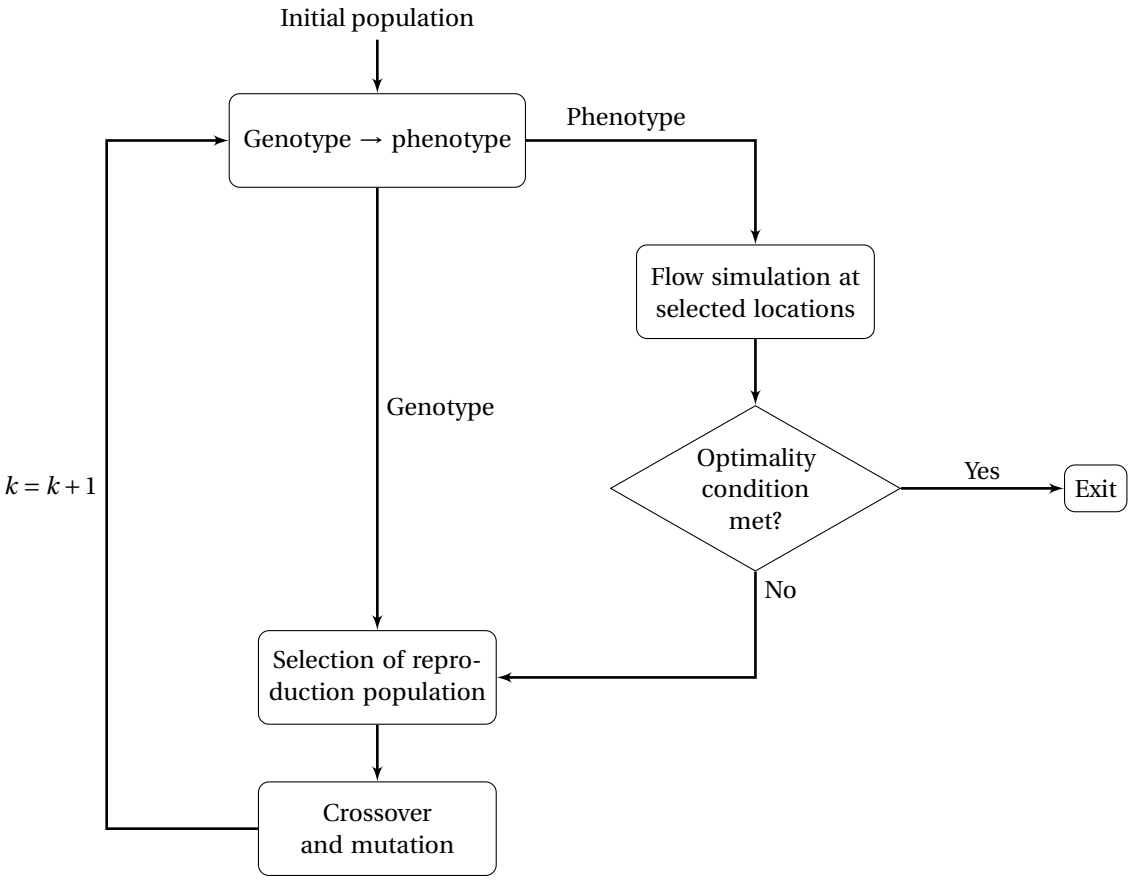


Figure 2.3: Evolutionary-based optimization [Skinner and Zare-Behtash 2018].

Main factors influencing the performance of evolutionary algorithms are population size, selection method (elitism, roulette wheel, tournament selection) and the reproduction (crossover and mutation) method. Many approaches exist to counter premature convergence, of which enlarging the population size is often applied.

Two main approaches exist for constraining global optimization problems, the penalty method and the augmented Lagrangian multiplier method. The penalty method adds a penalty term to the objective function which penalizes violated constraints. Constraint feasibility is thus not enforced directly. The Lagrangian method reformulates the optimization by adding Lagrangian multipliers and enforces the Karush–Kuhn–Tucker conditions of optimality, directly enforcing constraints.

HYBRID

Hybrid optimization algorithms combine a gradient-free and gradient-based algorithm, to benefit from their respective strengths. A popular hybrid approach is to initiate optimization with a global optimizer and switch to a gradient-based optimizer. This in order to leverage the exploration qualities of the global optimizer by identifying regions of interest, and the exploitation qualities of the gradient-based optimizers in order to accelerate convergence to $\boldsymbol{\alpha}^*$.

Three main hybridization policies are identified in order to couple the two methods [Skinner and Zare-Behtash 2018]. The first is pre-hybridization in which first a set of individuals is pre-optimized separately using a gradient-based approach, which are then further optimized using a gradient-free method [Azab and

Ollivier-Gooch 2012; Chernukhin and Zingg 2013]. The second is organic hybridization in which at each generation of a gradient-free optimization, members are improved by using a gradient-based method [Catalano et al. 2015; Châtel et al. 2020]. The third is post-hybridization, in which members of the final population of a gradient-free optimization are improved by a gradient-based optimization [Jansen et al. 2010; Kim et al. 2014; Pini et al. 2014].

SURROGATE-ASSISTED

Surrogate-assisted optimization is gaining interest within fluid-dynamic turbomachine design optimization, as shown in the review on related work in section 2.4. Surrogate-assisted optimization utilizes a surrogate model to provide function value estimations for candidate points in design space. The main objective of surrogate-assisted optimization is to optimally make use of the information on the design space given by the evaluated points [Stork et al. 2020]. Compared to genetic algorithms, the valiantly named Efficient Global Optimization (EGO) algorithm greatly reduces the number of costly function evaluations [Jeong et al. 2005]. However, difficulties arise when constructing interpolating surrogate models on noisy problems. Appendix B elaborates on the EGO algorithm, which is readily available in open-source optimization software.

2.1.3. FLOW SOLVER

Fluid-dynamic simulation serves as a proven indispensable tool in the design of geometries interacting with flows such as turbomachinery. In this work, the flow quantities are assumed to be in steady-state. This allows to increase the computational efficiency by a factor of 10–100, by not having to compute the flow solution at multiple time steps [Huang and Ekici 2013]. The two-dimensional compressible Reynolds-Averaged equations of Navier-Stokes (RANS) are applied as flow governing equations. The RANS equations are a system of coupled, non-linear partial differential equations. They are derived from the conserved flow quantities of mass, momentum and energy and the continuum hypothesis. The exact analytical solution to the NS equations are not available for all turbulent flows. For this reason, turbulence closure models are employed which provide heuristic relations to estimate the turbulent flow properties.

The non-linear partial differential equations of state can be compactly defined as in Equation 2.9. The exact formulation of these equations is deemed outside of the scope of the work. $\boldsymbol{R}(\boldsymbol{U})$ is the residual vector stemming from the estimation of the state. It is obtained by integrating a source term over the fluid control volume Ω and summing the convective and viscous fluxes over the edges of Ω. Solving Equation 2.9 requires for a separate discretization for the temporal and spatial derivatives.

$$\int_{\Omega} \frac{\partial \boldsymbol{U}}{\partial t} d\Omega + \boldsymbol{R}(\boldsymbol{U}) = 0 \tag{2.9}$$

Reynolds-averaging decomposes the flow quantities into a mean and fluctuating component and ignores the latter. This time-averaging of the flow quantities incurs modeling errors when inherently unsteady processes must be simulated. Unsteadiness describes flow phenomena that change with time, such as divergence and flutter.

Other flow solving approaches exist which do not apply time averaging, such as Direct Numerical Simulation (DNS) and Large-Eddy Simulation (LES). In DNS, all turbulent length scales are resolved. Unfortunately, DNS is intractable for most fluid-dynamic problems as of writing. LES aims to bridge the computational cost gap by resolving most of the length scales and modeling the remaining smallest.

RANS equations are solved using an iterative approach, as depicted by the fixed-point flow solver function $\mathcal{G}$. $\boldsymbol{U}^*$ represents the converged flow system of equations, at which the residual vector $\boldsymbol{R}(\boldsymbol{U}^*, \boldsymbol{X}_{\text{vol}})$ of the system is zero.

$$\boldsymbol{U}^{n+1} = \mathcal{G}(\boldsymbol{U}^n, \boldsymbol{X}_{\text{vol}}) \tag{2.10}$$

The Eddy viscosity hypothesis underpinning turbulence models such as Spalart-Allmaras [Spalart and Allmaras 1992] and SST $k-\omega$ [Menter 1993 1994], itself assumes that the Reynolds stress is proportional to the mean shear rate of the flow. As a result, anisotropic factors such as streamline curvature contribute to modelling error. Streamline curvature is significant for large flow-deflection flows in turbines and corner flows present in three-dimensional turbomachine problems [Iaccarino et al. 2017; Mehdizadeh et al. 2012].

An effective low-cost solution would be to implement an Explicit Algebraic Reynolds Stress Model type turbulence model [Mehdizadeh et al. 2012], which extends the (linear) Eddy viscosity model with non-linear relations between the deviation of Reynolds stress and mean shear rate. An alternative - actively studied - method is using machine learning to model the turbulence closure [Bru 2020; Schmelzer et al. 2020; Yin

et al. 2020; Zhao et al. 2020]. However, these forms of turbulence models are not readily implemented across open-source simulation codes at this moment.

2.2. OPTIMIZING FOR BLADE SHAPE

As described in the introduction, turbomachines are designed sequentially in roughly three steps. This work is situated between steps two and three. When the blade coordinates are determined, but not yet taking into account the full three-dimensional blade. However, the methods applied in this work can be extrapolated to optimize three-dimensional, multi-cascade and multi-discipline turbomachinery problems.

General turbomachine design requirements at this design stage are to obtain the required flow deflections with the least amount of lost work, while verifying that the design can bear mechanical stresses. In the detailed design stage, the design requirements may be extended by taking into account structural stiffness, ease of manufacturing and maintenance. This section elaborates on concepts regarding the numerical optimization of two-dimensional turbomachines in light of the former design requirements.

Figure 2.4 shows the frame of reference in the blade-to-blade plane of a two-dimensional turbine blade, as applied in this work. The kinematic components of a fluid particle are illustrated, aswell as a set of streamlines with arrows at constant time intervals, in order to provide an insight into a general flow field of a turbomachine cascade. The streamlines are colored according to the local fluid Mach number.

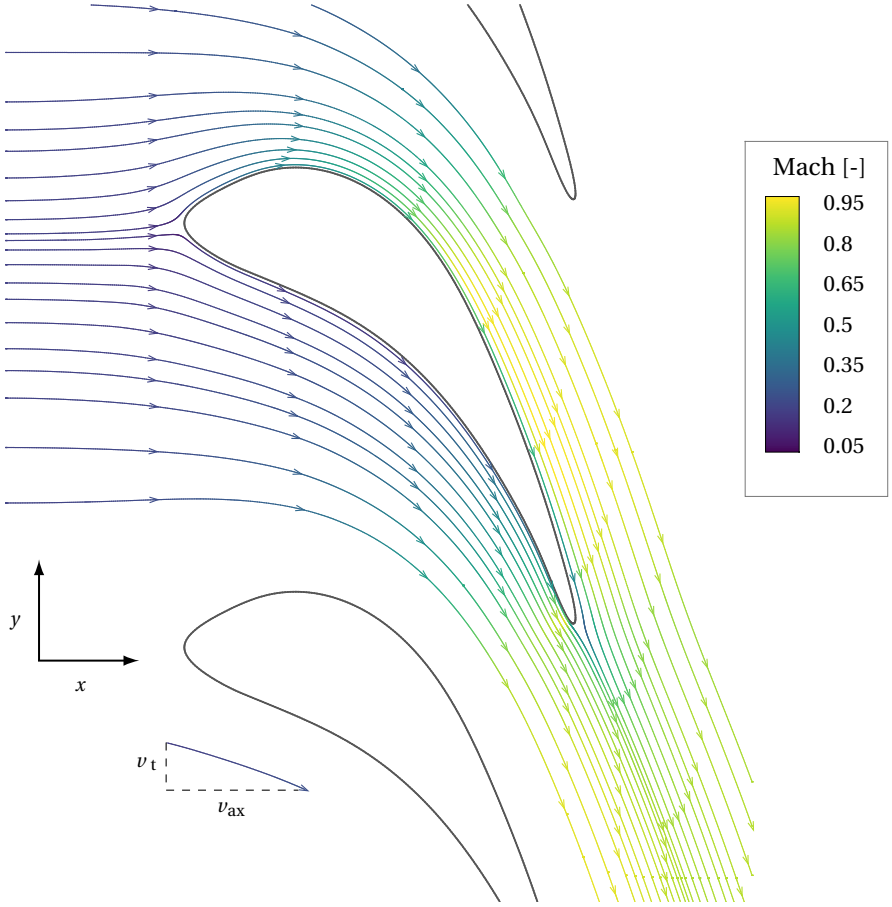


Figure 2.4: Frame of reference, kinematic components and stream traces in the blade-to-blade plane of a simulated fluid for an example turbine cascade. Stream traces are colored according to local Mach number of the fluid.

2.2.1. TWO-DIMENSIONAL LOSS MECHANISMS

Understanding the loss mechanisms in fluid-dynamic turbomachinery modeling and their fundamental physics aids in explaining the behavior of the optimized flow later on in this work. Three main loss mechanisms are identified in the two-dimensional adiabatic fluid-dynamic simulation of turbines [Denton 1993; Sieverding

and Manna 2020]. The first are losses around the blade profile such as viscous dissipation in the boundary layer and dissipation across shockwaves. Losses in the boundary layer are associated with shear forces resulting from the velocity differential of the flow, tangential to the blade surface. The second are mixing losses downstream of the trailing edge (TE) where the flows of the suction and pressure side coalesce. These losses can be linked to the static pressure at the TE, also termed base pressure. The third loss mechanism represents losses due to shockwaves downstream of the blade.

Viscous losses generally decrease for increasing Reynolds numbers (Re). The overall loss coefficients increase rapidly for increasing flow Mach numbers around unity, as is presented in Figure 2.5. At low subsonic numbers, viscous losses are dominant. While at transonic to supersonic speeds, losses due to shocks become dominant [Denton 1993; Mee et al. 1992]. At transonic Mach numbers, the losses downstream of the turbine blades are shown to be approximately 70% of the total two-dimensional losses [Xu and Denton 1988].

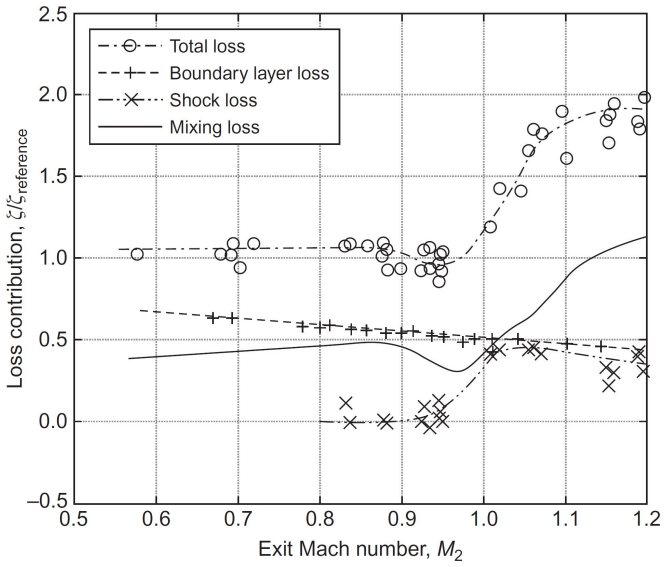


Figure 2.5: Experimental measurements of components to the loss coefficient of a turbine blade cascade for varying outflow Mach numbers (Re = 1.0×10^6) [Mee et al. 1992].

Flow uniformity at the outlet boundary is an important measure for turbine stator cascades when not simultaneously optimizing trailing cascades. Because it can promote separation of the boundary layer flow on downstream cascades and thus influence the fluid-dynamic efficiency of the turbine. When optimizing multiple cascades, downstream cascades can adapt the geometry in order to comply with the inflow characteristics. However, this measure is not included directly inside the objective function in order to restrict the scope of the work.

Entropy generation s_{gen} is an irreversible process and thus a useful indicator of lost work. The value of s is always relative, thus arbitrary. s is not measured directly, only inferred from other measured properties. For a calorically perfect gas, the entropy generation between two states can be expressed by Equation 2.11. This expression is obtained from the ideal gas thermodynamic equations of state. Where c_p is the specific heat for constant pressure, R is the universal gas constant, T is the temperature and p pressure.

$$s - s_{\text{ref}} = c_p \ln\left(\frac{T}{T_{\text{ref}}}\right) - R \ln\left(\frac{\rho}{\rho_{\text{ref}}}\right) \tag{2.11}$$

The loss in axial turbines may also be characterized by stagnation enthalpy loss or pressure loss. However, in rotating cascades of radial turbomachines, the relative stagnation pressure and enthalpy (rothalpy) of the fluid varies with the radius with respect to the axis of rotation, without an implied loss of work. Thus if the design problem consists of both rotating and stationary cascades, it is convenient to consider s_{gen} as a measure of loss.

2.2.2. NUMERICAL RESPONSES

This section describes the quantities of interest of the numerical fluid-dynamic optimization of turbomachines. These are the objective function and constraints on the geometry. Chapter 3 describes which quantities of interest are considered in this work. In addition, adjoint methods to obtain the flow sensitivity are described briefly.

OBJECTIVE FUNCTION

The decision on the performance metric for optimization depends on the design requirements of the study. A few examples of fluid-dynamic performance metrics are the kinetic energy loss ζ_K, total pressure loss ζ_P and the non-dimensional entropy generation s_{gen}.

s_{gen} is calculated from s at the inlet and outlet boundaries using mixed-out averaging $\langle\bar{\cdot}\rangle$. The components of the mixed-out averaging are mass, momentum and enthalpy. This method of averaging can be considered to produce a better representation of the flow than the non-weighted method, because non-uniformities of the flow quantities are emphasized. $T_{0,in}$ is the stagnation temperature at the inlet. v_0 is the spouting velocity, which is defined from the stagnation enthalpy at the inlet $h_{0,in}$ and the static enthalpy at the outlet $h_{is,out}$ as in Equation 2.13. $h_{is,out}$ is obtained from isentropic expansion. In this work, s at the inlet and outlet is calculated using Equation 2.11.

$$s_{gen} = \frac{T_{0,in}}{v_0^2}\left\langle \bar{s}_{out} - \bar{s}_{in} \right\rangle \tag{2.12}$$

$$v_0 = \sqrt{2(h_{t,in} - h_{is,out})} \tag{2.13}$$

It is worth noticing that solely focusing on s_{gen} during optimization can be detrimental for aerodynamic stability [Peeren and Vogeler 2017]. However, this is not an issue in the single-discipline fluid-dynamic optimization as performed in this work.

TURBINE CONSTRAINTS

In optimization of stator blades in single-cascade designs, the flow angle at the outlet boundary β_{out} is generally constrained to be larger than a set angle with respect to the meridional. This serves to ensure a sufficient flow turning. See Equation 2.14 for the expression of β_{out}, which is calculated from the mixed-out average of the flow velocity in tangential direction v_t and axial direction v_{ax}. Without constraining the outlet flow angle, the optimizer would reduce s_{gen} by nullifying the flow turning over the cascade. This is because flow turning causes flow acceleration, which increases viscous dissipation losses and hence increases s_{gen}.

$$\beta_{out} = \left\langle \overline{\arctan\left(\frac{v_t}{v_{ax}}\right)} \right\rangle \tag{2.14}$$

A constraint on the mass flow rate $\dot{m}$ can be utilized in order to keep the flow within the operating envelope of the machine [Châtel et al. 2020]. For instance, to ensure combustion of the fuel remains possible in gas turbines, or to avoid problem exploitation by the optimizer by reducing $\dot{m}$ to reduce s_{gen} [Montanelli et al. 2015].

When regarding multi-cascade design problems, a constraint on the shaft power can be applied in combination with a constraint on β_{out} of the downstream stator blade [Vitale et al. 2020]. The shaft power constraint ensures sufficient power transfer from the fluid and indirectly forces adequate flow deflections of the middle cascades.

ADJOINT METHOD FOR FLOW SENSITIVITY

When optimizing using gradient-based optimizers, the sensitivity of the objective function with respect to $\boldsymbol{\alpha}$ must be given to the optimizer. A trivial method for obtaining sensitivities is to perform finite-differencing, where the difference in quantity of interest is divided by the perturbation size. A frequently applied method to obtain these sensitivities in fluid-dynamic and structural problems is the adjoint method. In fluid-dynamic problems for instance, this approach enables an efficient calculation of the flow sensitivity. Of which the computational cost is linear in the number of dependent variables, i.e. flow quantities of interest. Hence, if the cumulative number of $\mathcal{J}$ and $\boldsymbol{c}$ is roughly more than the number of design variables, finite-differencing is preferred. This break-even is dependent on the compute-time factor of an adjoint over a direct flow simulation. Section 3.1.3 elaborates on the calculation steps of the adjoint method.

Two main approaches to obtaining the adjoint system of flow equations are identified; the continuous and discrete adjoint. In the continuous approach, the adjoint equations are derived from the continuous governing flow equations and discretized separately. Whereas in the discrete approach, the adjoint equations are derived from the already discretized flow equations, therefore being discretized analogously. As a result, the discrete approach returns sensitivities which are consistent with the direct flow solution. A drawback of the discrete approach is the increased difficulty of implementing turbulence models and a larger requirement on memory and computing time [Skinner and Zare-Behtash 2018]. However, the discrete method is readily implemented, and the computational complexity is deemed to not pose a time bottleneck for two-dimensional flow simulations. The discrete approach is also shown to be more robust because its accuracy is less sensitive to the coarseness of the mesh [Nadarajah and Jameson 2000].

2.2.3. GEOMETRY PARAMETRIZATION

An important aspect of design optimization of turbomachinery is the geometry parametrization approach. It is necessary to restrict the optimization complexity and cost. Selecting the parametrization method is a balance between cost, fidelity and robustness. On the one hand it must not restrict the optimal design too much. On the other, it should appropriately represent the design space with a minimal number of design variables. It influences the *design landscape*, the mathematical relations underpinning the optimization problem, because it determines what shapes and topologies can be considered during optimization [Anand et al. 2018]. Thus it affects the problem modality, linearity and continuity.

Parametrizing the two-dimensional blade for fluid-dynamic simulation constitutes two parts. These are parametrization of the surface and the mesh. See Figure 2.6 for the simplified parametrization chain as applied in this work. It is worth noticing that the pictured mesh is simplified for sake of clarity and thus does not portray the exact mesh parametrization approach adopted in this work.

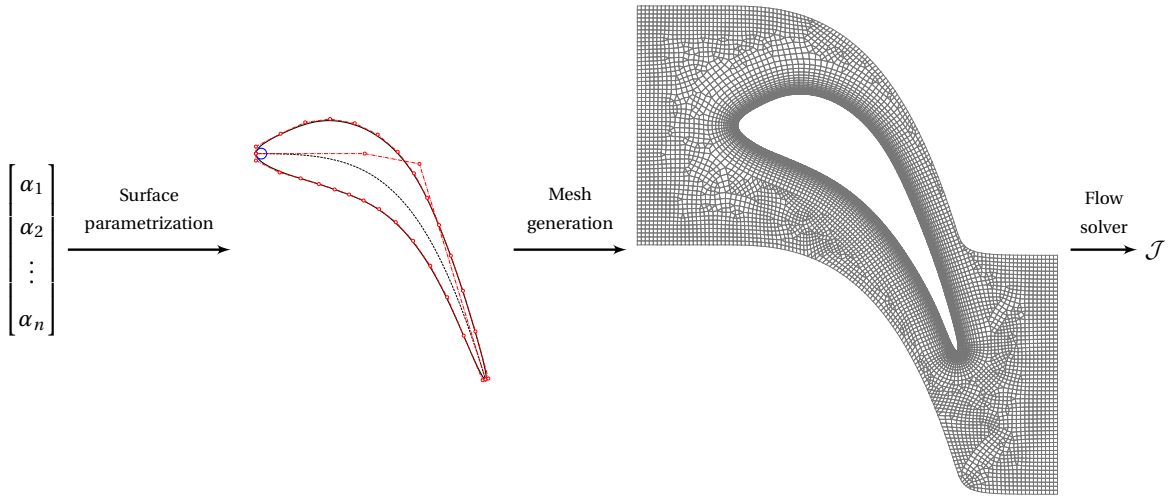


Figure 2.6: Simplified parametrization chain from design vector $\boldsymbol{\alpha}$ to objective function $\mathcal{J}$. Note that the fluid domain is meshed coarsely for illustration purposes.

SURFACE

Surface parametrization techniques can be classified into constructive and deformative methods. In turbomachinery, the two most frequently applied surface parametrization techniques are Computer-Aided Design (CAD) and free-form. These belong to the constructive and deformative classes respectively. Free-Form Deformation (FFD) [Sederberg and Parry 1986] is applied to the free-form parametrized design to adapt the design coordinates during optimization. The CAD and FFD methods are efficient and suitable for complex configurations [Samareh 2001].

In constructive methods, the body shapes are defined by functions such as splines and partial differential equations. CAD parametrization produces blade profiles with continuous curvature and rate of change of curvature, reducing the likelihood of flow separation [Korakianitis and Papagiannidis 1993]. This method also allows imposing of geometric constraints (for example minimum blade thicknesses) inherently by virtue of the constrained design variable being directly present in the design vector. CAD methods also retain com-

patibility with CAD software for post-processing, facilitating manufacturing.

In deformative methods, the coordinate values of the geometry form the design variables [Jameson et al. 1997], likely requiring significantly more design variables than constructive methods. Free-form parametrization allows to locally refine shapes and thus provides a larger freedom in design and application flexibility, at the cost of difficulties in satisfying geometrical constraints.
Summarizing the benefits of constructive and deformative parametrization methods:

Constructive:	+ Smoother shapes. + Natural geometric constraint imposition. + Post processing.
Deformative:	+ Larger design space. + Detailed design.

MESH

$\mathcal{J}$ is strongly dependent on the simulation domain discretization. The flow domain is discretized by a mesh, which is needed in a finite-volume approach. There are two main classes of methods to generating a mesh for the design iterations during optimization. The first is deforming an existing baseline mesh and the second is generating a mesh anew at each iteration, otherwise known as *integrated meshing*.

The deformation approach poses difficulties for maintaining consistency for the whole design space. In other words, it becomes difficult to reliably produce meshes of adequate quality when the subsequent designs during optimization differ greatly. The large deformations may produce negative volume cells or large-aspect cells which increase the spatial discretization error of the flow simulation.

The downside of integrated meshing, in contrast to FFD, is that each created mesh may not have an equivalent topology. This inconsistent topology does not ensure an equivalent flow problem being solved, increasing the numerical noise in the problem response and sensitivity between successive designs during optimization. However, this may be the only feasible meshing approach when using global optimization algorithms, due to its lower sensitivity of mesh quality to differences in design geometry [Samareh 2005]. This sensitivity may be large when utilizing large bounds on the design variables. Added, the deterioration in mesh quality for larger geometry deformations is shown to be more steep for grids containing triangular elements than structured hexahedral [Secco et al. 2021].

An approach to increase reliability of optimization is by employing adaptive meshing. This ensures mesh consistency by locally refining the mesh until a convergence property of the quantity of interest is satisfied, while being computationally cheaper and yielding improved optimized designs [Li and Hartmann 2015].

Meshes of different grid fidelity can be applied in multiple ways during optimization. One is the multigrid method, in which the flow simulation uses a sequence of grids of differing fidelity [Arnone 1994]. This increases convergence towards the flow solution of a design. Another is multi-level optimization, which performs optimization in stages on increasingly fine meshes [Lyu et al. 2014]. When implemented correctly, it accelerates convergence towards the optimized design, because a major part of the optimization occurs on the cheaper to evaluate coarser meshes.

The coarseness of the mesh elements has effects on fluid-dynamic losses in the simulated flow. The first is an incomplete resolution of viscous effects such as turbulence production, leading to a miscalculation of mixing losses. The second appears when using upwind finite volume schemes and is an increased numerical diffusion, stemming from the truncation error of the discretization method. This effectively increases the viscosity of the flow, increasing mixing losses. Under-resolving of turbulence decreases the boundary layer thickness, while increasing flow viscosity increases the boundary layer thickness.

An important metric of the mesh fidelity of the boundary layer for a particular flow is the dimensionless wall distance y^+. It is utilized to determine the mesh element height at the near domain walls. Different turbulence closure models recommend different values of y^+ for reasonably capturing the turbulent processes.

2.3. ROBUST DESIGN OPTIMIZATION

it is desirable for a design to still perform satisfactorily when conditions change. Robust optimization techniques are applied to account for uncertainty in design parameters. The process of finding robust solutions is

termed robust design optimization. In context of turbomachine design, sources of uncertainties can be manufacturing, systematic or operational. An example of systematic uncertainty are random effects in computer simulations. Operational uncertainty can be caused by disparate operating conditions or indirectly through response stochasticity resulting from sensory measurement errors [Jin and Branke 2005]. Verification of robust designs is performed *a posteriori* by varying uncertain variables according to an assumed distribution.

In complex systems design applications such as multidisciplinary design optimization, uncertainty can arise from having to independently optimize a certain discipline with incomplete information of the interactions of other disciplines [Kalsi et al. 2001]. The complexity of the whole system and time constraints affect the incompleteness of information.

An example of uncertainty quantification for robust multi-objective optimization is of minimizing the mean and standard deviation of $\mathcal{J}$ while varying a set of uncertain variables. However, uncertainty quantification as method of robust optimization is omitted from the scope due to increased cost of running extra numerical flow evaluations to resolve the statistical moments of uncertain variables.

Multi-objective (MO) optimization is a way to mitigate operational uncertainty by optimizing for multiple design objectives simultaneously. A desired and challenging task for compressors is an efficient and safe operation in all operating conditions [Dixon and Hall 2014b]. Specifically, a sufficient surge margin under inlet distortions and large tangential acceleration. Exceeding the stability limit can cause rotating stall or surge, frequently observed in the operation of compressors in automotive engines [Galindo et al. 2008]. These instability modes lead to loss in system performance and increase in mechanical stresses. Examples of multi-objective design problems for compressors are enlarging the range between choking and surge margins. Another is increasing the magnitude of inverse proportionality of pressure rise and $\dot{m}$ for stability reasons, and efficiency at a certain operational $\dot{m}$. Example competing objectives for turbines are minimizing s_{gen} while maximizing the stage loading at a fixed $\dot{m}$ and inlet and outlet conditions. Maximizing the stage loading reduces cascade solidity, reducing the number of blades and hence manufacturing and life-cycle cost [Dennis et al. 2001].

Implicit gradient smoothing is a method to increase the robustness of the design optimization process. It aims to stabilize the optimization by smoothing of the sensitivities of for example blade surface nodes in a region of surface non-smoothness. This smoothing acts to precondition the gradient vectors, reducing the risk of failing operations in the optimization algorithm [Jameson and Kim 2003]. Hence allowing for larger design steps and an implied reduction of design iterations, saving time.

Multi-point optimization is a subclass of multi-objective optimization and focuses on optimizing for a certain range of operational conditions. Multi-point objective function examples for turbine optimization include minimizing total pressure loss over multiple operating conditions [Montanelli et al. 2015],

2.3.1. MULTI-POINT OPTIMIZATION

A general multi-point optimization problem consists of optimizing a set of $l \geq 2$ objective functions $\mathcal{J}_1(\boldsymbol{\alpha})$, $\mathcal{J}_2(\boldsymbol{\alpha})$, ..., $J_l(\boldsymbol{\alpha})$. $\boldsymbol{\mathcal{J}}$ is thus a vector of l objective functionals $\boldsymbol{\mathcal{J}}$ as in Equation 2.15. The decision variables are the design variables $\boldsymbol{\alpha}$. The feasible design space is defined by the set of solutions which satisfy all constraints. In an optimization problem with merely inequality constraints, the feasible design space is defined as $\{\boldsymbol{\alpha} \mid g_j(\boldsymbol{\alpha}) \leq 0\}$. The criterion space, or objective space, contains the set of solutions and is spanned by the axes of objective functions, $\{\boldsymbol{\mathcal{J}}(\boldsymbol{\alpha}) \mid \boldsymbol{\alpha} \in \boldsymbol{y}\}$.

$$\min_{\boldsymbol{\alpha}} \boldsymbol{\mathcal{J}}(\boldsymbol{\alpha}) = \begin{bmatrix} \mathcal{J}_1(\boldsymbol{\alpha}) \\ \mathcal{J}_2(\boldsymbol{\alpha}) \\ \vdots \\ \mathcal{J}_l(\boldsymbol{\alpha}) \end{bmatrix} \tag{2.15}$$

In multi-criteria optimization, a solution $\boldsymbol{\mathcal{J}}$ is dominated by another solution $\boldsymbol{\mathcal{J}}^*$, *iff* $\boldsymbol{\mathcal{J}}^*$ is equally good or better than $\boldsymbol{\mathcal{J}}$ with respect to all objectives, and $\boldsymbol{\mathcal{J}}^*$ is better than $\boldsymbol{\mathcal{J}}$ for at least one objective [Haimes and Li 1987]. A solution is termed Pareto optimal if there is no other solution that improves upon *at least one* objective without deteriorating another objective [Pareto et al. 1975].

Pareto optimality is an important notion in multi-criteria optimization. Assuming it is not possible to obtain a single design solution which simultaneously minimizes all objectives, a decision has to be made on the basis of the Pareto front. Figure 2.7 shows an illustration of an example criterion space of two objectives

$\mathcal{J}_{1-2}$. The Pareto optimal points which form the Pareto front are marked in blue and dominated points in red. The utopia point $\mathcal{J}^*$ is the point in criterion space which is a minimum for all objectives, i.e. $\mathcal{J}_i^* = \min_{\boldsymbol{\alpha}} \{ \mathcal{J}_i(\boldsymbol{\alpha}) \mid \boldsymbol{\alpha} \in \boldsymbol{y} \}$ [Vincent and Grantham 1982]. And thus not attainable in practice.

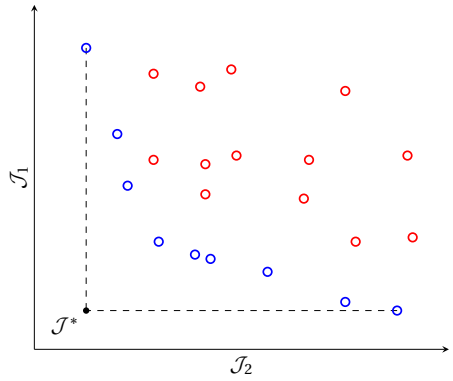


Figure 2.7: Example criterion space of a bi-objective optimization problem with competing objective functions $\mathcal{J}_{1-2}$. In multi-point optimization, $\mathcal{J}_{1-2}$ represent the objective function values at operating conditions 1 and 2. The Pareto optimal solutions are marked blue and the utopia point $\mathcal{J}^*$ is indicated.

Multi-criteria decision analysis has been an active field of research over the the past few decades. The most challenging problems are the identification or estimation of the Pareto front for problems of increasing dimensions [Figueira et al. 2009]. For which evolutionary algorithms are employed more frequently than exact methods.

It is not always straightforward to combine conflicting performance measures into one scalar value for optimization. A decision must generally be made which expresses a certain preference over the objectives in order to achieve this. In multi-criteria optimization, generally four classes of decision-making approaches are identified, as enumerated below [Hwang and Masud 1979]. In the following, a few methods are detailed which serve to demonstrate the concepts behind the decision making approaches. These examples form by no means an exhaustive treatment of this topic.

1. No-preference.
2. *A priori.*
3. *A posteriori.*
4. Interactive.

In the no-preference method, the objectives are combined in such a way that no weight is assigned to any objective. An example is the method of global criterion [Zeleny 1973] as shown in Equation 2.16. In which the multi-objective function response of a point in criterion space to be minimized is its worst performing objective.

$$\min_{\boldsymbol{\alpha}} \mathcal{J}_{\text{MO}} = \max_{1 \le i \le j} \left[\mathcal{J}_i(\boldsymbol{\alpha}) - \mathcal{J}_i^* \right] \tag{2.16}$$

An example of an a priori method is lexicographic ordering. In this method, the decision maker must rank the different objectives in order of importance. Then, a sequence of single-point optimizations are performed in the determined order of importance. The advantage of this method is its simplicity.

An a posteriori method is applying a weighted sum of the individual objectives to obtain a single objective function, as shown in Equation 2.17. The benefit of this method is ease of implementation on all optimization algorithms due to its differentiability, enabling the operation of gradient-based optimizers. The drawback is that it introduces an extra empirical degree of freedom in choosing suitable weights. Ideally, weights should be deducted from the objective functions to accurately approach a preference function [Messac 1996].

An example of an interactive method is the reference point method. It is an ad-hoc approach in which a reference point in criterion space is striven for, and iteratively adapted during the optimization process according to the state of the Pareto front. This method is desirable because it is easy for the decision maker to

understand, however no consistent strategy and defined convergence criterion with respect to a final solution.

A notable drawback of using Pareto front-based methods is difficulty of dealing with many-objective problems, especially for problems with more than three objectives. Besides the difficulty of visualization beyond three dimensions, the number of evaluations required to approximate the Pareto front scales exponentially with the dimension of the criterion space [Ishibuchi et al. 2008].

2.3.2. MULTI-POINT WEIGHTING METHODS

There are various methods of applying weights to $\mathcal{J}_i$ in order to obtain a scalar objective function value for MO optimization $\mathcal{J}_{\text{MO}}$. The most natural method is the weighted-sum method for which the composite objective function is presented in Equation 2.17.

$$\min_{\boldsymbol{\alpha}} \mathcal{J}_{\text{MO}} = \sum_{i=1}^{m} w_i \mathcal{J}_i(\boldsymbol{\alpha}) \qquad \text{for} \quad i \in [0,..,m] \tag{2.17}$$

In accordance with the *no free lunch theorem*, there exists no single best method of mathematically attributing weights to operating conditions for all problems [Marler and Arora 2004]. With gradient-based MO, in theory, optimizing the aggregate objective function yields solutions on the convex part of the Pareto front. However, even if the intention is to obtain a spread of points along the Pareto front, it is not possible to decide on a weighting method without knowing the shape of the Pareto front, which is not known a priori [Das and Dennis 1997].

The selection of operating condition weights also affects the obtained Pareto front, hence influencing the trade-off nature of the problem [Das and Dennis 1997]. Assigning weights to objectives directs the optimizer to solutions in the criterion space, which may lead to clustering of solutions. Thus possibly not capturing a Pareto set, and not giving full insight into all possible compromise solutions. However, the weighted sum method to obtain the Pareto front can be validated when also evaluating Pareto front-finding optimizers [Nemec et al. 2004], such as the Non-dominated Sorting-based multi-objective Genetic Algorithm II (NSGA2) [Deb et al. 2002] and Multi-Objective Particle-Swarm Optimization (MOPSO) [Moore et al. 2000].

Another method of allocating weights is to minimize Bayes' risk over the operating points [Huyse and Lewis 2001; Pini et al. 2014]. In this method, the expected risk ρ of the loss of the design over the operational lifetime is minimized. This weighting method is shown in Equation 2.18. Where $f_{\boldsymbol{C}}(C_i)$ is the probability density function of the set of operating conditions $\boldsymbol{C}$. This is an intuitive approach in that individual operating conditions C_i are weighted proportional to the fraction of operational lifetime. However, knowledge is needed about the operational lifetime before implementation.

$$\rho(\boldsymbol{\alpha}, \boldsymbol{C}) = \min_{\boldsymbol{\alpha}} \int_{\boldsymbol{C}} \mathcal{J}(\boldsymbol{\alpha}, C_i) f_{\boldsymbol{C}}(C_i) d\boldsymbol{C} \tag{2.18}$$

Multiple variations of applying the utopia point for weighting in criterion space exist. Examples range from using the summed distance to an exponential distance metric [Chankong and Haimes 2008], to local gradient in criterion space. The Utopia point weight method assigns weights to points which do not yield large increase in performance, thus attempting to increase the overall performance gain by focusing on points which might yield larger improvements when given more emphasis [Marler and Arora 2004]. A drawback is that each operating condition has to be optimized beforehand in order to obtain the performance at each operating condition. Another Utopia point method is to make use of the gradient of the design vector [Montanelli et al. 2015]. Here, weights are assigned to solutions in criterion space inversely proportional to the magnitude of its gradient in criterion space. Stagnant solutions are thus emphasized. The advantage of coupling this method of weight assignment with gradient-based optimization is the availability of the functional gradients.

$$w_i = \left| \frac{1}{\frac{d\mathcal{J}_i}{d\boldsymbol{\mathcal{J}}}} \right| \tag{2.19}$$

Tchebycheff scalarization does not utilize the utopia point, it uses the weighted objective function value of the worst performing operating condition as MO objective. See the expression in Equation 2.20. It has proven convergence properties towards the Pareto front, even for non-convex problems [Golovin and Zhang 2020]. However, it is not differentiable, and thus not usable for coupling with a gradient-based optimization algorithm.

$$\min_{\boldsymbol{\alpha}} \mathcal{J}_{\text{MO}} = \max_{i \le i \le j} \left[\, w_i \mathcal{J}_i(\boldsymbol{\alpha}) \, \right] \tag{2.20}$$

2.3.3. MULTI-POINT SELECTION METHODS

Optimizing a multi-point problem such that optimal performance is achieved over the desired operational range requires selecting a finite number of operating points from a continuous range of operating conditions. Selecting operating conditions for multi-point optimization is an important part of the formulation of the design problem, because it affects largely the computational time of the optimization. It must capture the operational range with a minimum number of sample points. For a robust design it is important to sample points not arbitrarily similar or not close to the nominal design condition [Huyse and Lewis 2001; Kenway and Martins 2016]. If the selected points do not represent competing operating conditions, the optimization result reduces to a single point result.

When the goal is to attain a constant performance metric throughout the operational range, selection of operating conditions can be performed iteratively during optimization [Li et al. 2002; Zingg and Elias 2006]. By sampling points in the operating range and adding the disparately performing conditions into the optimization problem until a satisfactorily low variation in $\mathcal{J}$ throughout the operational range is observed. In their approach, the added disparately performing operating conditions are weighted higher aswell.

The Gradient Span Analysis (GSA) method seeks to find operating points in the operational range which do not overlook potential other points which can be improved without degrading the existing ones [Montanelli et al. 2015]. The rationality behind it is to not miss operating points which could be included that improve the objective without degrading other operating points. This independency of $\mathcal{J}$ with respect to $\boldsymbol{\alpha}$ occurs between points with linearly independent gradient vectors. Thus selecting the operating points with disparate gradient vectors provides a cost function with largest sensitivity to the actual desired operating range.

2.3.4. NSGA2

Of the available Multi-Objective Genetic Algorithms (MOGA), the NSGA2 [Deb et al. 2002] is successfully implemented for multi-point fluid-dynamic shape optimization due to its favourable Pareto front resolving properties [Droandi and Gibertini 2015; Kim et al. 2014; Samad et al. 2009; Wang et al. 2011]. It emphasizes a diverse Pareto-front by taking into account a crowding factor during selection of offspring.

Strength Pareto Evolutionary Algorithm-II is another common MOGA [Zitzler et al. 2001]. However, it generally performs on par or slightly worse during later generations, and at a significant cost of computational time [Bui et al. 2004; Kaucic et al. 2019; King et al. 2016] compared to NSGA2.

NSGA2 largely follows the main steps of evolutionary algorithms illustrated in Figure 2.3. Non-dominated sorting is needed to sort individuals in a multi-objective criterion space. It orders a set of solutions by their nondomination *level*; by iteratively identifying the Pareto front and discounting it from the set of all solutions until all solutions are assigned a nondomination level. The procedure of non-dominated sorting of points in criterion space is illustrated in Figure 2.8. The domination levels of a set of points in the two-dimensional criterion space of $\mathcal{J}_{1-2}$ are given by the numbers.

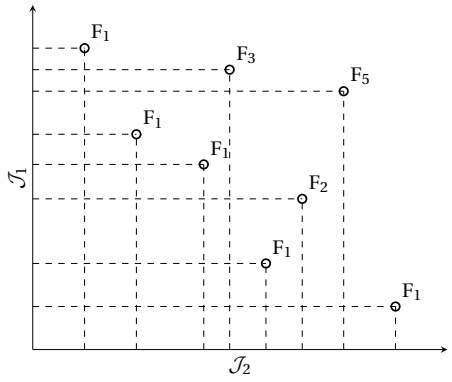


Figure 2.8: Ranking of individuals in criterion space according to nondomination level F_i of a bi-objective problem. The numbers in subscript indicate the nondomination rank of each individual. The rank number of an individual can be obtained pictorially by counting the number of other individuals inside its (hyper-)volume enclosed by its $\mathcal{J}_i$ values and the origin of the criterion space.

Figure 2.9 presents the procedure for selecting the configurations of the next iteration. Elitism selection is employed to select the best individuals of a population by combining the P_t and Q_t. For iteration t, P_t is the set of parent solutions and Q_t is the set of offspring solutions. $P_t \cup Q_t$, is reordered through non-dominated sorting after which each Pareto-front set is sorted by crowding distance sorting. Crowding distance sorting favours solutions in lesser crowded locations. The parent population of the next iteration, P_{t+1} is subsequently passed through selection, crossover and mutation. Crowding distance sorting is applied, based on mean cuboid side length with respect to all solutions in the objective function space. Solutions away from crowded areas are favoured.

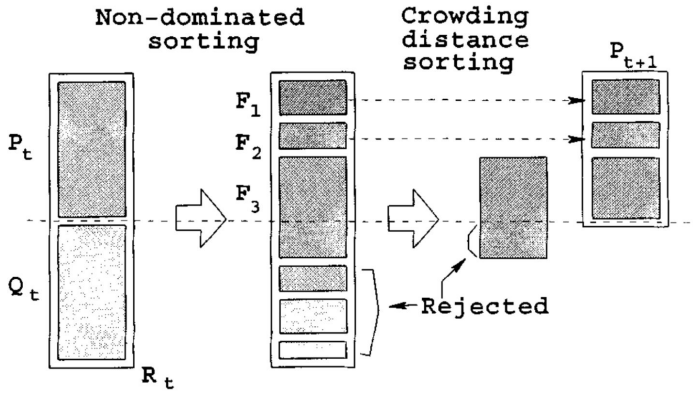


Figure 2.9: The NSGA2 algorithm [Deb et al. 2002]. Non-dominated sorting and crowding distance sorting are applied in sequence to rank the parent P and offspring Q population of iteration t, of which a fixed number of the best individuals form the basis of the parent population of the next iteration P_{t+1}.

Selection under constrained optimization is performed through binary tournament selection. Here, the constrained-dominant of two randomly picked solutions are chosen. A solution i is constrained-dominant over solution j, if either:

- Solution i is feasible and solution j not.
- Solution i has smaller overall constraint violation, in case solution j is also infeasible.
- Solution i dominates solution j, in case solution i and j are feasible.

2.4. RELATED WORK

This section serves to present a literature review of key research on multi-point turbomachinery optimization in the recent years. Emphasis is placed on the setup of the blade type, design parametrization and deformation, optimization algorithm and flow sensitivity calculation. Where possible, the comparison of design

performance of the single to multi-point results are detailed in order to present the merit of multi-point optimizations. Research is treated in ascending chronological order.

Mengistu and Ghaly [2008] employ a genetic algorithm to find the best design on a Artificial Neural Network (ANN) surrogate-model for a two-dimensional NACA65 subsonic compressor rotor. The objective function is formed by three terms. The first term contains the adiabatic efficiency of four equally weighted operating points of differing back-pressure. The second term penalizes the variation in adiabatic efficiency between the operating points. The third term penalizes three constraint violations. The constraint terms enforce a design within a certain reduced $\dot{m}$ and inlet and outlet flow angles. The blade profile camber line and thickness are parameterized by a Non-Uniform Rational B-Spline (NURBS) curve containing 10 and 7 control points respectively. The ANN has one input node for each design variable, 41 nodes in the single hidden layer and four output nodes, one for the adiabatic efficiency and one for each of the constraints. The ANN is trained with points selected through Latin Hypercube Sampling (LHS). The final feasible design displays a stable increase of 7% along all operating conditions.

Bonaiuti and Zangeneh [2009] perform an inverse optimization of the three-dimensional HP9 single-stage axial compressor using a MOGA guided by a quadratic Polynomial Regression (PR) response surface model. The objective is to minimize the offset to a target span-wise distribution of swirl velocity on the stage interface. Optimized geometries for two linear span-wise loading distributions are compared; hub loaded and shroud loaded. An inverse approach is selected in order to have a more direct control of the aerodynamic performance, which in turn allows for omitting conventional flow constraints in the objective function such as $\dot{m}$ and flow turning. Initial surrogate training points are selected using a fractional factorial design based on Taguchi orthogonal arrays. The meridional channel geometry and blade thickness are fixed, while the blade surface is parameterized by five direct geometric design variables. Varying $\dot{m}$ by 90% and 110% of nominal design represent the optimization operating conditions. Final design displays a wider operating range and improves stage efficiency by 1.3% to 2.3%.

Wang et al. [2010] optimize a three-dimensional rotor-stator configuration of an axial compressor stage for minimum entropy generation, while fixing the geometry of the upstream guide vane. Two operating points at the peak efficiency with weight 0.8 and near-choke with weight 0.2 are selected, and $\dot{m}$ and stagnation pressure ratio are constrained by additional weighted terms in the objective function. The continuous adjoint method is applied to limit memory and compute-time resources, providing gradients to the Steepest-Descent optimizer. Hicks-Henne shape functions are applied to parameterize the shape perturbation during optimization. The multi-point optimization increases peak efficiency by 1% and yields a better design than separate single point optimizations of the two operating conditions.

Luo et al. [2013] optimize adiabatic efficiency of the NASA 67 Rotor at three operating conditions denoting operation at different peak efficiency, stall and choke. $\dot{m}$ and total pressure ratio are equality-constrained. Blade shape is perturbed through a sum of weighted Hicks-Henne shape functions, located at 16 points at each side of the blade profile, at 9 span-wise locations. The weights of the shape functions constitute the design vector. The objective function is set as a weighted sum of operating points. Although the multi-point optimization yields lower peak efficiency gain than the single-point optimization, it yields larger adiabatic efficiency over the range of operating conditions. Efficiency improvements lie between 0.56% and 1.24%.

Pini et al. [2014] optimizes the uniformity of the exit Mach number of a two-dimensional supersonic ORC turbine stator cascade at three operating conditions of varying static back-pressure. Uncertainty quantification is performed to obtain the stochastic mean of the three operating points on which the objective function is based. A SD optimizer perturbs the blade surface consisting of NURBS curves, with 11 points at each side of the blade. The single point optimized design only performs better at the nominal operating condition, whereas the multi-point shows improved averaged performance over the range of operating conditions. The multi-point optimization reduces the averaged total pressure loss coefficient by 66% versus 32.5% for the single-point optimization. The multi-point optimization yields a more robust design aswell, by significantly reducing the variance of performance at the considered operating conditions.

Walther and Nadarajah [2015a] report a multi-point optimization of a three-dimensional rotor-stator setup of a transonic axial compressor, the Darmstadt Rotor No. 1, in order to minimize entropy generation.

The off-design points are characterized by a simultaneous increase and decrease of the static back-pressure and rotor rotational velocity by 5% with respect to the main operating point. The off-design operating points are chosen rather arbitrarily, due to the main focus of the work being on showcasing the functionality of the multi-point optimization framework. The main operating point accounts for half of the weight of the objective function, while the two off-design points have a weight of 0.25 each. The SNOPT [Gill et al. 2002] optimizer is applied to obtain the final set of design variables. The shape of the rotor and stator is perturbed at nine radial locations by a total of 162 Hicks-Henne bump functions. Entropy generation is reduced at all operating points, ranging between 11.04% and 16.4%.

Montanelli et al. [2015] optimize the two-dimensional LS89 for minimum total pressure loss at the outlet plane at multiple operating points determined by the GSA method. For a description of the GSA method, see section 2.3.3. The GSA method determines 5 optimal operating points within $M_{is} = [0.7, 1.1]$. A weighted objective function is minimized using a L-BFGS-B optimizer, which is an active-set optimizer using a limited-memory BFGS method [Zhu et al. 1997] The weights of which are obtained by the Utopia point method, as described in section 2.3.2. The outlet $\dot{m}$ is constrained by a lower limit to limit entropy production decrease by lowering $\dot{m}$. The blade surface is perturbed directly by altering sectional design variables such as chordwise thickness and maximum camber location, while freezing the leading and trailing edge geometry. A total of 25 variables constitute the design vector. The multi point setup reduces total pressure loss at all operating points between 0.16% and 0.90%. The single point setup yields a slightly favourable final performance at its design point, while increased total pressure loss at off-design conditions.

He and Zheng [2017] perform a bound-constraint optimization of the German Aerospace Center (DLR) three-dimensional transonic centrifugal impeller SRV2-O case with a genetic algorithm on an ANN surrogate-model. The objective function is a sum of the isentropic efficiencies at the peak efficiency and near surge points, and a penalty term enforcing a lower bound on $\dot{m}$ at the two operating points. The rotor camber line, sweep and lean are parameterized by a total of 14 Bezier control points. An ANN with one output value is constructed for each term in the objective function. Each ANN has an input neuron for each design variable, two hidden layer with 18 neurons and one output neuron. Initial 50 sample points are selected using Latin Hypercube Sampling (LHS). Peak efficiency is increased by 2.2% and choke $\dot{m}$ is increased by 8.1%, both averaged over the operating range.

Torreguitart et al. [2019] minimize the entropy generation of the LS89 turbine vane at isentropic Mach conditions of $M_{is} = [0.9, 0.955, 1.01]$. Equal weights are set to three operating conditions. Three constraints are enforced; the trailing edge thickness, axial chord length and the outlet flow angle. Algorithmic differentiation is applied to obtain the sensitivities for the grid and surface. A multi-block structured grid is generated at each design iteration. The L-BFGS-B optimizer is used, with bounds on the design variables. The optimal design decreases entropy generation by between 2% and 14% for the respective operating conditions.

Aissa and Verstraete [2019] perform an aero-thermal optimization of a radial compressor blade for maximum efficiency at two operating points, while constraining the maximum mechanical stress. Other constraints are imposed on the choke $\dot{m}$, the momentum of inertia and the pressure ratio. A bounded Kriging surrogate model is used. A differential evolution (DE) optimizer is considered to find optimal points on six surrogate models, one for each operating point and constraint. Bounded Kriging favours points close to known points, as to evade infeasible points. Ordinary Kriging, they find, does not provide satisfactory feasible designs during optimization, which leads to failed flow evaluations and premature termination of the optimization. Initial 129 flow evaluations for surrogate model training are selected using a factorial design. The weighted sum of objectives for the highly constrained optimization is improved by 2.59%.

Song et al. [2019] optimize a turbine of a cryogenic air separation unit at six uniformly spaced operating conditions of varying rotational velocity, inlet pressure and inlet temperature. A Cooperative Co-Evolution Algorithm (CCEA) finds optimal design points on a Kriging surrogate model. In CCEA, sub-populations optimize subsets of the design vector. For each sub-population at each generation, the ignored part of the design vector is copied from the fittest individuals of the other sub-populations. Using CCEA, they report, reduces the number of iterations until convergence. The objective function is formed by two terms; the first aims to maximize the isentropic efficiency at all equally weighted operating conditions. The second term aims to minimize the variation in isentropic efficiencies at the operating conditions, similarly to Mengistu and Ghaly

[2008]. The meridional channel geometry is deformed by NURBS curves through six control points, the blade surface is deformed by three blade twist angles. The multi-point optimized design consistently outperforms the single-point optimized design on all operating points by up to 1.5%.

Aissa et al. [2019] compare the convergence rate and final value of the SNOPT optimizer to a DE optimizer on a bounded Kriging surrogate model. The problem is a two-dimensional von Karman Institute LS-82 axial turbine vane profile with 19 geometric design variables, two operating points and two constraints. The gradient-based SNOPT combines the operating point values and gradients with equal weight. A central-composite design method is applied to select surrogate model training points. Three surrogate models are constructed; one for the weighted sum of the objective functions and one for each constraint value. The metaheuristic optimization requires roughly twice the number of CPU hours compared to SNOPT, while converging to a slightly improved design. A decrease in averaged total pressure loss is obtained of 22% for SNOPT versus 24% for Kriging combined with DE.

Châtel et al. [2020] minimize the entropy loss of the two-dimensional LS89 turbine cascade at multiple equally-weighted points across the operating range using a hybrid optimization approach. The operating points are the same as optimized in [Torreguitart et al. 2019]. The outlet flow angle is constrained to ensure a minimum flow turning and the $\dot{m}$ is constrained to ensure an upper bound. Constraints are imposed at one operating point of $M_{is} = 1.01$ to reduce computational time. Promising design points found by DE are improved locally by SD at each generation, in order to overcome the generally poor convergence performance of the evolutionary algorithm. An in-house multi-block grid generation tool generates the grid automatically during optimization. The losses of the optimized design are decreased at all points by more than 9.5% with respect to baseline, whereas the single point optimization yields a decrease of 21%.

Table 2.1 summarizes the design configuration, optimization algorithm, gradient calculation, geometry deformation and number of operating conditions, design variables and constraints of the regarded studies.

Author	Design config. †	Optimizer	Gradient calculation ‡	Geometry deformation	Number of		
					Points	DV §	C ¶
Châtel et al. [2020]	2D ATs	DE+SD	DA	Bezier+B-Spline	3	19	2
Aissa et al. [2019]	2D ATs	DE+Kriging, SNOPT	DA	B-Spline	2	19	2
Song et al. [2019]	3D RTr	CCEA+Kriging	n.a	Direct+B-Spline	6	9	0
Aissa and Verstraete [2019]	3D RCr	DE+Kriging	n.a.	Bezier+B-Spline	2	34	4
Torreguitart et al. [2019]	2D ATs	L-BFGS-B	DA	Bezier+B-Spline	3	24	3
He and Zheng [2017]	3D RCr	GA+ANN	n.a.	Bezier	2	14	1
Montanelli et al. [2015]	2D ATs	L-BFGS-B	DA	Direct	5	25	1
Walther and Nadarajah [2015a]	3D ACrs	SNOPT	DA	Hicks-Henne	3	162	0
Pini et al. [2014]	2D RTs	SD	DA	NURBS	3	22	0
Luo et al. [2013]	3D ACr	n.a.	CA	Hicks-Henne	3	288	2
Wang et al. [2010]	3D ACrs	Steepest Descent	CA	Hicks-Henne	2	341	2
Bonaiuti and Zangeneh [2009]	3D RCrs	MOGA+PR	n.a.	Direct	2	5	0
Mengistu and Ghaly [2008]	2D ACr	GA+ANN	n.a.	NURBS	4	17	3

† A: Axial, R: Radial, T: Turbine, C: Compressor, s: Stator, r: Rotor.

‡ Flow sensitivity; D: Discrete, C: Continuous, A: Adjoint.

§ DV: Design variables.

¶ C: Constraints.

Table 2.1: Optimization setup of studies on multi-point turbomachinery optimization.

The key takeaways of this chapter in relation to the first two research sub-questions are as below:

- When optimizing using global optimizers on problems where designs vary significantly, the integrated meshing approach provides a method which reduces the risk of mesh element inconsistencies compared to a mesh deformation approach.
- The performance increase on the nominal design point using multi-point optimization is lower as compared to single point optimization, but generally results in an increased performance on a wider operating range compared to single-point optimizations within the same range. This highlights the merit of multi-point optimization for generating designs displaying reduced sensitivity to variations in the operating conditions.
- A shift towards hybrid and surrogate-assisted optimization algorithms is noticeable. However, there is no consensus onto a single optimization algorithm for multi-point optimization. This is expected because the performance of the optimization algorithm is dependent on, among others, the number of operating conditions, design variables and constraints of the optimization problem.
- All works implement their own multi-point optimization code, which incurs an increased collective time cost due to duplication of created software. By providing an open-source optimization tool which is collectively maintained, this code duplication can be avoided, reducing the development time and costs of turbomachinery.
- Constructive parametrization is applied frequently in multi-point optimization studies. The more or less similar number of design variables is due to the applied geometry deformation approaches, affecting the number of design variables required. With works utilizing Hicks-Henne deformation employing a large number of design variables.
- The discrete adjoint approach is preferred over the continuous counterpart due to its consistency with respect to the discretization of the flow, despite the increased difficulty of implementing the turbulence models. This is also reflected in the frequency of Discrete (DA) over Continuous (CA) Adjoint approaches in Table 2.1.

3

METHODOLOGY

This chapter builds on the concepts introduced in the previous chapter and presents the methods implemented for performing the fluid-dynamic shape optimization of turbomachinery blades in ParaBlade. All flow simulations in this work are performed using the Stanford University Unstructured (SU2) computational suite [Economon et al. 2016]. SU2 is an open-source analysis and design tool for solving multi-disciplinary flow problems on unstructured meshes.

An alternative open-source software for aerodynamic shape optimization is the MACH-Aero framework by Martins et al. [2022], which is utilized for optimization of primarily external, but also internal [He et al. 2019], flow problems. It consists of a set of core modules for geometry parametrization and deformation, direct and adjoint flow simulation and optimization. However, this suite relies on their CAD-free surface parametrization module `pyGeo`. In addition, the grid generation module `pyHyp` does not currently support internal flow problems.

The basic chain of operations to obtain the objective function, and in context of optimization is introduced in section 3.1. The reasoning for the implementation of gradient-free, but also gradient-based optimizers for multi-point optimization is described in section 3.3. Although the optimization in this work is performed using a gradient-free optimizer, the method to obtain the design sensitivity is included because the code framework implements gradient-based optimizers aswell. The chapter is concluded by a section on the scaling procedure for the optimization responses and parameters for both single and multi-point optimization cases in section 3.2.

3.1. DESIGN CHAIN FOR FLUID-DYNAMIC SHAPE OPTIMIZATION

The main quantity of interest in a fluid-dynamic optimization is obtained from the flow solver module. The flow solver requires the computational domain to be discretized. Figure 3.1 shows the steps of obtaining the discretized volume, starting from the geometric design variables. The blade surface is constructed by the surface parametrization module $\mathcal{S}$, according to the design variables, using its parametrized specification as described in section 3.1.1. The resulting blade surface coordinates $\boldsymbol{X}_{\text{surf}}$ is input to the mesh generation module $\mathcal{M}_g$ which constructs the corresponding mesh coordinates $\boldsymbol{X}_{\text{vol}}$, as detailed in section 3.1.1. $\boldsymbol{X}_{\text{vol}}$ is fed into the flow solver $\mathcal{G}$ for evaluation into $\mathcal{J}$.

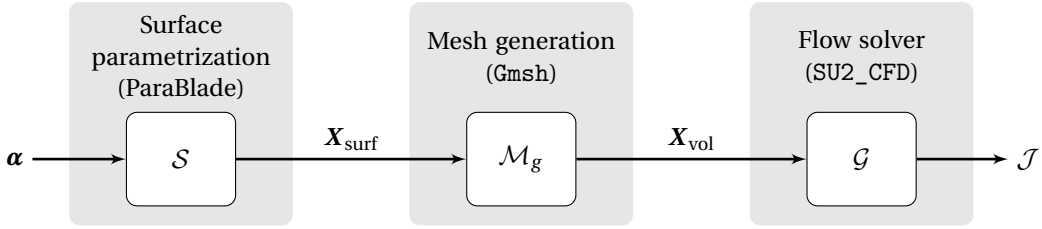


Figure 3.1: Simplified fluid-dynamic shape evaluation chain from design vector $\boldsymbol{\alpha}$ to objective function $\mathcal{J}$.

During optimization, the design is continuously changed and evaluated. Figure 3.2 pictures the Extended Design Structure Framework (XDSM) [Lambe and Martins 2012] of the optimization when using the geometry deformation approach. The XDSM highlights the iterative nature of the optimization. In the diagram, a thick gray line represents flow of variables and a thin black line depicts the sequence of the iterative process.

Before the optimization loop, the process is initialized in which the blade surface and mesh grid of the initial geometry is parametrized, which is detailed in section 3.1.1. $\boldsymbol{Q}$ is a matrix containing the surface coordinates of the reference blade.

The optimizer is the driver of the process. It determines the design vector to be evaluated at each iteration, based on the objective function value and sensitivities of the preceding iteration. It obtains the initial design vector $\boldsymbol{\alpha}^0$ and stops the optimization when the stopping criterion is met, returning the optimized $\boldsymbol{\alpha}^*$. At each flow evaluation, $\mathcal{J}$ and $\boldsymbol{c}$ are returned.

On the diagonal, the deformation blocks represent analysis modules. These modules pre-process the data required by the discipline blocks, embodied by the solver blocks. The output of the flow solver $\boldsymbol{U}$, represents the converged flow system. The multiplicity of the adjoint solver block illustrates that the adjoint solver executes more than once per iteration. This is because it is run once for the sensitivity of the objective function, and once to obtain the sensitivity of each constraint with respect to the design variables.

The iteration loop highlighted in Figure 3.2 resembles optimization using a gradient-based optimizer, because the adjoint solver discipline block is a component of the loop at step six.

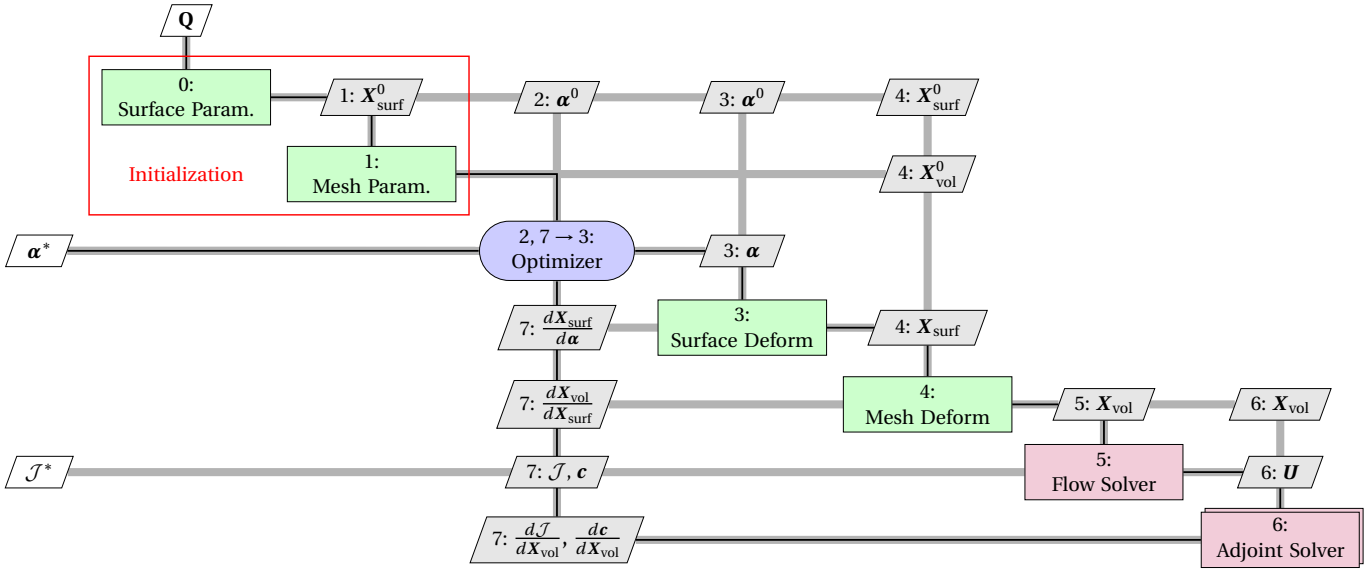


Figure 3.2: Extended design structure framework [Lambe and Martins 2012] for the fluid-dynamic shape optimization of turbomachinery. The numbers in each block indicate the order of execution. The thick gray line depicts a flow of variables, the black line depicts the process flow of a *gradient-based* optimization case.

When performing gradient-free optimization, the adjoint module is omitted from the loop, and the deformation modules are replaced by a single mesh generation analysis module, as illustrated in figure 3.3. The initialization is simplified by forgoing the mesh parametrization module, because the mesh generation inside the loop does not require an initial geometry.

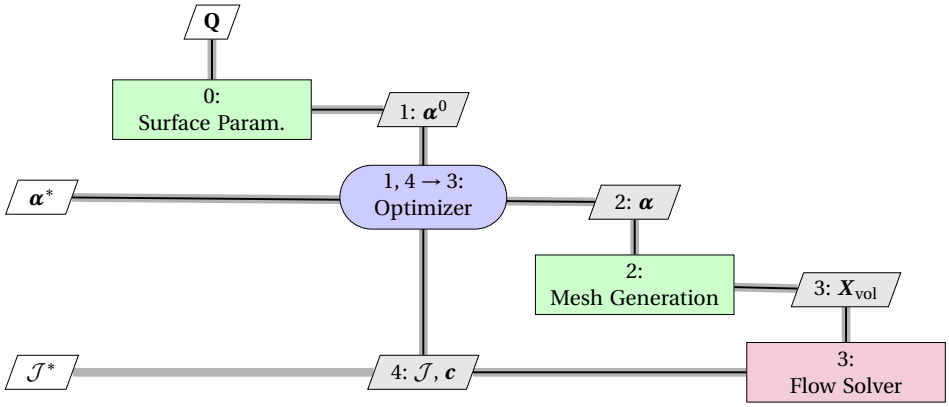


Figure 3.3: Extended design structure framework for the fluid-dynamic shape optimization of turbomachinery using a *gradient-free* optimizer.

3.1.1. PARAMETRIZING GEOMETRY

The geometry of the turbomachinery blades are parametrized in order to reduce the number of design variables. This reduces the problem complexity and hence decreases risks of non-convergence of the optimization. The definition of the geometry of the simulation problem consists of two components. These are the surface of the blade and the mesh which covers the computational domain of the flow simulation.

SURFACE

CAD parametrization is applied for parametrizing the blade surface due to its robustness to surface discontinuities, or non-smoothness, under deformation. Consequently, the problem design space is narrowed. Relaxing this constraint on the design space is considered beyond the scope of this research.

The blade surface is assembled using a camberline and thickness approach. In which a lower and upper thickness distribution $\mathbf{P}_i^{\text{u-l}}$ is imposed on a camber line $\mathbf{C}^{\text{c}}$. $\mathbf{C}^{\text{c}}$ is a cubic B-spline curve constructed using four control points $\mathbf{P}_{0\text{-}3}^{\text{c}}$, as shown in Figure 3.4a. c is the chord length, ξ is the stagger angle and θ_{in} and θ_{out} are the leading and trailing edge metal angles respectively. The intermediate camberline control points $\mathbf{P}_{1\text{-}2}^{\text{c}}$ are defined by the leading edge and trailing edge distances, tangent to the local metal angles. $\mathbf{P}_i^{\text{u-l}}$ are the control points for the upper and lower side NURBS curves $\mathbf{C}^{\text{u-l}}(u)$ and are imposed in the normal direction with respect to the local camber line, as shown in Figure 3.4b. The blade endpoint radii of curvature are imposed exactly through algebraic derivation of the parametric curve expressions. Also, there is no reliance on trimming or intersecting of curves, benefiting the simplicity of the optimization by reducing the number of degrees of freedom.

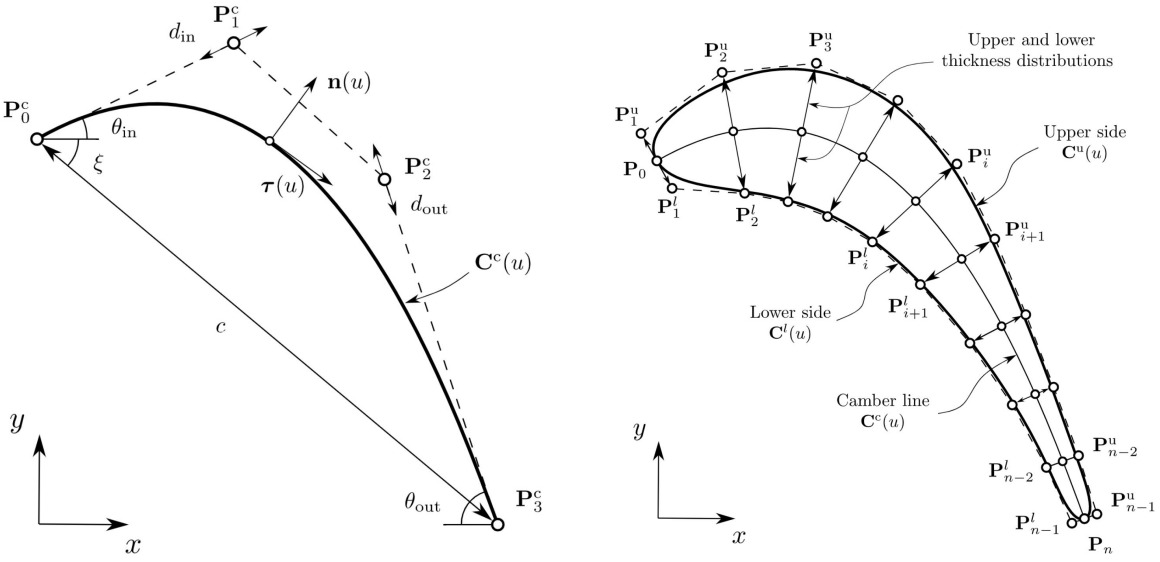
ParaBlade is utilized for parametrization [Anand and Agromayor 2020]. It is a unified blade parametrization tool for turbomachinery blades which is able to handle axial, radial, and mixed flow blade shapes [Agromayor et al. 2021]. Machine-accurate derivatives are obtained through utilizing the complex-step method, as described in section 3.1.3.

The CAD parametrization of the blade surface for optimization consists of two phases, assuming an initial guess for $\boldsymbol{\alpha}$. First, the *point projection* phase in which the reference blade surface coordinates are matched with those of a parametric model. Second, the *geometry update* phase which seeks the design variable values that best approximate the coordinates of the reference blade geometry $\mathbf{Q}$. This systematic approach allows for a consistent parametrization of the blade surface during optimization.

In the point projection phase, parametric values u_i are found which minimize the summed distance of each parametric point to the reference blade coordinates Q_i. The parametric values of the three-dimensional matching case are (u_i, v_i). The point-projection optimization problem of a two-dimensional blade is formulated as in Equation 3.1. Here, C^{b} is a NURBS curve which is constructed by joining $\mathbf{C}^{\text{u-l}}(\boldsymbol{u})$. This $\boldsymbol{u}, \boldsymbol{v}$-parametrization also allows to control the modeling fidelity of the reference blade.

$$\min_{\boldsymbol{u}\in\mathbb{R}} J(\boldsymbol{u}) = \frac{1}{2}\sum_{i=1}^{N_{\mathbf{Q}}} \| \mathbf{C}^{\text{b}}(u_i) - \mathrm{Q}_i \|^2, \quad \text{with} \quad 0 \le u_i \le 1 \tag{3.1}$$

The second phase is the geometry update as formulated in Equation 3.2. In this phase, the set of geometric design variable values $\boldsymbol{\alpha}$ is sought which minimizes the summed distance between $\mathbf{Q}$ and the previously



(a) Camber line representation [Agromayor et al. 2021]. (b) Profile representation [Agromayor et al. 2021].

Figure 3.4: Two-dimensional blade parametrization setup. The camber line (a) is a cubic B-spline spanned by four control points $\mathbf{P}_{0-3}^c$, d_{in} and d_{out} are the inlet and outlet tangent distances. The upper and lower thicknesses in (b) are imposed in the normal direction with respect to the camber line.

determined points of the blade matching. SLSQP is applied for solving both optimization sub-problems. This matching procedure produces parametrizations with errors in the order of magnitude of the manufacturing tolerances for axial gas turbine blades [Agromayor et al. 2021].

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} J(\boldsymbol{\alpha}) = \sum_{i=1}^{N_Q} \| \mathbf{C}^b(u_i, \boldsymbol{\alpha}) - \mathrm{Q}_i \|^2 \tag{3.2}$$

MESH

The two-dimensional computational domain is enclosed by two periodic boundaries and an inlet and outlet boundary, as shown in Figure 3.5. The periodic boundaries are copies of the camber line, translated in tangential direction and offset by a distance of half-pitch, while being constrained to zero slope at the inlet and outlet. The inlet and outlet boundaries are straight lines connecting the periodic boundaries.

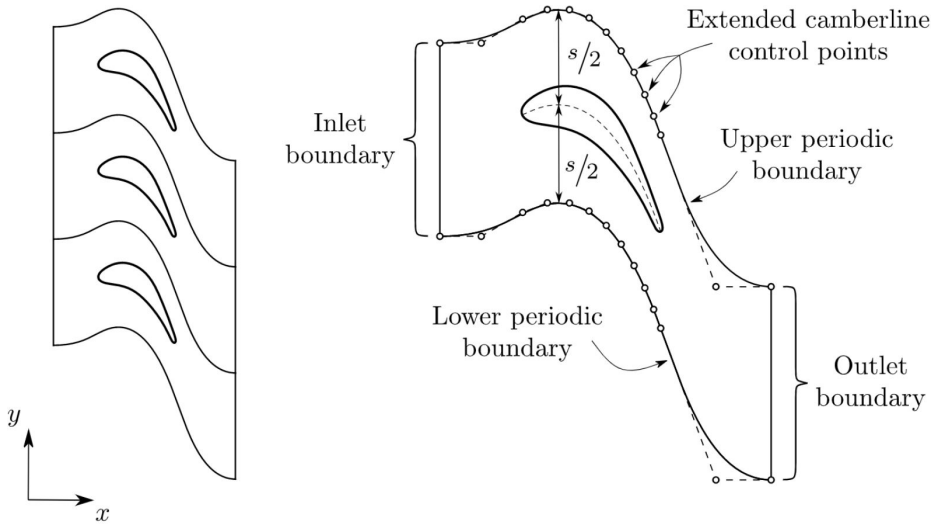


Figure 3.5: Two-dimensional flow domain setup [Agromayor et al. 2021].

The computational domain of the fluid simulation comprises two sections; the first consists of inflation layers close to the blade surface containing quadrilateral mesh elements. The second covers the rest of the domain which contains a mix of quadrilateral and triangle mesh elements. See Figure 3.6 for a zoom onto the inflation layers and flow domain of the baseline mesh. The inflation layers act to resolve the viscous effects due to flow gradients in the boundary layer near the surface of the blade. The main domain is meshed using a Frontal-Delaunay for quads algorithm using a simple recombination [Remacle et al. 2011]. After initial mesh generation, Laplacian smoothing is applied to increase the mesh quality. In order to ensure consistent mesh element topology along the periodic boundaries, the `SU2_PER` executable [Ghidoni 2017] is applied to modify the elements along the periodic boundaries such that the vertex positions are identical for both periodic boundaries.

The main domain and inflation layers are parametrized by the target size of the mesh elements in the domain and along the surface of the blade. The inflation layers are defined in `Gmsh` by the inflation ratio r, the height of the mesh elements along the blade surface h_0 and the total inflation layer thickness θ_T.

$$\theta_\mathrm{T} = \sum_{n=0}^{n_\theta - 1} h_0 \cdot r^n = h_0 \frac{1 - r^n}{1 - r} \tag{3.3}$$

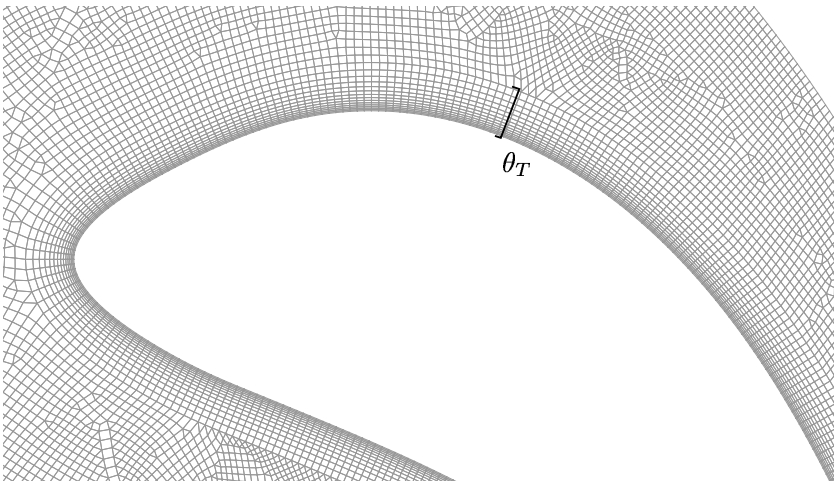


Figure 3.6: Detail of the mesh inflation layers and flow field of the baseline mesh, highlighting the total inflation layer thickness θ_T.

No inflation layers are generated if the target domain mesh size is smaller than h_0. The largest inflation layer height h_* is ensured to not be larger than the target domain mesh size $d_\mathrm{t,D}$. If the largest inflation layer height acquired from the initial parameters $h_{*,\mathrm{calc}}$ (r, θ_T, θ_0), is larger than $d_\mathrm{t,D}$, the cumulative inflation layer thickness parameter θ_final is reduced such that the largest inflation layer height is between $\frac{d_\mathrm{t,D}}{r} < h_* < d_\mathrm{t,D}$. See Algorithm 1 for the described steps for defining the inflation layer.

When ensuring $y^+ < 1$ while specifying a large mesh spacing along the blade surface, convergence issues during flow simulation and mesh deformation can arise [Dwight 2009]. An option for investigation can be to improve the mesh quality by ensuring mesh cell aspect ratios are close to unity along the blade surface.

Mesh generation is performed using `Gmsh` [Geuzaine and Remacle 2009]. `Gmsh` is an open-source finite element mesh generator under active development. It features a CAD engine and graphical user interface to interactively edit and visualize meshes. The CAD engine supports a native approach to specifying the CAD-parametrized curves in the domain. `Gmsh` allows for generating two- and three-dimensional unstructured meshes using various algorithms. The Advancing-front Delaunay for quads [Remacle et al. 2011] grid generation algorithm is selected, which emphasizes the production of quadrilateral mesh elements. Quadrilateral mesh elements reduce numerical diffusion issues in comparison to triangle mesh elements, as described in section 2.2.3.

No multigrid or multi-level optimization methods are applied, to limit the scope of the work and the complexity of the optimization, reducing the risk of time loss. However, a fine mesh is utilized to verify the results of the optimization performed on a coarser mesh, as described in section 5.5.

Algorithm 1 Computation of the inflation layer thickness θ_{final}.

Require: $h_0 < d_{\text{t,D}}$
  if $h_{*,\text{calc}}\,(r, \theta_{\text{T}}, \theta_0) < d_{t,D}$ **then** $\theta_{\text{final}} = \theta_{\text{T}}$
  else
    $\theta_{\text{final}} \leftarrow h_0$
    $h_* \leftarrow h_0$
    while $h_* r < d_{t,D}$ **do**
      $h_* = h_* \cdot r$
      $\theta_{\text{final}} = \theta_{\text{final}} + h_*$
    end while
  end if

3.1.2. DEFORMING GEOMETRY

Due to the changing blade design in the course of the optimization, the blade surface and fluid mesh need to be represented in a conforming way with as little of mesh quality degradation as possible. In the case of gradient-based optimization, the deformation approach is selected. In the case of global optimization, the mesh is regenerated at each iteration, as is pictured in Figure 3.3. Deforming the geometry ensures a consistent mesh topology and hence fosters a static underlying optimization problem. Figure 3.7 shows the fluid-dynamic deformation chain during optimization at iteration k.

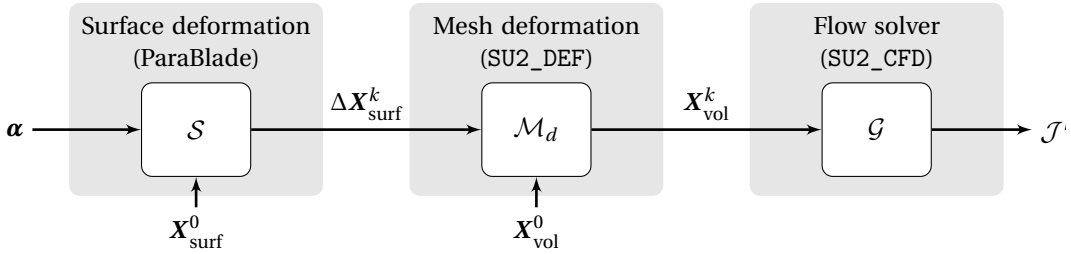


Figure 3.7: Simplified fluid-dynamic shape evaluation chain of the direct implementation, when utilizing mesh deformation. Adapted from Vitale et al. [2017].

SURFACE

The blade surface deformation matrix $\Delta \boldsymbol{X}_{\text{surf}}$ is obtained using the expression in Equation 3.4. It is the element-wise difference between the blade surface coordinate matrices at the current and previous design iteration. $\mathcal{S}_{\text{CAD}}$ represents the CAD parametrization module, outputting the blade surface coordinates from $\boldsymbol{\alpha}$.

$$\mathcal{S}(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^0) = \mathcal{S}_{\text{CAD}}(\boldsymbol{\alpha}^k) - \mathcal{S}_{\text{CAD}}(\boldsymbol{\alpha}^0) = \Delta \boldsymbol{X}^k_{\text{surf}} \tag{3.4}$$

MESH

To generate the mesh at each evaluation in the case of gradient-based optimizers, a baseline mesh is deformed. The deformation matrix of the mesh nodes $\Delta \boldsymbol{X}_{\text{vol}}$ is calculated using a spring-analogy method of FFD, based on linear elasticity equations [Dwight 2009; Economon et al. 2016]. Which applies weighting based on the inverse distance with respect to a deformed surface. In this approach, a system of linear equations is solved as shown in Equation 3.6. Here, $\Delta \boldsymbol{X}_{\text{surf}}$ is imposed, $\boldsymbol{K}$ is the mesh node stiffness matrix and $\boldsymbol{V}$ is a transformation matrix. The interested reader may look to Kenway et al. [2010] or Secco et al. [2021] for a more complete treatment of the FFD method. SU2's `SU2_DEF` executable is called to perform the mesh deformation during optimization.

$$\boldsymbol{X}^k_{\text{vol}} = \boldsymbol{X}^0_{\text{vol}} + \Delta \boldsymbol{X}^k_{\text{vol}} \tag{3.5}$$

$$\boldsymbol{K} \Delta \boldsymbol{X}^k_{\text{vol}} = \boldsymbol{V} \Delta \boldsymbol{X}^k_{\text{surf}} \tag{3.6}$$

The mesh element stiffness is modeled using the wall distance method. Mesh deformation is feasible for gradient-based optimization because the step sizes in subsequent designs during optimization are typically small enough to not yield inconsistently deformed meshes. Added, designs may not differ greatly from baseline, which strengthens the need for bounds on the design vector.

Integrated meshing is applied for the generation of meshes for gradient-free optimizers. This method is selected in order to ensure mesh quality during optimization. The caveat with integrated meshing is ensuring consistent mesh topology during optimization as described in section 2.2.3. However, this would not be an issue for an infinitely fine mesh, emphasizing the need for mesh discretization verification as conducted in subsection 5.4.1.

3.1.3. COMPUTING SENSITIVITY

In case of gradient-based optimizations, the optimizers implemented apply quasi-newton methods to construct an estimation to the Hessian of $\mathcal{J}$. This approach requires gradient information on the design variables to determine a suitable direction of the next evaluation of the design problem. See appendix A.2.1 for a detail on the quasi-Newton approach. The calculation of the total sensitivity of the objective function $\mathcal{J}$ with respect to the design vector $\boldsymbol{\alpha}$ is shown as a chained multiplication of its major components in Figure 3.8. These components represent the modules in the fluid-dynamic design chain as described previously in this chapter.

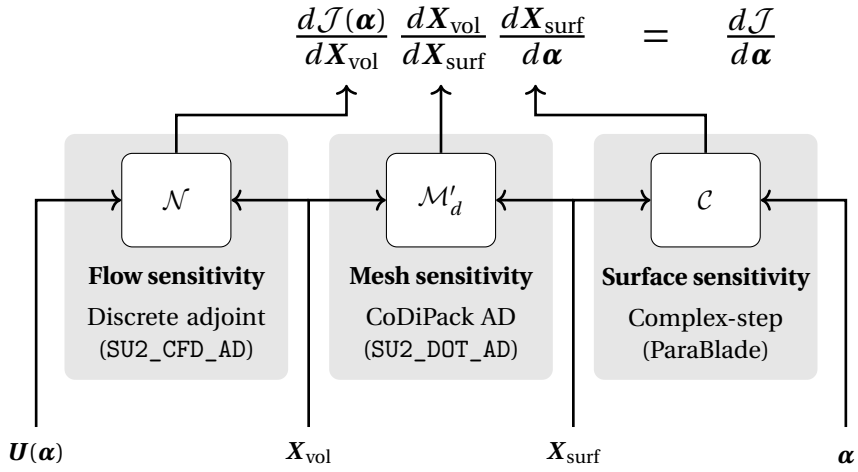


Figure 3.8: Simplified schematic for calculating the total design sensitivity, with emphasis on the components of the total objective function sensitivity $\frac{d\mathcal{J}}{d\boldsymbol{\alpha}}$. The discrete adjoint method of the executable `SU2_CFD_AD` generates the flow sensitivities, `SU2_DOT_AD` computes the mesh sensitivities using algorithmic differentiation (AD) and the complex-step method of ParaBlade computes the surface sensitivities.

The three components which are involved the total design sensitivity are the flow sensitivity $\frac{d\mathcal{J}}{d\boldsymbol{X}_{\text{vol}}}$, the mesh sensitivity $\frac{d\boldsymbol{X}_{\text{vol}}}{d\boldsymbol{X}_{\text{surf}}}$ and the surface sensitivity $\frac{d\boldsymbol{X}_{\text{surf}}}{d\boldsymbol{\alpha}}$. The following sections detail these components. Figure 3.8 shows the major calculation flow and applied methods and programs for the components.

FLOW SENSITIVITY

The flow sensitivity is obtained using the adjoint method provided by the `SU2_CFD_AD` executable of SU2, as detailed by Albring et al. [2016b]. This method is selected due to the low number of flow quantities of interest, $\mathcal{J}$ and β_{out}, compared to the number of design variables, as described in section 5.2.1.

The adjoint method allows to efficiently calculate the sensitivity of the objective function with respect to the volume coordinates $\boldsymbol{X}_{\text{vol}}$ without evaluating the expensive flow variable sensitivities $\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}}$. The intent of this section is to describe the concept of the adjoint-based gradient calculation. A detailed discussion of the method is beyond the scope because the focus of this work is towards optimization using a gradient-free approach.

The flow sensitivity is the total gradient of $\mathcal{J}$ with respect to the mesh points $\boldsymbol{X}_{\text{vol}}$, as expressed in Equation 3.7. Because $\mathcal{J}$ is a function of $\boldsymbol{X}_{\text{vol}}$ directly, but also indirectly through $\boldsymbol{U}(\boldsymbol{X}_{\text{vol}})$, see Equation 2.1. The

total gradient must then include the effect on $\mathcal{J}$ due to a change in $\boldsymbol{X}_{\text{vol}}$ on $\boldsymbol{U}(\boldsymbol{X}_{\text{vol}})$, which is represented by the second term in the right-hand side of Equation 3.7. To obtain the total gradient $\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}}$, the partial sensitivity of $\boldsymbol{U}$ must be calculated with respect to all geometric design variables $\boldsymbol{\alpha}$, because $\boldsymbol{X}_{\text{vol}}$ is function of $\boldsymbol{\alpha}$. This is very time-consuming as it requires a separate flow solution for each variable, meaning $n_{\boldsymbol{\alpha}}$ flow computations.

$$\frac{d\mathcal{J}}{d\boldsymbol{X}_{\text{vol}}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{X}_{\text{vol}}} + \frac{\partial\mathcal{J}}{\partial\boldsymbol{U}}\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}} \tag{3.7}$$

However, the adjoint method obtains $\frac{d\mathcal{J}}{d\boldsymbol{X}_{\text{vol}}}$ by computing a single adjoint flow solution. Incurring only a marginal time penalty with respect to a direct flow solution. This is achieved through recognizing that $\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}}$ also occurs in the total gradient of the residual vector $\boldsymbol{R}$ of the steady-state flow governing equations. Of which the value at convergence is zero, as in the case of iterative solving methods as considered in this work.

$$\boldsymbol{R}\big(\boldsymbol{X}_{\text{vol}}, \boldsymbol{U}(\boldsymbol{X}_{\text{vol}})\big) \tag{3.8}$$

$$\frac{d\boldsymbol{R}}{d\boldsymbol{X}_{\text{vol}}} = \frac{\partial\boldsymbol{R}}{\partial\boldsymbol{X}_{\text{vol}}} + \frac{\partial\boldsymbol{R}}{\partial\boldsymbol{U}}\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}} = 0 \tag{3.9}$$

Rewriting Equation 3.9 for $\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}}$ and substituting in this right-hand side into Equation 3.7 gives the adjoint vector $\boldsymbol{\psi}_k$ in Equation 3.11. Taking terms which are a function of the flow gives the equation in Equation 3.11 which forms the adjoint flow system of equations. Solving this provides $\boldsymbol{\psi}_k$.

$$\frac{d\boldsymbol{U}}{d\boldsymbol{X}_{\text{vol}}} = -\left[\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{U}}\right]^{-1}\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{X}_{\text{vol}}} \tag{3.10}$$

$$-\frac{\partial\mathcal{J}}{\partial\boldsymbol{U}}\left[\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{U}}\right]^{-1} = \boldsymbol{\psi}_k \quad \rightarrow \quad \frac{\partial\boldsymbol{R}}{\partial\boldsymbol{U}}\boldsymbol{\psi}_k = -\frac{\partial\mathcal{J}}{\partial\boldsymbol{U}} \tag{3.11}$$

Then, the total gradient $\frac{d\mathcal{J}}{d\boldsymbol{X}_{\text{vol}}}$ in terms of partial gradients and the adjoint vector $\boldsymbol{\psi}_k$ is computed using the simplified expression in Equation 3.12. This expression only contains partial derivative terms which are readily available from the solved adjoint flow system of equations.

$$\frac{d\mathcal{J}}{d\boldsymbol{X}_{\text{vol}}} = \frac{\partial\mathcal{J}}{\partial\boldsymbol{X}_{\text{vol}}} + \boldsymbol{\psi}_k\frac{\partial\boldsymbol{R}}{\partial\boldsymbol{X}_{\text{vol}}} \tag{3.12}$$

The discrete adjoint approach is selected in this work, in light of the considerations between the continuous and discrete adjoint methods as described in section 2.2.2. Black-box algorithmic differentiation (AD) is applied to linearize the discretized flow equations [Albring et al. 2016b].

Other applications of the flow sensitivity are to drive mesh refinements in an error estimation study by identifying regions to which the objective function value is more sensitive [Page et al. 2015].

A limitation in the adopted adjoint method is an assumption of constant turbulent eddy viscosity [Albring et al. 2016a], simplifying the derivation of the adjoint equations in order to lower the computational time and memory [Marta and Shankaran 2013]. Also termed the Constant Eddy Viscosity (CEV) assumption. This simplification poses a source of inconsistency, because it ignores the turbulent eddy viscosity derivative. Which is a significant term in regions of large adverse pressure gradients such as shockwaves. Kim et al. [2003] show that the CEV approximation can cause inaccurate gradients in steady-state simulations when strong shockwaves are present. However, for the studied engineering design problem, the CEV approximation caused a minor difference in optimized design. Though, when optimizing multi-modal problems, the final design may differ greatly due to inaccurate path traversal resulting from inaccurate gradients. Additionally, the sensitivity discrepancy issue is shown to be large for unsteady flow simulations aswell [Rubino et al. 2020].

MESH SENSITIVITY

To obtain the sensitivity of the volume nodes with respect to the blade surface nodes, here termed the mesh sensitivity $\frac{d\boldsymbol{X}_{\text{vol}}}{d\boldsymbol{X}_{\text{surf}}}$, the SU2's routine `SU2_DOT_AD` is used. The function $\mathcal{M}'()$ represents the mapping performed by `SU2_DOT_AD`. $\mathcal{M}'()$ calculates the mesh sensitivity using AD [Albring et al. 2015]. The value of the differential is obtained by back-propagating through the linear operations of $\mathcal{M}$ performed by `SU2_DEF` by applying the chain rule.

SURFACE SENSITIVITY

ParaBlade uses the complex-step method [Lyness and Moler 1967] to obtain the surface sensitivities $\frac{d\boldsymbol{X}_{\text{surf}}}{d\boldsymbol{\alpha}}$. The complex-step method performs a Taylor series expansion in the imaginary axis and rearranges the imaginary part. In context of this work, this expression for the partial sensitivity with respect to design variable j takes the form of 3.14. h_j represents the perturbation of $\boldsymbol{\alpha}$ by solely perturbing design variable j. Using the complex-step method, the derivatives can be calculated with arbitrarily small step sizes without running into round-off errors stemming from floating point arithmetic.

$$\mathcal{S}_{\text{CAD}}(\boldsymbol{\alpha} + ih_j) = \mathcal{S}_{\text{CAD}}(\boldsymbol{\alpha}) + ih_j \frac{\partial \mathcal{S}_{\text{CAD}}}{\partial \alpha_j} - \frac{h_j^2}{2!} \frac{\partial^2 \mathcal{S}_{\text{CAD}}}{\partial \alpha_j^2} + \mathcal{O}(ih_j^3) \tag{3.13}$$

Rearranging the imaginary part of the expression yields the surface sensitivity as in Equation 3.14.

$$\frac{\partial \mathcal{S}_{\text{CAD}}}{\partial \alpha_j} = \frac{\text{Im}\left[\mathcal{S}_{\text{CAD}}(\boldsymbol{\alpha} + ih_j)\right]}{h_j} + \mathcal{O}(ih_j^2) \tag{3.14}$$

3.2. SCALING OPTIMIZATION PARAMETERS AND RESPONSES

Scaling of the optimization parameters, $\boldsymbol{\alpha}$, objective function and constraint responses is performed during optimization. In the case of the parameters and constraints, scaling is required to ensure that the variables are in similar order of magnitude and thus equally sensitive to absolute perturbations as executed by the optimizer. In exact computations for parameter scaling-invariant optimization algorithms, scaling does not affect optimization problem and thus the optimal solution [Moré 1983]. However, in practice, for the limited-precision calculations as performed by computers, scaling can mitigate numerical errors for small-scale functional values. Other numerical quantities of interest such as the sensitivities of $\mathcal{J}$ and $\boldsymbol{c}$ are scaled as a consequence of the scaling on the responses.

Since the optimizer operates in the scaled parameter space, a change in coordinates of the unscaled optimization parameters must be performed, or *scaling*. The general case of changing coordinates is described here, to provide a background on the algebra applied for performing the various scaling operations in this section.

To calculate the gradient of a general objective function f required by the optimizer, $\frac{df}{d\mathbb{\alpha}}$, using the non-scaled $\boldsymbol{\alpha}$ gradient $\frac{df}{d\boldsymbol{\alpha}}$, which is received from the adjoint method, the chain rule as in Equation 3.16 is used. Here, $g(\boldsymbol{\alpha})$ is a function performing the change of variable $\boldsymbol{\alpha} \mapsto \mathbb{\alpha}$.

$$\mathbb{\alpha} = g(\boldsymbol{\alpha}) = \frac{1}{\boldsymbol{\alpha}^0}\boldsymbol{\alpha} \tag{3.15}$$

$$\frac{df}{d\boldsymbol{\alpha}} = \frac{df}{d\mathbb{\alpha}}\frac{d\mathbb{\alpha}}{d\boldsymbol{\alpha}} = \frac{df}{d\mathbb{\alpha}}\frac{dg(\boldsymbol{\alpha})}{d\boldsymbol{\alpha}} = \frac{df}{d\mathbb{\alpha}}\frac{1}{\boldsymbol{\alpha}^0} \tag{3.16}$$

3.2.1. SINGLE POINT

During optimization of both single and multi-point problems, the design vector $\boldsymbol{\alpha}$ and objective function responses $\mathcal{J}(\boldsymbol{\alpha})$ and sensitivities $\frac{d\mathcal{J}(\boldsymbol{\alpha})}{d\boldsymbol{\alpha}}$ are scaled. This section describes the scaling steps which are applied in the case of single-point optimizations.

PARAMETERS

The optimization parameters, or design vector, at evaluation number k, $\mathbb{\alpha}^k$, is scaled with respect to the baseline design vector, as in Equation 3.17. Thus, the initial design vector is a vector of ones. $\boldsymbol{\alpha}^0$ is a constant during optimization.

$$\mathbb{\alpha}^k = f(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^0) = \frac{\boldsymbol{\alpha}^k}{\boldsymbol{\alpha}^0} \qquad \rightarrow \qquad \mathbb{\alpha}^0 = \mathbb{1} \tag{3.17}$$

OBJECTIVE FUNCTION

The normalized responses passed to the optimizer, $\mathbb{J}(\boldsymbol{\alpha}^k)$, are scaled with respect to the initial response value, as in Equation 3.18. Consequently, the initial scaled response value $\mathbb{J}^0 = 1$. It is worth noticing that despite

the different notation, the values of $\mathcal{J}(\mathbb{\alpha}^k)$ and $\mathbb{J}(\mathbb{\alpha}^k)$ are the same as $\mathcal{J}(\boldsymbol{\alpha}^k)$ and $\mathbb{J}(\boldsymbol{\alpha}^k)$ respectively, because $\mathbb{\alpha}$ is re-scaled to $\boldsymbol{\alpha}$ before shape parametrization.

$$\mathbb{J}(\boldsymbol{\alpha}^k) = f\big(\mathcal{J}(\boldsymbol{\alpha}^k),\ \mathcal{J}(\boldsymbol{\alpha}^0)\big) = \frac{\mathcal{J}(\boldsymbol{\alpha}^k)}{\mathcal{J}(\boldsymbol{\alpha}^0)} \qquad \rightarrow \qquad \mathbb{J}^0 = 1 \tag{3.18}$$

When scaling the function response, the function sensitivities must be scaled accordingly. The calculation obtaining the sensitivities $\frac{d\mathbb{J}(\mathbb{\alpha}^k)}{d\mathbb{\alpha}^k}$ from the corresponding un-scaled sensitivities $\frac{d\mathcal{J}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k}$ as received from the gradient assembly is shown in Equation 3.19. Here, the scaling in Equation 3.17 and Equation 3.18 are applied to the denominator and numerator respectively.

$$\frac{d\mathbb{J}(\mathbb{\alpha}^k)}{d\mathbb{\alpha}^k} = \frac{d\mathcal{J}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\frac{1}{\mathcal{J}(\boldsymbol{\alpha}^0)}}{\frac{1}{\boldsymbol{\alpha}^{0,\mathrm{T}}}} \qquad = \frac{d\mathcal{J}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{\mathcal{J}(\boldsymbol{\alpha}^0)} \tag{3.19}$$

CONSTRAINTS

The constraint response and sensitivities are also scaled during optimization. The response is scaled for numerical error considerations and the sensitivities as a consequence for consistency. In this work, all constraints equally weighted, because they are either feasible or infeasible. i.e. same order as seen by the optimizer. Equation 3.20 shows how the constraint responses are normalized, analogous to how the objective function responses are normalized. $c(\boldsymbol{\alpha}^0)$ is considered a constant during optimization.

$$\mathbb{c}(\boldsymbol{\alpha}^k) = f\big(c(\boldsymbol{\alpha}^k), c(\boldsymbol{\alpha}^0)\big) = \frac{c(\boldsymbol{\alpha}^k)}{c(\boldsymbol{\alpha}^0)} \qquad \rightarrow \qquad \mathbb{c}^0 = 1 \tag{3.20}$$

When scaling the objective function sensitivities, the constraint sensitivities $\frac{dc(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k}$ should also be scaled in order to not emphasize either objective or constraint in the dual optimization problem. See the augmented Lagrangian method for reference. Obtaining the scaled constraint sensitivities from the un-scaled counterpart is shown in Equation 3.21.

$$\frac{d\mathbb{c}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} = \frac{dc(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\frac{1}{c(\boldsymbol{\alpha}^0)}}{\frac{1}{\boldsymbol{\alpha}^{0,\mathrm{T}}}} \qquad = \frac{d\mathcal{J}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{c(\boldsymbol{\alpha}^0)} \tag{3.21}$$

3.2.2. MULTI-POINT

Since the gradient-based optimizers in the scope of this work require the objective function to be a scalar, weighting is needed in multi-point optimization to reduce the operating point response vector into a single value.

OBJECTIVE FUNCTION

The weighted normalized objective function $\mathbb{J}_w(\boldsymbol{\alpha}^k)$ at iteration k for operating points $1 < i \leq n$ is obtained similarly to the single point case, by extending the calculation into a vector weighted sum expression.

$$\begin{aligned}
\mathbb{J}_w(\boldsymbol{\alpha}^k) &= w_1 \frac{\mathcal{J}_1(\boldsymbol{\alpha}^k)}{\mathcal{J}_1(\boldsymbol{\alpha}^0)} + w_2 \frac{\mathcal{J}_2(\boldsymbol{\alpha}^k)}{\mathcal{J}_2(\boldsymbol{\alpha}^0)} + \cdots + w_n \frac{\mathcal{J}_n(\boldsymbol{\alpha}^k)}{\mathcal{J}_n(\boldsymbol{\alpha}^0)} \\
&= w_1 \mathbb{J}_1(\boldsymbol{\alpha}^k) + w_2 \mathbb{J}_2(\boldsymbol{\alpha}^k) + \cdots + w_n \mathbb{J}_n(\boldsymbol{\alpha}^k) \\
&= \boldsymbol{w}\, \boldsymbol{\mathbb{J}}^{\mathrm{T}}(\boldsymbol{\alpha}^k)
\end{aligned} \tag{3.22}$$

Accordingly, the expression in Equation 3.23 displays the gradient vector for response- and parameter-scaled $\mathbb{J}_w(\mathbb{\alpha})$, as expressed by its unscaled counterpart.

$$\begin{aligned}
\frac{d\mathbb{J}_w(\mathbb{\alpha}^k)}{d\mathbb{\alpha}^k} &= w_1 \left[\frac{d\mathcal{J}_1(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{\mathcal{J}_1(\boldsymbol{\alpha}^0)} \right]^{\mathrm{T}} + w_2 \left[\frac{d\mathcal{J}_2(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{\mathcal{J}_2(\boldsymbol{\alpha}^0)} \right]^{\mathrm{T}} + \quad \ldots \quad + w_n \left[\frac{d\mathcal{J}_n(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{\mathcal{J}_n(\boldsymbol{\alpha}^0)} \right]^{\mathrm{T}} \\
&= \boldsymbol{w} \left[\frac{d\boldsymbol{\mathcal{J}}(\boldsymbol{\alpha}^k)}{d\boldsymbol{\alpha}^k} \frac{\boldsymbol{\alpha}^{0,\mathrm{T}}}{\boldsymbol{\mathcal{J}}^{\mathrm{T}}(\boldsymbol{\alpha}^0)} \right]^{\mathrm{T}}
\end{aligned} \tag{3.23}$$

CONSTRAINTS

Multi-point constraint response and sensitivity reduction is performed analogously to how the multi-point objective functional response and sensitivity, respectively, are reduced.

3.3. MULTI-POINT OPTIMIZATION

The code framework allows for various gradient-based and gradient-free optimizations. This section describes implementation details of the various optimizers of the framework.

3.3.1. GRADIENT-BASED

When a scalar optimizer is used, in contrast to a Pareto front-based optimizer, the array of objective function values of the operating conditions needs to be reduced into one value. The weighted sum method is selected for this scalarization. This method is selected due to the absence of a decision maker in the design process, for scope-limiting reasons. The weighted-sum method takes priority over the no-preference and other multi-point weighting methods as discussed in section 2.3.2 due to its simplicity. Nonetheless, the other methods present viable candidates for future turbomachinery optimization studies. These weights are assigned in the optimization configuration file, as described in the next chapter.

In order to not exit the optimization prematurely, a failure continuation method is indispensable to mitigate the unusable responses of non-converging flow simulations. If a direct flow simulation does not converge, a penalized response is assigned of $1.0 \times 10^2 \times \mathcal{J}^0$. As advised by Nocedal and Wright [2006], the sensitivities of the latest converging adjoint simulation are used. If an adjoint flow simulation does not converge, the previous sensitivities are passed to the optimizer aswell.

3.3.2. GRADIENT-FREE

Failed simulations are penalized with an equivalently large objective function value. This can pose problems in the case of surrogate-assisted optimization, where all evaluated points are considered in the model output. There is no consensus on this type of failure handling [Stork et al. 2020].

4

CODE FRAMEWORK

This chapter elaborates on the structure of the code written in context of this work, of which the objective is to enable multi-point optimization for turbomachinery fluid-dynamic shape design. The code design principles are detailed in section 4.1, the features implemented in order to achieve these principles are described in section 4.2. This chapter continues with a description of the code input and structure.

4.1. DESIGN PRINCIPLES

A few design principles are set out in the effort of enabling multi-point optimization for turbomachinery fluid-dynamic shape design; speed, robustness, extensibility, user-friendliness. This section describes the features implemented in context of the design principles.

- *Speed*: In order to realize the speed goal, parallel computing options are made available to single- and multi-point optimizations to reduce the diminishing computational efficiency on increasing CPU core count of flow evaluation using the SU2 solvers. Porting the code from the Python language to for instance C would not significantly reduce the computation time, because the fraction of time spent in the Python code modules during optimization is negligible.

- *Robustness*: Pursued by allowing a range of optimization algorithms and problem types such as harmonic balance, multi-zone and no limit on the number of operating points. Another element is robustness against unresponsive flow simulations during initialization or termination, stemming from the Message-Passing Interfacing (MPI) protocol utilized by SU2. A routine is implemented which waits a conservatively determined time, after which the flow simulation is restarted. This feature increases the likelihood of successful completion of the optimization. Also, failures are caught during mesh generation and flow solving. Meshing failures can happen due to inconsistent geometries as a result of, for instance, too lenient bounds on the design vector. An example is when the boundary layer crosses a periodic boundary. For the approach to handling of failed flow solutions, see section 3.3.1.

- *Extensibility*: The code is written in accordance with object-oriented programming principles. This allows for a structured approach of adding optimizers from different libraries using class-inheritance. Also, modules may be deleted which may be found of minimal value. Additionally, a regression testing suite is available which allows to automatically check the correct functioning during code extensions. It also provides a standardized approach for extension by future code regression tests, using the Python builtin unit testing framework.

- *User-friendliness*: User friendliness is an important goal because the framework enables various optimization approaches to be compared, aiding the user in determining the approach best suited to their specific problem type. To this end, the program updates a history file during optimization. Containing the response values and design variables, following each flow solution for both single and (weighted) multi-point optimization. Error messages are generated, displaying inconsistent options or diagnostics and tips for resolution. Recipe scripts are available which automatically install the ParaBlade development environment and dependencies such as SU2, Gmsh and the various Python optimization

libraries. Debugging options of varying levels are available to generate diagnostics and plots of flow simulation convergence, blade geometry and mesh at each evaluation.

4.2. CAPABILITIES

This section elaborates on the major capabilities of the code framework. A list of features is displayed here and major elements are detailed in the following subsections.

- Multi-point designs.
- Multi-zone problems.
- Gradient-based and -free optimizers.
- Flow simulation (`SU2`).
- Integrated meshing (`Gmsh`).
- Harmonic-balance problems.
- Scaling of $\mathcal{J}$, $\boldsymbol{\alpha}$ and $\boldsymbol{c}$.
- Testing suite for continuous code integration.
- Restarting optimizations.
- Automatic restart of stale simulations.
- Recipe installation scripts.

4.2.1. OPTIMIZATION PROBLEMS

The framework enables optimization of many-point and many-zone problems due to the agnostic implementation of data structures containing the response and sensitivity of objective function and constraints. The weighted sum approach allows for an arbitrary number of points to be considered during optimization. Hard-coding is avoided entirely, allowing an arbitrary number of operating conditions and simulation zones.

The Harmonic Balance (HB) simulation method is available in the SU2 suite and enabled for operation using this framework. HB is a reduced-order method for periodic unsteady flow simulation which reduces the computational cost in simulating time-accurate aero-elastic automated design problems in turbomachinery [Anand et al. 2020; Chahine et al. 2019; Engels-Putzka et al. 2019; Moffatt and He 2003]. The output files of the various simulation instances are read automatically.

4.2.2. OPTIMIZERS

This section briefly documents the optimizers available for selection in both single and multi-point optimization. A detailed treatment on the theory behind all algorithms is outside the scope of this work. However, some concepts behind gradient-based optimization are treated in appendix A, and appendix B elaborates on Efficient Global Optimization (EGO). The optimization algorithms implemented in this framework are shown in Table 4.1, in abbreviated notation and full.

The optimizers are selected according to two criteria. The first is frequency of occurrence in literature on multi-point turbomachinery optimization studies, which is addressed in section 2.4. The second is availability in optimization libraries. The local optimizers, or gradient-based, are SLSQP, SNOPT and IPOPT. The options for global optimizers are DE, NSGA2, ALPSO, MOPSO and EGO and its variations. All global optimizers are gradient-free optimizers, with exception for Gradient-Enhanced Kriging-EGO (EGO-GEK) which utilizes gradient information for constructing its surrogate model.

This framework does not implement optimization algorithms from scratch, but interfaces with optimization modules readily implemented by external libraries. This is due to scope restrictions and in order to ensure the optimizer code modules are kept up to-date by the broader community. The libraries in which the optimizers are implemented are shown in Table 4.1. Some algorithms are available from multiple libraries. Of these options, the default library is shown first.

The local optimizers are variations of Sequential Quadratic Programming (SQP) methods, see appendix A.1 for details on SQP. Sequential-Least-Squares Quadratic Programming (SLSQP) [Kraft 1988; Schittkowski 1982] is one the most frequently considered optimizers in engineering optimization. It is implemented by the Python scientific computing optimization library SciPy [Virtanen et al. 2020]. SLSQP transforms the direction finding sub-problem into a least-squares problem. See appendix A.3 for details. Sparse Nonlinear OPTimizer (SNOPT) [Gill et al. 2002] is a software package which enables efficient optimization of problems with thousands of design variables and constraints. SNOPT is interfaced through either

PyGMO [Biscani and Izzo 2020] or PyOptSparse [Perez et al. 2012]. Interior Point OPTimizer [Wächter and Biegler 2005] is another well-known local optimizer which tends to preserve feasibility of the design during optimization.

The general steps of the evolutionary algorithms Differential Evolution (DE) and NSGA2 are described in section 2.1.2 and section 2.3.4 respectively. Section 5.3 covers the implementation details of NSGA2 as used in this work.

Augmented Lagrangian Particle Swarm Optimization (ALPSO) [Jansen and Perez 2011; Sedlaczek and Eberhard 2006] utilizes the augmented Lagrangian approach to handle constraints. NSGA2 is detailed in section 2.3.4. Multi-Objective Particle-Swarm Optimization (MOPSO) [Coello Coello et al. 2004; Moore et al. 2000; Peeren and Vogeler 2017] is a PSO adapted to multi-objective optimization using the domination rules in section 2.3.4 applied to the fitness values of PSO. It is implemented by JMetalPy [Benítez-Hidalgo et al. 2019].

Optimized-MOPSO [Sierra and Coello 2005] utilizes of the crowding distance as employed in NSGA2 to select solutions and a combination of two mutation operators to accelerate the convergence towards an optimal region in design space. The OMOPSO algorithm utilizes ϵ-dominance [Laumanns et al. 2002] to select particles on the Pareto-set, provably augmenting convergence to a set of Pareto-optimal solutions and diversity among the solutions.

EGO is gaining attention in fluid-dynamic design optimization for its favourable exploration and exploitation properties. See appendix B for the concepts regarding EGO. the Surrogate Modeling Toolbox [Bouhlel et al. 2019] and DAKOTA [Adams et al. 2021] suites enable optimization using EGO. The Kriging surrogate model can be supplied with gradient information, which is performed in Gradient-Enhanced Kriging (GEK) [Bouhlel and Martins 2019]. The complexity of the Kriging approach can be reduced by pruning the Kriging spatial correlation matrix using K-means Partial-Least Squares (KPLSK). An EGO with Expected Volume-Gain (EVG) infill criterion, adapted from Backhaus et al. [2017], is developed in context of this framework within SMT. EVG is a statistical prediction of the volume gain of a candidate point with respect to the Pareto front within a multi-criteria optimization problem. Allowing for selection of candidate points based on estimated volume gain mean and spread, see appendix B.2.2. Additionally, a novel EGO-GEK optimizer is implemented and accepted for incorporation into the SMT suite.

Type	Abbreviation	Name	Optimization library
Local	SLSQP	Sequential-Least-Squares Quadratic Programming	SciPy, PyGMO, PyOptSparse
	SNOPT	Sparse Nonlinear OPTimizer	PyGMO, PyOptSparse
	IPOPT	Interior Point OPTimizer	PyGMO, PyOptSparse
Global	DE	Differential Evolution	PyGMO, SciPy
	NSGA2	Non-dominated Sorting-based Genetic Algorithm-II	PyOptSparse, JMetalPy
	ALPSO	Augmented Lagrangian Particle-Swarm Optimization	PyOptSparse
	MOPSO	Multi-Objective PSO	JMetalPy
	EGO	Efficient Global Optimization (-GE †) (-KPLS §)	SMT, DAKOTA
	EGO-EVG	EGO + Expected Volume-Gain infill criterion	SMT

† GE: Gradient-Enhanced

§ KPLS: K-means Partial Least-Squares

Table 4.1: Optimization algorithms and corresponding source libraries implemented for multi-point turbomachinery optimization in this framework.

4.2.3. PARALLEL COMPUTING

Parallel computing is implemented on two levels; the first is by specifying the number of cpu cores for performing the flow simulation. The second is by specifying the number of parallel flow evaluations. Parallel computing setup is available by defining how many points will be evaluated simultaneously with the 'N_BATCH' option.

When parallelizing over multiple simultaneous flow evaluations, the number of assigned cores are divided by the number of parallel jobs. Thus, the total number of utilized cpu cores does not increase. The ability to parallelize multiple flow evaluations reduces the mean flow evaluation time by reducing the number of cores per flow evaluation. This is because the flow evaluation time-efficiency reduces for larger core counts.

For single-point optimization, only the non-sequential optimizers are available for parallel computing. This is performed by distributing the available parallel slots over every individual in the population or swarm. For multi-point optimization, all optimizers are parallel-capable. For gradient-based optimizers and EGO, each operating point is allocated a parallel slot. For the other optimizers, the slots are shared between the individuals of the population. For each individual, the operating points are evaluated in sequence.

Total speedup when performing multiple flow simulations in parallel is dependent on the total number of cpu cores and number of parallel flow simulations. This is because the parallel efficiency drops for large core counts, being 70% for evaluating a direct flow evaluation at 256 cores [Albring et al. 2016b]. Thus by performing multiple flow evaluations in parallel and lowering the cpu cores per flow simulation, this drop in parallel efficiency can be mitigated.

4.2.4. INTEGRATED MESHING

Integrated meshing serves as a feature of the code framework which enables the optimization using global optimizers. This section describes the major design choices, features and limitations of the meshing approach in the framework.

Integrated meshing is the default meshing approach for optimization using global optimizers. For gradient-based optimization, the mesh deformation approach is used, see section 3.1.2 for the motivation behind this approach. Mesh generation occurs in two steps, as described in section 3.1.1. `Gmsh` constructs the mesh, then `SU2_PER` ensures a shared periodic element topology for the periodic boundaries.

The mesh generation and recombination algorithms and smoothing steps are not hard coded, being free to vary. After generation, the meshing logs are saved to disk. An option is available which enables printing a wireframe of the mesh to disk using the offscreen rendering feature of `Gmsh`. Additionally, a module is implemented which enables the automatic visualization of mesh quality diagnostics by utilizing the open-source scientific visualization tool `ParaView` [Ahrens et al. 2005].

In the case of multi-zone optimization, meshes are generated separately per-zone and subsequently joined. This is in accordance with the requirements of SU2 for the computational grid, which require a single mesh for the computational domain.

A limitation of the current meshing approach is that mesh generation of three-dimensional geometries is not supported. However, this may be implemented analogously to the current two-dimensional mesh generator code.

4.3. SPECIFYING INPUT

Three input files, as enumerated here and shown in the Unified Modeling Language (UML) diagram of Figure 4.1, are required to fully specify the shape optimization.

- *Optimization:* Containing parameters specifying the other configuration files and optimizer-related parameters such as objective function, constraint, bound and optimizer type.
- *Blade parametrization:* Containing the baseline blade design variables and mesh generation parameters such as outer domain dimensions and target cell sizes.
- *SU2:* Containing parameters which specify the flow simulation. The parameters specify the flow problem type, solver, boundary condition, input files and solution monitoring as required per SU2. This file is adapted for the various SU2 analyses during optimization, such as `SU2_DEF` and `SU2_CFD_AD`.

Another input file defining the coordinates for the geometry of the matched blade is also required. This is an indirect input file because this is an output of the blade matching, regarded as separate to the optimiza-

tion. These coordinates are utilized for deforming the mesh in gradient-based optimization, by providing a reference for the surface deformations.

Restarting a previous optimization is possible by enabling it in the optimization configuration file. Restarting for EGO-based optimizers in SMT is only possible when the existing number of evaluations is equal or more than the size of the requested DoE. This is because the DoE data points must be preloaded into the SMT as a whole.

4.4. CODE STRUCTURE

This section gives an outline of the structure of the code, divided into two parts. The first part describes the single-point optimization code structure, the second reports the multi-point code structure. For sake of clarity, the UML diagrams show only a high-level overview of the main classes, methods and attributes. They highlight the modularity of the framework, allowing for extension by optimization algorithms which are implemented by other optimization libraries.

Figure 4.1 shows the UML diagram of the single-point optimization code. The main class is the `ShapeOptimization` class. This class contains methods to perform flow-based objective function and sensitivity evaluations. It also handles the printing of the optimization progress to disk and reading pre-existing optimization results. The configuration files define the design optimization, parametrization and flow evaluation and are read during the class initialization. The blade parametrization can consist of multiple files, each file defines the blade shape of corresponding with a zone. `ShapeOptimization` is stereotyped in italics to emphasize it being a metaclass. It has one abstract method `optimize()` which must be implemented by the extending classes, which are related through realization.

The design vector is contained in the attribute `X`, which is an m-by-n array containing the design variables for m zones and n design variables per zone. `obj` Is a scalar value representing the objective function response. The sensitivity of the design is contained in a two-dimensional array, in which each row holds the sensitivity vector corresponding with a zone.

The `SciPyOptimizer`, `PyOptSparseOptimizer`, `PyGmoOptimizer`, `SmtOptimizer` and `DakotaOptimizer` classes implement the abstractly defined `optimize()` method in `ShapeOptimization` for their respective optimization libraries, hence the realization relationship. The angle brackets highlight their interfacing role with the external optimization libraries.

`Su2Runner` Is a class which contains methods to perform direct, adjoint flow evaluations and mesh deformation by interfacing with SU2. It also contains methods which ensure that the required configuration files are available to SU2 at each evaluation. `ShapeOptimization` Is related by a two-way composition, because it shares the evaluation counter attribute and folder structure with `Su2Runner`, among others.

`GradientValidator` Contains methods which perform gradient validation of the adjoint method. `ShapeOptimization` is related by a two-way composition, because `GradientValidator` needs access to `Su2Runner` flow simulation functions which are accessed through `ShapeOptimization`, and `GradientValidator` is stored as an object on `ShapeOptimization` in order to re-use the direct and adjoint results for the first evaluation of the optimization.

`PyGmoProblem` Contains methods which return the fitness and gradient of the problem. These methods are placed in a separate class primarily in order to adhere to `PyGMO`'s particular style of defining optimizations by separating the problem and optimization classes.

`DakotaApiMixin` Acts interfaces with the `DAKOTA` library, containing methods to create the `DAKOTA` optimization input file, call the `DAKOTA` optimizer and read the output file. It extends the `DakotaOptimizer` with its methods and attributes, for simple access from `DakotaOptimizer`.

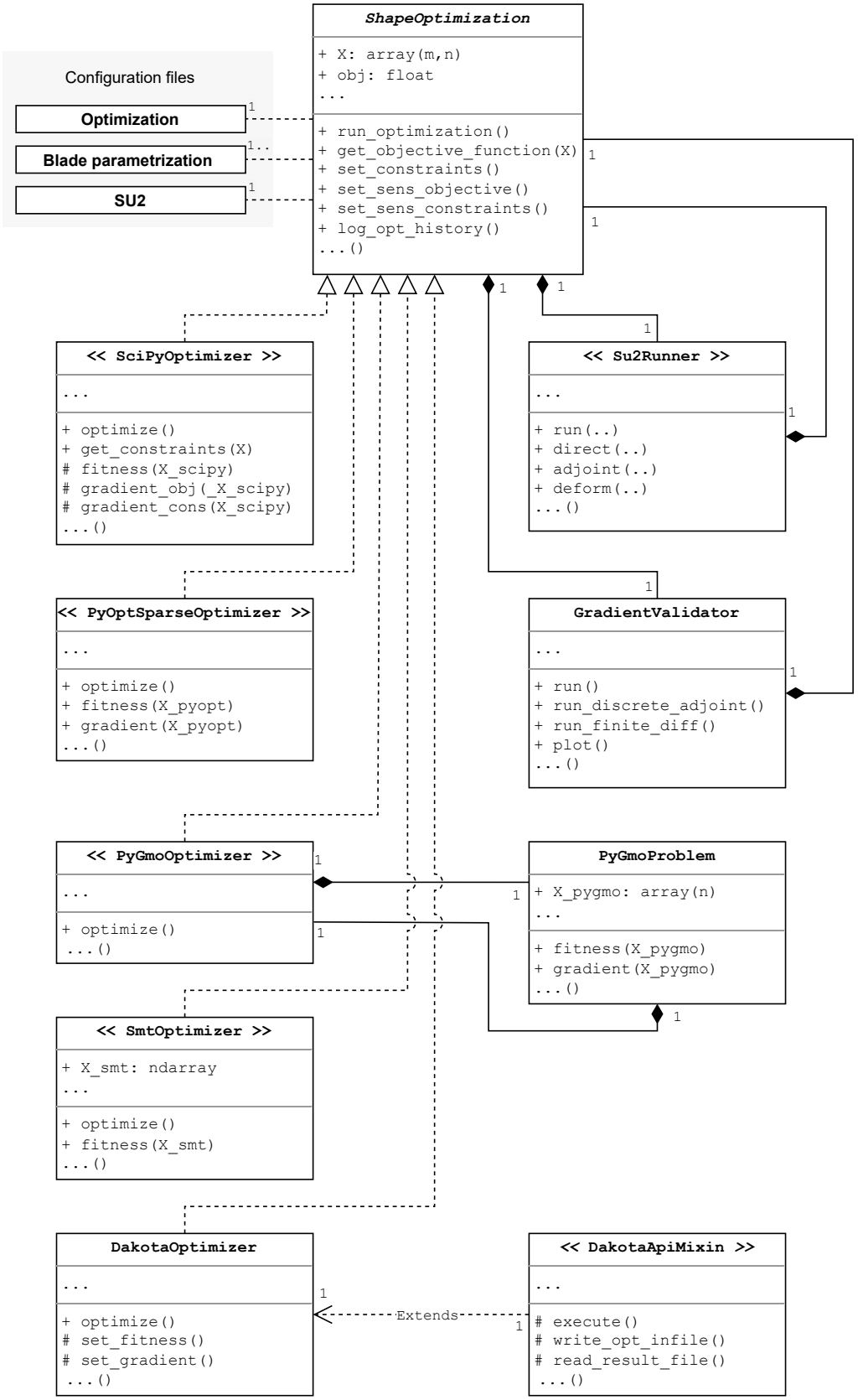


Figure 4.1: Class diagram depicting the high-level elements of the code concerned with single point optimization. Protected methods are declared by '#'. Chevrons (< >) are utilized to stereotype classes which act as interfaces between applications.

Figure 4.2 displays the UML diagram of the code performing multi-point optimization. The input specification is largely unchanged as compared to the single-point case, except for the SU2 configuration file multiplicity. Each operating condition is specified by a separate SU2 flow configuration file.

The state of each operating condition during optimization is contained in a separate `ShapeOptimization` class, located in an array of `ShapeOptimization` on `MultiPointOptimizer`. This relationship is chosen to maximize code re-use. The analysis methods of these classes are accessed by `MultiPointOptimizer`.

The `MultiPointOptimizer` class contains all the methods which handle the multi-point optimization. Each extending optimization class from Figure 4.1 has its own corresponding method. These methods utilize the single-point analysis code specific to the various extension optimization classes to aggregate the objective function responses and sensitivities of the various operating conditions. These methods are contained in `MultiPointOptimizer` as opposed to in their respective classes due to their overall individuality and brevity. However, this might be subject to change in case when functionality and hence code length is extended.

The `weighted_sum()` method reduces the multi-point responses into a scalar by applying a weighted sum according to the specified weights in the optimization configuration file. The sensitivities are reduced analogously. This method is required for optimizers which operate on a scalar response value. `normalize_gradient_vectors()` scales the gradient vectors of the various operating conditions according to the procedure described in section 3.2.2. `batched_eval()` Handles the batched optimization of the multi-point problem. When batched optimization is selected, each operating condition is run concurrently, with the flow simulations per operating condition in sequence. To this end, each operating condition is assigned a separate batch slot. `batched_eval()` Aggregates the results of the parallel processes and ensures that the mesh which is shared between the operating conditions, is generated once per optimization iteration.

`PyGmoMultiPointProblem` and `JMetalPyPointProblem` are self-contained classes for the same reason as why `PyGmoProblem` of the single-point case is separated from its extension class `PyGMO`.

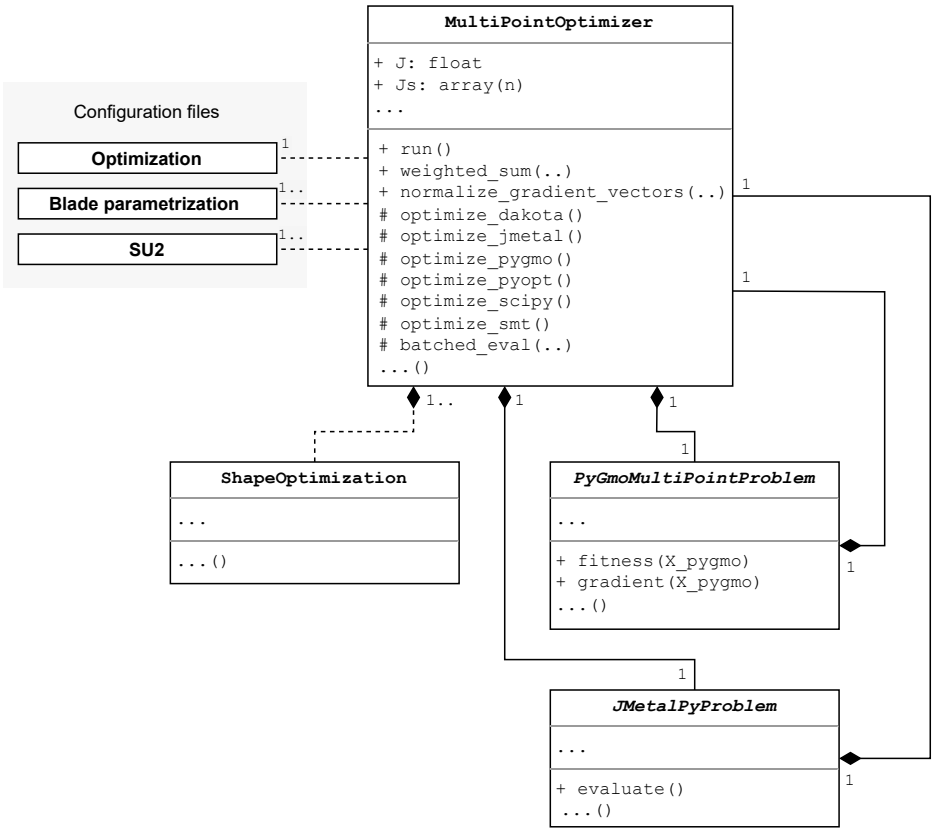


Figure 4.2: Class diagram of code involved with multi point optimization.

4.5. CODE VERIFICATION

A regression testing suite is developed for continuous code integration purposes through verifying extensions to the code. This module mitigates coding errors by ensuring that no functionality is broken upon addition of

new code. A standardized approach for code regression testing is implemented, by using the Python builtin unit testing framework, see appendix E.

5

PROBLEM FORMULATION

This chapter documents experimental setup of the multi-point optimization of this work, with a focus on the selected blade geometry study case. The turbomachine blade specification is documented in section 5.1. The simulation of the multi-point optimization is summarized in section 5.2. Section 5.4 describes the verification and validation of the simulation problem of the selected blade. This chapter is concluded with a section on the integrated mesh generation details of the optimization.

5.1. AACHEN AXIAL TURBINE STATOR

A two-dimensional axial turbine blade is selected as optimization test case. The selection of this configuration is motivated mainly because of its simplicity and low risk of simulation errors, reducing the computational time of the optimization. Simplicity, because the main focus of this work is on exhibiting the functionality of the multi-point optimization framework. Computational time plays a crucial role in optimization studies of global optimizers due to the large number of required evaluations. The absence of tip, hub and shroud mesh boundaries reduce the complication of the mesh topology of a two-dimensional case compared to three-dimensional. This reduces the risk of non-converging flow simulations due to the reduced risk of non-physical mesh deformations.

The selected blade is the first stator row of the Aachen turbine [Gallus et al. 1993; Walraevens et al. 1998]. It is a widely studied 1.5 stage transonic turbine [Rubino et al. 2020; Vitale et al. 2020] introduced by the RWTH Aachen University. See Figure 5.1 for an overview of the blade-to-blade geometry. The first blade row has a Traupel profile, followed by a Von Karman Institute profile and the third row has an identical Traupel profile. The subsonic operating condition of the turbine reduces the risk of flow solver instabilities due to the absence of strong shock-induced discontinuities or flow separation.

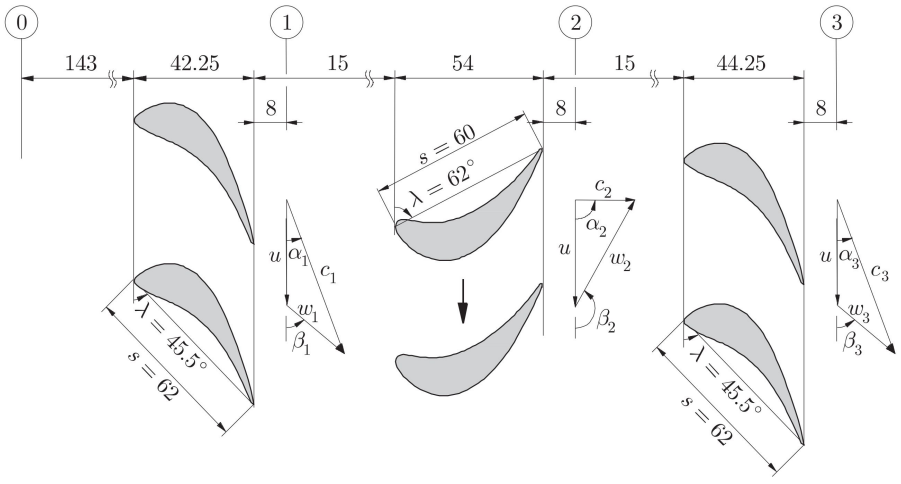


Figure 5.1: Cross-section of the 1.5 stage Aachen turbine [Walraevens et al. 1998].

5.2. SIMULATION

This section details how the two dimensional Aachen stator blade is parametrized, how the flow is simulated and which operating conditions are regarded during the multi-point optimization.

5.2.1. GEOMETRY

The Aachen turbine blade geometry is parametrized using the camberline-thickness approach as described in section 3.1.1. The geometric design variable values of the matched blade are summarized in Table 5.1.

Besides the design variables displayed in Table 5.1, other parameters are required to fully determine the geometry. These are kept constant during optimization. The blade pitch is kept constant because optimizing the cascade solidity is deemed outside of the scope of a detailed design stage such as in this work. The axial chord length is kept constant to inhibit the optimizer from reducing s_{gen} by eliminating the blade wake through extending the trailing edge downstream. t_5^{l} is kept constant due to the solver's inability to accurately model its contribution to s_{gen}. As experienced by the rate of spurious sensitivities during optimization, supported by its partial sensitivity in the adjoint gradient verification in Figure 5.6a.

Index	Name	Symbol	Value	Unit	Bounds $\{u, l\}$ [%]
1	Stagger angle	ξ	-45.785	[°]	20
2	LE metal angle	θ_{in}	1.954×10^{-3}	[°]	20
3	TE metal angle	θ_{out}	-72.988	[°]	20
4	LE radius	r_{in}	1.276×10^{-3}	[m]	20
5	TE radius	r_{out}	1.546×10^{-4}	[m]	**40**
6	LE distance	d_{in}	2.997×10^{-1}	[m]	20
7	TE distance	d_{out}	6.005×10^{-1}	[m]	20
8–12	Thickness lower 0–4	$t_{0\text{-}4}^{\text{l}}$	2.485×10^{-3}	[m]	20
			5.043×10^{-3}	[m]	20
			6.434×10^{-3}	[m]	20
			3.425×10^{-3}	[m]	20
			1.375×10^{-3}	[m]	20
13–18	Thickness upper 0–5	$t_{0\text{-}5}^{\text{u}}$	1.0×10^{-9}	[m]	20
			7.752×10^{-3}	[m]	20
			1.049×10^{-2}	[m]	20
			2.333×10^{-3}	[m]	20
			1.554×10^{-3}	[m]	20
			6.077×10^{-4}	[m]	20

Table 5.1: Design variables of the parameterized two-dimensional Aachen turbine blade surface.

The lower and upper bounds are set conservatively at a uniform 20%. The limiting design variable is the TE metal angle θ_{out}, which has a baseline value of -73°. Enlarging it by 23.3% would increase its value to 90°, which is perpendicular to the axial direction. Exceeding or approaching this angle would pose difficulties for flow solution convergence and hence jeopardize successful optimization.

The optimization bound on r_{out} is enlarged to 40%. This increase is determined in order to extend the design space. After observing frequent active bounds, in combination with the large sensitivity of s_{gen} with respect to r_{out}. Important to point out is the possibility of manufacturing issues at the TE due to the smaller r_{out} of the matched blade in combination with the more lenient optimization bounds.

The reference Aachen turbine blade geometry and the ParaBlade-matched geometry is displayed in Figure 5.2. LE and TE are shown in the detail plots on the right. A slight deviation in geometry of the matched blade from reference is noticeable around the LE and TE locations. However, the matching is deemed acceptable due to the small magnitude of the deviations in comparison to the blade surface coordinate order of magnitude.

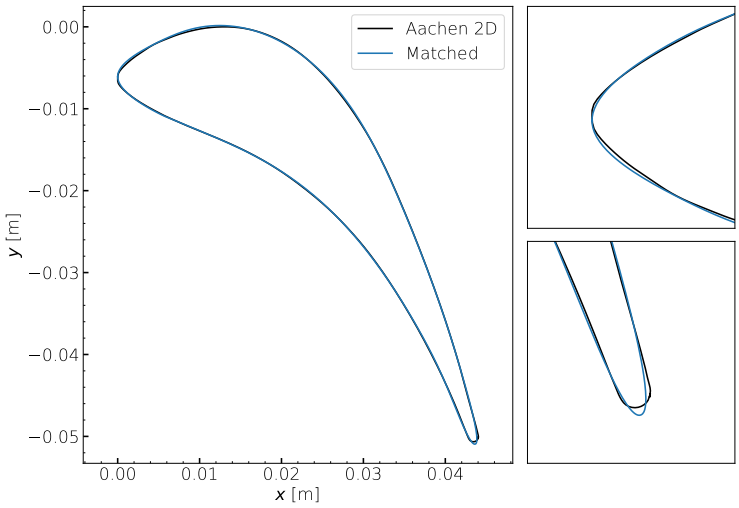


Figure 5.2: Matching of the two-dimensional Aachen blade surface using ParaBlade.

5.2.2. FLOW

The RANS equations of state are selected due to its favourable compromise between accuracy and computational cost and widespread availability. The computational domain is spatially discretized and solved using a cell-centered finite-volume approach. Convective terms are discretized using the second order method by Roe [1981], which enables simulation of transonic and supersonic flows. An Euler implicit time marching scheme is used, with a variable Courant-Friedrichs-Lewy (CFL) number between 5 and 10. The discretized equations are solved using a linear Newton-Krylov approach. The Krylov solver is preconditioned by the Lower-Upper Symmetric Gauss-Seidel (LUSGS). In order to aid flow convergence in presence of shocks, oscillations near shocks are suppressed by using a van-Albada type limiter [Venkatakrishnan 1993].

At the inlet boundary, the stagnation pressure P_0 and temperature T_0, axial flow, turbulence intensity, and viscosity ratio are specified. Whereas the static pressure P is prescribed at the outlet. See Table 5.3 for the values of the inlet and outlet boundary conditions. Along the blade surface, an adiabatic no-slip boundary condition is set. The far-field periodic boundaries are imposed with periodic boundary conditions.

Non-reflecting boundary conditions are applied at inlet, outlet and periodic boundaries to avoid non-physical wave reflections. Increasing accuracy and reducing computational efficiency because a coarser mesh may be used. At the inlet and outlet boundaries, the flow quantity of interest is calculated through weighted mixed-out averaging of density, momentum and enthalpy. The respective weights are 1.0, 1.0×10^{-5} and 15.

The one-equation turbulence closure model by Spalart and Allmaras [1992] (SA) is chosen because of its reliability in yielding physical results. Minimizing the risk of flow convergence errors during optimization. The maximum y^+ value along the blade surface is ensured to be below 20. This value is selected following a compromise between ensuring sufficient residual reduction of the conserved quantities and computational time. y^+ is controlled by the cell height at the blade surface. The determination of the target mesh element size as depicted in Table 5.2, is a result of the mesh convergence study in subsection 5.4.1.

The flow simulation is terminated when either a reduction of the order of magnitude of the residual of the conserved flow quantities of 7 is met or 2k flow solver iterations are performed. The iteration threshold is set to limit the time expense of the flow simulations.

Flow	State equations	Compressible RANS
	Spatial scheme	Upwind second-order Roe
	Time marching	Euler implicit
	Temporal gradient smoothing	Modified van-Albada
	Turbulence model	Spalart-Allmaras
	CFL number	5–10 [-]
	Linear solver	Krylov + FMGRES
	Preconditioner	LUSGS
Mesh	Topology	Mixed
	Inlet and outlet meridional length	2.0×10^{-2} m
	y^+ (Blade surface)	< 20 [-]
	Target element size	3.0×10^{-4} m
	Inflation ratio	1.15 [-]
	Inflation thickness	2.343×10^{-3} m
	Blade surface cell height	5.0×10^{-5} m

Table 5.2: Specification of the flow simulation and mesh parameters.

5.2.3. OPERATING POINTS

The nominal operating condition boundary condition values at the inlet and outlet are derived from the conditions at mid-span of the Aachen stator blade [Walraevens et al. 1998]. The off-design condition is characterized by 105.5% of total-to-static expansion ratio $\beta = \frac{P_{0,\text{in}}}{P_{\text{out}}}$. The backpressure is varied because turbine efficiency is sensitive to this quantity [Pini 2013]. The off-design backpressure perturbation magnitude is similar to the operational variability in backpressure of the ORC as reported by Casella et al. [2013]. The minor increase also reduces the risk of flow solver instabilities resulting from the presence of strong shockwaves.

		Nominal	Off-design	
Inlet	P_0	0.14	0.14	[MPa]
	T_0	300	300	[K]
	I_{tur}	0.03	0.03	[-]
	$\frac{\mu_{\text{tur}}}{\mu_{\text{lam}}}$	100	100	[-]
Domain	β	1.4036	1.4775 (**+5.5%**)	[-]

Table 5.3: Specification of the multi-point inlet and outlet flow boundary conditions for the Aachen turbine.

5.3. OPTIMIZATION

The NSGA2 optimizer is chosen due to its performance on multi-objective optimization problems in comparison to other global optimizers, and its ability in successfully optimizing the specific turbomachine problem. The general problem setup for the optimization in this work is summarized in Table 5.4.

The flow angle at the outlet boundary β_{out} is constrained to be larger than 70%, in order to ensure sufficient flow turning of the stator. No constraint on $\dot{m}$ is imposed because $\dot{m}$ is not found to vary more than one percent during optimization of this problem. See section 2.2.2 for an elaboration on turbine constraints.

Optimization	Algorithm	NSGA2
	Convergence criterion	100 generations or 100 hours
Objective function		s_{gen} [-]
Multi-Point weighting		Uniform
Constraints		$\beta_{\text{out}} > 70\ ^\circ$
Bounds (default)		±20%
Blade	Design	2D Aachen first stator
	Parametrization	CAD

Table 5.4: Setup of the numerical optimization.

The NSGA2 specific parameters are summarized in Table 5.5. A population size of 40 is selected, by applying roughly a factor of 2 over the number of design variables. This factor is shown to give acceptable convergence rate and compromise between computational cost and accuracy [Tüchler et al. 2018]. He and Zheng [2017] also adopted a population of similar size of 50 individuals, with 14 design variables on two objectives. The probabilities for crossover and mutation are kept as the default value as implemented by the PyOptSparse optimization library. Tuning of these parameters is overly time-consuming and deemed outside of the scope of this work.

Population size	50	[-]
$P_{\text{crossover}}$	0.6	[-]
P_{mutation}	0.2	[-]

Table 5.5: NSGA2-specific parameters.

5.4. VERIFICATION AND VALIDATION

In order to ensure confidence in the results of the optimization, verification and validation is needed of the numerical methods and models. If the models do not provide an accurate representation, the optimizer may exploit these errors and yield false optimized designs.

The American Society of Mechanical Engineers describe verification as the "process of determining that a model implementation accurately represents the developers conceptual description of the model", also known as *solving governing equations right*. Validation is described as the "process of determining the degree to which the model implementation is an accurate representation of the real world for the intended uses of the model", or *Solving the right equations* [ASME 2009]. Roache [1998] describes verification as a mathematics and computer science issue. Whereas validation as a physical sciences and mathematics issue.

Validation of the fluid-dynamic simulation model is not carried out in this work, due to this topic being deemed outside the scope of work. Of which the focus is to present the capability of the framework and demonstrating its application to the multi-point optimization of turbomachinery.

Within CFD verification there are two sub classes, the first is code verification and the second is solution verification [ASME 2009]. Code verification aims to ensure that the code solution accurately represents the mathematical models. Solution verification aims to ensure the numerical accuracy of the code solution for the target problem. Strictly speaking, there is a need to quantify accuracy using methods from both classes. For example, ensuring numerical convergence as part of solution verification does not ensure accuracy of physical constants. Likewise, small deviations in gradient limiters may pass code verification but may not result in numerical convergence for the problem at hand.

An approach to CFD solution verification of time-averaged simulations is comparing the solution with that of methods which capture the dynamics of turbulent flows such as LES or Unsteady-RANS [Speziale 1998]. Enabling quantifying the error incurred from ignoring unsteady processes such as vortex shedding.

Fluid-dynamic code verification is not performed here because of time constraints and the existence of verification studies within SU2 for the Aachen profile [Rubino et al. 2020; Vitale et al. 2020], albeit for more elaborate turbulence models and three-dimensional geometry. Fluid-dynamic solution verification is performed in the form of a grid convergence study to ensure grid-independent results.

A rigorous method for quantifying the discretization uncertainty incurred by using a specific mesh is the Grid Convergence Index (GCI) as described by Roache [1998] and extended by Eça and Hoekstra [2004]. The method provides a conservative uncertainty estimation U_i for grid i, reported to guarantee a 95% confidence level Eça et al. [2004]. However, such an approach is deemed outside of the scope of this work due to time constraints. For the interested, the calculation steps of U_i using the GCI, see Appendix D.

A form of numerical solution verification is performed to assess if the optimizer managed to find an improved design, verifying the sanity of the optimized design in a numerical sense. This is performed through checking if the optimized design constitutes a stationary point in design space. In unconstrained optimization, if the magnitude of the gradient vector of the optimized design is not zero, the design is not located on a minimum and thus can be improved.

5.4.1. GRID CONVERGENCE

A grid convergence study is performed to quantify the influence of mesh fidelity on the objective function value. The convergence of s_{gen}, target mesh size and flow simulation time for increasing number of mesh elements in the domain is displayed in Figure 5.3.

Mesh generation is performed on a single core of Intel(R) Xeon(R) CPU E5-2640v4 at 2.4GHz. `Gmsh` enables mesh generation on multiple cores in parallel, however no significant speedup is observed for the baseline mesh in this work. The CFD setup in this grid convergence study is equivalent to the one utilized during optimization, except for the flow solver iteration number termination condition. This number is set at 10k in order to mitigate solution errors due to incomplete flow convergence. The finest mesh, also referred to as the continuum mesh, is determined on the basis of time budget considerations; evaluating finer meshes requires more than an hour of computation, which is infeasible taking into account the evaluation frequency of this optimization study.

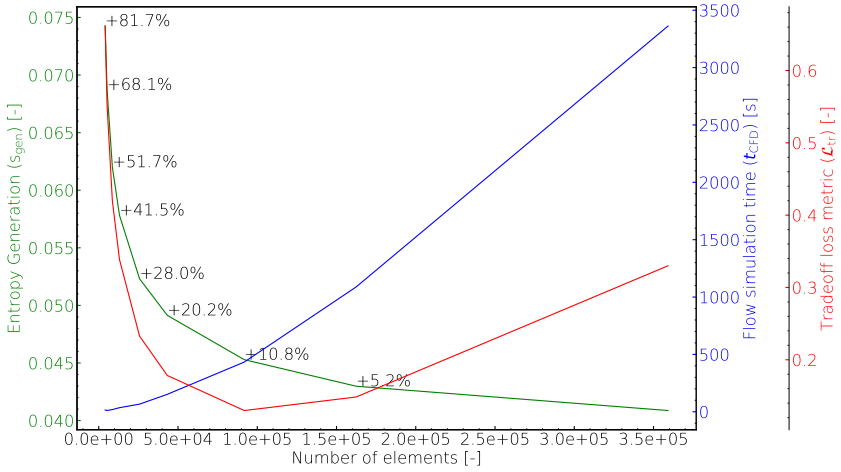


Figure 5.3: Convergence of entropy generation s_{gen}, flow computation time and computed trade-off metric for finer mesh sizes for the baseline design.

Solely considering error values, the mesh with 5.2% error with respect to the finest mesh would reasonably satisfy the accuracy requirement. However, the flow computational time of 1088 seconds is prohibitively large. The mesh with target element size of $3 \times 10^{-4} m$ is chosen, following a weighted tradeoff loss metric $\mathcal{L}_{\text{tr}}$ between flow computational time and discretization error. The trade-off loss metric value $\mathcal{L}_{\text{tr},i}$ for mesh i is calculated as the weighted sum of relative discretization error ϵ_i and computational time $t_{\text{CFD},i}$ as in Equation 5.1. ϵ_i and $t_{\text{CFD},i}$ are scaled with respect to their respective largest values. Weights are applied in order to emphasize ϵ_i, with $w_\epsilon = 0.66$ and $w_{t_{\text{CFD}}} = 0.33$. The solution error of the selected mesh at 2k flow iterations is negligibly low, as detailed in appendix C.

$$\mathcal{L}_{\text{tr},i} = w_\epsilon \frac{\epsilon_i}{\max(\epsilon)} + w_{t_{\text{CFD}}} \frac{t_{\text{CFD},i}}{\max(t_{\text{CFD}})} \tag{5.1}$$

Of all evaluated meshes, the minimal order of magnitude reduction of the flow residual is 5.771, and average at 7.21. Assuming the flow to be sufficiently converged, the steep increase in ϵ_i for coarser meshes can be attributed to the diminishing resolution capacity of the entropy-generating viscous effects.

Based on the results, the mesh corresponding with the lowest trade-off loss, with a target mesh size of 3.0×10^{-4} is selected. This mesh represents a reasonable compromise between accuracy and computational time.

An accurate *change* in $\mathcal{J}$ resulting from a perturbation in $\Delta\boldsymbol{\alpha}$ is of importance for argumentation on the performance of designs, during the design optimization process. Thus in this case, the actual $\mathcal{J}$ is of less importance than an accurate change in $\mathcal{J}$ due to $\Delta\boldsymbol{\alpha}$. Hence, if the spread in ϵ is negligible for a set of geometries evaluated on constant mesh fidelity, the change in $\mathcal{J}$ due to $\Delta\boldsymbol{\alpha}$ can be accurately derived, despite an existing significant ϵ. However, this specific assessment is only valid for constant flow boundary conditions. To assess this accuracy in *change* of $\mathcal{J}$, Figure 5.4 displays the convergence of ϵ for meshes of decreasing target mesh element sizes of ten designs of which the design vectors are sampled from a truncated normal distribution. All α_i's of each design are sampled separately from the aforementioned distribution, within truncation bounds of ±20%, reflecting the bounds on the design vector as applied in the optimization of this work.

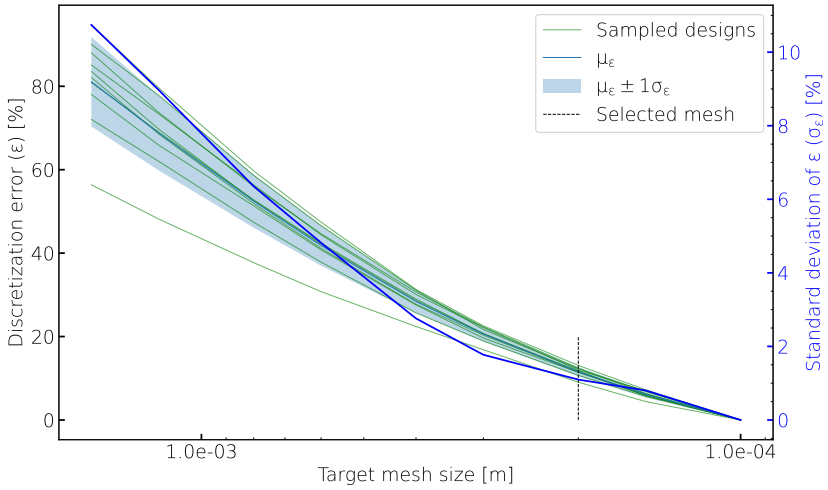


Figure 5.4: Convergence of mesh discretization error ϵ for meshes of decreasing target mesh element sizes, for ten designs initialized by a truncated normal distribution, with truncation of the normal distribution at ±20%. The standard deviation of ϵ for each mesh, σ_ϵ, is plotted on the right y-axis. The mean of ϵ, μ_ϵ, is indicated by the light blue line and the target mesh size of the selected mesh for optimization is indicated by the dashed line.

Of the mesh employed during optimization, the mean discretization error and standard deviation are $\mu_\epsilon = 11.57\%$ and $\sigma_\epsilon = 1.10\%$. This low spread in discretization error indicates a low sensitivity of relative Δs_{gen} to changes in $\boldsymbol{\alpha}$. Supporting argumentation on the differences in performance between blade shapes of the initial design (and operating condition) by using simulation results from the selected coarser mesh.

5.4.2. ADJOINT GRADIENTS

Since the optimization framework enables optimization with gradient-based optimizers, it implements the adjoint method to obtain the flow sensitivity. Correct gradients are of high importance because gradients are a main source of information for gradient-based optimizers. With incorrect gradient information, optimizers might traverse a longer path before converging on the final design or yield non-optimal final designs. Additionally, the design stationarity assessment performed in this work relies on the adjoint method for obtaining the design gradients.

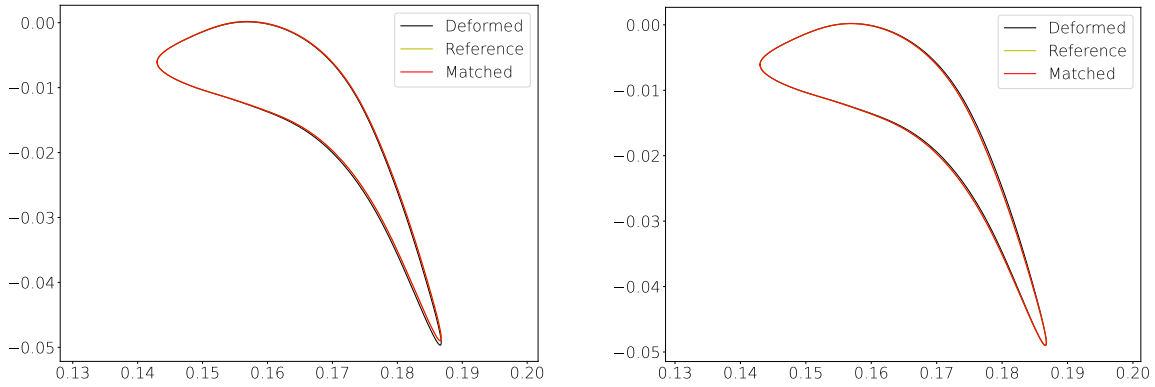
A gradient validation module is available to assess the error incurred from using the discrete adjoint method with respect to the primal CFD solver. To test the working of the gradient validation module, the gradients obtained with the adjoint method are verified on the baseline Aachen turbine testcase. This is performed by comparing the adjoint gradients to the gradients obtained from finite differencing the results of the direct flow solver.

Sources of modelling error can stem from the CEV assumption of the adjoint method. However, the flow field at the optimal design point is expected to be relatively well-behaved, meaning a less complicated turbulent flow structure, lowering the error due to the CEV assumption. Added, the accuracy loss is limited for blades operating in transonic conditions, such as in this work [Vitale et al. 2020]. This is due to a smaller share of viscous losses when compared to low subsonic. Therefore lower sensitivity to the ignored turbulent flow quantities of the CEV [Rubino et al. 2020]. Another source of modelling error can be from the eddy viscosity assumption of the SA flow turbulence model as described in section 2.1.3.

The finite difference values are calculated with a forward approach. As described in the expression in Equation 5.3 for design variable i. As shown in Equation 5.2, $\boldsymbol{\delta}_i$ is a Kronecker delta vector which perturbs the scaled design vector with Δh at the index of variable i. To illustrate the difference in blade geometry by varying individual design variables, the geometry of the baseline blade before and after perturbing ξ and θ_{out} by $\Delta h = 0.01$ is shown in Figure 5.5.

$$\boldsymbol{\delta}_i = \mathbf{1}_i(u_j) = \begin{cases} 1 \cdot \Delta h & \text{when } j = i \\ 0 & \text{else} \end{cases} \tag{5.2}$$

$$\frac{\partial \mathcal{J}}{\partial \alpha_i}\Big|_{FD} = \frac{\mathcal{J}(\alpha^0 + \boldsymbol{\delta}_i) - \mathcal{J}(\alpha^0)}{||\boldsymbol{\delta}_i||} \tag{5.3}$$



(a) $\xi^0\,(1+\Delta h)$. (b) $\theta^0_{\text{out}}\,(1+\Delta h)$.

Figure 5.5: Deformed two-dimensional Aachen blade w.r.t. stagger ξ (a) and trailing edge metal angle θ_{out} (b).

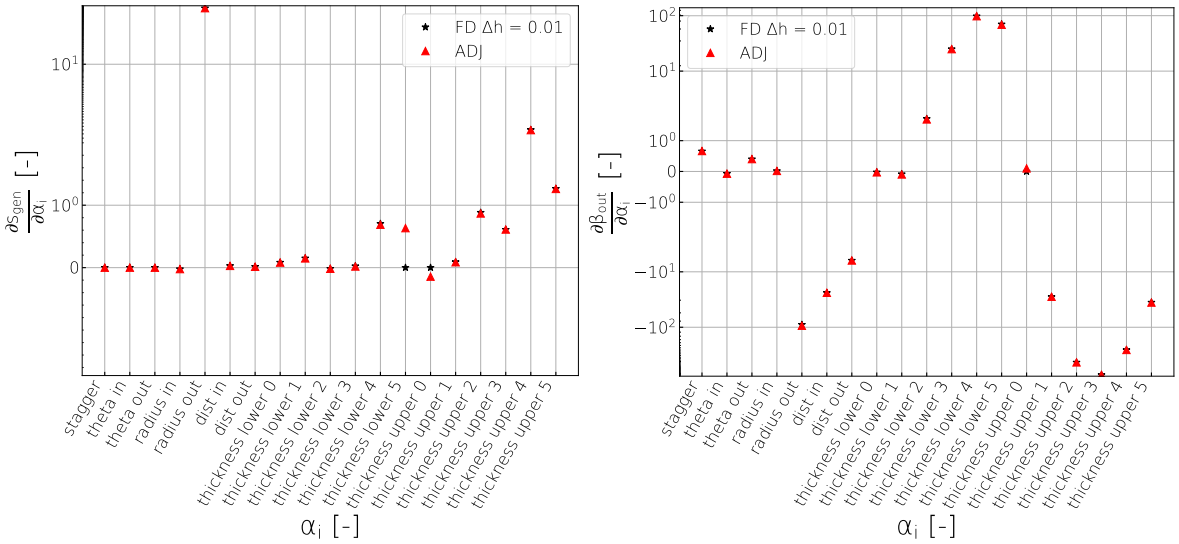


Figure 5.6: Partial sensitivities with of entropy generation respect to the design variables $\frac{\partial s_{\text{gen}}}{\partial \alpha_i}$ (a) and flow angle at the outlet $\frac{\partial \beta_{\text{out}}}{\partial \alpha_i}$ (b) for the baseline two-dimensional Aachen turbine blade.

Note, the correct functioning of the discrete adjoint method is agnostic of the nominator to which the sensitivity is computed, in other words, validating the adjoint method gradient on one flow quantity of interest suffices for validation.

The zero finite-difference sensitivities for t_5^l and t_0^u for s_{gen} in Figure 5.6a and for t_0^u for β_{out} in Figure 5.6b are due to rounding errors. Performing finite difference with a step of 0.1 yields non-zero gradients. However, FD sensitivities of nonlinear responses lose accuracy for larger step sizes.

Observing the partial sensitivities of s_{gen}, an increase in sensitivity is apparent for the thicknesses located near the TE. A thicker TE increases flow accelerations at the TE where the boundary layer flow exits into the wake and hence increases losses due to flow mixing. $\frac{\partial s_{\text{gen}}}{\partial \alpha_i}$ is lower for the lower blade thicknesses compared to the upper thicknesses, because the boundary layer at the pressure side is relatively thin due to accelerating flow. Thus a lower impact on s_{gen}.

A distinct trend in magnitude of the partial sensitivities of β_{out} with respect to thickness is visible towards the mid-aft-chord locations. This is expected, because changing thickness at this location has an increased effect on the mean flow angle at the outlet boundary. The sensitivities of the lower thicknesses are positive, whereas those of the upper thicknesses are negative. This also agrees with what is expected, because a more convex pressure side would drive the mean flow angle in the wake more off-axis, decreasing the outlet flow angle and vice-versa for the lower blade thicknesses.

The harmonic mean of differences $\mu'_{\Delta,i}$ between the finite difference and adjoint methods is applied to indicate the gradient validation error per design variable, as displayed in Table 5.6. $\mu'_{\Delta,i}$ is employed because it is a measure which is agnostic to the denominator in calculating differences between values.

$$\mu'_{\Delta,i} = \frac{2}{\frac{1}{\left|\frac{\frac{\partial \mathcal{J}}{\partial \alpha_i}\big|_{FD} - \frac{\partial \mathcal{J}}{\partial \alpha_i}\big|_{ADJ}}{\frac{\partial \mathcal{J}}{\partial \alpha_i}\big|_{FD}}\right|} + \frac{1}{\left|\frac{\frac{\partial \mathcal{J}}{\partial \alpha_i}\big|_{ADJ} - \frac{d\mathcal{J}}{\partial \alpha_i}\big|_{FD}}{\frac{\partial \mathcal{J}}{\partial \alpha_i}\big|_{ADJ}}\right|}} \tag{5.4}$$

	s_{gen}			β_{out}		
Design variable	FD	ADJ	$\mu'_{\Delta,i}$ [%]	FD	ADJ	$\mu'_{\Delta,i}$ [%]
ξ	-3.40e-04	-4.15e-04	19.87	6.56e-01	6.63e-01	1.06
θ_{in}	1.38e-04	1.38e-04	0.07	-7.26e-02	-7.25e-02	0.13
θ_{out}	-5.75e-04	-6.93e-04	18.67	4.00e-01	4.01e-01	0.39
r_{in}	-2.28e-02	-2.32e-02	1.70	1.56e-02	1.72e-02	10.09
r_{out}	3.15e+01	3.16e+01	0.04	-8.98e+01	-9.39e+01	4.48
d_{in}	2.84e-02	2.81e-02	1.21	-2.39e+01	-2.39e+01	0.17
d_{out}	1.43e-02	1.42e-02	0.69	-6.25	-6.25	0.06
t^{l}_{0-4}	8.21e-02	8.13e-02	0.98	-2.55e-02	-2.89e-02	12.36
	1.50e-01	1.48e-01	1.36	-8.56e-02	-9.72e-02	12.70
	-1.68e-02	-1.79e-02	6.31	1.71	1.69	1.25
	2.48e-02	1.84e-02	29.41	2.50e+01	2.49e+01	0.36
	7.04e-01	6.87e-01	2.39	9.79e+01	9.77e+01	0.20
t^{u}_{0-5}	0	-1.44e-01	-	0	1.00e-01	-
	9.09e-02	8.64e-02	5.07	-2.83e+01	-2.82e+01	0.29
	8.82e-01	8.68e-01	1.58	-4.31e+02	-4.30e+02	0.12
	6.09e-01	6.11e-01	0.39	-7.22e+02	-7.22e+02	0.07
	2.61	2.59	0.58	-2.56e+02	-2.57e+02	0.11
	1.26	1.26	0.28	-3.57e+01	-3.59e+01	0.39

Table 5.6: Relative gradient validation errors for the flow quantities of interest entropy generation s_{gen} and outlet flow angle β_{out}, with respect to the design variables. Errors are expressed in terms of the harmonic mean of differences $\mu'_{\Delta,i}$ between the finite difference (FD) and adjoint (ADJ) methods.

Note the consistent sign of the FD and ADJ gradients. This instills confidence in the adjoint method in providing correct directions to the optimizer. The larger $\mu'_{\Delta,i}$ values are observed for the partial sensitivities with smaller magnitude, resulting in small absolute errors. Nonetheless, the average relative $\mu'_{\Delta,i}$ is 5.3% and 2.57% for s_{gen} and β_{out} respectively, which is deemed satisfactory.

5.5. INTEGRATED MESH GENERATION

Integrated meshing is selected as mesh generation method due to NSGA2 being a global optimizer. The baseline mesh contains 41.6k mesh elements, while the mesh employed in mesh deformation contains 26.9k mesh elements. Regarding compute-time of mesh generation for the Aachen blade, the deformation approach requires less time; around one second. The integrated meshing approach requires around 17 seconds for mesh generation and periodic boundary node correction. This time required for mesh generation is a factor of ten over the deformation approach, correcting for the increased element count of the integrated meshing generated mesh.

The fine mesh is constructed with a target mesh size of the finest mesh from the discretization study. The height of the first cell at the blade surface is set to ensure that the first confidence interval (1-σ) y^+ is smaller than 1. This approach is chosen above ensuring that the maximum $y^+ < 1$, due to the required exceedingly tiny mesh height along the blade, inhibiting convergence of the flow simulation.

The fine mesh is employed for post-optimization verification of the fluid-dynamic results of optimized blade. More specifically, the results of s_{gen}, M_{is} along the blade surface, and outlet Mach distribution are compared between the optimization and fine mesh.

5.6. OPTIMIZED DESIGN STATIONARITY

The magnitude of the gradient vector of the optimized design is compared with that of the baseline design. The results of this assessment are shown in the subsequent chapter in which the results are presented. The expectation is that the magnitude of the gradient vector of the optimized design is reduced with respect to that of the initial design. The difference in magnitude of the design variable i is expressed in Equation 5.5. This value allows to evaluate each design variable separately. Comparing the gradient vector magnitudes of the baseline with the corresponding optimized gradient vectors is performed by comparing the L^2 norms of the gradient vectors.

$$\Delta \left| \frac{\partial s_{\text{gen}}}{\partial \alpha_i} \right| = \left| \left(\frac{\partial s_{\text{gen}}}{\partial \alpha_i} \right)_{\text{final}} \right| - \left| \left(\frac{\partial s_{\text{gen}}}{\partial \alpha_i} \right)_{\text{baseline}} \right| \tag{5.5}$$

The gradient norm of the optimized design may not be zero due to active flow constraints, active bounds on the design variables, due to the stochasticity of the optimization problem response value or incomplete optimization due to termination criteria such as time or number of iterations. These termination criteria do not take into account the convergence of the objective function value, and thus do not warrant a stationary point in design space. The optimization problem stochasticity is due to the noise in the response value due to modelling errors.

For optimization problems with active bounds on the optimized design variables, an extra check is performed to assess whether certain variables can be excluded from the assessment. For example, if a variable is bounded by its lower bound and its sensitivity is positive, its value can not be moved in the direction of decreasing s_{gen}.

6

RESULTS AND DISCUSSION

The results of the multi-point optimization are presented in this chapter. This chapter concludes with a discussion of the challenges experienced with regards to numerical optimization.

The experiments in the following sections are performed on two 10-core, 20-thread Intel(R) Xeon(R) CPU E5-2640v4 2.4GHz CPUs on the High Performance Computing cluster of the TU Delft.

6.1. MESH GENERATION

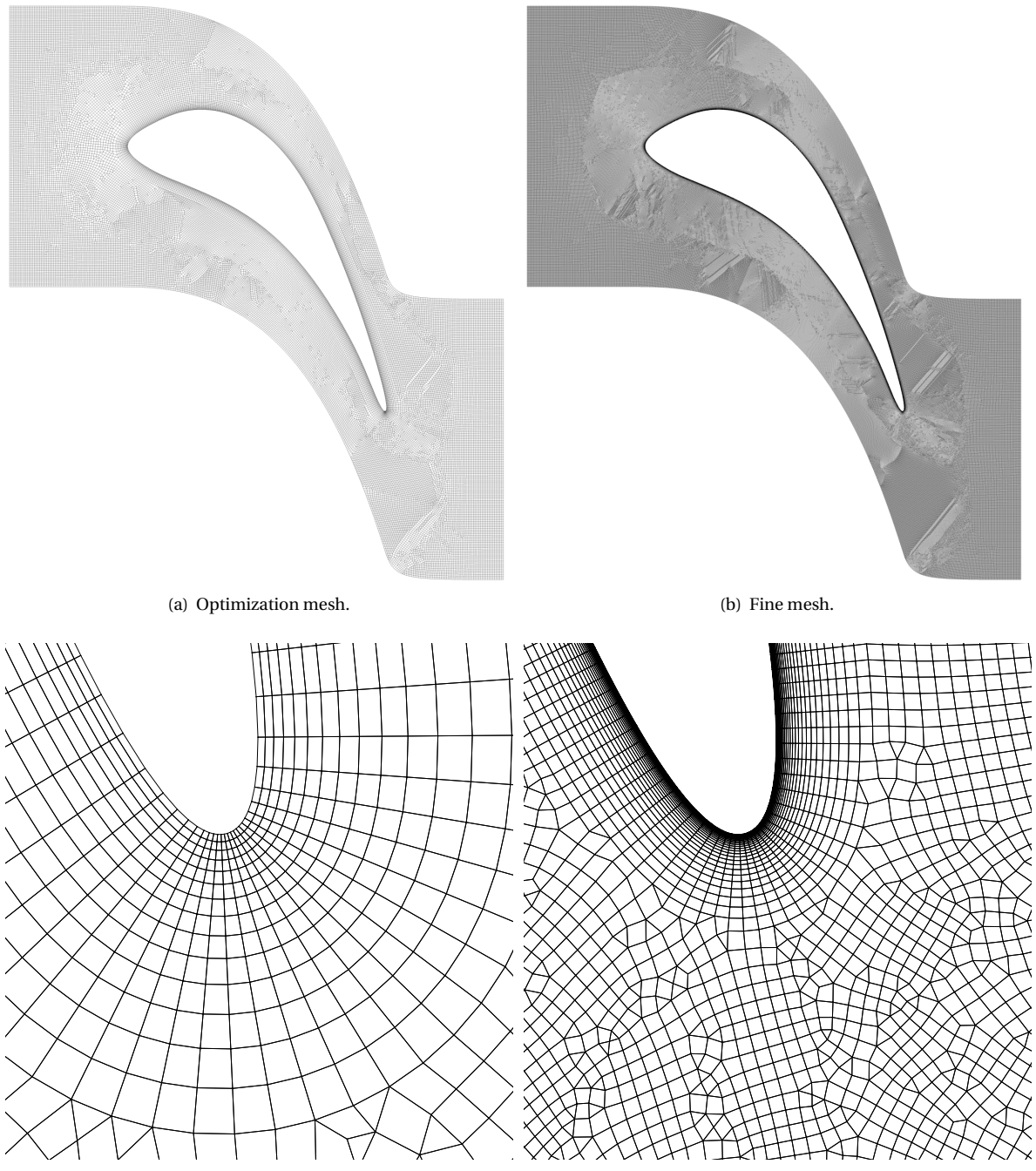
The mean time required for generating the mesh which is employed during optimization with `Gmsh` is close to 12 seconds on a single core, with 210Mb of RAM used. Afterwards, `SU2_PER` requires 5 seconds to reposition the mesh vertices in proximity of the periodic boundaries.

The fine mesh contains circa 400k cells, roughly a ten-fold of mesh cells of the mesh employed during optimization. See Table 6.1. The arithmetic mean y^+ for the fine mesh of the baseline and optimized blades is 0.66, with a standard deviation of 0.21. The 1σ bound for y^+ is below unity, thus it is assumed that turbulence in the boundary layer is adequately resolved in the fine mesh.

		Element count			y^+	
Mesh	Design	Quads	Triangles	$\sum$	μ	σ
Optimization	Baseline	41,588	1,729	43,317	13.94	3.48
	Optimized	42,576	1,591	44,167	13.46	3.89
Fine	Baseline	382,873	14,127	397,000	0.68	0.21
	Optimized	386,871	13,904	400,775	0.65	0.22

Table 6.1: Mesh element counts and y^+ arithmetic mean (μ) and standard deviation (σ) for the mesh applied during optimization and fine mesh. Statistics for the meshes of baseline and optimized blade.

Figure 6.1 pictures the mesh topology of the mesh employed during optimization and fine mesh, by showing an overview and a zoom onto the blade TE. Apparent is the smaller mesh element size in the domain of the fine mesh, causing a reduced height of the inflation layers. This is expected because the largest inflation layer height is ensured to be smaller than the target domain mesh element size, as described by the inflation layer computation algorithm in Algorithm 1. Also visible is the smaller height of the first mesh element layer of the fine mesh along the surface of the blade. The effects of which are described in the following sections of this chapter.



(a) Optimization mesh.

(b) Fine mesh.

(c) Zoom onto trailing edge, optimization mesh.

(d) Zoom onto trailing edge, fine mesh.

Figure 6.1: Mesh element comparison of the baseline design between the mesh employed during optimization (a, c) and the fine mesh (b, d). The overviews are pictured in (a, b) and a zoom onto the blade surface at the trailing edge is shown in (c, d).

6.2. OPTIMIZATION

The optimization of the two dimensional Aachen turbine blade yields a considerable reduction of entropy generation s_{gen} of -3.74% at the nominal operating condition and -4.25% at off-design. See section 5.2.3 for the specification of the operating conditions. The optimization is halted after around 2900 flow evaluations. The total run time on 20 cores with hyper-threading is 102 hours, and required 4.68 Gb of memory for the direct flow evaluations. The design iteration with lowest s_{gen} is obtained in flow evaluation 2728.

Figure 6.2 displays the s_{gen} of all flow evaluations during optimization. The spread of s_{gen} decreases sharply during the first 200 flow evaluations, after which remains at a constant level. This *steady-state* spread is due to the exploratory property of the mutation operation in the NSGA2 optimizer. Mutation ensures that designs randomly perturbed from current best are evaluated, which is reflected by the increased s_{gen}. Another source for the spread in s_{gen} can be numerical noise, despite the time-averaged flow simulation. This can be a result of simulation errors, for example due to non-converged flow solutions or mesh discretization errors.

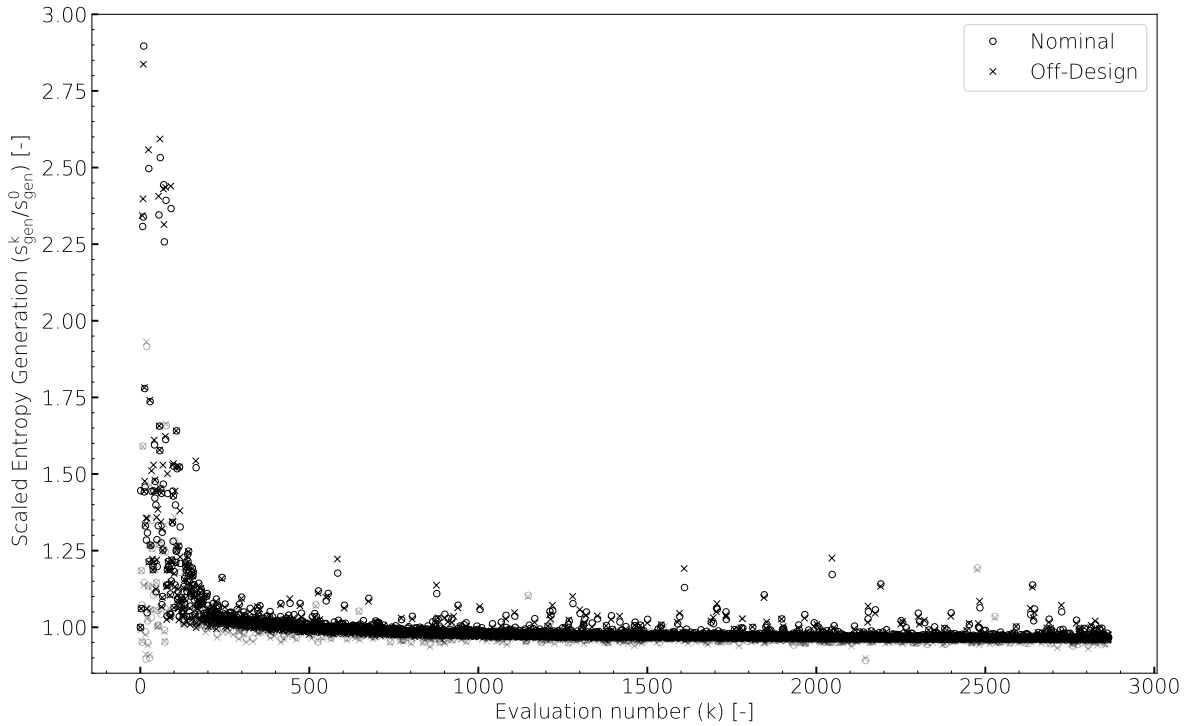


Figure 6.2: Entropy generation of all evaluated designs during the the two-point shape optimization of the Aachen two-dimensional turbine test case. The translucent markers indicate infeasible evaluations.

Figure 6.3 presents the optimization history of s_{gen} for the nominal and off-design operating conditions. The weighted s_{gen} reduction over the two operating conditions is 4.00%. The optimized design is feasible for both operating conditions and meshes, as also summarized in Table 6.2. At around 1100 evaluations, the compute-time efficiency is most optimal. Here, at an expense of roughly 40% of computational time, 75% of final s_{gen} reduction is achieved.

The optimized design evaluated on the fine mesh yields a reduction in s_{gen} at both operating conditions, exceeding the s_{gen} reduction of the mesh employed during optimization, with a weighted s_{gen} reduction of 7.57%. This result similarity verifies the result obtained from the coarser mesh used during optimization, hereby justifying this approach.

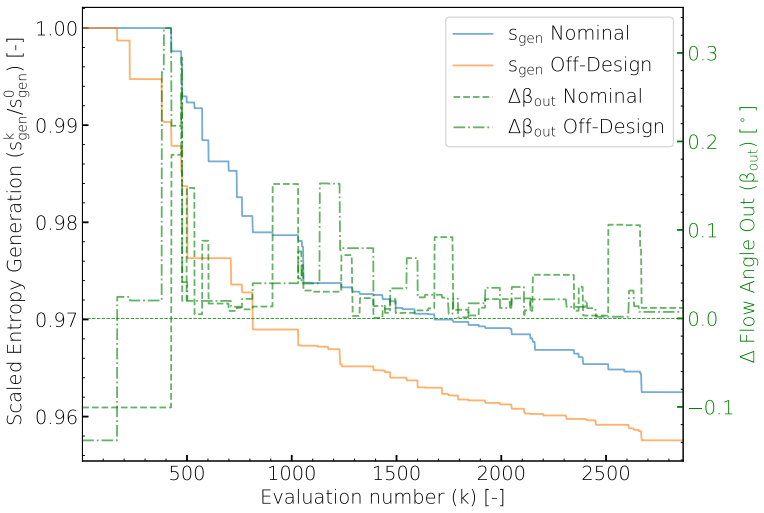


Figure 6.3: Optimization history of the scaled s_{gen}, and constraint on β_{out} for the two-dimensional Aachen turbine blade for nominal and off-design points. A positive $\Delta\beta_{out}$ depicts a satisfied constraint.

		Optimization mesh			Fine mesh		
		Baseline	Final	Δ [%]	Baseline	Final	Δ [%]
Nominal	s_{gen} [-]	4.915×10^{-2}	4.731×10^{-2}	-3.74	2.107×10^{-2}	1.968×10^{-2}	-6.57
	β_{out} [°]	-69.899	-70.012	-	-70.390	-70.485	-
Off-design	s_{gen} [-]	4.881×10^{-2}	4.674×10^{-2}	-4.25	2.242×10^{-2}	2.050×10^{-2}	-8.57
	β_{out} [°]	-69.861	-70.007	-	-70.307	-70.454	-

Table 6.2: Flow quantities of interest entropy generation s_{gen} and outlet flow angle β_{out} of the coarser mesh employed during optimization and the fine mesh.

Figure 6.4 shows the criterion space for the two-point optimization. The two selected operating conditions do not represent strongly competing objectives. This is demonstrated by the positive linear correlation of all evaluated points during optimization, between the s_{gen} of the operating conditions. A positive linear correlation indicates that increasing performance on one operating condition is expected to be accompanied by an increase in performance on the other aswell, and vice-versa. In competing objectives, increasing the performance of one operating condition means a deterioration of the performance at other operating conditions.

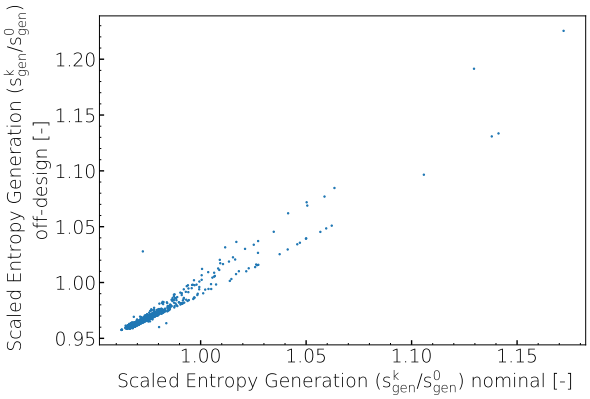


Figure 6.4: Criterion space for the two-point shape optimization of the Aachen single row turbine test case.

6.3. FINAL DESIGN

Figure 6.5 pictures the baseline and optimized blade surface. The optimized blade shape is noticeably thinner at mid chord, includes a smaller LE radius and flatter upstream surfaces on both sides. Both sides of the blade have an increased curvature, of which th suction side peak curvature is located closer to the LE. The length of the blade is not altered significantly, showing a reduction of only 0.92%. The increased surface length due to the increased curvature on the pressure side offsets the surface length decrease of the shortened chord length resulting from a reduced stagger angle. The increase in camber in the profile of the optimized blade compensates for a decreased stagger angle, in order to satisfy the outlet flow angle constraint.

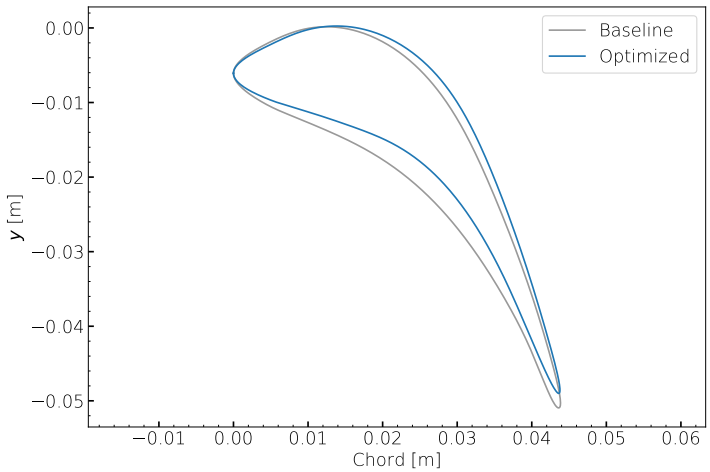


Figure 6.5: Blade surface of baseline and optimized designs.

Figure 6.6 presents the scaled variable values of the optimized design. Some design variable bounds are active for the optimized design, meaning the design variable is located at either its lower or upper bound. The lighter band in Figure 6.6 indicates the design space enclosed by the default bounds. The bounded design variables are r_{in}, r_{out}, t_3^{l}, t_4^{l} and t_4^{u}. The optimizer exploits the extended bounds of ±40% on r_{out}, this custom bound extension is detailed in section 5.2.1. Relaxing the bounds for these design variables would enlarge the design space and provide an opportunity for the optimizer to reduce s_{gen} more. However, extending the bounds increases the risk of failing flow evaluations due to inconsistent blade shapes, as detailed in section 5.2.1. Moreover, tuning the bounds by trial-and-error requires increased time and resources.

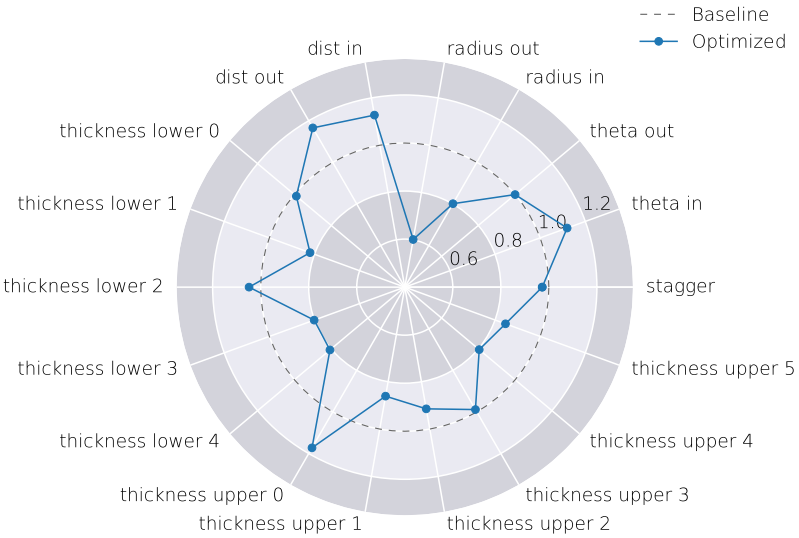


Figure 6.6: Scaled optimal design variables. The dashed line illustrates the baseline design configuration. The white band indicates the design space enclosed by the default bounds on the design vector.

Table 6.3 presents an assessment of the difference in gradient magnitude between baseline and optimized design, in order to verify that the optimized design constitutes a stationary point in design space and thus the optimizer succeeded in improving the design in a numerical sense. A reduction in partial derivative magnitude during optimization is indicated by a green cell, while a magnitude increase by a red cell. Refer to Equation 5.5 for the expression of the partial derivative magnitude difference $\Delta\left|\frac{\partial s_{\text{gen}}}{\partial \alpha_i}\right|$.

Most of the partial sensitivities of the fine mesh of the final design show a reduction in magnitude with respect to baseline. Some optimized variables are actively bounded, for example t_2^{l}, t_3^{l} and t_4^{u}. Refer to Table 5.1 for the corresponding design variable names. These bounded variables are located at their lower bounds, with positive partial sensitivities, thus not possible to move the values of these variables in the direction of decreasing s_{gen}. Consequently, these variables will not be included in this assessment. For the design variables which have an increased sensitivity magnitude as evaluated on the optimization mesh, only stagger ξ and the thicknesses t_2^{l} and t_3^{u} are either not bounded nor kept constant during optimization.

	Baseline	Optimized				
			Fine mesh		Optimization mesh	
Design variable	$\frac{\partial s_{\text{gen}}}{\partial \alpha_i}$	Bound active	$\frac{\partial s_{\text{gen}}}{\partial \alpha_i}$	$\Delta\left\|\frac{\partial s_{\text{gen}}}{\partial \alpha_i}\right\|$	$\frac{\partial s_{\text{gen}}}{\partial \alpha_i}$	$\Delta\left\|\frac{\partial s_{\text{gen}}}{\partial \alpha_i}\right\|$
ξ	-4.150×10^{-4}	-	-1.323×10^{-3}	9.08×10^{-4}	-8.778×10^{-4}	4.63×10^{-4}
θ_{in}	1.380×10^{-4}	-	9.695×10^{-5}	-4.10×10^{-5}	1.315×10^{-4}	-6.55×10^{-6}
θ_{out}	-6.930×10^{-4}	-	-4.821×10^{-4}	-2.11×10^{-4}	-6.038×10^{-4}	-8.92×10^{-5}
r_{in}	-2.320×10^{-2}	Yes	1.543×10^{-2}	-7.78×10^{-3}	3.817×10^{-2}	1.50×10^{-2}
r_{out}	3.160×10^{1}	Yes	1.948×10^{1}	-1.21×10^{1}	2.094×10^{1}	-1.07×10^{1}
d_{in}	2.810×10^{-2}	-	1.459×10^{-2}	-1.35×10^{-2}	2.615×10^{-2}	-1.95×10^{-3}
d_{out}	1.420×10^{-2}	-	8.466×10^{-3}	-5.73×10^{-3}	1.151×10^{-2}	-2.69×10^{-3}
$t_{0\text{-}4}^{\text{l}}$	8.130×10^{-2}	-	3.086×10^{-2}	-5.04×10^{-2}	6.497×10^{-2}	-1.63×10^{-2}
	1.480×10^{-1}	-	6.538×10^{-2}	-8.26×10^{-2}	1.115×10^{-1}	-3.65×10^{-2}
	-1.790×10^{-2}	-	2.908×10^{-2}	1.12×10^{-2}	1.070×10^{-3}	-1.68×10^{-2}
	1.840×10^{-2}	Yes	1.714×10^{-1}	1.53×10^{-1}	1.600×10^{-1}	1.42×10^{-1}
	6.870×10^{-1}	Yes	8.749×10^{-1}	1.88×10^{-1}	1.021	3.34×10^{-1}
$t_{0\text{-}5}^{\text{u}}$	-1.440×10^{-1}	-	-2.619×10^{-2}	-1.18×10^{-1}	-4.697×10^{-2}	-9.70×10^{-2}
	8.640×10^{-2}	-	5.529×10^{-2}	-3.11×10^{-2}	4.852×10^{-2}	-3.79×10^{-2}
	8.680×10^{-1}	-	4.887×10^{-1}	-3.79×10^{-1}	7.140×10^{-1}	-1.54×10^{-1}
	6.110×10^{-1}	-	6.647×10^{-1}	5.37×10^{-2}	8.712×10^{-1}	2.60×10^{-1}
	2.590	Yes	1.883	-7.07×10^{-1}	1.981	-6.09×10^{-1}
	1.260	-	1.086	-1.74×10^{-1}	1.163	-9.72×10^{-2}

Table 6.3: Magnitudes of partial sensitivity for entropy generation s_{gen} with respect to design variable i α_i, between the baseline and optimized design for the fine mesh and mesh employed during optimization. Partial sensitivities of the optimized design which have increased in magnitude with respect to baseline are marked in red, while sensitivities which have decreased in magnitude are marked in green. Refer to Table 5.1 for the corresponding design variable names.

The existence of variables with un-bounded and non-decreasing sensitivity magnitude indicates that an improvement of design is possible. However, this may also be expected for an optimized design due to some

combination of the termination criterion being a time threshold or the noisiness of the problem or the outlet flow angle flow constraint.

Table 6.4 displays the L^2 norm of the gradient vectors of the optimized design with respect to the baseline design, for the fine mesh. The L^2 norm of the gradient vector of the final design displays a 18% reduction in magnitude with respect to the baseline gradient norm. This reduction is what is expected for an improved design of a bound-constrained optimization problem. Contrarily, when no flow and geometric constraints are applied, a reduction closer to 100% is expected because the optimizer is free to find a design vector in the unrestricted design space. A reduction of exactly 100% is unlikely due to numerical noise or modelling errors in the response value or the time- or iteration-based termination criterion of the optimization.

Design	$\left\| \frac{ds_{\text{gen}}}{d\boldsymbol{\alpha}} \right\|_2$	Δ w.r.t. baseline [%]
Baseline	1.665	-
Final	1.3673	-17.9

Table 6.4: L^2 norm of the gradient vectors $\left\| \frac{ds_{\text{gen}}}{d\boldsymbol{\alpha}} \right\|_2$ of the optimized design with respect to the baseline design. For the optimized design, both gradient vector norms of the fine mesh and mesh employed during optimization are displayed. Variables which are either kept constant or actively bounded are excluded from this result.

6.4. FLOW

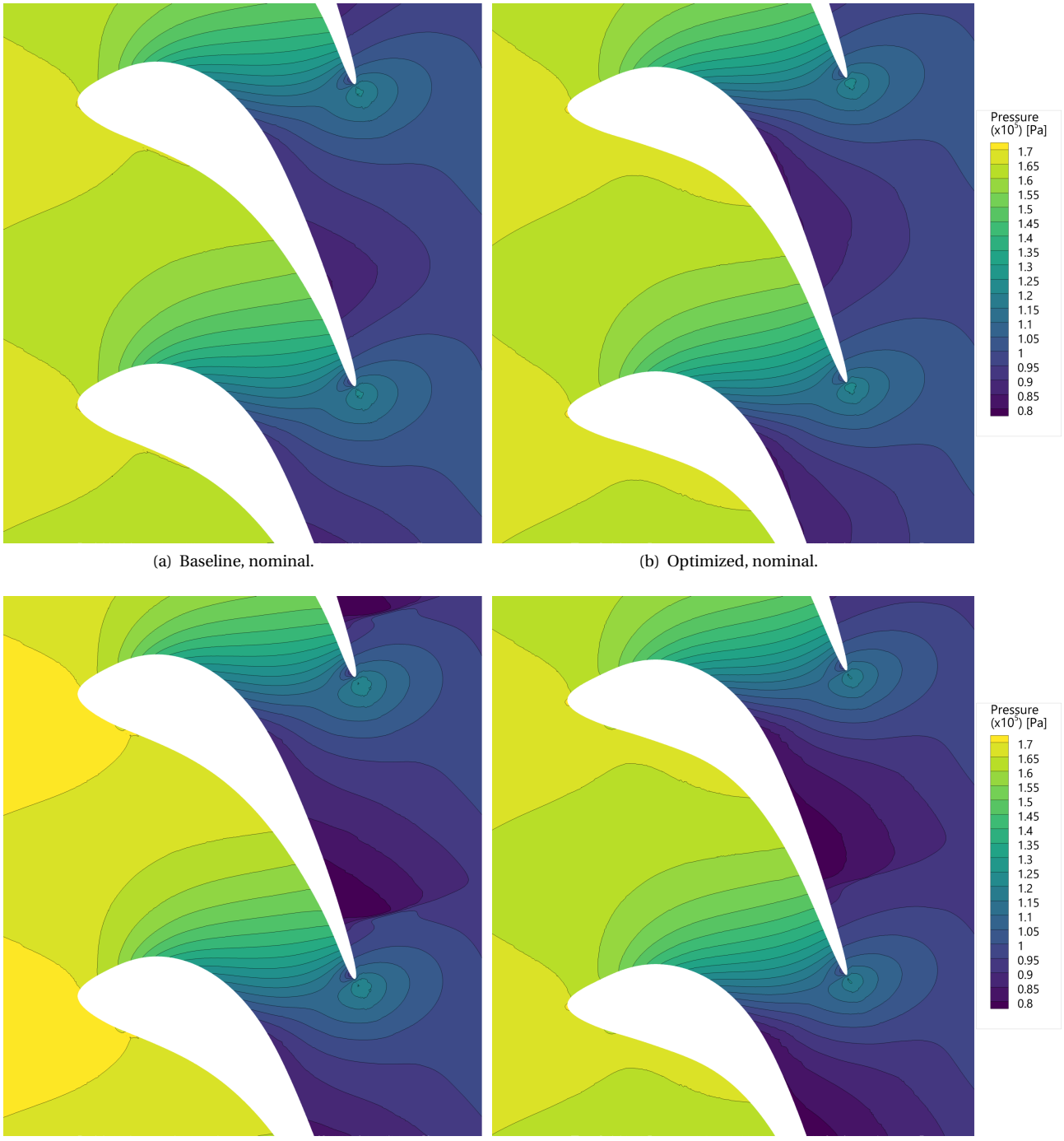
This section presents and discusses comparisons of flow quantities such as pressure and entropy in the blade-to-blade plane, isentropic Mach numbers along the surface and Mach number along the outlet boundary, between the baseline and optimized blades at the two operating conditions. These quantities are interpreted in order to describe what fluid-dynamic loss mechanisms as described in section 2.2.1 affect the entropy generation of the two-dimensional turbine blade.

6.4.1. BLADE-TO-BLADE

Figure 6.7 shows the blade-to-blade pressure contours of the nominal operating condition (a, b) and off-design (c, d) for the initial (a, c) and optimized blade. The optimized blade displays two main points of assessment. Firstly, an upstream shift of low pressure on the suction side, which is also supported by the upstream shift in M_{is} peak in Figure 6.11 of the next section. Secondly, a reduced adverse pressure gradient along the aft suction side. Looking at the off-design contours, a shock is visible on the aft suction side. The shock is weakened in the optimized design as seen by the reduced adverse pressure gradient.

A shockwave thickens the boundary layer locally. However, the impact on s_{gen} reduction due to the attenuation of this shock is negligible in this Mach regime, which is supported by the absence of significant entropy at this location in the domain and boundary layer in the entropy contour plots in Figure 6.8.

As anticipated, the main locations of entropy in the flow field are the boundary layer on the suction side and the wake of the blade. Observing the blade-to-blade entropy contours for the baseline and optimized blades in nominal and off-design condition in Figure 6.8, it is clear that the differences in domain entropy between the baseline and optimized blade are similar for nominal and off-design conditions.



(a) Baseline, nominal.

(b) Optimized, nominal.

(c) Baseline, off-design.

(d) Optimized, off-design.

Figure 6.7: Pressure contours of nominal operating condition (a, b) and off-design (c, d) operating condition for the baseline (a, c) and optimized (b, d) blade.

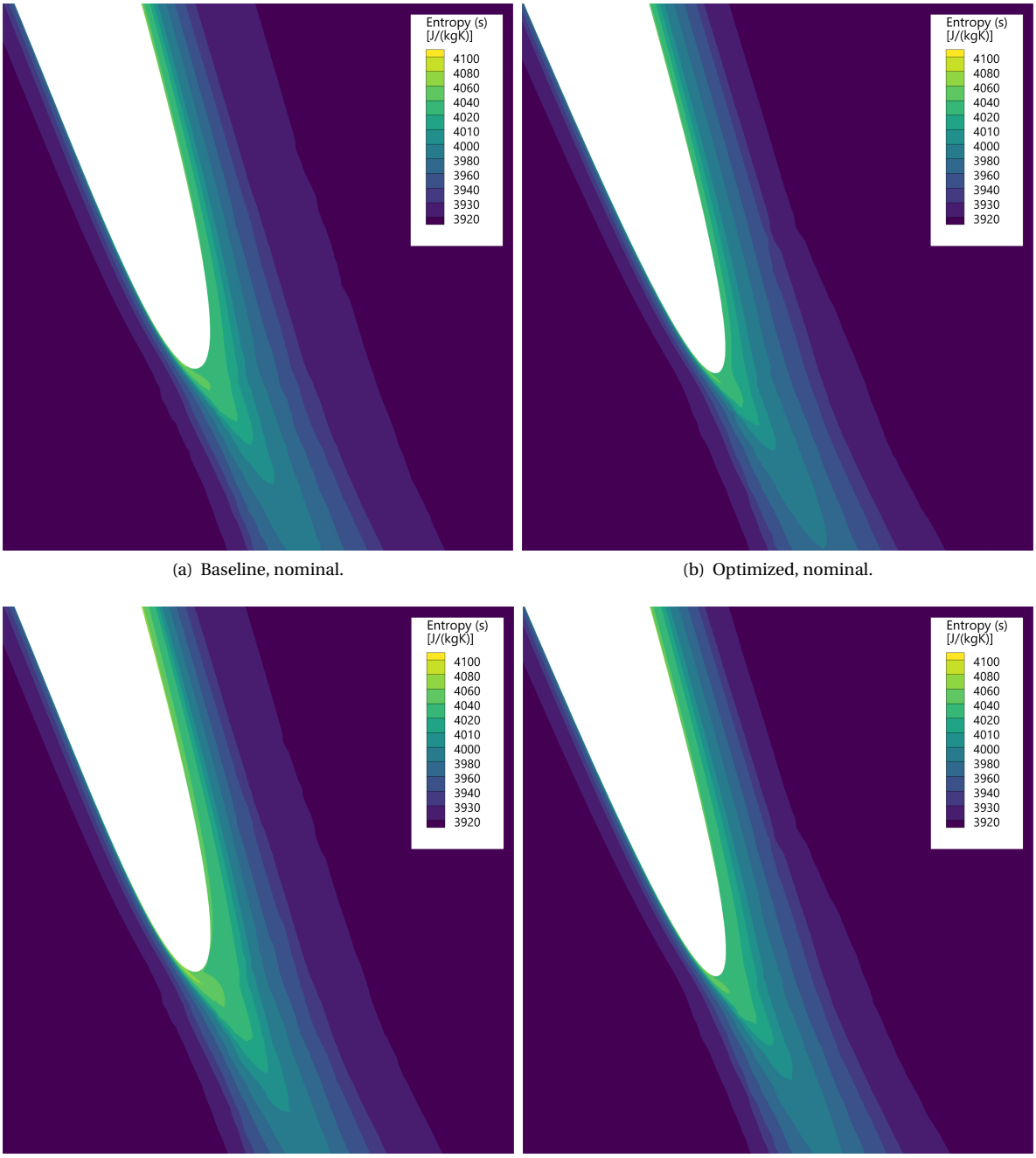
(a) Baseline, nominal.

(b) Optimized, nominal.

(c) Baseline, off-design.

(d) Optimized, off-design.

Figure 6.8: Entropy contours in the blade-to-blade plane for the nominal operating point (a, b) and off-design (c, d), for the baseline (a, c) and optimized (b, d) blade. The contour levels reflect the expected locations of high entropy; in the wake and along the aft suction side.



(a) Baseline, nominal.

(b) Optimized, nominal.

(c) Baseline, off-design.

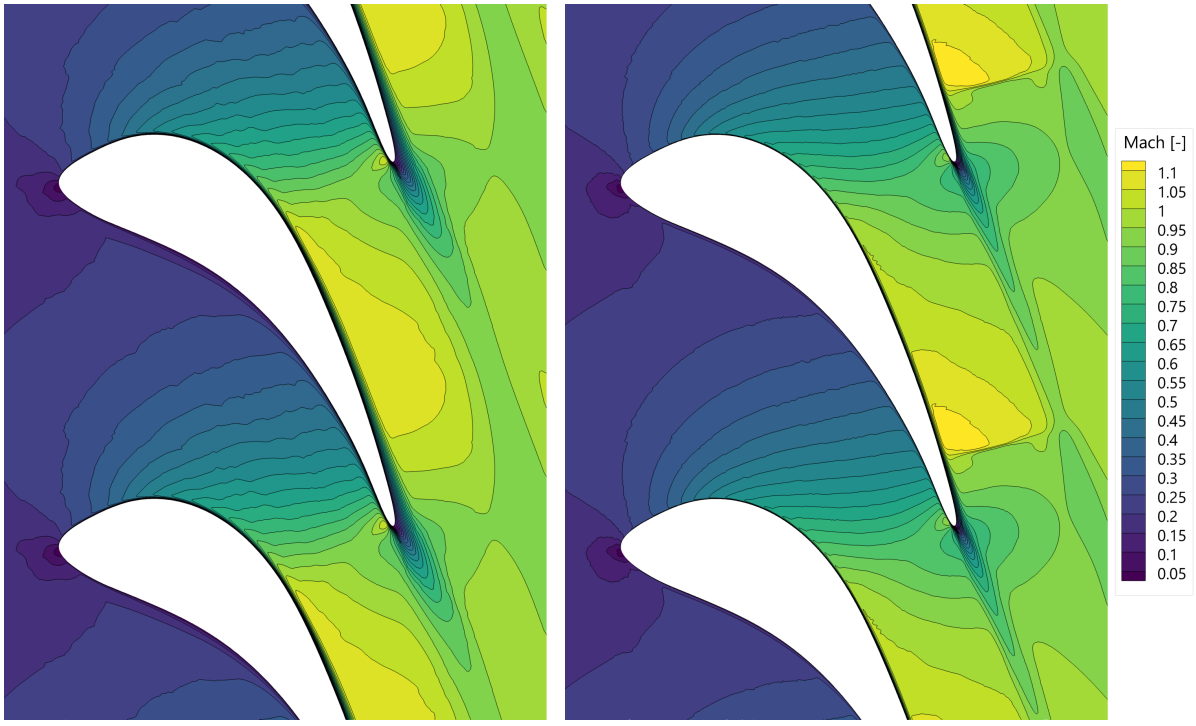
(d) Optimized, off-design.

Figure 6.9: Zoom onto the trailing edge entropy contours for nominal point (a, b) and off-design (c, d), for baseline (a, c) and optimized (b, d) blade. Note the reduced levels of entropy for the optimized blade compared to baseline and off-design compared to nominal.

Figure 6.9 shows the entropy contour plots, zoomed onto the TE of the blades. It displays a reduced entropy in the boundary layer at the pressure side TE for the optimized design for both operating conditions. The optimized design shows a reduction of entropy along the blade surface, most notably around the TE at both operating conditions. The profile losses at the suction side are lowered due to smoother flow accelerations as a result of the smaller LE radius, which is detailed in Figure 6.11 in the following section.

In Figure 6.10, sub-figure (a) presents the Mach number contours of the baseline blade of the mesh employed during optimization and (b) of the fine mesh. The profile losses of the fine mesh are less than those of the optimization mesh, evidenced by the reduced boundary layer size at the suction side of the TE.

The utilization of a coarser mesh such as the one employed during optimization incurs two main effects on the boundary layer, as examined in section 2.2.3. On the one hand, the incomplete resolution of viscous effects such as turbulence production *decreases* the boundary layer thickness. On the other, the increased numerical diffusion effectively increases the viscosity of the flow and hence *increases* the boundary layer thickness, as demonstrated by the larger boundary layer at the suction side at TE in Figure 6.10a as compared to in Figure 6.10b. The overestimation of viscous losses in the boundary layer corresponds with increased s_{gen}. The Mach contours of both meshes in Figure 6.10 also display an emerging TE shock, however not strong enough to impinge on the suction side of the blade, hence no boundary layer losses resulting from shock-induced separation.



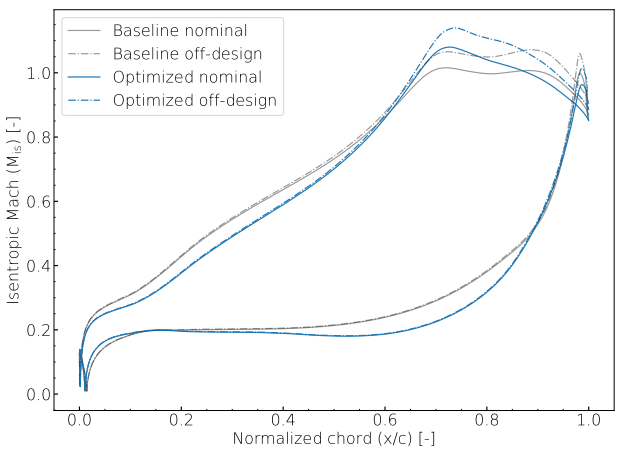
(a) Optimization mesh.

(b) Fine mesh.

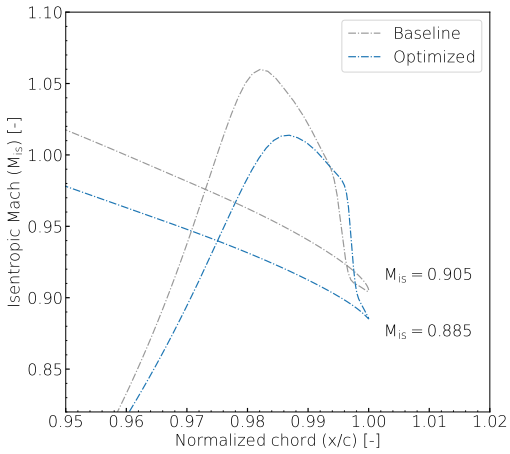
Figure 6.10: Mach contours of the baseline blade at off-design operating condition between the mesh employed during optimization (a) and the fine mesh (b).

6.4.2. SURFACE

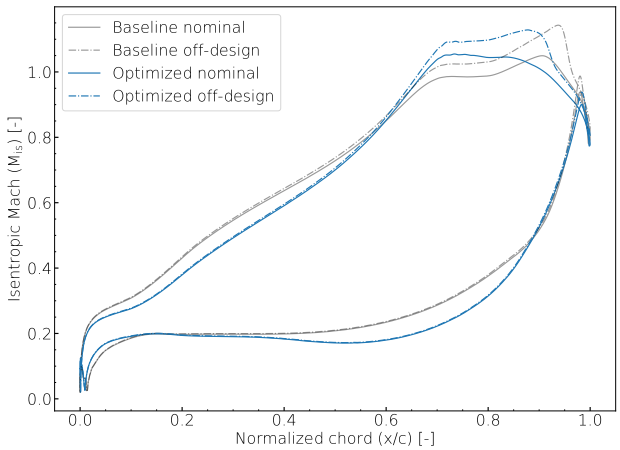
Figure 6.11 shows the M_{is} along the surface of the blade for the baseline and optimized designs. M_{is} plots provide an accurate non-dimensional indication of blade loading. Two main observations are visible in comparing the baseline with the optimizes blade. The first is that the blade loading is roughly constant, which is a result of the outlet flow angle constraint. The second is that the optimized blade is more aft-loaded; the difference in M_{is} between suction and pressure side is larger for the optimized blade as compared to baseline. The optimized blade displays a smoother flow acceleration at the LE of the suction side, which postpones boundary layer transition and hence reduces profile losses downstream of the blade surface. The upstream pressure side M_{is} is largely unaltered, however a slight recompression is noticeable at mid-chord between $0.3 \leq x/c \leq 0.55$ in Figure 6.11a and 6.11c. This is due to the increased curvature of the pressure side surface of the optimized design. An explanation for this optimized pressure side shape can be that the fluid-dynamic losses downstream in the boundary layer due to this recompression are limited, in such a way that it is negligible to the optimizer. This is because the boundary flow downstream passes through only a short section of adverse pressure at the TE.



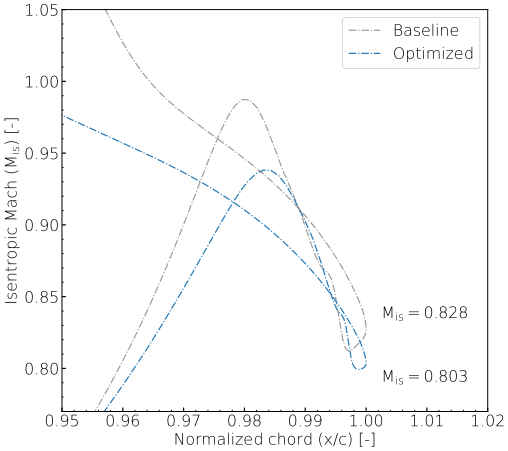
(a) Isentropic Mach distribution, optimization mesh.



(b) Trailing edge detail. Off-design, optimization mesh.



(c) Isentropic Mach distribution, fine mesh.



(d) Trailing edge detail. Off-design, fine mesh.

Figure 6.11: Isentropic Mach number M_{is} for baseline and optimized blade along the surface for the complete blade (a) and zoom onto the trailing edge for the off-design operating condition.

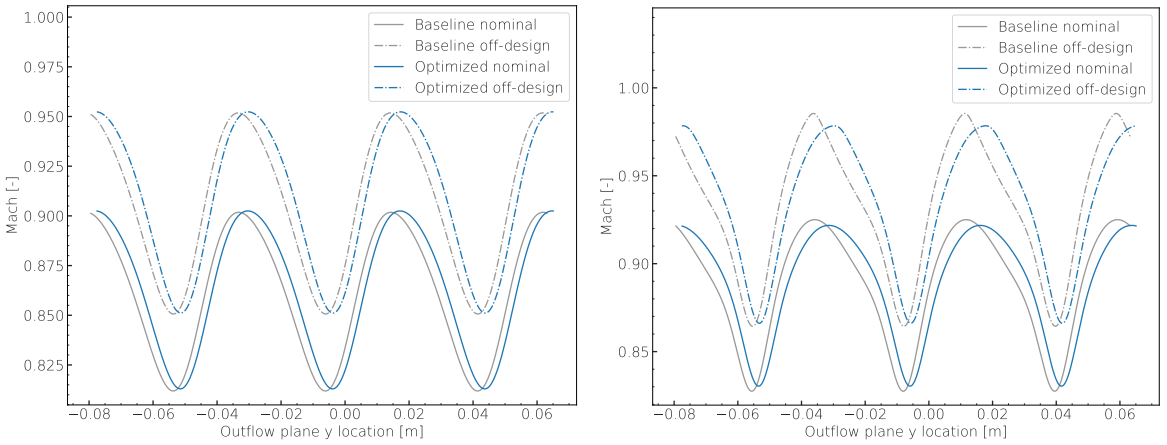
Figure 6.11b and 6.11d detail the TE M_{is} for the off-design condition for the optimization and fine mesh respectively. The TE exit M_{is} is decreased for both operating conditions; a reduction of 2.21% for M_{is} is observed for the optimized design of the optimization mesh and a reduction of 3.01% for the fine mesh. At nominal operating condition, this reduction is 1.84% and 2.57% for the optimization and fine mesh respectively. This reduced M_{is} corresponds with an increased base pressure, which suggests a reduction in s_{gen} due to reduced mixing losses in the wake downstream of the TE. The flow approaching the TE on the pressure side accelerates strongly before leaving the blade. These accelerations are reduced for the optimized blade. This is due to the smaller TE radius of the optimized blade, which allows for a smoother exit of the boundary layer flow into the wake. The larger increase of base pressure for the off-design point compared to nominal agrees with the larger s_{gen} reduction at off-design as shown in Table 6.2.

Observing the mesh employed during optimization in Figure 6.11a, the increase in s_{gen} resulting from the increased peak of M_{is} downstream of the suction side for the optimized blade is offset by the reduced M_{is} acceleration upstream, and at mid-chord of the pressure side.

Looking at the M_{is} distribution on the fine mesh in Figure 6.11c, a rapid deceleration is evident for the baseline blade at off-design condition at the aft suction side. A similar, yet less pronounced rapid deceleration is present in the nominal operating condition. This deceleration, and preceding acceleration of the flow is reduced for the optimized blade at both operating conditions, reducing profile losses. The larger reduction in M_{is} peak and reduced flow decelerations of the fine mesh as compared to the mesh used during optimization agrees with the larger s_{gen} reduction of the fine mesh.

6.4.3. OUTLET FLOW

Figure 6.12 presents the Mach number profile along the outlet boundary for the mesh employed during optimization (a) and fine mesh (b). The optimized design yields a decreased spread of the kinematic quantities at the outlet boundary by -4.91% for the nominal operating condition and -2.45% for off-design, as presented in the statistics in Table 6.5. The decreased spatial spread of the Mach number at the outlet of the optimized design indicates an improved flow uniformity downstream of the optimized blade. A more uniform outlet flow indicates less flow mixing in the wake, lowering s_{gen} and reducing the fluid-dynamic losses in the cascade downstream.



(a) Optimization mesh.

(b) Fine mesh.

Figure 6.12: Mach number profile along the outlet boundary of baseline versus optimized blade for the mesh employed during optimization (a) and the fine mesh (b).

The main contribution to the reduction of s_{gen} of the optimization problem is the reduction of losses in the boundary layer of the blade. This is indicated by the increase in the spread of the kinematics at the outlet of the optimization mesh and the negligible losses due to shocks, as shown by the absence of shocks in the flow of the baseline blade in Figure 6.10a and entropy contours of the baseline blade in Figure 6.8a and 6.8c.

Mesh	Operating condition	Baseline		Optimized		Δ (%)	
		μ	σ	μ	σ	μ	σ
Optimization	Nominal	0.861	3.15×10^{-2}	0.863	3.15×10^{-2}	0.21	0.12
	Off-design	0.906	3.53×10^{-2}	0.908	3.56×10^{-2}	0.24	0.91
Fine	Nominal	0.888	3.17×10^{-2}	0.889	3.02×10^{-2}	0.11	-4.91
	Off-design	0.933	3.76×10^{-2}	0.935	3.67×10^{-2}	0.17	-2.45

Table 6.5: Mach number statistics at the outlet for the optimization and fine mesh. Arithmetic mean (μ) and standard deviation (σ) of nominal and off-design conditions.

The arithmetic mean of the Mach numbers at the outlet plane of the mesh employed during optimization is lower than that of the fine mesh. This is a result of increased viscosity originating from numerical diffusion due to the coarseness of mesh. The optimized design shows a positive y-shift of the Mach distribution compared to the baseline design, caused by the decreased stagger angle by -2.77% of the optimized blade. This increases the y location of the TE and hence translates the wake in positive y direction. The minima of the Mach number distributions of the optimized blade are consistently larger than the initial blade counterparts. This is due to the reduced size of the wake of the optimized blade, hence increasing the mean Mach number in the section at the outlet which is downstream of the wake.

The non-sinusoidal shape of the Mach number at the outlet plane on the fine mesh is related to the severity of the suction side shock. Deviating from this sinusoidal shape increases Mach disparities downstream of the blade passage, increasing fluid-dynamic losses due to increased diffusion. Observing the outlet kinematics of the fine mesh in Figure 6.12b, its *sinusoidalness* increases when moving from baseline to optimized and from off-design to nominal. Both situations are characterized by a reduction in shock strength. The same is observed when comparing the results of the optimization mesh with the fine mesh – the sinusoidalness of the Mach distribution increases from fine mesh to optimization mesh, where the shock reduces. Thus, an increase in suction side shock strength is accompanied with losses downstream of the blade passage, increasing s_{gen}.

Important to mention is that the Mach numbers presented in this section are a result of solving the steady state RANS equations, whereas vortex shedding is an unsteady process. This can be a cause of disparities in flow quantities which are a result of turbulent processes such as flow mixing, such as the Mach numbers at the outlet. A validation of the optimization results using unsteady methods such as URANS or LES or by comparison to experiments would shed light on the inaccuracies. However, this particular study is deemed outside of the scope of this work.

6.5. CHALLENGES AND PERSPECTIVES IN OPTIMIZATION

A number of other optimizers have been considered alongside NSGA2, see the available optimization algorithms in Table 4.1 for an overview. Most of which encountered difficulties in returning an improved design or traversing the design space in a desirable manner. Directions for further investigation into this behaviour are towards, among others, the tuning of optimizer-specific parameters, errors stemming from assumptions in the flow solver or discretization, and shape parametrization. This section describes some challenges experienced, and options to mitigate these, in the numerical optimization of the two-dimensional Aachen stator blade.

Figure 6.13 presents the optimization history of the two-dimensional Aachen turbine blade on the nominal operating condition using EGO. Visible are the 50 evaluations in the design of experiments phase, after which the optimizer focuses on a location in design space estimated to contain a minimum of s_{gen}. However, no improved design is found.

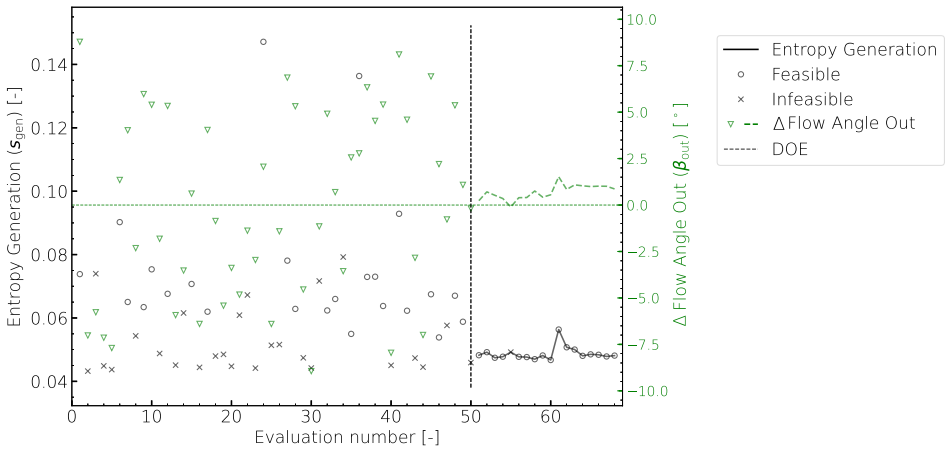


Figure 6.13: Single-point optimization history overlaid with constraint values of flow angle at the outlet β_{out}. Optimized using Efficient Global Optimization.

This behaviour can be a result of the stochasticity of the problem response, or termed here as *numerical noise*. This can be introduced by simulation errors such as inconsistent mesh topology or non-converged flow fields [Azab and Ollivier-Gooch 2012; Venter and Sobieszczanski-Sobieski 2004]. Another potential source, however not observed in the flow results here, are strong shocks in the flow domain. If a sufficiently large and random variability exists in the s_{gen} between similar designs, it is impossible to make a prediction of the s_{gen} of designs in the vicinity, when the output estimates of the model must *intercept* all sampled points exactly.

A few options to mitigate this settlement on non-optimal designs are available in the code. One is to obtain a different set of available sample points by expanding the number of evaluations in the DoE, or changing the space sampling method from Latin hypercube sampling to random or full factorial. Another approach is to add a *nugget* to the spatial correlation matrix of the Kriging model, which alters the surrogate model from

interpolating to regressing over the sampled points, as described in the section on Kriging in appendix B.2. This smoothens-out the predictions of the surrogate model and hence makes it more robust to numerical noise in the data points. In addition, the geometry parametrization may be simplified, simplifying the design landscape.

Figure 6.14 presents the s_{gen} history of the single–point optimization using the DE optimizer. The optimization is terminated at an arbitrary evaluation number due to inability of convergence upon an improved design. The constraints are enforced through the penalty method, which is reflected by the increasingly larger ratio of feasible solutions towards later evaluation numbers. However, no feasible improved design is returned. The numerical noise level in the response value may be a contributing factor in DE's inability to provide an improved design. Options to fine-tune DE to converge on improved design are to relax the penalty factor of the constraints, favouring a reduction in s_{gen} over satisfying the constraints. Or reducing the probability of mutation, favouring exploitation of known performant regions in the design space. However, tuning these parameters is deemed outside of the scope of this work.

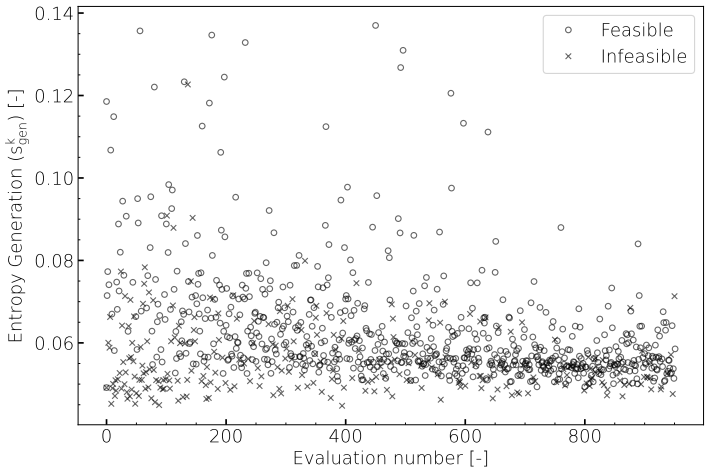


Figure 6.14: Single point optimization history using the Differential Evolution optimizer.

An approach to increasing the likelihood of a *successful* optimization is to reduce the numerical noise of the problem by only varying the n design variables with smallest numerical noise. The degree of numerical noise of a design variable can be estimated by analysis of variance (ANOVA) statistical models. These procedures estimate the variance in the response value through varying the design variables. By doing this, the certainty in the response value is increased, benefitting the effectiveness of the optimizer. Appendix F elaborates on the approach for the calculation of the noise variance.

Figure 6.15 presents the s_{gen} history of the same two–point optimization using the SLSQP optimizer, with equal weights of the operating points. Optimization using two different kinds of design vectors are shown. One using all described design variables as in this work, and one in which only the ten *least-noisy*, or with lowest variance in the response noise, design variables. For each kind of design vector, 20 baseline designs are initialized using LHS sampling between ±10% bounds and optimized. The median μ and spread $\mu \pm 1\sigma$ of these 20 optimizations is plotted for the two kings of design vectors. Apparent is the larger $\mathcal{J}$ decrease for the optimization using the least-noisy design vector, aswell as the larger convergence rate and lower spread. Noteworthy is that no design evaluated during the optimizations performed on the complete design vector satisfy the constraint on β_{out}.

Figure 6.16 shows the optimization history of three variants of EGO. The left figure depicts baseline EGO. The middle depicts EGO by incorporating a nugget in the Kriging surrogate model. The right figure pertains to baseline EGO applied to the same *least-noisy* design vector as in Figure 6.15. The nugget decreases the fluctuation in $\mathcal{J}$ during the first 20 iterations after the DoE. These fluctuations are non-existent when optimizing the problem with the *least-noisy* design vector.

The reduced fluctuations in $\mathcal{J}$ for the two differing EGO variants are a result of the increased prediction accuracy of the surrogate model, as shown in Figure 6.17. Figure 6.17 shows the prediction error for the same three variants of EGO. The prediction error magnitude is inversely correlated with the fluctuation of $\mathcal{J}$.

The figures 6.16 and 6.17 show that adding a nugget and selecting design variables on the basis of numerical noise are two effective approaches towards a successful optimization. By respectively making the optimization process more robust to or mitigate numerical noise of the problem.

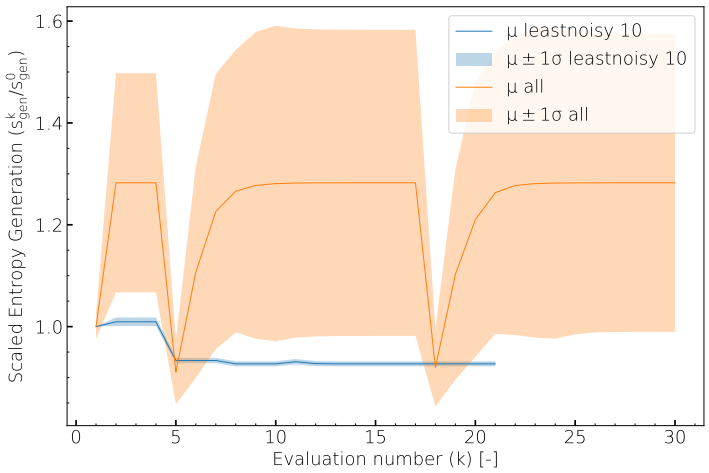


Figure 6.15: Two-point optimization history using the Sequential-Least-Squares Quadratic Programming (SLSQP) optimizer. Median μ and spread ($\mu\pm$1-standard deviation σ) of 20 LHS-sampled designs are displayed.

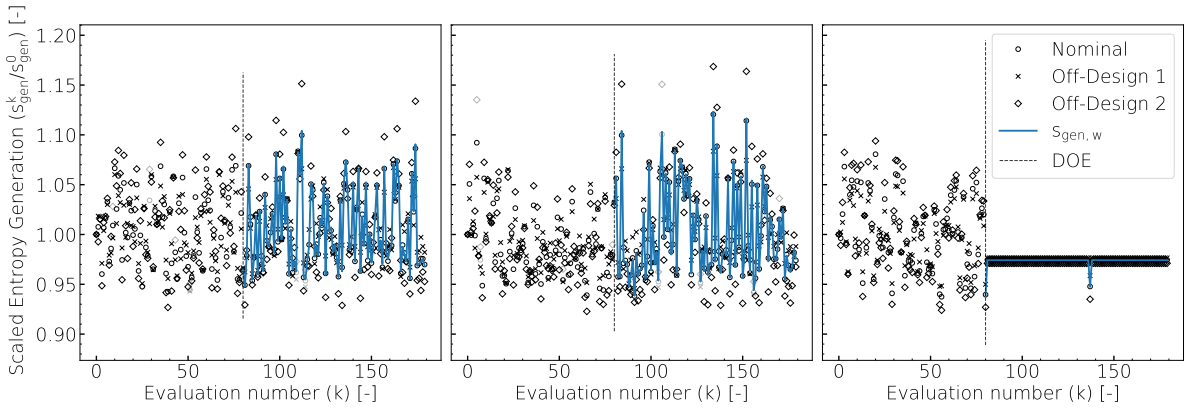


Figure 6.16: Optimization history for three variants of Efficient Global Optimization (EGO); from left to right respectively baseline EGO, EGO with nugget in the Kriging surrogate model, and baseline EGO with a *low-noise*-selected design vector.

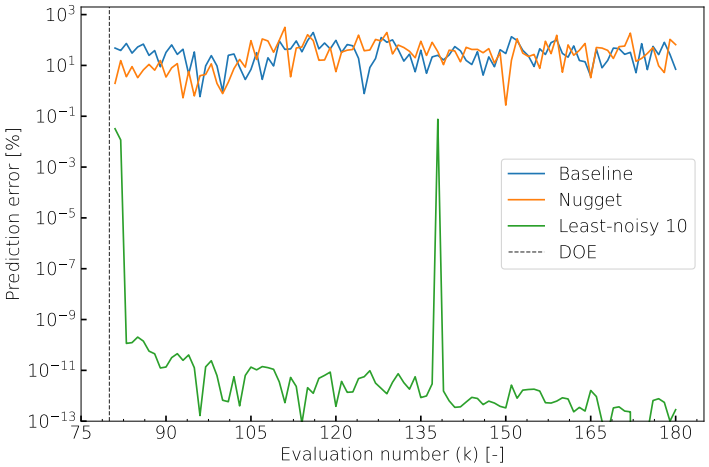


Figure 6.17: Prediction error for three variants of Efficient Global Optimization (EGO); respectively baseline EGO, EGO with nugget in the Kriging surrogate model and baseline EGO with a *low-noise*-selected design vector.

7

CONCLUSION AND RECOMMENDATIONS

This final chapter finalizes the thesis report by providing conclusions and recommendations. The scientific and technical implications of the two-point fluid-dynamic optimization of the two-dimensional Aachen turbine blade within the framework as described in this work are presented in light of the research questions and objective. Section 7.1 summarizes the conclusions, which is followed by the recommendations for future work in section 7.2.

7.1. CONCLUSION

The objective of this work is to develop an optimization library capable of multi-point fluid-dynamic shape optimization for the design of turbomachinery blades, by analyzing the components requiring implementation and performing a multi-point optimization case study. The novelty of this work lies in that this framework enables the multi-point, multi-zone fluid-dynamic optimization of turbomachine blades by utilizing local and global optimization algorithms.

Throughout the development of the code framework, a few considerations were dominant; to create a tool capable of optimization of turbomachine problems using global and local optimization approaches, being extensible in order to easily change the code as new developments happen and striving for parallel computing efficiency.

The key contributions of this work are summarized as follows:

1. The feasibility and effectiveness of ParaBlade as a multi-point design optimization framework is demonstrated by the fluid-dynamic shape optimization of stator blade on two operating conditions. The fluid-dynamic entropy generation of a two-dimensional Aachen turbine blade is successfully reduced by an average of 7.57%, while satisfying the imposed constraint on flow turning. The NSGA2 optimization algorithm required around 2700 flow evaluations to converge onto the optimized blade.

2. Three key code modules are implemented in this work in order to enable the optimization of turbomachine shapes using global optimization algorithms, realizing their advantages over local optimizers in design problems with multiple local optima. These are the single- and multi-point optimization classes and the integrated meshing module. The code structure enables the application and straightforward extension of this framework to the automated shape design of many–point and multi–zone turbomachinery problems. To the present knowledge, this work forms the first implementation of such a tool supporting integrated meshing in an open-source fashion.

3. The literature review highlights a trend towards hybrid and surrogate-assisted optimization approaches in multi–point turbomachine optimization studies. However, no improved design is obtained by using the Efficient Global Optimizer in this work, potentially due to the stochasticity of the objective function value. Further problem tuning is required to obtain sensible results, for which options are available in the code.

4. Two verification methods are performed in this work. The first is a grid convergence study to quantify the modeling error stemming from the mesh discretization. A grid is chosen which performs best on a

trade-off loss metric defined by the computational cost and entropy generation error for the baseline blade with respect to a fine mesh. The second is a design vector stationarity study of the optimized design which demonstrates a reduction in the norm of the gradient of the optimized design by 18% with respect to baseline. This verifies that the optimized design constitutes an improved design in a numerical sense.

5. In context of the mesh discretization error study, changes in blade shape have a small effect on the discretization error, indicating a low sensitivity of exact objective function on changes in shape. Supporting the potential argumentation on the basis of *changes* between blade designs during optimization.

6. A loss analysis shows the major sources of fluid-dynamic loss are at the suction side towards the trailing edge and in the wake of the blade. The upstream shift of peak isentropic Mach along the pressure side allows for a smoother recompression of the boundary layer flow downstream towards the trailing edge. Reducing the *viscous dissipation losses in the boundary layer* in the mesh employed during optimization and flattening the flow velocity peak when simulating using a fine mesh. The increased base pressure and reduced spread of outlet boundary kinematics of the optimized blade agree with the reduction in s_{gen} through a reduction in *mixing losses downstream of the trailing edge*. The major sources of entropy in the computational domain agree with theory on thermodynamic losses.

7. The fluid-dynamic simulations show that the flow simulation result is impacted significantly by the mesh discretization. A significant numerical diffusion is present when using the optimization mesh, leading to over-valuing of viscous effects, resulting in an overestimation of entropy generation.

8. The low-fidelity mesh fails to accurately predict the correct entropy generation. However, optimization yields an improved design when evaluated on a fine mesh. Therefore, the results demonstrate that the predictive capability of the flow simulation with a low-fidelity mesh is adequate to design turbomachinery blades for multiple operating conditions simultaneously.

9. The adjoint gradient validation and sensitivity magnitude results demonstrate that the design variables affecting the blade geometry near the trailing edge have the largest effect on entropy generation of the baseline and optimized blade.

10. The magnitude of sensitivity of the entropy generation of the optimized blade shows an expected reduction with respect to baseline. However, the non-zero magnitude of the optimized design sensitivity demonstrates that the optimized design has potential to improve further.

11. Numerical optimization is challenging, demonstrated by the difficulties experienced in yielding improved designs by a number of optimization algorithms.

12. An approach to increasing the probability of an improved design by fluid-dynamic shape optimization is to disregard the most numerically noisy design variables.

13. Disregarding design variables which cause a large numerical noise in the response, increases the likelihood of successful optimization.

7.2. RECOMMENDATIONS

The work performed in this thesis constitutes a step towards automated end-to-end open-source turbomachinery optimization capabilities. However, due to the limited time and resources associated with a master's thesis, the framework in its current form represents by no means a feature-complete tool. A few topics on which future research and improvement can be focused are identified in the following.

1. Extending the current two-dimensional integrated meshing module in order to support three-dimensional geometries. Coding-wise, this can readily be implemented because the required code modules may be written analogously to the current modules. A focus can be applied onto the scaling performance of the mesh generator, allowing to simulate problems with larger mesh element counts.

2. Optimizing three-dimensional turbomachine blades and/or multi-zone and multi-disciplinary such as structural or heat-transfer problems.

3. Applying other optimization algorithms implemented by the code framework. Surrogate-assisted optimization algorithms such as Efficient Global Optimization, extended by gradient information, promise efficient optimization approaches for fluid-dynamic turbomachinery problems.

4. Integrating the code into SU2's Python API. This integration draws on open-source collaboration to further the capabilities and ensure that a wider audience can benefit from the work

5. Optimizing problems constituting true competing objectives. Capturing the tradeoff in optimizing for competing multi-point objectives, leveraging the supporting capabilities to multi-criterion decision-making.

6. Implementing SU2's *multi-objective adjoint* capability for reducing computational time of generating the reduced gradient vector of objective function and constraints (for application in Gradient-Enhanced Kriging). See section 3.4 in Kline [2017] for an elaboration on the multi-objective adjoint approach.

BIBLIOGRAPHY

Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 1 2020. ISSN 0066-4189. doi:10.1146/annurev-fluid-010719-060214.

B. M. Adams, W. J. Bohnhoff, et al. Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.14 User's Manual. Technical report, Sandia National Laboratories, 5 2021.

R. Agromayor, N. Anand, et al. A Unified Geometry Parametrization Method for Turbomachinery Blades. *CAD Computer Aided Design*, 133:102987, 4 2021. doi:10.1016/j.cad.2020.102987.

J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large-data visualization. *Visualization Handbook*, pages 717–731, 1 2005. doi:10.1016/B978-012387582-2/50038-1.

M. H. Aissa and T. Verstraete. Metamodel-Assisted Multidisciplinary Design Optimization of a Radial Compressor. *International Journal of Turbomachinery, Propulsion and Power*, 4(4):35, 11 2019. doi:10.3390/ijtpp4040035.

M. H. Aissa, R. Maffuli, et al. Optimisation of a turbine inlet guide vane by gradient-based and metamodel-assisted methods. *International Journal of Computational Fluid Dynamics*, 33(6-7):302–316, 8 2019. doi:10.1080/10618562.2019.1683168.

M. F. Akorede, H. Hizam, and E. Pouresmaeil. Distributed energy resources and benefits to the environment. *Renewable and Sustainable Energy Reviews*, 14:724–734, 2 2010. doi:10.1016/J.RSER.2009.10.025.

T. Albring, M. Sagebaum, and N. R. Gauger. A consistent and robust discrete adjoint solver for the su2 framework - validation and application. In *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, volume 132, pages 77–86, 2016a. doi:10.1007/978-3-319-27279-5_7/FIGURES/7.

T. Albring, M. Sagebaum, and N. R. Gauger. Efficient aerodynamic design using the discrete adjoint method in SU2. In *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016b. doi:10.2514/6.2016-3518.

T. A. Albring, M. Sagebaum, and N. R. Gauger. Development of a Consistent Discrete Adjoint Solver in an Evolving Aerodynamic Design Framework. In *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2015. doi:10.2514/6.2015-3240.

N. Anand and R. Agromayor. ParaBlade: v1.0, 6 2020. doi:10.5281/zenodo.3894778.

N. Anand, S. Vitale Propulsion, and M. Pini Propulsion. Assessment of FFD and CAD-based shape parametrization methods for adjoint-based turbomachinery shape optimization. In *Proceedings of Montreal 2018 Global Power and Propulsion Forum*, Montreal, 5 2018. doi:10.5281/zenodo.1344595.

N. Anand, S. Vitale, et al. Design methodology for supersonic radial vanes operating in nonideal flow conditions. *Journal of Engineering for Gas Turbines and Power*, 141, 2 2019. doi:10.1115/1.4040182.

N. Anand, A. Rubino, et al. Adjoint-based aeroelastic design optimization using a harmonic balance method. In *ASME Turbo Expo 2020*, 9 2020. doi:10.1115/GT2020-16208.

I. Andrić, A. Pina, et al. Techno-economic analysis of waste heat recovery with orc from fluctuating industrial sources. *Energy Procedia*, 129:503–510, 2017. doi:10.1016/J.EGYPRO.2017.09.170.

A. Arnone. Viscous analysis of three-dimensional rotor flow using a multigrid method. *Journal of Turbomachinery*, 116:435–445, 7 1994. doi:10.1115/1.2929430.

ASME. *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*. The American Society of Mechanical Engineers, New York, 9 2009. ISBN 9780791832097.

M. B. Azab and C. Ollivier-Gooch. High order aerodynamic optimization using new hybrid sequential quadratic programing-particle swarm intelligence technique. *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2012. doi:10.2514/6.2012-730.

J. Backhaus, M. Aulich, et al. Gradient enhanced surrogate models based on adjoint CFD methods for the design of a counter rotating turbofan. In *Proceedings of the ASME Turbo Expo*, volume 8, pages 2319–2329. American Society of Mechanical Engineers Digital Collection, 7 2012. doi:10.1115/GT2012-69706.

J. Backhaus, A. Schmitz, et al. Application of an algorithmically differentiated turbomachinery flow solver to the optimization of a fan stage. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization, 2017*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2017. doi:10.2514/6.2017-3997.

S. A. I. Bellary, A. Samad, et al. A comparative study of kriging variants for the optimization of a turbomachinery system. *Engineering with Computers*, 32(1):49–59, 1 2016. doi:10.1007/s00366-015-0398-x.

A. Benítez-Hidalgo, A. J. Nebro, et al. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:100598, 12 2019. doi:10.1016/j.swevo.2019.100598.

F. Biscani and D. Izzo. A parallel global multiobjective framework for optimization: pagmo. *Journal of Open Source Software*, 5(53):2338, 2020. doi:10.21105/joss.02338.

D. Bonaiuti and M. Zangeneh. On the coupling of inverse design and optimization techniques for the multiobjective, multipoint design of turbomachinery blades. *Journal of Turbomachinery*, 131(2), 4 2009. doi:10.1115/1.2950065.

M. A. Bouhlel and J. R. Martins. Gradient-enhanced kriging for high-dimensional problems. *Engineering with Computers*, 35(1):157–173, 1 2019. doi:10.1007/s00366-018-0590-x.

M. A. Bouhlel, J. T. Hwang, et al. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, page 102662, 2019. doi:10.1016/j.advengsoft.2019.03.005.

L. Bui, D. Essam, et al. Performance analysis of evolutionary multi-objective optimization methods in noisy environments. In *Proceedings of the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pages 29–39, 12 2004. ISBN 0646443402.

F. Casella, T. Mathijssen, et al. Dynamic Modeling of Organic Rankine Cycle Power Systems. *Journal of Engineering for Gas Turbines and Power*, 135(4), 4 2013. doi:10.1115/1.4023120.

L. A. Catalano, D. Quagliarella, and P. L. Vitagliano. Aerodynamic shape design using hybrid evolutionary computing and multigrid-aided finite-difference evaluation of flow sensitivities. *Engineering Computations*, 32:178–210, 4 2015. doi:10.1108/EC-02-2013-0058.

C. Chahine, T. Verstraete, and L. He. A comparative study of coupled and decoupled fan flutter prediction methods under variation of mass ratio and blade stiffness. *Journal of Fluids and Structures*, 85:110–125, 2 2019. doi:10.1016/J.JFLUIDSTRUCTS.2018.12.009.

V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making: Theory and Methodology*. Courier Dover Publications, New York, 2008. ISBN 0486462897.

A. Châtel, T. Verstraete, and G. Coussement. Multipoint optimization of an axial turbine cascade using a hybrid algorithm. *Journal of Turbomachinery*, pages 1–15, 2 2020. doi:10.1115/1.4046231.

Y. Chen, J. Wang, and P. D. Lund. Thermodynamic performance analysis and multi-criteria optimization of a hybrid combined heat and power system coupled with geothermal energy. *Energy Conversion and Management*, 210, 4 2020. doi:10.1016/J.ENCONMAN.2020.112741.

O. Chernukhin and D. W. Zingg. Multimodality and global optimization in aerodynamic design. *AIAA Journal*, 51:1342–1354, 5 2013. doi:10.2514/1.J051835.

C. A. Coello Coello. Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17(4):319–346, 2000. doi:10.1080/02630250008970288.

C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 6 2004. doi:10.1109/TEVC.2004.826067.

P. Colonna, E. Casati, et al. Organic rankine cycle power systems: From the concept to current technology, applications, and an outlook to the future. *Journal of Engineering for Gas Turbines and Power*, 137, 10 2015. doi:10.1115/1.4029884/373802.

S. E. Cox, R. T. Haftka, et al. A Comparison of Global Optimization Methods for the Design of a High-speed Civil Transport. *Journal of Global Optimization*, 21(4):415–432, 2001. doi:10.1023/A:1012782825166.

L. Cozzi, T. Gül, S. Bouckaert, et al. Net zero by 2050. Technical report, International Energy Agency (IEA), Paris, 10 2021. URL https://www.iea.org/reports/net-zero-by-2050.

L. Da, L. Hanan, et al. Optimization of a transonic axial-flow compressor under inlet total pressure distortion to enhance aerodynamic performance. *Engineering Applications of Computational Fluid Mechanics*, 14(1): 1002–1022, 1 2020. doi:10.1080/19942060.2020.1792348.

I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997. doi:10.1007/BF01197559.

K. Deb, A. Pratap, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 4 2002. doi:10.1109/4235.996017.

B. H. Dennis, I. N. Egorov, et al. Multi-objective optimization of turbomachinery cascades for minimum loss, maximum loading, and maximum gap-to-chord ratio. *International Journal of Turbo and Jet Engines*, 18 (3):201–210, 9 2001. doi:10.1515/TJJ.2001.18.3.201.

J. D. Denton. Loss mechanisms in turbomachines. *ASME Journal of Turbomachinery*, 2, 2 1993. doi:10.1115/93-GT-435.

S. Dixon and C. Hall. Chapter 1 - introduction: Basic principles. In *Fluid Mechanics and Thermodynamics of Turbomachinery (Seventh Edition)*. Butterworth-Heinemann, Boston, seventh edition edition, 2014a. doi:10.1016/B978-0-12-415954-9.00003-6.

S. Dixon and C. Hall. Chapter 3 - two-dimensional cascades. In *Fluid Mechanics and Thermodynamics of Turbomachinery (Seventh Edition)*, pages 69–117. Butterworth-Heinemann, Boston, seventh edition edition, 2014b. doi:10.1016/B978-0-12-415954-9.00003-6.

G. Droandi and G. Gibertini. Aerodynamic blade design with multi-objective optimization for a tiltrotor aircraft. *Aircraft Engineering and Aerospace Technology*, 87:19–29, 1 2015. doi:10.1108/AEAT-01-2013-0005.

R. P. Dwight. Robust Mesh Deformation using the Linear Elasticity Equations. In *Computational Fluid Dynamics 2006*, pages 401–406. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-540-92779-2_62.

R. Eberhart and J. Kennedy. New optimizer using particle swarm theory. In *Proceedings of the International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE, 1995. doi:10.1109/mhs.1995.494215.

L. Eça and M. Hoekstra. A Verification Exercise For Two 2-D Steady Incompressible Turbulent Flows. In *European Congress on Computational Methods in Applied Sciences and Engineering ECCOMAS*, Jyvaskyla, 7 2004. URL https://www.mit.jyu.fi/eccomas2004/proceedings/pdf/148.pdf.

L. Eça, G. B. Vaz, et al. Verification of calculations of the potential flow around two-dimensional foils. *AIAA Journal*, 42(12):2401–2407, 5 2004. doi:10.2514/1.782.

T. D. Economon, F. Palacios, et al. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 12 2016. doi:10.2514/1.J053813.

A. Engels-Putzka, J. Backhaus, and C. Frey. Forced response sensitivity analysis using an adjoint harmonic balance solver. *Journal of Turbomachinery*, 141, 3 2019. doi:10.1115/1.4041700/475746.

European Commission and Directorate-General for Communications Networks, Content and Technology, B. Beckert, et al. *The economic and social impact of software & services on competitiveness and innovation : final report.* Publications Office, 2017. doi:doi/10.2759/949874.

European Commission and Directorate-General for Mobility and Transport and Directorate-General for Research and Innovation. *Flightpath 2050 : Europe's vision for aviation: maintaining global leadership and serving society's needs.* Publications Office, 2011. doi:doi/10.2777/50266.

Eurostat. Electricity prices for non-household consumers. https://ec.europa.eu/eurostat/databrowser/view/nrg_pc_205, 2022. data retrieved from NRG_PC_205.

Exergy. Radial outflow turbine working scheme. https://www.exergy-orc.com/technology/radial-outflow-turbine, 2017. Accessed: 02-05-22.

M. Eyidogan, F. C. Kilic, et al. Investigation of organic rankine cycle (orc) technologies in turkey from the technical and economic point of view. *Renewable and Sustainable Energy Reviews*, 58:885–895, 5 2016. doi:10.1016/J.RSER.2015.12.158.

J. R. Figueira, A. Liefooghe, et al. A Parallel Multiple Reference Point Approach for Multi-objective Optimization. Research Report RR-6938, INRIA, 2009.

D. E. Finkel. Global Optimization With The Direct Algorithm. Technical report, NCSU, 2 2005. URL https://www.lib.ncsu.edu/resolver/1840.16/4738.

J. Galindo, J. R. Serrano, et al. Experiments and modelling of surge in small centrifugal compressor for automotive engines. *Experimental Thermal and Fluid Science*, 32:818–826, 1 2008. doi:10.1016/J.EXPTHERMFLUSCI.2007.10.001.

H. E. Gallus, C. A. Poensgen, and J. Zeschky. Three-Dimensional Unsteady Flow in a Single Stage Axial-Flow Turbine and Compressor. In *Unsteady Aerodynamics, Aeroacoustics, and Aeroelasticity of Turbomachines and Propellers*, pages 487–505. Springer New York, 1993. doi:10.1007/978-1-4613-9341-2_24.

GE Aviation. Ge9x turbofan engine cutaway. https://www.geaviation.com, 2018. Accessed: 02-05-22.

C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 9 2009. doi:10.1002/NME.2579.

A. Ghidoni. Algorithm to create periodic data structure (su2 format), 9 2017. URL https://github.com/su2code/MeshTools/blob/master/SU2_PER/su2_periodic.f90.

P. Gill, W. Murray, and M. Saunders. User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming. Technical report, Stanford University, Stanford, 6 2008.

P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 7 2002. doi:10.1137/S1052623499350013.

D. Golovin and Q. R. Zhang. Random hypervolume scalarizations for provable multi-objective black box optimization. *37th International Conference on Machine Learning, ICML 2020*, PartF168147-15:11030–11039, 6 2020. doi:10.48550/arxiv.2006.04655.

Y. Haimes and D. Li. Hierarchical Multiobjective Analysis for Large-Scale Systems: Current Status. *IFAC Proceedings Volumes*, 20(9):27–39, 8 1987. doi:10.1016/s1474-6670(17)55679-7.

P. He, C. A. Mader, et al. Aerothermal optimization of a ribbed u-bend cooling channel using the adjoint method. *International Journal of Heat and Mass Transfer*, 140:152–172, 9 2019. doi:10.1016/J.IJHEATMASSTRANSFER.2019.05.075.

X. He and X. Zheng. Performance improvement of transonic centrifugal compressors by optimization of complex three-dimensional features. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 231(14):2723–2738, 12 2017. doi:10.1177/0954410016673395.

M. J. Heron, V. L. Hanson, and I. Ricketts. Open source and accessibility: advantages and limitations. *Journal of Interaction Science*, 1:1–10, 5 2013. doi:10.1186/2194-0827-1-2.

J. C. Hoarau, P. Cinnella, and X. Gloerfelt. Large eddy simulation of turbomachinery flows using a high-order implicit residual smoothing scheme. *Computers and Fluids*, 198:104395, 2 2020. doi:10.1016/j.compfluid.2019.104395.

M. Hoekstra and L. Eça. An Evaluation of Verification Procedures for CFD Applications. In *24th Symposium on Naval Hydrodynamics*, pages 568–587, Fukuoka, 7 2002. Marin.

J. H. J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence / John H. Holland.* Complex adaptive systems. MIT press, Cambridge, Mass., 1992. ISBN 0262581116.

H. Huang and K. Ekici. An efficient harmonic balance method for unsteady flows in cascades. *Aerospace Science and Technology*, 29:144–154, 8 2013. doi:10.1016/J.AST.2013.02.004.

L. Huyse and M. R. Lewis. Aerodynamic shape optimization of two-dimensional airfoils under uncertain operating conditions. Technical report, Institute for Computer Applications in Science and Engineering (ICASE), Hampton, 2001. URL https://dl.acm.org/doi/book/10.5555/870428.

C.-L. Hwang and A. S. M. Masud. *Multiple Objective Decision Making – Methods and Applications*, volume 164. Springer Berlin Heidelberg, 1979. doi:10.1007/978-3-642-45511-7.

G. Iaccarino, A. A. Mishra, and S. Ghili. Eigenspace perturbations for uncertainty estimation of single-point turbulence closures. *Physical Review Fluids*, 2(2):024605, 2 2017. doi:10.1103/PhysRevFluids.2.024605.

H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pages 2419–2426, 2008. doi:10.1109/CEC.2008.4631121.

A. Jameson and S. Kim. Reduction of the adjoint gradient formula in the continuous limit. *AIAA Journal*, 41, 11 2003. doi:10.2514/1.487.

A. Jameson, N. A. Pierce, and L. Martinelli. Optimum aerodynamic design using the navier-stokes equations. *35th Aerospace Sciences Meeting and Exhibit*, 1997. doi:10.2514/6.1997-101.

P. W. Jansen and R. E. Perez. Constrained structural design optimization via a parallel augmented lagrangian particle swarm optimization approach. *Computers and Structures*, 89:1352–1366, 7 2011. doi:10.1016/J.COMPSTRUC.2011.03.011.

P. W. Jansen, R. E. Perez, and J. R. Martins. Aerostructural optimization of nonplanar lifting surfaces. *https://doi.org/10.2514/1.44727*, 47:1490–1503, 5 2010. doi:10.2514/1.44727.

S. Jeong, M. Murayama, and K. Yamamoto. Efficient optimization design method using kriging model. *Journal of Aircraft*, 42(2):413–420, 5 2005. doi:10.2514/1.6386.

Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005. doi:10.1109/TEVC.2005.846356.

D. R. Jones and J. R. Martins. The direct algorithm: 25 years later. *Journal of Global Optimization*, 79:521–566, 3 2021. doi:10.1007/S10898-020-00952-6/FIGURES/27.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 10 1993. doi:10.1007/BF00941892.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998. doi:10.1023/A:1008306431147.

M. Kalsi, K. Hacker, and K. Lewis. A comprehensive robust design approach for decision trade-offs in complex systems design. *Journal of Mechanical Design, Transactions of the ASME*, 123(1):1–10, 2001. doi:10.1115/1.1334596.

M. Kaucic, M. Moradi, and M. Mirzazadeh. Portfolio optimization by improved NSGA-II and SPEA 2 based on different risk measures. *Financial Innovation*, 5(1):26, 12 2019. doi:10.1186/s40854-019-0140-6.

G. K. Kenway and J. R. R. A. Martins. Aerodynamic Shape Optimization of the CRM Configuration Including Buffet-Onset Conditions. In *54th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics (AIAA), 1 2016. doi:10.2514/6.2016-1294.

G. K. Kenway, G. J. Kennedy, and J. R. Martins. A cad-free approach to high-fidelity aerostructural optimization. *13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference 2010*, 2010. doi:10.2514/6.2010-9231.

C. S. Kim, C. Kim, and O. H. Rho. Feasibility study of constant eddy-viscosity assumption in gradient-based design optimization. *Journal of Aircraft*, 40:1168–1176, 5 2003. doi:10.2514/2.7206.

J.-H. Kim, B. Ovgor, et al. Optimization of the aerodynamic and aeroacoustic performance of an axial-flow fan. *AIAA Journal*, 52(9):2032–2044, 2014. doi:10.2514/1.J052754.

R. King, K. Deb, and H. Rughooputh. Comparison of NSGA-II and SPEA2 on the Multiobjective Environmental/Economic Dispatch Problem. *University of Mauritius Research Journal*, 16(1), 2 2016. URL https://www.ajol.info/index.php/umrj/article/view/131124.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 5 1983. doi:10.1126/SCIENCE.220.4598.671.

H. Kline. *The continuous adjoint method for multi-fidelity hypersonic inlet design*. PhD thesis, Stanford University, Palo Alto, CA, 04 2017. URL https://purl.stanford.edu/mm280hp6972.

M. Klun, Z. Guzović, and P. Rašković. Innovative small axial multistage turbine with partial admission for bottoming orc. *Energy Reports*, 7:9069–9093, 11 2021. doi:10.1016/J.EGYR.2021.11.229.

T. Korakianitis and P. Papagiannidis. Surface-curvature-distribution effects on turbine-cascade performance. *Journal of Turbomachinery*, 115:334–341, 4 1993. doi:10.1115/1.2929239.

D. Kraft. A software package for sequential quadratic programming. Technical report, Deutsche Forschungs- und Versuchsanstalt fuer Luft- und Raumfahrt e.V. (DFVLR), Koln, 1988. URL http://hdl.handle.net/10068/147127.

A. B. Lambe and J. R. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46 (2):273–284, 8 2012. doi:10.1007/s00158-012-0763-y.

M. Laumanns, L. Thiele, et al. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, 10:263–282, 9 2002. doi:10.1162/106365602760234108.

S. Lemmens and S. Lecompte. Case study of an organic rankine cycle applied for excess heat recovery: Technical, economic and policy matters. *Energy Conversion and Management*, 138:670–685, 2017. doi:10.1016/J.ENCONMAN.2017.01.074.

T. M. Leung and D. W. Zingg. Aerodynamic shape optimization of wings using a parallel newton-krylov approach. *AIAA Journal*, 50:540–550, 3 2012. doi:10.2514/1.J051192.

R. I. Lewis. *Turbomachinery Performance Analysis*. Elsevier Science, 5 1996. ISBN 9780340631911. URL https://www.elsevier.com/books/turbomachinery-performance-analysis/lewis/978-0-340-63191-1.

D. Li and R. Hartmann. Adjoint-based airfoil optimization with discretization error control. *International Journal for Numerical Methods in Fluids*, 77:1–17, 1 2015. doi:10.1002/FLD.3971.

W. Li, L. Huyse, and S. Padula. Robust airfoil optimization to achieve drag reduction over a range of mach numbers. *Structural and Multidisciplinary Optimization 2002 24:1*, 24:38–50, 8 2002. doi:10.1007/S00158-002-0212-4.

J. Luo, C. Zhou, and F. Liu. Multipoint design optimization of a transonic compressor blade by using an adjoint method. *Journal of Turbomachinery*, 136(5), 9 2013. doi:10.1115/1.4025164.

J. N. Lyness and C. B. Moler. Numerical differentiation of analytic functions. *SIAM Journal on Numerical Analysis*, 4:202–210, 7 1967. doi:10.1137/0704019.

Z. Lyu, G. K. Kenway, and J. R. Martins. Aerodynamic shape optimization investigations of the common research model wing benchmark. *AIAA Journal*, 53:968–985, 10 2014. doi:10.2514/1.J053318.

H. Ma, H. Wei, et al. A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints. *Information Sciences*, 560:68–91, 6 2021. doi:10.1016/j.ins.2021.01.029.

M. H. A. Madsen, F. Zahle, et al. Multipoint high-fidelity CFD-based aerodynamic shape optimization of a 10 MW wind turbine. *Wind Energy Science*, 4(2):163–192, 4 2019. doi:10.5194/wes-4-163-2019.

R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering, 4 2004.

A. C. Marta and S. Shankaran. On the handling of turbulence equations in RANS adjoint solvers. *Computers and Fluids*, 74:102–113, 3 2013. doi:10.1016/j.compfluid.2013.01.012.

J. R. Martins et al. Mach-aero framework. `https://github.com/mdolab/MACH-Aero`, 2022.

V. Masson-Delmotte, P. Zhai, H. Pörtner, D. Cynthia Roberts, et al. Sixth assessment report on climate change: Mitigation of climate change. Technical report, Working Group Intergovernmental Panel on Climate Change (IPCC), 10 2021. URL `https://www.ipcc.ch/report/ar6/wg3/`.

M. D. Mckay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 42(1):55, 2 1979. doi:10.2307/1271432.

D. J. Mee, N. C. Baines, et al. An examination of the contributions to loss on a transonic turbine blade in cascade. *Journal of Turbomachinery*, 114:155–162, 1 1992. doi:10.1115/1.2927979.

O. Z. Mehdizadeh, L. Temmerman, et al. Applications of earsm turbulence models to internal flows. In *Proceedings of the ASME Turbo Expo*, volume 8, pages 2079–2086. American Society of Mechanical Engineers Digital Collection, 7 2012. doi:10.1115/GT2012-68886.

T. Mengistu and W. Ghaly. Aerodynamic optimization of turbomachinery blades using evolutionary methods and ANN-based surrogate models. *Optimization and Engineering*, 9(3):239–255, 9 2008. doi:10.1007/s11081-007-9031-1.

F. Menter. Zonal Two Equation k- Turbulence Models For Aerodynamic Flows. In *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*, Orlando, FL, 7 1993. American Institute of Aeronautics and Astronautics. doi:10.2514/6.1993-2906.

F. R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32 (8):1598–1605, 5 1994. doi:10.2514/3.12149.

A. Messac. Physical programming: Effective optimization for computational design. *AIAA Journal*, 34(1): 149–158, 5 1996. doi:10.2514/3.13035.

V. Michelassi, L. W. Chen, et al. Compressible direct numerical simulation of low-pressure turbines-part II: Effect of inflow disturbances. *Journal of Turbomachinery*, 137(7), 7 2015. doi:10.1115/1.4029126.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

S. Moffatt and L. He. Blade forced response prediction for industrial gas turbines: Part 1 - methodologies. In *Turbo Expo: Power for Land, Sea, and Air*, volume 4, pages 407–414. American Society of Mechanical Engineers, 2 2003. doi:10.1115/GT2003-38640.

H. Montanelli, M. Montagnac, and F. Gallard. Gradient span analysis method: Application to the multipoint aerodynamic shape optimization of a turbine cascade. *Journal of Turbomachinery*, 137(9), 3 2015. doi:10.1115/1.4030016.

J. Moore, R. Chapman, and G. Dozier. Multiobjective particle swarm optimization. *Proceedings of the Annual Southeast Conference*, 2000-April:56–57, 7 2000. doi:10.1145/1127716.1127729.

J. J. Moré. Recent developments in algorithms and software for trust region methods. *Mathematical Programming The State of the Art*, pages 258–287, 1983. doi:10.1007/978-3-642-68874-4_11.

L. Morgan and P. Finnegan. Benefits and drawbacks of open source software: An exploratory study of secondary software firms. In *Open Source Development, Adoption and Innovation*, pages 307–312, Boston, MA, 2007. Springer US. ISBN 978-0-387-72486-7. doi:10.1007/978-0-387-72486-7_33.

S. K. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *38th Aerospace Sciences Meeting and Exhibit*, 2000. doi:10.2514/6.2000-667.

M. Nemec, D. W. Zingg, and T. H. Pulliam. Multipoint and multi-objective aerodynamic shape optimization. *AIAA Journal*, 42(6):1057–1065, 5 2004. doi:10.2514/1.10415.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, 2 edition, 2006. doi:10.1007/978-0-387-40065-5.

J. H. Page, R. Watson, et al. Advances of turbomachinery design optimization. In *53rd AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2015. doi:10.2514/6.2015-0629.

V. Pareto, A. S. Schwier, and A. N. Page. Manual of political economy. *Economic Development and Cultural Change*, 23(3):551–553, 4 1975. doi:10.1086/450817.

C. Peeren and K. Vogeler. Geometrical modification of the unsteady pressure to reduce low-pressure turbine flutter. *Journal of Turbomachinery*, 139, 9 2017. doi:10.1115/1.4036343.

R. E. Perez, P. W. Jansen, and J. R. Martins. PyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Structural and Multidisciplinary Optimization*, 45(1):101–118, 1 2012. doi:10.1007/s00158-011-0666-3.

M. Pini. *Turbomachinery Design Optimization using Adjoint Method and Accurate Equations of State*. PhD thesis, Politecnico di Milano, Milano, 12 2013.

M. Pini, G. Persico, and V. Dossena. Robust adjoint-based shape optimization of supersonic turbomachinery cascades. In *Proceedings of the ASME Turbo Expo*, volume 2B. American Society of Mechanical Engineers (ASME), 9 2014. doi:10.1115/GT2014-27064.

M. Pini, G. Persico, et al. Adjoint method for shape optimization in real-gas flow applications. *Journal of Engineering for Gas Turbines and Power*, 137(3), 3 2015. doi:10.1115/1.4028495.

J. Praß, J. Weber, et al. *Smart Energy and Grid: Novel Approaches for the Efficient Generation, Storage, and Usage of Energy in the Smart Home and the Smart Grid Linkup*. John Wiley & Sons, Ltd, 3 2017. doi:10.1002/9781119226444.CH20.

J. Qian, J. Yi, et al. A sequential constraints updating approach for Kriging surrogate model-assisted engineering optimization design problem. *Engineering with Computers*, 36(3):993–1009, 7 2020. doi:10.1007/s00366-019-00745-w.

J. F. Remacle, F. Henrotte, et al. A frontal delaunay quad mesh generator using the L^∞ norm. In *Proceedings of the 20th International Meshing Roundtable, IMR 2011*, pages 455–472. Springer, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-24734-7_25.

E. Rinaldi, R. Pecnik, and P. Colonna. Unsteady operation of a highly supersonic organic rankine cycle turbine. *Journal of Turbomachinery*, 138, 12 2016. doi:10.1115/1.4033973/473250.

P. J. Roache. Perspective: A method for uniform reporting of grid refinement studies. *Journal of Fluids Engineering, Transactions of the ASME*, 116(3):405–413, 9 1994. doi:10.1115/1.2910291.

P. J. Roache. Verification of codes and calculations. *AIAA Journal*, 36(5):696–702, 5 1998. doi:10.2514/2.457.

P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 10 1981. doi:10.1016/0021-9991(81)90128-5.

A. Rubino, M. Pini, et al. A look-up table method based on unstructured grids and its application to non-ideal compressible fluid dynamic simulations. *Journal of Computational Science*, 28:70–77, 9 2018. doi:10.1016/j.jocs.2018.08.001.

A. Rubino, S. Vitale, et al. Fully-turbulent adjoint method for the unsteady shape optimization of multi-row turbomachinery. *Aerospace Science and Technology*, 106:106132, 11 2020. doi:10.1016/j.ast.2020.106132.

A. Samad, K. Y. Kim, and K. S. Lee. Multi objective optimization of a turbomachinery blade using nsga-ii. In *Proceedings of the 5th Joint ASME/JSME Fluids Engineering Summer Conference*, pages 885–891. American Society of Mechanical Engineers Digital Collection, 3 2009. doi:10.1115/FEDSM2007-37434.

J. A. Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA Journal*, 39:877–884, 5 2001. doi:10.2514/2.1391.

J. A. Samareh. Geometry and grid/mesh generation issues for cfd and csm shape optimization. *Optimization and Engineering*, 6:21–32, 3 2005. doi:10.1023/B:OPTE.0000048535.08259.A8.

M. Schireson and D. Thakker. The money in Open Source Software, 2016. URL https://techcrunch.com/2016/02/09/the-money-in-open-source-software/.

K. Schittkowski. The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function - Part 2: An efficient implementation with linear least squares subproblems. *Numerische Mathematik*, 38(1):115–127, 2 1982. doi:10.1007/BF01395811.

M. Schmelzer, R. P. Dwight, and P. Cinnella. Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression. *Flow, Turbulence and Combustion*, 104(2):579–603, 2020. doi:10.1007/s10494-019-00089-x.

M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, ON, Canada, 1997. URL https://uwspace.uwaterloo.ca/handle/10012/190.

T. Scott and A. E. Rung. Memorandum for the Heads of Departments and Agencies, 7 2016. URL https://www.cio.gov/2016/08/11/peoples-code.html.

N. R. Secco, G. K. Kenway, et al. Efficient mesh generation and deformation for aerodynamic shape optimization. *AIAA Journal*, 59:1151–1168, 2 2021. doi:10.2514/1.J059491.

T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*, pages 151–160. Association for Computing Machinery, Inc, 8 1986. doi:10.1145/15922.15903.

K. Sedlaczek and P. Eberhard. Using augmented Lagrangian particle swarm optimization for constrained problems in engineering. *Structural and Multidisciplinary Optimization*, 32(4):277–286, 10 2006. doi:10.1007/s00158-006-0032-z.

B. O. Shubert. A Sequential Method Seeking the Global Maximum of a Function. *SIAM Journal on Numerical Analysis*, 9(3):379–388, 7 1972. doi:10.1137/0709036.

M. R. Sierra and C. A. C. Coello. Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. *Lecture Notes in Computer Science*, 3410:505–519, 2005. doi:10.1007/978-3-540-31880-4_35.

C. Sieverding and M. Manna. A review on turbine trailing edge flow. *International Journal of Turbomachinery, Propulsion and Power 2020, Vol. 5, Page 10*, 5:10, 5 2020. doi:10.3390/IJTPP5020010.

S. N. Skinner and H. Zare-Behtash. State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing Journal*, 62:933–962, 1 2018. doi:10.1016/j.asoc.2017.09.030.

P. Song, J. Sun, and C. Huo. Enhancing refrigeration capacity of turbo expander by means of multipoint design optimization. *International Journal of Refrigeration*, 108:60–78, 12 2019. doi:10.1016/j.ijrefrig.2019.08.035.

P. R. Spalart and S. R. Allmaras. One-equation turbulence model for aerodynamic flows. In *30th Aerospace Sciences Meeting and Exhibit*, pages 5–21, 1992. doi:10.2514/6.1992-439.

C. G. Speziale. Turbulence modeling for time-dependent rans and vles: A review. *AIAA Journal*, 36:173–184, 2 1998. doi:10.2514/2.7499.

J. Stork, M. Friese, et al. Open Issues in Surrogate-Assisted Optimization. In *High-Performance Simulation-Based Optimization*, pages 225–244. Springer Verlag, 2020. doi:10.1007/978-3-030-18764-4_10.

W. T. Tiow, K. F. C. Yiu, and M. Zangeneh. Application of simulated annealing to inverse design of transonic turbomachinery cascades. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 216(1):59–73, 2 2002. doi:10.1243/095765002760024845.

I. Torreguitart, T. Verstraete, and L. Mueller. CAD and adjoint based multipoint optimization of an axial turbine profile. In *Computational Methods in Applied Sciences*, volume 49, pages 35–46. Springer, 2019. doi:10.1007/978-3-319-89890-2_3.

S. Tüchler, Z. Chen, and C. D. Copeland. Multipoint shape optimisation of an automotive radial compressor using a coupled computational fluid dynamics and genetic algorithm approach. *Energy*, 165:543–561, 12 2018. doi:10.1016/j.energy.2018.09.076.

R. A. Van den Braembussche. *Numerical Optimization for Advanced Turbomachinery Design*, pages 147–189. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-72153-6. doi:10.1007/978-3-540-72153-6_6.

G. N. Vanderplaats. *Numerical Optimization Techniques For Engineering Design*. Vanderplaats Research and Development, Inc, Colorado Springs, 3 edition, 1999. ISBN 9780944956007.

G. N. Vanderplaats and H. Sugimoto. A general-purpose optimization program for engineering design. *Computers and Structures*, 24(1):13–21, 1 1986. doi:10.1016/0045-7949(86)90331-7.

V. Venkatakrishnan. On the accuracy of limiters and convergence to steady state solutions. In *31st Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics (AIAA), 1 1993. doi:10.2514/6.1993-880.

G. Venter and J. Sobieszczanski-Sobieski. Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization. *Structural and Multidisciplinary Optimization*, 26:121–131, 1 2004. doi:10.1007/S00158-003-0318-3.

G. Venter and J. Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA Journal*, 41:1583–1589, 5 2012. doi:10.2514/2.2111.

T. L. Vincent and W. J. Grantham. *Optimality in Parametric Systems*, volume 18. Wiley, New York, 5 1982. ISBN 0471083070.

P. Virtanen, R. Gommers, and T. E. Oliphant. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 3 2020. doi:10.1038/s41592-019-0686-2.

S. Vitale, M. Pini, et al. Extension of the SU2 Open Source CFD code to the simulation of turbulent flows of fluids modelled with complex thermophysical laws. In *22nd AIAA Computational Fluid Dynamics Conference*, 2015. doi:10.2514/6.2015-2760.

S. Vitale, T. A. Albring, et al. Fully turbulent discrete adjoint solver for non-ideal compressible flow applications. *Journal of the Global Power and Propulsion Society*, 1, 11 2017. doi:10.22261/jgpps.z1fvoi.

S. Vitale, M. Pini, and P. Colonna. Multistage Turbomachinery Design Using the Discrete Adjoint Method Within the Open-Source Software SU2. *Journal of Propulsion and Power*, pages 1–14, 3 2020. doi:10.2514/1.B37685.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 2005 106:1*, 106:25–57, 4 2005. doi:10.1007/S10107-004-0559-Y.

R. E. Walraevens, H. E. Gallus, et al. Experimental and computational study of the unsteady flow in a 1.5 stage axial turbine with emphasis on the secondary flow in the second stator. In *Proceedings of the ASME Turbo Expo*, volume 1. American Society of Mechanical Engineers (ASME), 12 1998. doi:10.1115/98-GT-254.

B. Walther and S. Nadarajah. An Adjoint-Based Multi-Point Optimization Method for Robust Turbomachinery Design. In *Proceedings of ASME Turbo Expo 2015: Turbine Technical Conference and Exposition*. ASME International, 6 2015a. doi:10.1115/gt2015-44142.

B. Walther and S. Nadarajah. Adjoint-based constrained aerodynamic shape optimization for multistage turbomachines. *Journal of Propulsion and Power*, 6 2015b. doi:10.2514/1.B35433.

D. X. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage turbomachines-Part I: Methodology and verification. *Journal of Turbomachinery*, 132(2), 4 2010. doi:10.1115/1.3072498.

D. X. Wang, L. He, et al. Adjoint aerodynamic design optimization for blades in multistage turbomachines-Part II: Validation and application. *Journal of Turbomachinery*, 132(2), 4 2010. doi:10.1115/1.3103928.

P. Wang and M. Zangeneh. Aerodynamic and aeroacoustic optimization of a transonic centrifugal compressor. In *Proceedings of the ASME Turbo Expo*, volume 2A. American Society of Mechanical Engineers (ASME), 9 2014. doi:10.1115/GT2014-26813.

X. D. Wang, C. Hirsch, et al. Multi-objective optimization of turbomachinery using improved NSGA-II and approximation model. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):883–895, 2 2011. doi:10.1016/j.cma.2010.11.014.

A. P. Wheeler, R. D. Sandberg, et al. Direct numerical simulations of a high-pressure turbine vane. *Journal of Turbomachinery*, 138(7), 7 2016. doi:10.1115/1.4032435.

P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11:226–235, 1969. doi:10.1137/1011036.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi:10.1109/4235.585893.

L. Xu and J. D. Denton. The base pressure and loss of a family of four turbine blades. *Journal of Turbomachinery*, 110:9–17, 1 1988. doi:10.1115/1.3262174.

Y. Yin, P. Yang, et al. Feature selection and processing of turbulence modeling based on an artificial neural network. *Physics of Fluids*, 32(10):105117, 10 2020. doi:10.1063/5.0022561.

Y. Yu, Z. Lyu, Z. Xu, and J. R. Martins. On the influence of optimization algorithm and initial design on wing aerodynamic shape optimization. *Aerospace Science and Technology*, 75:183–199, 4 2018. doi:10.1016/j.ast.2018.01.016.

M. Zeleny. Compromise Programming. *Multiple Criteria Decision Making*, pages 262–301, 1973.

Y. Zhao, H. D. Akolekar, et al. RANS turbulence model development using CFD-driven machine learning. *Journal of Computational Physics*, 411:109413, 6 2020. doi:10.1016/j.jcp.2020.109413.

C. Zhu, R. H. Byrd, et al. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 12 1997. doi:10.1145/279232.279236.

D. W. Zingg and S. Elias. Aerodynamic optimization under a range of operating conditions. *AIAA Journal*, 44 (11):2787–2792, 11 2006. doi:10.2514/1.23658.

D. W. Zingg, M. Nemec, and T. H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *River Publishers*, 17:103–126, 2012. doi:10.3166/REMN.17.103-126.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *TIK*, 103, 2001. doi:10.3929/ethz-a-004284029.

A

GRADIENT-BASED OPTIMIZATION

The contents of this appendix serve to provide background on some core concepts behind gradient-based optimization algorithms. Additionally, a description on a selection of frequently applied algorithms is given.

A.1. SQP

Sequential Quadratic Programming (SQP) presents an elegant approach to handle constraints by direct inclusion in the design problem without the need to determine the Lagrange multipliers. The Lagrangian is set up as in Equation A.1, λ_j is the j-th Lagrange multiplier.

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \mathcal{J}(\boldsymbol{\alpha}) + \sum_{j=1}^{m_{\text{ineq}}} \lambda_j c_{\text{ineq}_j}(\boldsymbol{\alpha}) + \sum_{l=1}^{m_{\text{eq}}} \lambda_{(m_{\text{ineq}}+l)} c_{\text{eq}_l}(\boldsymbol{\alpha}) \tag{A.1}$$

Applying the Karush–Kuhn–Tucker conditions leads to the system of equations for quadratic programming in Equation A.2.

$$\nabla_{\boldsymbol{\alpha}} \mathcal{L} = 0\,,\ \nabla_{\boldsymbol{\lambda}} \mathcal{L} = 0 \quad \Rightarrow \quad \begin{bmatrix} \boldsymbol{g}(\boldsymbol{\alpha}) + \boldsymbol{y}(\boldsymbol{\alpha})\boldsymbol{\lambda} \\ \boldsymbol{c}(\boldsymbol{\alpha}) \end{bmatrix} = 0 \tag{A.2}$$

Applying Newton's iterative method for root finding in Equation A.3 and using $\Delta\boldsymbol{\lambda} = \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k$, Equation A.4 is obtained which describes the general system of equations in SQP.

$$\boldsymbol{\alpha}^{k+1} = \boldsymbol{\alpha}^k - \frac{\mathcal{J}(\boldsymbol{\alpha}^k)}{\nabla \mathcal{J}(\boldsymbol{\alpha}^k)} \tag{A.3}$$

$$\begin{bmatrix} \boldsymbol{W}(\boldsymbol{\alpha}^k, \boldsymbol{\lambda}^k) & \boldsymbol{y}(\boldsymbol{\alpha}^k) \\ \boldsymbol{y}(\boldsymbol{\alpha}^k)^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\alpha} \\ \boldsymbol{\lambda}^{k+1} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{q}(\boldsymbol{\alpha}^k) \\ \boldsymbol{c}(\boldsymbol{\alpha}^k) \end{bmatrix} \tag{A.4}$$

Quadratic programming sub problem is defined to find the search direction $\boldsymbol{S}$ for constrained optimization in Equation A.5. The last term ensures that the problem is solvable by ensuring consistent constraints, making it a (n+1)-dimensional problem.

$$\min_{S \in \mathbb{R}^n} \frac{1}{2} \boldsymbol{S}^{\mathrm{T}} \boldsymbol{B}^k \boldsymbol{S} + \nabla \mathcal{J}(\boldsymbol{\alpha}^k) d + \frac{1}{2} \rho^k (\delta^k)^2 \tag{A.5}$$

Subject to the linearized constraints:

$$\nabla \boldsymbol{g}_j(\boldsymbol{\alpha}^k) \boldsymbol{S} + \delta_j^k \boldsymbol{g}_j(\boldsymbol{\alpha}^k) \geq 0, \quad j = 1, \ldots, m_{\text{eq}} + m_{\text{ineq}} \tag{A.6}$$

$$0 \leq \delta^k \leq 1 \tag{A.7}$$

Where:

$$\delta^k = \begin{cases} 1 & \text{if } \boldsymbol{g}_j(\boldsymbol{\alpha}^k) > 0 \\ \sigma^k & \text{else} \end{cases} \tag{A.8}$$

In Equation A.5, ρ is a penalty parameter which aims to keeping δ^k as small as possible. $\boldsymbol{B}^k$ relates to the Hessian of $\mathcal{L}$. Which in Quasi-Newton methods is approximated from first-order gradient terms of $\mathcal{J}$. The Broydon-Fletcher-Goldfarb-Shanno (BFGS) algorithm is a popular method to obtain this approximation, which is described by the expression in Equation A.9.

$$\boldsymbol{B}^{k+1} = \boldsymbol{B}^k + \frac{\boldsymbol{q}^k(\boldsymbol{q}^k)^{\mathrm{T}}}{(\boldsymbol{q}^k)^{\mathrm{T}}\boldsymbol{s}^k} - \frac{\boldsymbol{B}^k\boldsymbol{s}^k(\boldsymbol{s}^k)^{\mathrm{T}}\boldsymbol{B}^k}{(\boldsymbol{s}^k)^{\mathrm{T}}\boldsymbol{B}^k\boldsymbol{s}^k} \tag{A.9}$$

with $\boldsymbol{s}^k$, $\boldsymbol{q}^k$ and $\boldsymbol{\eta}^k$ computed as:

$$\boldsymbol{s}^k = \boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k = d^k\boldsymbol{S}^k \tag{A.10}$$

$$\boldsymbol{q}^k = \theta^k\boldsymbol{\eta}^k + (1-\theta^k)\boldsymbol{B}^k\boldsymbol{S}^k \tag{A.11}$$

$$\boldsymbol{\eta}^k = \nabla_{\boldsymbol{\alpha}}\mathcal{L}(\boldsymbol{\alpha}^{k+1}, \boldsymbol{\lambda}^k) - \nabla_{\boldsymbol{\alpha}}\mathcal{L}(\boldsymbol{\alpha}^k, \boldsymbol{\lambda}^k) \tag{A.12}$$

And θ^k is selected a value in order to guarantee positive definiteness of the updated B matrix in the case of active constraints.

A.2. NEWTON'S METHOD

Newton's method is a second order method to obtain the search direction in gradient-based optimization. The second-order Taylor series of the objective function in Equation A.13, is minimized for the search direction that minimizes $\mathcal{J}$.

$$\mathcal{J}\left(\boldsymbol{\alpha}^{k-1} + \boldsymbol{S}(\boldsymbol{\alpha}^k)\right) \approx \mathcal{J}(\boldsymbol{\alpha}^{k-1}) + \nabla\mathcal{J}^{\mathrm{T}}(\boldsymbol{\alpha}^{k-1})\,\boldsymbol{S}(\boldsymbol{\alpha}^k) + \frac{1}{2}\boldsymbol{S}(\boldsymbol{\alpha}^k)^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{\alpha}^{k-1})\,\boldsymbol{S}(\boldsymbol{\alpha}^k) \tag{A.13}$$

$\frac{d\mathcal{J}}{d\boldsymbol{S}} = 0$ minimizes $\mathcal{J}(\boldsymbol{\alpha}^k) = \mathcal{J}\left(\boldsymbol{\alpha}^{k-1} + \boldsymbol{S}(\boldsymbol{\alpha}^k)\right)$, leading to Equation A.14 for the search direction, defined by the function gradient and Hessian at the previous iteration.

$$\boldsymbol{S}(\boldsymbol{\alpha}^k) = -\frac{\nabla\mathcal{J}(\boldsymbol{\alpha}^{k-1})}{\frac{1}{2}\boldsymbol{H}(\boldsymbol{\alpha}^{k-1})} \tag{A.14}$$

A.2.1. QUASI NEWTON'S METHOD

In the quasi-Newton method, the Hessian $\boldsymbol{H}$, is replaced by an approximation matrix $\boldsymbol{A}$ as in Equation A.15.

$$\boldsymbol{S}(\boldsymbol{\alpha}^k) = -\boldsymbol{A}(\boldsymbol{\alpha}^k)\,\nabla\mathcal{J}(\boldsymbol{\alpha}^{k-1}) \tag{A.15}$$

$\boldsymbol{A}$ is obtained by a difference matrix $\boldsymbol{D}$, which is a function of the difference in design and gradient vector between the current and previous iteration and the previous approximation matrix, as in Equation A.16. In BFGS, $\boldsymbol{D}$ is represented by the two terms in the right-hand side of Equation A.9.

$$\boldsymbol{A}(\boldsymbol{\alpha}^{k+1}) = \boldsymbol{A}(\boldsymbol{\alpha}^k) + \boldsymbol{D}(\boldsymbol{\alpha}^k) \quad \Rightarrow \quad \boldsymbol{D}(\boldsymbol{\alpha}^k) = f\left(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1},\, \nabla\mathcal{J}(\boldsymbol{\alpha}^k) - \nabla\mathcal{J}(\boldsymbol{\alpha}^{k-1}),\, \boldsymbol{A}(\boldsymbol{\alpha}^k)\right) \tag{A.16}$$

A.3. SLSQP

Sequential-Least-Squares Quadratic Programming (SLSQP) [Kraft 1988; Schittkowski 1982] uses a transformation of the quadratic sub-problem for the search direction in Equation A.5 into a least-squares sub-problem, see in Equation A.17. Constraints are linearized and treated analogously as in the previous section, therefore omitted for brevity.

$$\min_{d\in\mathbb{R}^n} \| (\boldsymbol{D}^k)^{\frac{1}{2}}(\boldsymbol{L}^k)^{\mathrm{T}}\boldsymbol{d} + (\boldsymbol{D}^k)^{-\frac{1}{2}}(\boldsymbol{L}^k)^{-1}\nabla f(\boldsymbol{\alpha}^k) \| \tag{A.17}$$

$\| \boldsymbol{M} \|$ denotes the Euclidean induced matrix norm of matrix $\boldsymbol{M}$. In other terms, it is the square root of the largest eigenvalue of $\boldsymbol{M}^*\boldsymbol{M}$, where $\boldsymbol{M}^*$ is the conjugate transpose of $\boldsymbol{M}$, hence least-*squares*. See Equation A.18.

$$\begin{aligned} \| \boldsymbol{M} \| &= \sup\{ \| \boldsymbol{M}\boldsymbol{\alpha} \| : \| \boldsymbol{\alpha} \| = 1\} \\ &= \sqrt{\lambda_{\max}(\boldsymbol{M}^* \boldsymbol{M})} \\ &\left(= \sigma_{\max}(\boldsymbol{M})\right) \end{aligned} \tag{A.18}$$

$\boldsymbol{L}$ is a lower triangular matrix with unit diagonals and $\boldsymbol{D}$ is a diagonal matrix of d_i, such that $\boldsymbol{L}$ and $\boldsymbol{D}$ are the components of a stable factorization of $\boldsymbol{B}$ as in Equation A.19.

$$\boldsymbol{B}^k = \boldsymbol{L}^k \boldsymbol{D}^k (\boldsymbol{L}^k)^{\mathrm{T}} \tag{A.19}$$

A.4. SNOPT

Sparse Nonlinear OPTimizer (SNOPT) is a program for general linear and nonlinear function optimization [Gill et al. 2002]. It allows for thousands of constraints and variables, and as mentioned in the user's guide, works most efficiently under many active constraints [Gill et al. 2008].

A.5. IPOPT

Interior Point OPTimization (IPOPT) applies an indirect method of constrained optimization by incorporating constraints in the objective function through penalty methods. The merit function for Sequential Unconstrained Minimization Techniques (SUMT) containing the objective and constraint penalty terms is shown in Equation A.20. Here, r_p is a scalar penalty parameter.

$$\Phi(\boldsymbol{\alpha}, r_p) = \mathcal{J}(\boldsymbol{\alpha}) + r_p P(g_j(\boldsymbol{\alpha}), h_k(\boldsymbol{\alpha})) \tag{A.20}$$

The merit function for interior penalty methods is shown in Equation A.21. G should move to infinity when constraint are approached and H should move to infinity when r_p moves to zero.

$$\Phi(\boldsymbol{\alpha}, r_p) = \mathcal{J}(\boldsymbol{\alpha}) + r_p \sum_{j=1}^{m_{\text{ineq}}} \boldsymbol{g}_j(\boldsymbol{\alpha}) + H(r_p) \sum_{k=1}^{m_{\text{eq}}} \boldsymbol{h}_k^2(\boldsymbol{\alpha}) \tag{A.21}$$

B

EFFICIENT GLOBAL OPTIMIZATION

Efficient Global Optimization (EGO) is a surrogate-assisted optimization algorithm. EGO has an approach similar to as shown in figure B.1. In this figure, the genetic algorithm is replaced by a different optimizer. In DAKOTA's implementation of EGO, the DIRECT algorithm developed by the North Carolina State University (NCSU) is applied as optimizer, and the Gaussian Process (GP) Kriging is used as surrogate model. The DIRECT algorithm is used to find the candidate points for evaluation, based on maximizing the Expected Improvement (EI) [Mockus et al. 1978] infill criterion. DIRECT is preferred by DAKOTA over the general branch-and-bound [Jones et al. 1998] because of reported convergence issues. Latin Hypercube Sampling (LHS) is used to sample the initial distribution from which the surrogate model is constructed. The process of identifying the sampling points is called design of experiments.

EGO is an on-line Kriging method, meaning it updates its surrogate model on each function evaluation. Optimization stops when the EI is sufficiently small. The EI function is applied to guide the selection of next points to be evaluated by the surrogate model. Exploration can be traded for exploitation by selecting new points with larger uncertainty. The design vector which maximizes the EI is selected as next evaluation point, as in Equation B.1. EI is elaborated on further in subsection B.3.1.

$$\boldsymbol{\alpha}^{k+1} = \arg\max_{\boldsymbol{\alpha}} \left(EI\left(\hat{G}(\boldsymbol{\alpha})\right)\right) \tag{B.1}$$

B.1. SURROGATE MODELLING

There are many ways of constructing a surrogate model, also referred to as metamodel. Metamodels are classified according to whether they are interpolating or regressing over the data points. Examples of metamodels are artificial neural networks, support vector machines, radial basis functions, polynomial regression and Kriging. Of which Kriging tends to be considered most frequently within fluid-dynamic shape optimization [Skinner and Zare-Behtash 2018].

Surrogate models are especially suited for optimizing expensive flow simulation problems such as turbomachines, which are characterized by a slow rate of convergence due to the complexity of the flow [Bellary et al. 2016]. Reducing the number of function evaluations by using for example Kriging models thus benefits hugely the optimization cost.

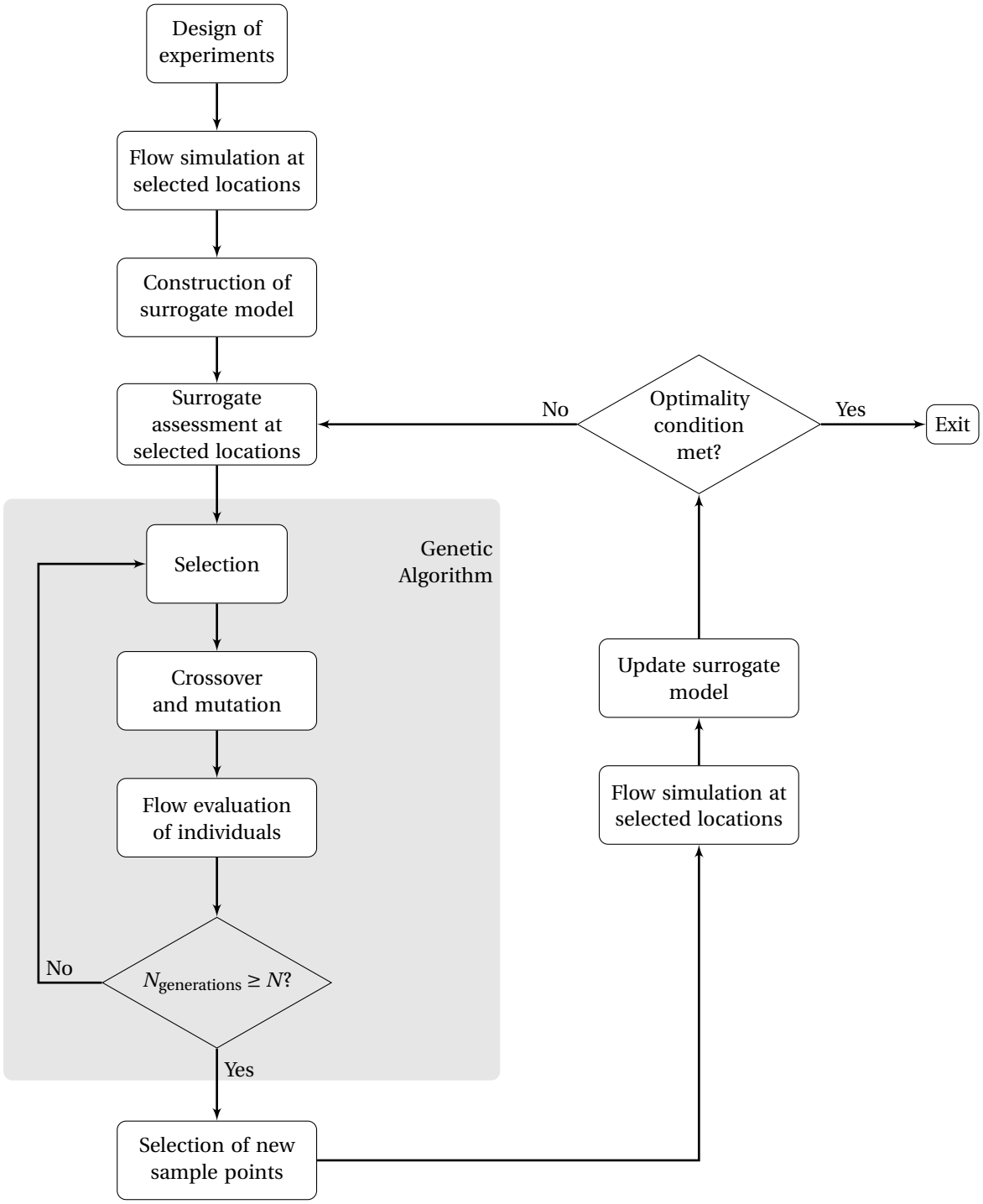


Figure B.1: Surrogate-based optimization combined with a genetic algorithm [Skinner and Zare-Behtash 2018].

Design of experiments

The goal of design of experiments is to provide the maximum information on the design space with minimum number of evaluated data points. Latin Hypercube Sampling (LHS) [Mckay et al. 1979] is most commonly applied in implementing design of experiments for surrogate-based optimization. LHS is a method which samples from a design space where each design variable contains one sample for each M number of equal probable intervals of value. The total number of samples is also M. Design variables in computational optimization are generally assumed having a uniform prior distribution, so each design variable contains one sample for each constant interval. The advantage over random sampling is that each design variable, irrespective of importance, is guaranteed to be sampled at multiple values. This provides information on how the objective function value responds to change in design variables, reducing the metamodel estimation error and thus aids optimization efficiency.

B.2. KRIGING

Kriging is a popular method to represent the surrogate model in EGO applications. Kriging in its basic form is an interpolation method, it interpolates data points. It is developed by geologists aiming to predict ground mineral concentrations using limited measurements.

The Gaussian Process (GP) model is desired because of its ability to not only provide an estimate of the average value of an unevaluated point, but also the prediction variance, which is a measure of the uncertainty in the GP. Mean and standard deviation are also necessary inputs to the EI infill criterion frequently applied in EGO. Each point in the design space of the GP metamodel is a realization of a stochastic Gaussian process, as in Equation B.2.

$$\hat{G}(\boldsymbol{\alpha}) \sim \mathcal{N}(\mu_G(\boldsymbol{\alpha}), \sigma_G^2(\boldsymbol{\alpha})) \tag{B.2}$$

Kriging assumes a spatial correlation between sampled points, expressed through a spatial correlation function. Further assumptions are stationarity and ergodicity. Stationarity is assumed, enabling the utilization of a constant μ and σ_Z for the whole estimated domain. The ergodicity assumption enables generalizing parameters obtained from a small section of the domain, to the whole domain.

The Kriging interpolant is modeled as in Equation B.3. $\boldsymbol{h}(\boldsymbol{\alpha})$ is the trend function of the model and $\boldsymbol{\beta}$ is the vector of trend coefficients. In Universal Kriging, $\boldsymbol{h}(\boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{\beta}$ is obtained through a least-squares polynomial fit of evaluated points. $Z(\boldsymbol{\alpha})$ is the GP error model which acts to correct the trend function, such that $G(\boldsymbol{\alpha})$ interpolates the evaluated points with zero uncertainty. For evaluated points $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ in the design space, the covariance function of the error model is shown in Equation B.4.

$$G(\boldsymbol{\alpha}) = \boldsymbol{h}(\boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{\beta} + Z(\boldsymbol{\alpha}) \tag{B.3}$$

$$Cov[Z(\boldsymbol{\alpha}), Z(\boldsymbol{\alpha}')] = \sigma^2 r(\boldsymbol{\alpha}, \boldsymbol{\alpha}') \tag{B.4}$$

$r(\boldsymbol{\alpha}, \boldsymbol{\alpha}')$ is the spatial correlation function between points $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$, in the form of a Gaussian function. For the ordinary Kriging considered here, $\sum_{i=1}^{n}\theta_i = 1$ and $\boldsymbol{h}(\boldsymbol{\alpha})^{\mathrm{T}}\boldsymbol{\beta} = \mu$, which is constant. d is the design vector dimensionality.

$$r(\boldsymbol{\alpha}, \boldsymbol{\alpha}') = exp\Big[-\sum_{i=1}^{d}\theta_i(\alpha_i - \alpha_i')^2\Big] \tag{B.5}$$

The correlation terms θ_i, mean μ and variance σ of the stationary Gaussian process Z, are the surrogate model's tuning parameters and are obtained by maximum likelihood estimation. The maximum likelihood function in logarithm form is given in Equation B.6.

$$-\frac{1}{2}\Big[n\,ln(2\pi\sigma^2) + ln(det\boldsymbol{R}) + (\boldsymbol{y} - \mathbf{1}\mu)^{\mathrm{T}}\boldsymbol{R}^{-1}\frac{(\boldsymbol{y} - \mathbf{1}\mu)}{\sigma^2}\Big] \tag{B.6}$$

Where $\boldsymbol{y}$ is the matrix of all observed points $[\mathcal{J}(\boldsymbol{\alpha}^{(1)}), \ldots, \mathcal{J}(\boldsymbol{\alpha}^{(n)})]^{\mathrm{T}}$ and $\boldsymbol{R}$ is the spatial correlation matrix as in Equation B.7, with size $n \times n$.

$$\boldsymbol{R} = \begin{bmatrix} r(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(1)}) & \dots & r(\boldsymbol{\alpha}^{(1)}, \boldsymbol{\alpha}^{(n)}) \\ \vdots & \ddots & \vdots \\ r(\boldsymbol{\alpha}^{(n)}, \boldsymbol{\alpha}^{(1)}) & \dots & r(\boldsymbol{\alpha}^{(n)}, \boldsymbol{\alpha}^{(n)}) \end{bmatrix} \tag{B.7}$$

In order to construct the Kriging model, μ and σ are obtained by setting the partial derivatives of the MLE to zero. Due to a lack of an analytical solution for determining the θ_i hyper-parameters, a separate numerical optimization has to be performed which maximizes the likelihood function. For bounded Kriging, a bounded numerical optimization is used [Aissa and Verstraete 2019; Aissa et al. 2019].

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}} \boldsymbol{R}^{-1} \boldsymbol{y}}{\mathbf{1}^{\mathrm{T}} \boldsymbol{R}^{-1} \mathbf{1}} \tag{B.8}$$

$$\hat{\sigma}^2(\boldsymbol{\alpha}) = \frac{1}{n} (\boldsymbol{y} - \mathbf{1}\hat{\mu})^{\mathrm{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \mathbf{1}\hat{\mu}) \tag{B.9}$$

The best linear unbiased predictor of candidate point $\boldsymbol{\alpha}$, $\hat{\boldsymbol{\alpha}}$ is given by Equation B.10. With $\boldsymbol{r}(\boldsymbol{\alpha}, \boldsymbol{\alpha}') = [r(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(1)}), \dots, r(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(n)})]^{\mathrm{T}}$. In other words, $\hat{y}$ is a linear combination of evaluated points with mean approximation error of zero. In constructing the Kriging model, calculating the determinant of $\boldsymbol{R}$ becomes expensive for large design vectors and large number of performed flow evaluations.

$$\hat{\mathcal{J}}(\boldsymbol{\alpha}) = E\big(\hat{G}(\boldsymbol{\alpha}) \big| \mathcal{J}(\boldsymbol{y})\big) = \hat{\mu} + \boldsymbol{r}(\boldsymbol{\alpha}, \boldsymbol{\alpha}')^{\mathrm{T}} \boldsymbol{R}^{-1} (\boldsymbol{y} - \mathbf{1}\hat{\mu}) \tag{B.10}$$

B.2.1. GRADIENT-ENHANCED KRIGING

Gradient-Enhanced Kriging (GEK) introduces sensitivity information in the data points to construct the approximation of the objective function. GEK is demonstrated to obtain a decrease of 60% in the number of flow evaluations [Backhaus et al. 2017], as depicted in Figure B.2. This is mainly due to a reduction in prediction error, which is an order of magnitude is lower [Backhaus et al. 2012 2017]. Additionally, the rate of feasible designs is considerably larger and lie closer to the Pareto front than the standard Kriging approach. The addition of sensitivity introduces a property in the provided information which acts to extrapolate to the fitness function in the vicinity of evaluated data points. Because gradients are cheaply obtainable by using the adjoint method and of GEK's superior performance, GEK is implemented in the code framework.

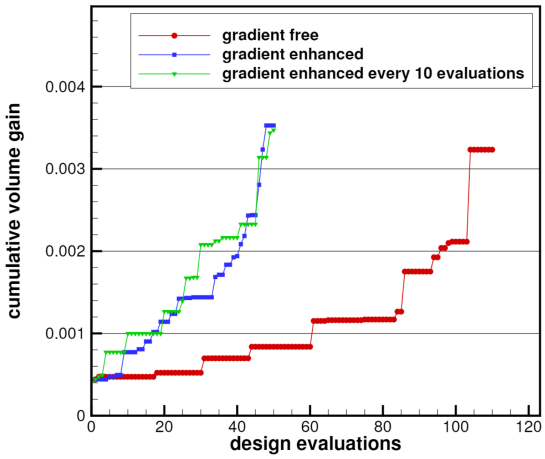


Figure B.2: Number of flow evaluations for Ordinary Kriging and gradient enhanced Kriging-based multi-point optimization [Backhaus et al. 2017]. For a description of cumulative volume gain, see expected volume gain in subsection B.2.2.

The are two main methods of introducing gradients into the Kriging model. The first is *indirect* GEK, in which the correlation matrix $\boldsymbol{R}$ is augmented by artificial points, perturbed slightly from existing points in all dimensions by using first-order Taylor series approximations. The drawback of this method is a high risk of an ill-conditioned $\boldsymbol{R}$, hampering the required invertibility of $\boldsymbol{R}$. Ill-conditioning is a result of data points being

located in proximity of each other, which tend to have similar response values, leading to linearly dependent columns in $\boldsymbol{R}$. The second method is *direct* GEK, in which the $\boldsymbol{y}$ matrix is augmented by the derivative values as in Equation B.11. The size of $\boldsymbol{y}$ is $n(d+1)\times 1$.

$$\boldsymbol{y} = \left[\mathcal{J}(\boldsymbol{\alpha}^{(1)}), \ldots, \mathcal{J}(\boldsymbol{\alpha}^{(n)}), \frac{\partial\mathcal{J}(\boldsymbol{\alpha}^{(1)})}{\partial\alpha_1}, \ldots, \frac{\partial\mathcal{J}(\boldsymbol{\alpha}^{(1)})}{\partial\alpha_d}, \ldots, \frac{\partial\mathcal{J}(\boldsymbol{\alpha}^{(n)})}{\partial\alpha_d}\right]^{\mathrm{T}} \tag{B.11}$$

To obtain the correlation matrix for the gradient enhanced method $\boldsymbol{R}_\nabla$, $\boldsymbol{R}$ is augmented with derivatives as in Equation B.12. $\boldsymbol{R}_\nabla$ contains for any pair of evaluated points, the auto correlation of evaluated points, cross correlation of evaluated points with gradients, and auto correlation of gradients. Its size is $d(n+1)\times d(n+1)$ and its complexity is thus quadratic in the number of dimensions d and evaluation points n. This is the *curse of dimensionality* which causes GEK to be prohibitively expensive to calculate for larger problems.

$$\boldsymbol{R}_\nabla = \begin{bmatrix} r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(1)}) & \cdots & r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(n)}) & \frac{\partial r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(1)})}{\partial\boldsymbol{\alpha}^{(1)}} & \cdots & \frac{\partial r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(n)})}{\partial\boldsymbol{\alpha}^{(n)}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(1)}) & \cdots & r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(n)}) & \frac{\partial r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(1)})}{\partial\boldsymbol{\alpha}^{(1)}} & \cdots & \frac{\partial r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(n)})}{\partial\boldsymbol{\alpha}^{(n)}} \\ \frac{\partial r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(1)})}{\partial\boldsymbol{\alpha}^{(1)}}^{\mathrm{T}} & \cdots & \frac{\partial r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(n)})}{\partial\boldsymbol{\alpha}^{(1)}}^{\mathrm{T}} & \frac{\partial^2 r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(1)})}{\partial^2\boldsymbol{\alpha}^{(1)}} & \cdots & \frac{\partial^2 r(\boldsymbol{\alpha}^{(1)},\boldsymbol{\alpha}^{(n)})}{\partial\boldsymbol{\alpha}^{(1)}\partial\boldsymbol{\alpha}^{(n)}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(1)})}{\partial\boldsymbol{\alpha}^{(n)}}^{\mathrm{T}} & \cdots & \frac{\partial r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(n)})}{\partial\boldsymbol{\alpha}^{(n)}}^{\mathrm{T}} & \frac{\partial^2 r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(1)})}{\partial\boldsymbol{\alpha}^{(n)}\partial\boldsymbol{\alpha}^{(1)}} & \cdots & \frac{\partial^2 r(\boldsymbol{\alpha}^{(n)},\boldsymbol{\alpha}^{(n)})}{\partial^2\boldsymbol{\alpha}^{(n)}} \end{bmatrix} \tag{B.12}$$

The expected values $\hat{\mathcal{J}}$ and $\hat{\sigma}$ are obtained using the same equations as in the non-gradient enhanced method.

B.2.2. EXPECTED VOLUME GAIN

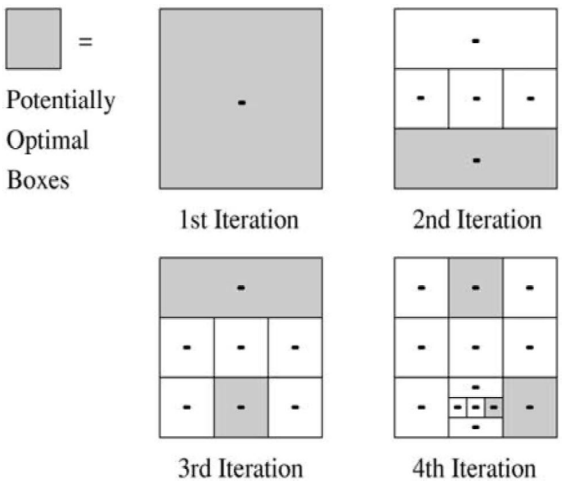
The expected volume-gain (EVG) is an infill criterion for multi-objective surrogate based optimization. It is statistical prediction estimating the volume gain enclosed by a candidate point $\mathcal{J}$ with respect to the Pareto front Y by Monte-Carlo sampling from the n-dimensional normal distributions obtained from the local values of mean and variance from the surrogate model [Backhaus et al. 2017]. The expression for expected volume gain is presented in Equation B.13. If no point on the Pareto front is dominated by $\mathcal{J}$, the difference to a lower limit value of the objective is used. The attractiveness of this pareto-based infill criterion is that candidate points can be selected on the basis of favourable mean volume gain, emphasizing exploitation, or points with slightly worse mean estimated value and large uncertainty, emphasizing exploration.

$$\mathcal{J}_{\mathrm{MO}} = \prod_i^n \left(\mathcal{J}_i - \max\{y_i,\ y\in Y \mid y_i < \mathcal{J}_i\}\right) \tag{B.13}$$

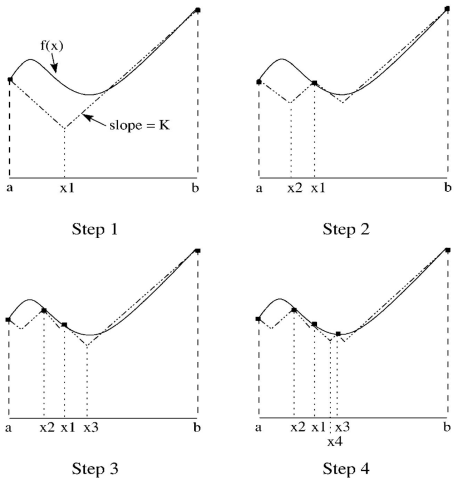
B.3. OPTIMIZER

DAKOTA's optimizer for EGO is the Dividing Rectangles (DIRECT) algorithm [Jones et al. 1993]. DIRECT is a variation on Lipschitzian optimization, of which the steps are pictured in Figure B.3b. Here, th design space is divided into rectangles wherein the optimal of each rectangle is determined by the objective function value at middle of the rectangle in combination with its largest side length. Optimal boxes are selected by the estimated minimum value at their centers, according to the gradient at their vertices. Optimal boxes, on the Pareto front of value and box dimension size, are divided during subsequent iterations. On the one hand, dividing large boxes yields exploration, on the other hand, dividing better value boxes yields exploitation. Varying Lipschitz constants are applied throughout optimization. Multiple boxes can be selected simultaneously for division, allowing for parallelization.

The modified DIRECT algorithm locates regions in design space of promising $\mathcal{J}$ and then uses a local optimizer for final convergence. This to leverage the fast initial convergence of DIRECT mitigate the slow final convergence [Jones and Martins 2021], and has been shown to require less function evaluations than multi-start SQP [Cox et al. 2001].



(a) DIRECT algorithm box division.



(b) Lipschitzian optimization.

Figure B.3: DIRECT algorithm box division steps (a) and Lipschitzian optimization steps using Shubert's algorithm [Shubert 1972] (b). Illustrations from Cox et al. [2001].

B.3.1. INFILL CRITERION

The expected improvement (EI) function is defined as the expectation of a point in the design space to have a better objective function value than the best solution known, using the estimated function value of the metamodel. The exploration and exploitation trade-off is introduced by balancing function values versus function value uncertainty of candidate points. The EI function is defined as in Equation B.14 [Jones et al. 1998]. Where $\mathcal{J}_{\min}$ is the response with lowest value from the existing sample points.

$$EI\big(\hat{G}(\boldsymbol{\alpha})\big) \equiv E\Big[\max\big(\mathcal{J}_{\min} - \hat{G}(\boldsymbol{\alpha}), 0\big)\Big] \tag{B.14}$$

A closed form of EI can be expressed as in Equation B.15, obtained through reformulating Equation B.14 as an integral and integrating by parts. Here, Φ and ϕ are the cumulative and probability density functions respectively of the normal distribution $\mathcal{N}(0,1)$.

$$EI(\boldsymbol{\alpha}) = \big(\mathcal{J}_{\min} - \hat{\mu}(\boldsymbol{\alpha})\big)\Phi\left(\frac{\mathcal{J}_{\min} - \hat{\mu}(\boldsymbol{\alpha})}{\hat{\sigma}(\boldsymbol{\alpha})}\right) + \hat{\sigma}(\boldsymbol{\alpha})\phi\left(\frac{\mathcal{J}_{\min} - \hat{\mu}(\boldsymbol{\alpha})}{\hat{\sigma}(\boldsymbol{\alpha})}\right) \tag{B.15}$$

Using $\arg\max_{\boldsymbol{\alpha}}\big(EI(\boldsymbol{\alpha})\big)$ as infill criterion is attractive as it is proven to find the global optimum in a finite number of iterations [Schonlau 1997].

B.3.2. HANDLING CONSTRAINTS

Handling of constraints for metamodel based optimization is generally performed through sub problem recasting such as estimating constraint violations or estimating feasible regions [Qian et al. 2020]. The simplest approach to incorporating constraints in the optimization problem is to reject infeasible solutions in the construction of the metamodel [Coello Coello 2000]. The DAKOTA implementation of EGO handles constraints through a L1-penalty method [Finkel 2005]. The resulting merit function to be minimized is shown in Equation B.16. The constraint penalty parameters c_J can be set by the user, but are in practice overlooked when utilizing the DAKOTA library.

$$\mathcal{J} + \sum_{i=1}^{m} c_j g_j(\boldsymbol{\alpha}) \tag{B.16}$$

C

FLOW SIMULATION CONVERGENCE

Figure C.1 displays the convergence of s_{gen} and the residuals of the fluid-dynamic simulation of the baseline blade using the mesh selected by the grid convergence study in section 5.4.1. The solution error at 2000 flow iterations, the error of s_{gen} with respect to the s_{gen} at the largest iteration number, is 0.26%.

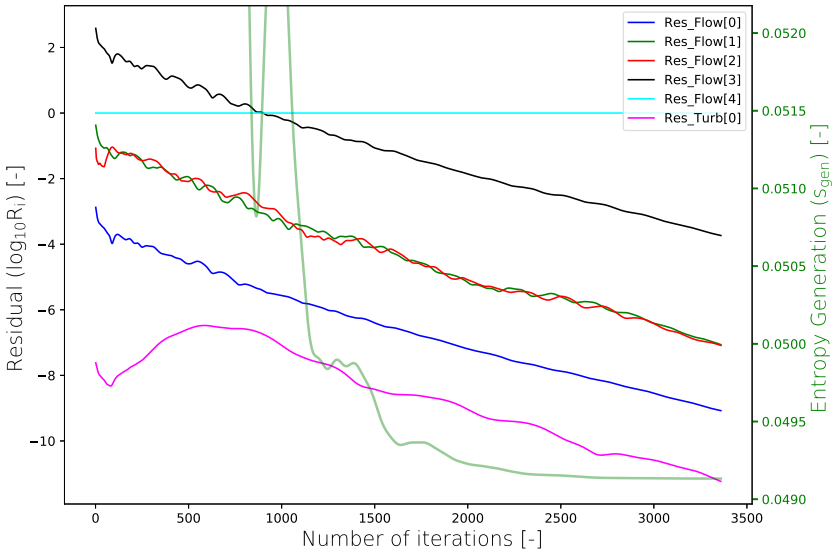


Figure C.1: Convergence of entropy generation and the residuals of the fluid-dynamic simulation of the mesh selected for optimization.

D

GRID CONVERGENCE INDEX METHOD

The Grid Convergence Index (GCI) method as described by Roache [1998] and extended by Eça and Hoekstra [2004] provides a conservative uncertainty estimation U_i for grid i, reported to guarantee a 95% confidence level [Eça et al. 2004].

The s_{gen} of three meshes with different fidelity is evaluated to calculate the error with respect to the estimated entropy generation for an infinitely fine mesh, the continuum solution value f_c. The mesh discretization error is calculated with the GCI method as described in Roache [1998].

For a fine grid 1 and coarse grid 2, the estimation of f_c is obtained using the generalized Richardson extrapolation for mixed, or p-th, order methods [Roache 1994], as in Equation D.1. r is the grid refinement ratio h_2/h_1, p represents the apparent order of accuracy of the time and space numerical schemes, f_1 and f_2 are the values of grid 1 and 2 respectively.

$$f_c = f_1 + \frac{f_1 - f_2}{r^p - 1} \tag{D.1}$$

Richardson extrapolation assumes that the solution values have a series representation in the grid size h and that the two grids are in the asymptotic range of the limit.

D.1. QUANTIFYING MESH DISCRETIZATION UNCERTAINTY

The mesh discretization uncertainty U_i for grid i is obtained by combining a calculated safety factor F_s and the discretization error δ_i, described in general terms by Equation D.2.

$$U_i = F_s \left|\delta_i\right| \tag{D.2}$$

For grid convergence studies with more than three grids, the least-squares root approach is applied to calculate U_i [Eça and Hoekstra 2004]. The estimation of δ_i is obtained using the grid solution value, otherwise known as the quantity of interest, ϕ_i, in this work observed as the entropy generation value of grid i. δ_i is assumed to satisfy a (first-order) power series expansion.

$$\delta_i = \phi_i - \phi_c \approx g h_i^p \tag{D.3}$$

ϕ_c is the continuum solution value corresponding with an infinitely fine grid, g is a constant term of the power series, h_i is the cell size of grid i, p represents the apparent order of accuracy of the time and space numerical schemes. When three grids are available, ϕ_c is obtained using the generalized Richardson extrapolation for mixed, or p-th, order methods [Roache 1994]. With more available grids, ϕ_c, g and p are obtained by taking the derivative of the function $S(\phi_c, g, p)$ in Equation D.4 with respect to ϕ_c, g and p and setting to zero.

$$S(\phi_c, g, p) = \sqrt{\sum_{i=1}^{n_g} \left(\phi_i - \left(\phi_c + g h_i^p\right)\right)^2} \tag{D.4}$$

Setting the partial derivatives of $S(\phi_c, g, p)$ with respect to ϕ_c, g and p to zero leads to the expressions in Equation D.5-D.7 respectively [Hoekstra and Eça 2002]. p is calculated iteratively using the implicit Equation D.7.

$$\phi_c = \frac{\sum_{i=1}^{n_g} \phi_i - g \sum_{i=1}^{n_g} h_i^p}{n_g} \tag{D.5}$$

$$g = \frac{n_g \sum_{i=1}^{n_g} \phi_i h_i^p - \left(\sum_{i=1}^{n_g} \phi_i\right)\left(\sum_{i=1}^{n_g} h_i^p\right)}{n_g \sum_{i=1}^{n_g} h_i^{2p} - \left(\sum_{i=1}^{n_g} h_i^p\right)\left(\sum_{i=1}^{n_g} h_i^p\right)} \tag{D.6}$$

$$\sum_{i=1}^{n_g} \phi_i h_i^p \log(h_i) - \phi_c \sum_{i=1}^{n_g} h_i^p \log(h_i) - g \sum_{i=1}^{n_g} h_i^{2p} \log(h_i) = 0 \tag{D.7}$$

Given monotonic convergence of the quantity of interest for the sequence of finer meshes, the mesh discretization uncertainty U_i is determined according to the cases in Equation D.8 [Eça and Hoekstra 2004]. This distinction in safety factors is in order to ensure a conservative U for disparate apparent and theoretical orders of accuracy.

$$U_i = \begin{cases} 1.25 \cdot |\delta_i| + U_s & \text{if} \quad 0.5 < p \leq 2 \\ 1.25 \cdot \max\left(|\delta_i|,\ |\delta_i'|\right) + U_s & \text{if} \quad 2 < p \leq 3 \\ 3 \cdot \max\left(|\phi_j - \phi_{j+1}| \quad \text{for} \, j = 1, .., n_g - 1\right) & \text{else, or if oscillatory convergence} \end{cases} \tag{D.8}$$

In the case of $2 < p \leq 3$, the extended discretization error δ_i' is assumed to be represented by a three-term power series expansion with fixed exponents, as in Equation D.9.

$$\delta_i' = \phi_i - \phi_c \quad \approx g_1 h_i^2 + g_2 h_i^3 + g_3 h_i^3 \tag{D.9}$$

Where the coefficients are determined by regression through a least-squares root approach from minimizing the cost function $S(\phi_c, g_1, g_2, g_3)$, as in Equation D.10. For reference, see Hoekstra and Eça [2002].

$$S(\phi_c, g_1, g_2, g_3) = \sqrt{\sum_{i=1}^{n_g} \left(\phi_i - \left(\phi_c + g_1 h_i^2 + g_2 h_i^3 + g_3 h_i^4\right)\right)^2} \tag{D.10}$$

For a power series with fixed exponents as in Equation D.9, g_1, g_2 and g_3 alongside ϕ_c are thus obtained by solving the following system of equations in Equation D.11.

$$\begin{bmatrix} n_g & \sum_{i=1}^{n_g} h_i^2 & \sum_{i=1}^{n_g} h_i^3 & \sum_{i=1}^{n_g} h_i^4 \\ \sum_{i=1}^{n_g} h_i^2 & \sum_{i=1}^{n_g} h_i^4 & \sum_{i=1}^{n_g} h_i^5 & \sum_{i=1}^{n_g} h_i^6 \\ \sum_{i=1}^{n_g} h_i^3 & \sum_{i=1}^{n_g} h_i^5 & \sum_{i=1}^{n_g} h_i^6 & \sum_{i=1}^{n_g} h_i^7 \\ \sum_{i=1}^{n_g} h_i^4 & \sum_{i=1}^{n_g} h_i^6 & \sum_{i=1}^{n_g} h_i^7 & \sum_{i=1}^{n_g} h_i^8 \end{bmatrix} \begin{bmatrix} \phi_c \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n_g} \phi_i \\ \sum_{i=1}^{n_g} \phi_i h_i^2 \\ \sum_{i=1}^{n_g} \phi_i h_i^3 \\ \sum_{i=1}^{n_g} \phi_i h_i^4 \end{bmatrix} \tag{D.11}$$

In Equation D.8, the standard deviation of the fit U_s is given by Equation D.12. n_f denotes the number of degrees of freedom of the least squares fit. The expression in D.8 used is dependent on whether $|\delta_i|$ or $|\delta_i'|$ respectively is used in the final calculation of U_i.

$$U_s = \sqrt{\frac{\sum_{i=1}^{n_g} \left(\phi_i - \left(\phi_c + g h_i^p\right)\right)^2}{n_g - n_f}} \quad \text{or} \quad U_s = \sqrt{\frac{\sum_{i=1}^{n_g} \left(\phi_i - \left(\phi_c + g_1 h_i^2 + g_2 h_i^3 + g_3 h_i^4\right)\right)^2}{n_g - n_f}} \tag{D.12}$$

E

INTEGRATED CODE TESTING

A regression testing suite is developed to verify extensions to the code. This module mitigates coding errors by ensuring that no functionality is broken upon addition of new code. It is implemented using a standardized approach for code regression testing, by using the Python builtin unit-testing framework.

Three main testing classes are implemented. One contains test functions covering single-point optimizers. The second covers the multi-point optimizers. The third tests the integrated meshing module. Testing full-scale optimization studies is too expensive for unit testing purposes. For this reason the most time consuming functions are mocked, such as the flow simulation methods. Mocking means that the SU2 executables are not called, but instead a dummy result file is copied to take the place of the flow evaluation result.

F

ANALYSIS OF NUMERICAL NOISE VARIANCE

To determine the noise level of the response corresponding to the variation of design variable i, the following approach is used to calculate the variance of the noise. Here, the median of absolute deviations (MAD) is used as proxy for the noise level, being a robust estimator with respect to outliers. Figure F.1 illustrates the calculation of the MAD for the thickness design variables of the parametrized Aachen turbine geometry as considered in this work.

1. The $\mathcal{J}$ is simulated for a set of design variables by varying design variable i between ±20%.
2. A response trend is fitted using a fourth-order polynomial.
3. The MAD is calculated with respect to this fitted trend.

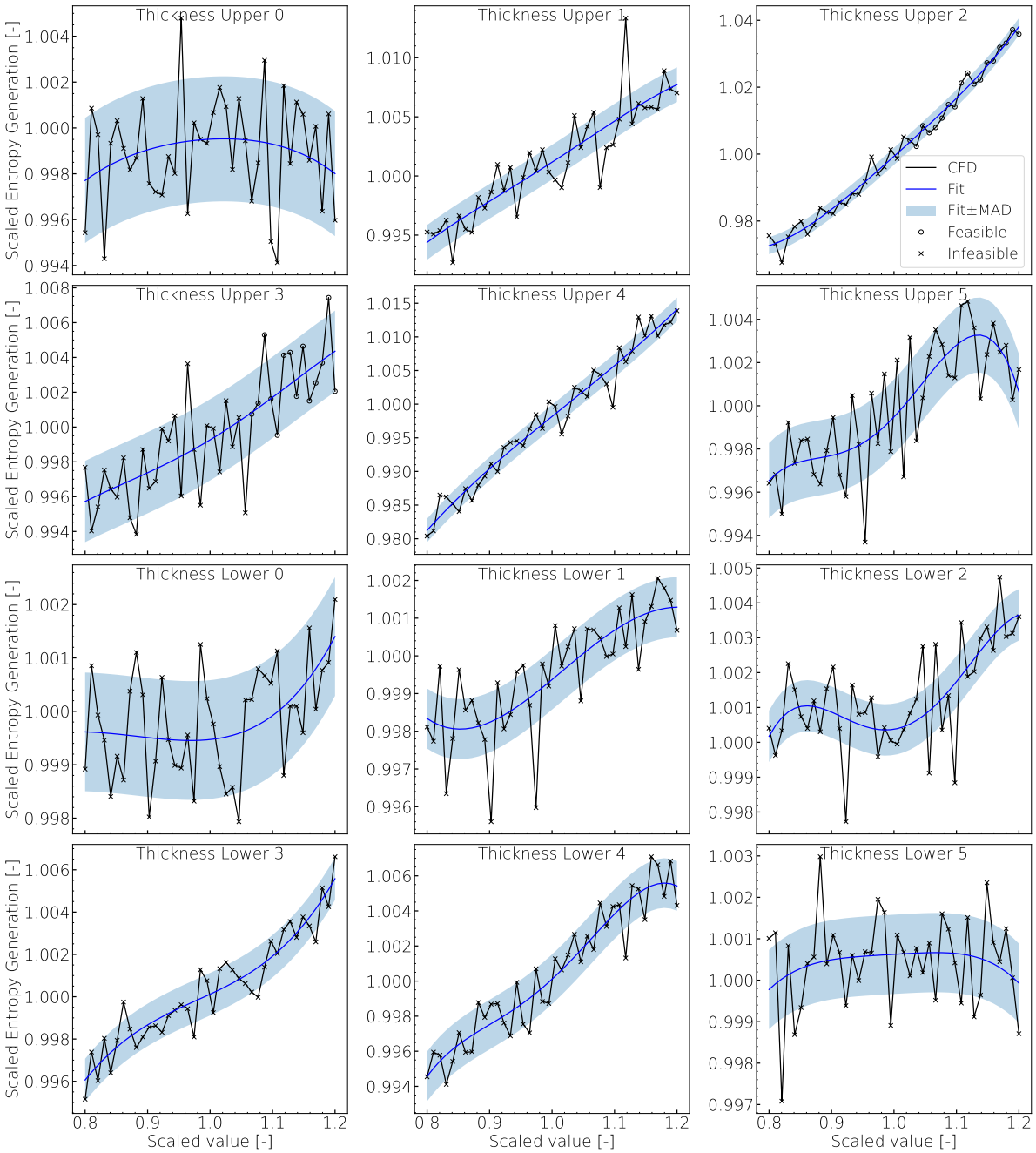


Figure F.1: Visualization of the the numerical noise *level*, for which the median of absolute deviation (MAD) with respect to a fourth-order polynomial is taken as estimate. The shaded area illustrates the noise level for the various design variables. Only blade thickness design variables are shown for purpose of brevity.