

**Efficiency Improvement of Evolutionary
Multiobjective Optimization
Methods for CFD-Based Shape Optimization**

Vom Fachbereich Maschinenbau
an der Technischen Universität Darmstadt
zur
Erlangung des Grades eines Doktor-Ingenieurs
(Dr.-Ing.)
genehmigte Dissertation

von

Hongtao Sun, M. Sc.

aus Liaoning, V. R. China

Berichterstatter:	Prof. Dr. rer. nat. M. Schäfer
Mitberichterstatter:	Prof. Dr. rer. nat. S. Ulbrich
Tag der Einreichung:	24. November 2009
Tag der mündlichen Prüfung:	14. April 2010

Darmstadt 2010

D17

Preface

This thesis contains the outcome of my research in the last five years at the institute of Numerical Methods in Mechanical Engineering at TU Darmstadt.

There are many people that helped, inspired and encouraged me to progress and complete this thesis, to whom I am deeply thankful. First and foremost, I would like to express my gratitude and appreciation towards Prof. Dr. rer. nat. Michael Schäfer for his great supervision, support and encouragement during the whole work. I also thank Prof. Dr. rer. nat. Stefan Ulbrich for kindly accepting to become the co-advisor of this thesis.

My gratitude is extended to all the colleagues at our institute for the support and friendship that created a wonderful and motivating working environment. Particularly I would like to thank Dr.-Ing. Zerrin Harth for the pleasant collaboration, thank Michael Kornhass, Plamen Pironkov, Gerrit Becker, Johannes Siegmann, Dr.-Ing. Markus Heck, Yu Du, Dr.-Ing. Dörte Sternel for the fruitful discussions. Special thanks go to our system administrator Michael Fladerer for his availability and willingness to solve all kinds of software problems and our secretary Monika Müller for her kind help on a lot of things. Furthermore I would like to thank Dr. Andreas Schönfeld for his valuable suggestions on the efficient computing on HHLR, Gary Hachadorian for the extensively grammatical and linguistic correction of this thesis.

I am also profoundly thankful to my parents who did all they could to support me. Last, but not the least, I thank my husband, Yinghua Wang for all his enormous affection and incredible patient during these five years. This dissertation is dedicated to my parents and my husband.

Hongtao Sun

Darmstadt, Germany
November 2009

Table of Contents

Table of Contents	i
List of Tables	iii
List of Figures	iv
1 Introduction	1
1.1 Motivation	1
1.2 State of the Art	2
1.3 Scope of the Work	4
1.4 Overview of the Work	4
2 Foundations of Flow Shape Optimization	6
2.1 Numerical Flow Simulation	7
2.2 Shape Variation	8
2.2.1 General Aspects	8
2.2.2 Free Form Deformation	9
2.3 Optimization Fundamentals	11
2.3.1 Optimization Problem	11
2.3.2 Optimization Methods	13
2.4 Automated Shape Optimization Loop	14
3 Multiobjective Optimization Methods	16
3.1 Multiobjective Optimization Problem	16
3.1.1 Pareto-optimal Concepts	16
3.1.2 Classical Methods	18
3.1.3 Evolutionary Algorithms	21
3.2 Modified NSGA-II	22
3.2.1 Initialization	24
3.2.2 External Population and Final Selection	24
3.2.3 Parallel Structure	25
3.2.4 Optimization Procedure	27
4 RBFN-Based Approximation Model	28
4.1 Introduction	28
4.1.1 Network Structure	28

4.1.2	Radial Basis Functions	29
4.2	Network Training	30
4.2.1	Determination of Training Size	31
4.2.2	Determination of Output Coefficients	32
4.2.3	Determination of Network Centers	33
4.3	RBFN Summary	36
5	Hybrid Optimization Technique	38
5.1	Global Search	39
5.1.1	Global Search Procedure	39
5.1.2	Control Generation	39
5.2	Local Search	41
5.2.1	Starting Points of Local Search	41
5.2.2	Multiobjective Problems	43
5.2.3	Deterministic Optimization Methods	45
5.2.4	Local Optimization Procedure	47
5.3	Test Cases	48
5.3.1	Analytical Test Case 1 - ZDT1	50
5.3.2	Analytical Test Case 2 - FON	55
5.3.3	Numerical Test Case 1 - Pipe Junction	58
5.3.4	Numerical Test Case 2 - Heat Exchanger	65
6	Proper Orthogonal Decomposition (POD)-Based Reduced-Order Model	74
6.1	Proper Orthogonal Decomposition	75
6.2	Combined Interpolation Approach	76
6.3	Optimization Procedure	78
6.4	Test cases	79
6.4.1	Test Case 1 - Pipe Junction	79
6.4.2	Test Case 2 - Heat Exchanger	83
7	Conclusions	91
	Bibliography	94

List of Tables

4.1	Radial Basis Functions	30
5.1	Global optimization parameters	42
5.2	Approximation control parameters (ZDT1)	52
5.3	Performance comparison of the optimal solutions after global search (ZDT1)	53
5.4	Performance comparison of Pareto solutions with different p_0 (ZDT1)	54
5.5	Approximation control parameters (FON)	57
5.6	Performance comparison of the optimal solutions after global search (FON)	58
5.7	Approximation control parameters (pipe - 8 DVs)	62
5.8	Optimization performance comparison (pipe - 8 DVs)	64
5.9	Optimal solution obtained by NSGA-II+CONDOR (pipe - 8 DVs)	65
5.10	Approximation control parameters (heat exchanger - 4 pipes)	68
5.11	Performance comparison after global search (heat exchanger - 4 pipes)	71
5.12	Final Pareto-optimal solutions (heat exchanger - 4 pipes)	71
6.1	Comparison of CPU time (pipe - 4 DVs)	82
6.2	Comparison of optimization results (pipe - 4 DVs)	83
6.3	Comparison of CPU time (fin-tube heat exchanger)	87
6.4	Four exemplary optimal solutions (fin-tube heat exchanger)	90
6.5	Performance comparison of two optimization runs (fin-tube heat exchanger)	90

List of Figures

2.1	Methodology of numerical flow shape optimization	6
2.2	Illustration of original (left) and deformed shape (right) using FFD	11
2.3	A general flowchart of CFD-based shape optimization	15
3.1	Illustration of dominance concept, Pareto-optimal and reference vectors	17
3.2	Illustration of weighted sum method	19
3.3	Illustration of weighted metric method	20
3.4	Illustration of ε -constraint method	21
3.5	Illustration of crowding distance	23
3.6	Generation of four sampling points in 2D design space using LHS	25
3.7	Archive population A_g	26
3.8	Master-slave model for parallel function evaluations	26
3.9	Flowchart of modified NSGA-II	27
4.1	RBFN architecture	29
4.2	Generation of regression tree	35
4.3	Structure of regression tree	36
5.1	Evolutionary optimization procedure	39
5.2	Working procedures in the control generation	40
5.3	Clustering method	43
5.4	Local search using pseudo-weights	44
5.5	Determination of nadir point, ideal point and scale region	45
5.6	Illustration of a local search case	45
5.7	Local search procedure	48
5.8	Illustration of hypervolume of a bi-objective optimization problem	49
5.9	Approximation error of the 1st objective against RBFN models (ZDT1)	52
5.10	Approximation error of the 2nd objective against RBFN models (ZDT1)	53
5.11	Approximation error and q in control generations (ZDT1)	54
5.12	Optimization results comparison after global search (ZDT1)	55
5.13	Comparison of approximation error and q with different p_0 (ZDT1)	56
5.14	Comparison of final optimization results (ZDT1)	56
5.15	Approximation error and q in control generations (FON)	57
5.16	Optimization results comparison after global search (FON)	58
5.17	Comparison of final optimization results (FON)	59
5.18	Sketch of the initial geometry configuration (pipe - 8 DVs)	60

5.19	Shape box discretization and the selected control points (pipe - 8 DVs)	60
5.20	Deformation directions and the corresponding DVs (pipe - 8 DVs)	60
5.21	Pressure contour of the initial configuration (pipe - 8 DVs)	61
5.22	Approximation error and q in control generations (pipe - 8 DVs)	63
5.23	Convergence history of all optimization runs (pipe - 8 DVs)	64
5.24	Pressure contour of initial and optimal configuration in xy -plane (pipe - 8 DVs)	65
5.25	Pressure contour of initial and optimal configuration in xz -plane (pipe - 8 DVs)	66
5.26	Recirculation of initial and optimal configuration in xz -plane (pipe - 8 DVs) . .	66
5.27	Sketch of the initial geometry (heat exchanger - 4 pipes)	67
5.28	Approximation error of pressure drop (heat exchanger - 4 pipes)	69
5.29	Approximation error of Nusselt number (heat exchanger - 4 pipes)	70
5.30	Approximation error and q in control generations (heat exchanger - 4 pipes) . .	70
5.31	Optimization results comparison after global search (heat exchanger - 4 pipes) .	71
5.32	Temperature contour comparison (heat exchanger - 4 pipes)	72
5.33	Pressure contour comparison (heat exchanger - 4 pipes)	73
5.34	Comparison of final optimization results (heat exchanger - 4 pipes)	73
6.1	Shape box discretization and the selected control points (pipe - 4 DVs)	80
6.2	Deformation directions and the corresponding DVs (pipe - 4 DVs)	80
6.3	Average reconstruction errors of 256 and 625 snapshots (pipe - 4 DVs)	81
6.4	Pressure contour comparison using 256 snapshots (pipe - 4 DVs)	81
6.5	Pressure contour comparison using 625 snapshots (pipe - 4 DVs)	82
6.6	Comparison of optimization history (pipe - 4 DVs)	83
6.7	Top view of a fin-tube heat exchanger	84
6.8	Selected optimization domain (fin-tube heat exchanger)	84
6.9	Shape boxes and selected control points (fin-tube heat exchanger)	85
6.10	Deformation directions and the corresponding DVs (fin-tube heat exchanger) .	85
6.11	Average reconstruction error of 200 snapshots (fin-tube heat exchanger)	87
6.12	Pressure contour comparison (fin-tube heat exchanger)	88
6.13	x -velocity contour comparison (fin-tube heat exchanger)	88
6.14	Temperature contour comparison (fin-tube heat exchanger)	88
6.15	Pareto solutions achieved by POD evaluations (fin-tube heat exchanger)	89
6.16	Four exemplary optimal shapes (fin-tube heat exchanger)	89
6.17	Pareto front comparison (fin-tube heat exchanger)	90

List of Symbols and Acronyms

Latin Symbols

A	archive population
A_s	area of the temperature surface
\mathbf{A}	autocorrelation matrix used in POD
B	splitting boundary of the regression tree
c_p	specific heat
c_i	constraints of the optimization problem
\mathbf{c}	RBFN center
C	cost function for RBFN training
C	cluster
d	crowding distance
d	normalized Euclidean distance between two objective solutions
$d_{i,\min}$	minimum distance of the i -th Pareto solution to all the other solutions in the Pareto front
$d_{\text{ave},\min}$	average $d_{i,\min}$ of all solutions in the Pareto front
d_C	distance between two clusters
$\mathbf{d}^{\text{ini}}, \mathbf{d}^{\text{def}}$	coordinate vector of the initial and deformed grid point in the physical domain
D	distribution function of the design variable in the design space
D_h	hydraulic diameter
e	internal energy
e_{ave}	average percentage approximation error
e_{max}	maximum allowed approximation error
$f_{m,\text{max}}, f_{m,\text{min}}$	maximum and minimum values of the m -th objective
\tilde{f}	interpolation model used in the trust-region method
\mathbf{f}, f_i	volume force per mass unit
f_i	objective functions of the optimization problem
\mathbf{F}, \mathbf{f}^i	snapshots matrix
\mathbf{g}	deformation vector
\mathbf{g}	POD basis vector
h	heat transfer coefficient
\mathbf{h}, h_i	heat flux
\mathbf{h}, h_m	hidden layer of RBFN
HV	hypervolume

I	sorted indices vector
\mathbf{I}_K	identity matrix of size K
K	size of the RBFN training set
K_{\max}, K_{\min}	maximum and minimum size of the RBFN training set
K_L, K_R	number of data points in the subsets S_L and S_R
M	truncation degree
l_p	l_p metrics with p varying from 1 to ∞
N_{con}	number of constraints
N_{dv}	number of design variables
N_{dv_b}	number of binary design variables
N_{dv_r}	number of real design variables
N_e	number of recalculated solutions in the control generation
$N_{\text{fun},e}$	total number of required exactly evaluated functions
$N_{\text{gen},e}$	total number of required exactly evaluated generations
N_J	number of solutions on Pareto front \mathcal{F}_J
N_{local}	number of starting points for the local search
N_{max}	maximum allowed times for design vector regenerations
N_{obj}	number of optimization objectives
N_p	number of parallel runs
N_{pop}	population size
N_P	number of solutions in the Pareto front
Nu	Nusselt number
p	pressure
p	generation control frequency
p_c	recombination probability
p_m	mutation probability
p_0	number of initial exactly evaluated generations
$\bar{p}_{\text{in}}, \bar{p}_{\text{out}}$	mean pressure of the inlet or outlet cross-section
P	parent population
Pr	Prandtl number
\mathbf{P}	projection matrix
q	scalar heat source
q_{ini}	initial exactly evaluated generations in p generations
q_s	general source term
q_{min}	minimum number of exactly evaluated generations in p generations
q_{ave}	average number of exactly evaluated generations calculated in all control generations
Q	child population
Q	total heat transfer
r_n^m	RBFN radius corresponding to the n -th dimension of input vector and m -th network center
R	combined parent-child population
Re	Reynolds number
\mathbf{s}	coordinate vector of all grid points

$\mathbf{s}^{\text{ini}}, \mathbf{s}^{\text{def}}$	initial and deformed coordinate vector of all grid points
S	size of the database
S	Pareto solution
SCM	set coverage metric
S_L, S_R	subsets obtained by splitting the node of the regression tree
SP	spacing
t	time
$t_{\text{serial}}, t_{\text{parallel}}$	time required by serial and parallel computing
\mathbf{t}^i	shape basis vector casued by a initial displacement of i -th control point
T	temperature
T_{in}	inlet temperature
$\bar{T}_{\text{in}}, \bar{T}_{\text{out}}$	mean temperature of the inlet or outlet cross-section
\mathbf{T}, T_{ij}	Cauchy stress tensor
$u_{x,\text{in}}$	x -component of inlet velocity \mathbf{u}
\mathbf{u}, u_i	velocity vector
$x_{e,\text{h}}$	hydrodynamic entrance region
$x_{e,\text{t}}$	thermal entrance region
x_i^L, x_i^U	low and upper bound of design variable x_i
$x_i^{\text{Lb}}, x_i^{\text{Ub}}$	low and upper bound of binary design variable x_i
$x_i^{\text{Lr}}, x_i^{\text{Ur}}$	low and upper bound of real design variable x_i
\mathbf{x}, x_i	cartesian coordinate
\mathbf{x}, x_i	input layer of RBFN
\mathbf{x}, x_i	design vector
y	output layer of RBFN
\bar{y}_L, \bar{y}_R	mean value of subset S_L and S_R
\mathbf{y}	output vector of a set of training points
z	weighted sum objective.
z	distance between input \mathbf{x} to the RBFN center \mathbf{c}
z_{ave}	average distance between input \mathbf{x} and the point in the database
\mathbf{z}^I	ideal vector
\mathbf{z}^U	utopian vector
\mathbf{z}^N	nadir vector

Greek Symbols

α_m	sharpness coefficient
$\alpha_n, \beta_n, \gamma_n$	total number of control points in three directions
δ_{ij}	Kronecker delta operator
$\boldsymbol{\varepsilon}, \varepsilon_i$	constraint vector defined in ε -constraint method
η_p	efficiency of pressure drop reduction
$\boldsymbol{\eta}$	coordinate vector of the grid point in the logical domain
$\boldsymbol{\eta}^{\text{ini}}, \boldsymbol{\eta}^{\text{def}}$	coordinate vector of the initial and deformed grid point in the logical domain
κ	heat conductivity

λ	eigenvalue
λ	regularization parameter
μ	dynamic viscosity
$\boldsymbol{\pi}^{\text{ini}}, \boldsymbol{\pi}^{\text{def}}$	initial and deformed coordinate of the control point
ρ	material density
$\boldsymbol{\sigma}$	eigenvector
τ	tournament size
ϕ	fundamental polynomial used to construct interpolation model in the trust region method
ϕ	radial basis function
$\boldsymbol{\omega}$	RBFN coefficient
$\boldsymbol{\omega}$	pseudo-weight vector
Γ_{Φ}	diffusion coefficient
Δp	pressure drop
Δp_{ini}	intial pressure drop
Δ	trust radius
ΔT	log-mean temperature difference
$\boldsymbol{\Theta}, \theta^i$	emperical coefficient matrix
Λ	regularization matrix
$\boldsymbol{\Phi}, \phi^i$	RBFN interpolation matrix
Ω	coefficient matrix of RBF interpolation

Other Symbols

∞	infinity
\mathbb{R}	set of rational numbers
\mathcal{D}	design variable space
\mathcal{F}	Pareto front
\mathcal{P}	solution set
\mathcal{S}	search space
\mathcal{Z}	objective space

Acronyms and Abbreviations

ANN	artificial neural network
BIC	Bayesian information criterion
CAD	computer aided design
CFD	computational fluid dynamic
DOE	design of experiments
DV	design variable
EA	evolutionary algorithm
EP	evolutionary programming
ES	evolutionary strategy
FFD	free form deformation
FEM	finite element method
FVM	finite volume method

GA	genetic algorithm
GCV	generalized cross-validation criterion
HHLR	Hessian high performance computer
LHS	Latin hypercube sampling
MLP	multilayer perceptron
MOEA	multiobjective evolutionary algorithm
MOOP	multiobjective optimization problem
MSC	model selection criteria
NSGA	non-dominated sorting genetic algorithm
PAES	Pareto-archived evolutionary strategy
PCX	parent-centric recombination operators
PDE	partial differential equation
POD	proper orthogonal decomposition
PSO	particle swarm optimization
RBF	radial basis function
RBFN	radial basis function network
RSM	response surface model
SA	simulated annealing
SBV	shape basis vector
SBX	simulated binary crossover
SMP	symmetric multiprocessor
SOOP	single-objective optimization problem
SPEA	strength Pareto evolutionary algorithm
SPX	simplex crossover
SQP	sequential quadratic programming
SSE	sum-squared-error
UNDX	unimodal normal distribution crossover

Chapter 1

Introduction

With the development of computer hardware and the availability of parallel computers, computational simulation is playing an ever important role in the early phase of the product development process. The first area of research for computer-aided design optimization of a particular shape was the field of structure analysis applications [14, 48, 54]. In the last decade, computational fluid dynamic (CFD) has also been successfully combined with modern optimization tools for a variety of engineering design applications, e.g., in the fields of aerospace, automotive, turbomachinery and heat transfer. The simulation based optimization is able to fulfill design demands with much less investment on time and money. However, for models with large and complex geometries, a single flow simulation can take days or weeks. There is an ever increasing demand for high performance computers, more efficient flow solvers and optimization methods. This work will address itself to develop an efficient optimization framework that is particularly appropriate for solving CFD-based shape design optimization problems.

1.1 Motivation

CFD based shape design is a highly multidisciplinary problem. Based on an exact examination of the component or system that is to be optimized, one must first derive a CFD model and set up an optimization problem including the design variables, optimization objectives and reasonable constraints. Then appropriate methods should be selected to perform numerical simulation, shape variation and finally optimization to complete the whole computational design process. Therefore, one requires of knowledge of various fields such as mathematics, computer science and engineering. Among all of these factors, the correct choice of optimization method is especially important because it often influences the performance of the optimization strongly.

The selection of the appropriate optimization method is dependent on the specific problem. In fundamental research or engineering applications, the flow shape optimization problems usually involve multiple and concurrent objectives. Without designating a desired result, the optimization results will yield a set of solutions instead of a single one. Another property of flow shape optimization problems is that there is usually no directly available derivative information and the approximation of these derivatives can be very time consuming. Moreover, the optimization problems are usually nonlinear and can be nonconvex or have multiple local

optima. Out of all these considerations, the evolutionary algorithm (EA), employing the principle of natural evolution, seems to be a promising choice. It works on a set of solutions in the design space, which enables the calculations to localize all of the optimal solutions in a single optimization run. As the optimization is conducted, it utilizes stochastic operators to guide the optimization process. Therefore it does not require any derivative information and verifies the process with a global search at the same time. In the last few decades EAs have been successfully applied to various design optimization problems. However, when compared to deterministic optimization methods, the population-based EAs require a much larger number of function evaluations, especially when the design space or the objective space is high dimensional. Usually, they have a poor convergence rate at the regions close to the optima. This limits the application of EA in the CFD field because the flow simulations are usually computationally expensive.

The analysis of pros and cons of the evolutionary optimization method motivates the present work, which will focus on the investigation of different possibilities to improve the optimization efficiency. Moreover, based on the proposed efficient evolutionary optimization methodology, a complete optimization process that will enable the solution of the complex CFD-based shape optimization problems is developed.

1.2 State of the Art

EA belongs to the stochastic optimization method and includes a variety of algorithms based on the mechanisms of nature evolution, i.e., reproduction, recombination, mutation and selection. The main representatives are evolutionary programming (EP), genetic algorithm (GA) and evolutionary strategy (ES), which were first introduced by Fogel[39] in the 1960s, and Holland [58] and Rechenberg [94] in the 1970s. Now they have been developed successfully to the practical search and optimization processes for more than a decade. The first real application of EA for solving multiobjective optimization problems (MOOPs) can be found in [98]. In 1989, Goldberg [45] proposed a multiobjective evolutionary algorithm (MOEA) based on the Pareto dominance concept. After that, different EA schemes have been developed by introducing the non-dominated concept to maintain the diversity. An overview of these methods is given in [22]. Later on, elitism is added in MOEA as another important function. The well-known methods are non-dominated sorting genetic algorithm-II (NSGA-II) [26], Pareto-archived evolutionary strategy (PAES) [69] and strength Pareto evolutionary algorithm 2 (SPEA2) [117], etc.. The convergence properties of EAs are investigated in [15, 42, 53, 95], different selection schemes are compared in [46] and a new adaptive mutation strategy is suggested by Blum in [10]. Furthermore, instead of traditional binary GA, several real-coded GA operators have been proposed to provide a better way to explore the continuous design spaces. The commonly used are the parent-centric recombination operators, e.g., simulated binary crossover (SBX) [23], parent-centric recombination (PCX) [24], and the mean-centric recombination operators, e.g., unimodal normal distribution crossover (UNDX) [90], simplex crossover (SPX) [108], as well as a number of real-coded mutation operators. An overview and a performance study of the real-coded operators can be found in [25, 93]. In [21] it has been argued that choosing parent-centric recombination is better than mean-centric recombination for a steady and reliable search. Besides, the issues about controlling various EA parameters are studied in [32].

Analytical benchmark test cases for the evaluation of different EAs are designed by Deb [20] and a systematic comparison of several MOEAs is performed by Zitzler [116]. The applications combining EAs and CFD solver for solving the shape design optimization problems can be found in [6, 36, 55, 56, 82, 107].

The limitation of the EA's applications lies in the expensive computational cost due to the large number of required flow simulations for objective function evaluations. A great deal of research has been conducted in the interest of overcoming this problem; using cheap and low-fidelity models to substitute the costly high-fidelity flow simulations required by EAs is quite recent and has been receiving increasing interest. In [5], flow models with different discretization levels are utilized as the approximate and high-fidelity models by a GA optimization. In [31, 33], a multilevel shape parameterization is suggested, which is inspired by the multigrid method and varies the number of design variables to reduce the computational cost during the optimization process. Another type of low-fidelity models is the approximation model constructed on a set of database containing the information of the relations between the design variables and optimization objectives. The commonly used approximation techniques are response surface models (RSM), artificial neural networks (ANN) and Gaussian process (kriging models). These models are globally or locally constructed and used to substitute whole or parts of the expensive high-fidelity function evaluations. The overview of the models and the data sampling methods are given in [65, 103]. The most popular form of RSM is the second order polynomial models. In [78] a strategy for coupling EA and quadratic response surface is proposed. However, when working on the problems with a larger number of variables, the accuracy of quadratic models may become questionable. Wang improved the state of art by creating an adaptive quadratic model which approximates the objective function in a gradually reduced design space [110]. Another pseudo response surface model is suggested recently in [85], which deals with each objective independently and is designed so that accuracy is only critical in the optimal regions. Besides, different variations of ANN have been applied to approximate objective functions in evolutionary optimizations, examples can be found in [44, 57, 67, 83]. In [66] a criterion was proposed to decide the frequency at which the approximation models should be used when coupling multilayer perceptron (MLP) network and ES. The applications of employing kriging models with different EA schemes can be found in [11, 34, 35, 105], where the proposed optimization procedures have been applied on standard test functions as well as on the applications such as the optimization of stationary gas turbine compressor profiles, the multipoint airfoil design in aerodynamics and turbine blade firtrrees, etc..

Furthermore, the existing strategies for improving EA efficiency also include the hybridization of local search methods. In [88] a GA is combined with a deterministic hill-climbing method to optimize the rear of a simplified car shape. Similar studies are presented in [28, 30]. However, these works only concentrate on solving the single-objective optimization problem (SOOP). When solving MOOPs, there are more challenges. Recently, the issues with respect to MOOP are considered in [62, 71, 102], but these studies are restricted to solving mathematical optimization functions. Moreover, most of the combined local searches are dependent on the gradient information of first or second order, and the estimation of the information requires additional computational cost that should be avoided in computationally expensive flow optimization problems.

1.3 Scope of the Work

This work is dedicated to provide an efficient optimization methodology based on EA. Without sacrificing the optimization accuracy, it attempts to improve the optimization efficiency from different aspects, such as reducing the number of required function evaluations, the computational cost for a single evaluation or shortening the computational time. The principle idea consists of utilizing parallel function evaluations, employing approximation models and combining the deterministic methods to accelerate the local search. Various issues on the selection of approximation models and deterministic methods, the control of approximation accuracy, and the multiobjective local search are considered and investigated.

The complete optimization framework, which combines the proposed hybrid optimization methodology with the CFD solver and a shape variation technique, i.e., free form deformation (FFD), is designed to solve the flow shape optimization problems. The fluid model is restricted to steady flow and the optimization problem can be multiobjective, nonlinear, nonconvex with local optima and continuous design variables. The flow solver FASTEST, an in-house developed, high performance software to simulate 3D complex flow, is employed to perform the objective function evaluation and calculate sampling points as the database for the training of approximation models. FFD provides a way to change the shape locally and efficiently by moving a limited number of control points. The proposed optimization framework is applied to different test cases. The influences of using different approximation models, different approximation control parameters as well as different local search optimization methods are studied. The optimization performance is investigated through the comparisons of the results obtained by employing approximation models, the final results after the hybridization of local search with the reference results.

Another contribution of the present work is to provide a methodology to construct the approximation model by combining the interpolation methods (spline interpolation or radial basis function interpolation) with the proper orthogonal decomposition (POD) technique with the purpose to approximate the complete flow region in an efficient manner. Applied in the optimization process, this kind of surrogate model has the ability not only to predict the objective functions but also to provide a detailed estimation of the underlying flow region. Also, other design objectives of solutions (both intermediate and final solutions) can be easily accessed during or after the optimization process.

1.4 Overview of the Work

The remainder of the thesis is organized as follows.

Chapter 2 gives the theoretical basis relative to solving flow shape optimization problems. Section 2.1 introduces the governing equations of flow simulation and the flow solver FASTEST. Section 2.2 gives an overview of shape variation methods and details the one applied in this work, i.e., FFD methods. Then Section 2.3 introduces the optimization fundamentals including the optimization problems and methods. Finally a general automated shape optimization process, which employs flow solver FASTEST, FFD and derivative free optimization methods, is presented in Section 2.4.

Chapter 3 first introduces some special properties of multiobjective optimization problems

and gives an overview of the classical and evolutionary optimization methods (Section 3.1). Then it focuses on the employed GA in Section 3.2.

In Chapter 4, the employed approximation model RBFN is presented, which includes a general introduction of the network structure (Section 4.1) and the detailed network training methods (Section 4.2). The last section summarizes the properties and applications of RBFN in this work.

Chapter 5 presents the efficient hybrid optimization methodology (Section 5.1 and 5.2), which includes global search and local search for the purpose of exploring the design space as well as accelerating the optimization convergence. In Section 5.3 two analytical optimization test cases and two numerical shape optimization problems are solved and the optimization results are quantitatively compared.

The construction of the reduced-order model based on POD and the corresponding interpolation methods are detailed in the first two sections of Chapter 6. Then the evolutionary optimization procedure employing this reduced-order model is given in Section 6.3. The issues concerning the efficiency and accuracy of the approximation models as well as the quality of the optimization results are investigated by two shape optimization test cases in the last section.

Finally, Chapter 7 summarizes the main results and contributions of the thesis. Meanwhile, some remarks and prospects for further research are concluded.

Chapter 2

Foundations of Flow Shape Optimization

Due to the high computational expense required for flow simulations around realistic 3D configurations and the improvement of computational fluid dynamic (CFD) techniques, CFD tools are increasingly applied for the shape design and optimization in industry. Flow shape optimization is an interdisciplinary task which requires a good understanding of physics governing each problem. It also involves mathematical knowledge such as the theory of partial differential equations (PDEs), numerical approximation methods such as finite volume method (FVM) and finite element method (FEM), as well as the optimization theory. Basically, as shown in Figure 2.1, the simulation-based flow shape optimization is a combination of three major aspects: the shape parameterization and variation, the efficient and accurate flow solver and a suitable optimization strategy.

In the following sections, introductions of the numerical flow simulation and shape variation as well as an overview of optimization problems and optimization methods are presented. Furthermore, Section 2.4 illustrates a general automated shape optimization procedure.

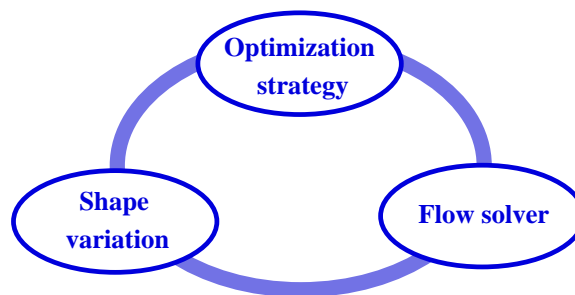


Figure 2.1: Methodology of numerical flow shape optimization

2.1 Numerical Flow Simulation

Fundamental Equations

The motion of a fluid in three dimensions is described by a system of partial differential equations: continuity equation (2.1), momentum equation (2.2) and energy equation (2.3).

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_j)}{\partial x_j} = 0 \quad (2.1)$$

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial T_{ij}}{\partial x_j} + \rho f_i \quad (2.2)$$

$$\frac{\partial(\rho e)}{\partial t} + \frac{\partial(\rho u_i e)}{\partial x_i} = T_{ij} \frac{\partial u_j}{\partial x_i} - \frac{\partial h_i}{\partial x_i} + \rho q \quad (2.3)$$

In the above equations, $\rho, t, q, e, u_i, f_i, T_{ij}, h_i$ denotes the density, the time, the scalar heat source, the internal energy, the components of velocity \mathbf{u} , the components of volume force per mass unit \mathbf{f} , the components of the *Cauchy stress tensor* \mathbf{T} and the components of the heat flux \mathbf{h} in Cartesian coordinates x_i , respectively. For a more detailed description and derivation of the equations one is referred to [7, 8, 38, 97]

This work mainly restricts the optimization model to a steady, incompressible, isotropic Newtonian flow with or without heat transfer. In a Newtonian fluid the viscous stresses are proportional to the rates of deformation. The material law for the Cauchy stress tensor \mathbf{T} is defined as

$$T_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - p \delta_{ij} \quad (2.4)$$

with the pressure p and the dynamic viscosity μ . δ_{ij} is the Kronecker delta operator. For a steady and incompressible fluid, time dependence is not involved, the density ρ is constant and equation (2.1) becomes:

$$\frac{\partial u_i}{\partial x_i} = 0. \quad (2.5)$$

Thus, the last divergence term in equation (2.4) vanishes. Substituting the Cauchy stress tensor into equations (2.2) and (2.3) and omitting the time derivative terms yields the new momentum and energy equations:

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] - \frac{\partial p}{\partial x_i} + \rho f_i, \quad (2.6)$$

$$\frac{\partial(\rho u_i e)}{\partial x_i} = \mu \frac{\partial u_i}{\partial x_j} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \frac{\partial h_i}{\partial x_i} + \rho q. \quad (2.7)$$

Equations (2.5 - 2.7) can be written into a general form

$$\underbrace{\frac{\partial(\rho u_i \Phi)}{\partial x_i}}_{\text{convection}} - \underbrace{\frac{\partial}{\partial x_i} \left(\Gamma_\Phi \frac{\partial \Phi}{\partial x_i} \right)}_{\text{diffusion}} = \underbrace{q_s}_{\text{source}} \quad (2.8)$$

by setting a general variable Φ to 1, u_i , e and selecting appropriate values for the diffusion coefficient Γ_Φ and the source term q_s .

For heat transfer problems in an incompressible fluid, since there is no density variation, the energy equation can be solved separately to the mass conservation and momentum equation. Employing the Fourier's law (for isotropic material)

$$h_i = -\kappa \frac{\partial T}{\partial x_i} \quad (2.9)$$

with the heat conductivity κ and the temperature T , neglecting the work performed by pressure and friction forces and assuming further that the specific heat c_p is constant, the energy conservation equation (2.7) can be simplified to a transport equation for the temperature as follows:

$$\frac{\partial(\rho c_p u_i T)}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\kappa \frac{\partial T}{\partial x_i} \right) + \rho q. \quad (2.10)$$

Flow Solver FASTEST

The flow solver FASTEST (Flow Analysis Solving Transport Equations with Simulated Turbulence) based on the FVM is employed in this work for the numerical flow simulation. It works on 3D block-structured, boundary-fitted hexahedral grids with non-staggered cell-centered grid arrangement [77]. The pressure-velocity coupling is established by using a special variant of the SIMPLE algorithm [76, 91]. Within the pressure-correction scheme, the linear equation system is solved by an ILU method [106]. FASTEST also provides a nonlinear multigrid scheme [60] and the possibility of parallelization for the convergence acceleration. A detailed description about the solver can be found in [1].

2.2 Shape Variation

2.2.1 General Aspects

One of the important issues of the CFD-based shape optimization is shape altering. A simple and straightforward method is called the direct discrete deformation, which deforms the grids directly and defines the deformation on each grid point as the design variable (DV) of the optimization problem. For a given mesh of m grid points, the shape of the model is defined by a $3m$ -dimensional vector \mathbf{s} , which contains the coordinates vector \mathbf{d}^j , $j = 1, \dots, m$, of all grid points, i.e., $\mathbf{s} = [\mathbf{d}^1, \mathbf{d}^2, \dots, \mathbf{d}^m]^T$. Denoting the vector of initial grid points with \mathbf{s}^{ini} and the vector of deformed grid points with \mathbf{s}^{def} , this process can be summarized by

$$\mathbf{s}^{\text{def}} = \mathbf{s}^{\text{ini}} + \mathbf{g}, \quad (2.11)$$

where \mathbf{g} is the deformation vector consisting of $3m$ components and defines the deformation of all grid points. Obviously, when employing this deformation method the number of design variables is dependent on the number of grid points, which would be very computationally expensive for problems requiring a large number of grid points since the optimization cost is

highly dependent on the number of design variables. Furthermore, although it has an efficient grid deformation process, it cannot ensure a smoothly deformed geometry and there is no direct connection to the computer-aided design (CAD) model. Therefore, the selection of an appropriate shape parameterization method is quite important. A good one is usually able to use as few design variables as possible to represent and deform the shape while maintaining the smoothness of the resulting shapes.

The parameterization can be basically divided into two types: CAD-based parameterization and CAD-free parameterization. A CAD-based parameterization method defines the shape by the geometry parameters in a CAD-system such as ProEngineer or CATIA. It is usually applied followed along an automated parametric grid generation which generates the initial grids by a set of CAD-related parameters. During the optimization iterations, the grids are completely regenerated every time when the geometry is modified and the design variables are usually the CAD parameters. The CAD-free parameterization generally employs Bezier or B-spline surfaces to represent the geometry and parameterized directly the discrete surface. The shape is deformed by modifying the position of the control points on the discrete surfaces, whose displacements are employed as the design variables. The distinct advantage of CAD-free parameterization is that the computational grids are deformed simultaneously with the shape variation and therefore the costly remeshing procedure is omitted. CAD-free parameterization method also guarantees a smooth grid deformation and allows conducting only a local shape modification. A disadvantage lies in the difficulty of transferring the optimized shape back into a CAD environment. On the contrary, when using a CAD-based parameterization the computational grids need to be totally regenerated, which can be time consuming for complex geometries and may cause failure by the automated grid generation process. But it provides all the CAD parameters of the deformed shape and is convenient for engineering applications. A summary of the shape variation techniques can be found in [96], and in [41] the CAD-free and CAD-based methods are compared in details regarding aspects such as methodology, parameterization, geometry generation, design variables selection and grid deformation. Generally speaking, the parameterization method defines the formulation of the optimization problem as well as the deformed region and therefore has a major effect on the optimization results. The choice of a proper shape parameterization method is dependent on the individual problem. The decision should be made by taking all the positive and negative aspects into account.

2.2.2 Free Form Deformation

In the present work, the free form deformation (FFD) technique, a CAD-free parameterization technique, is selected to represent and deform the flow shapes. FFD was initially conceived by Sederberg and Parry [101] and has been extended and generalized by Coquillard [18]. It is a powerful tool for a high-level grid deformation known from low-level geometric parameter manipulation. It embeds the object to be deformed into a parametric shape box and instead of modifying the object directly, modifies the shape box based on Bézier or B-spline polynomial parameterization. A set of control points defined on the spline surfaces determine the degree of deformation, namely the deformation flexibility. FFD is successfully implemented combining with the flow solver FASTEST by Harth [51]. It makes a change on the computational grids by working directly on the mesh file generated by FASTEST. The following gives an short overview on its working principle.

In FFD the shape deformation is accomplished by

$$\mathbf{s}^{\text{def}} = \mathbf{s}^{\text{ini}} + \sum_{i=1}^N x_i \mathbf{t}^i, \quad (2.12)$$

where x_i determinates the displacement of the control point and acts as a design variable of the optimization problem. \mathbf{t}^i is the shape basis vector (SBV) giving the deformation direction and the default magnitude of all related grids caused by a initial displacement of the i -th control point. The shape box is first transformed into a unit cube in a logical coordinate system. Before the optimization process begins, the SBVs will be generated once for all by deforming the unit cube. In the logical domain, each point $\boldsymbol{\eta}^{\text{ini}}$ within the cube corresponding to a point $\mathbf{d}^{\text{ini}} = [d_1^{\text{ini}}, d_2^{\text{ini}}, d_3^{\text{ini}}]^T$ in the physical domains is denoted by

$$\boldsymbol{\eta}^{\text{ini}} = \begin{bmatrix} \eta_1^{\text{ini}}(d_1^{\text{ini}}, d_2^{\text{ini}}, d_3^{\text{ini}}) \\ \eta_2^{\text{ini}}(d_1^{\text{ini}}, d_2^{\text{ini}}, d_3^{\text{ini}}) \\ \eta_3^{\text{ini}}(d_1^{\text{ini}}, d_2^{\text{ini}}, d_3^{\text{ini}}) \end{bmatrix}, \quad \eta_1^{\text{ini}}, \eta_2^{\text{ini}}, \eta_3^{\text{ini}} \in [0, 1]. \quad (2.13)$$

The control points $\boldsymbol{\pi}_{\alpha\beta\gamma}^{\text{ini}}$ are generated by equidistantly dividing the unit cube in all directions:

$$\boldsymbol{\pi}_{\alpha\beta\gamma}^{\text{ini}} = \begin{bmatrix} \alpha/\alpha_n \\ \beta/\beta_n \\ \gamma/\gamma_n \end{bmatrix}, \quad \alpha = 0, \dots, \alpha_n, \quad \beta = 0, \dots, \beta_n, \quad \gamma = 0, \dots, \gamma_n, \quad (2.14)$$

where $\alpha_n, \beta_n, \gamma_n$ represent the total number of control points in three directions. The points inside the cube are deformed through the movements of the control points from their initial positions, i.e., $\boldsymbol{\pi}_{\alpha\beta\gamma}^{\text{ini}} \rightarrow \boldsymbol{\pi}_{\alpha\beta\gamma}^{\text{def}}$. The deformed points in cube $\boldsymbol{\eta}^{\text{def}}$, are defined in this work by the product of three Bernstein polynomials $a_\alpha^{\alpha_n}(\eta_1^{\text{ini}})$, $a_\beta^{\beta_n}(\eta_2^{\text{ini}})$ and $a_\gamma^{\gamma_n}(\eta_3^{\text{ini}})$:

$$\boldsymbol{\eta}^{\text{def}} = \sum_{\alpha=0}^{\alpha_n} \sum_{\beta=0}^{\beta_n} \sum_{\gamma=0}^{\gamma_n} a_\alpha^{\alpha_n}(\eta_1^{\text{ini}}) a_\beta^{\beta_n}(\eta_2^{\text{ini}}) a_\gamma^{\gamma_n}(\eta_3^{\text{ini}}) \boldsymbol{\pi}_{\alpha\beta\gamma}^{\text{def}}. \quad (2.15)$$

A general form for the Bernstein polynomial $a_n^m(\eta)$ is

$$a_n^m(\eta) = \binom{m}{n} (1-\eta)^{m-n} \eta^n = \frac{m!}{n!(m-n)!} (1-\eta)^{m-n} \eta^n. \quad (2.16)$$

After the deformation, the grid point $\boldsymbol{\eta}^{\text{def}}$ is mapped back to the physical domain and the corresponding point \mathbf{d}^{def} is obtained. This inverse transformation is carried out with the help of the coordinates of the fixed shape box corners in both the logical and the physical domains. The coordinate difference between the deformed points \mathbf{d}^{def} and the initial points \mathbf{d}^{ini} gives the shape basis vector \mathbf{t} .

An example of 3D shape deformation using FFD is given in Figure 2.2. In this figure, the original geometry is surrounded by a cubic shape box, which is uniformly discretized using three points in x, y, z -direction, respectively. Moving 8 control points yields the deformed shape on the right side.

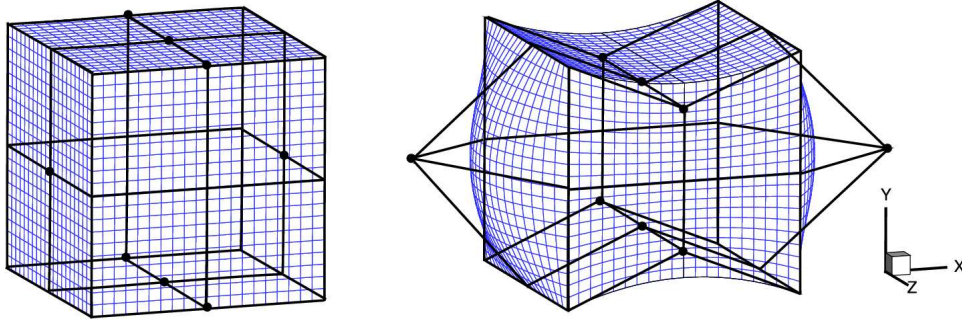


Figure 2.2: Illustration of original (left) and deformed shape (right) using FFD

FFD enables a highly flexible deformation by using fewer numbers of design variables and the deformation is independent on shape complexity and singularities. It also provides the possibility of local shape modification since only the grid points inside the shape box needs to be calculated. The successful applications can be found in [29, 52, 84].

2.3 Optimization Fundamentals

2.3.1 Optimization Problem

The purpose of engineering optimization is to seek the best solution or solutions for a product or process design according to certain measurement criteria within a given set of requirements. The solution is defined by a set of design variables. These criteria and requirements are the objectives and constraints in the optimization problems. A mathematical formulation of the optimization problem is

$$\begin{aligned}
 & \min && f_i(\mathbf{x}), \quad i = 1, \dots, N_{\text{obj}}, \\
 & \text{with} && \mathbf{x} = [x_1, x_2, \dots, x_{N_{\text{dv}}}]^T, \\
 & \text{subject to} && c_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, N_{\text{con}}, \\
 & && x_i^L \leq x_i \leq x_i^U, \quad i = 1, \dots, N_{\text{dv}}.
 \end{aligned} \tag{2.17}$$

In the above formulation, f_i are N_{obj} functions that should be minimized and \mathbf{x} is the design vector with N_{dv} components. c_i are N_{con} equality and inequality constraints that restrict the choice of design vector \mathbf{x} . x_i^L and x_i^U define the lower and upper bound for each design variable x_i and constitute the design variable space \mathcal{D} . The solutions that satisfy all of the inequality, equality and bound constraints are known as the feasible solutions, which constitute the feasible region and also the search space \mathcal{S} in this work. A maximization problem or a *great-than-equal-to* constraint can be considered by multiplying the objective function f_i and constraint c_i by -1.

According to the properties of design variables, objectives and constraints, an optimization problem can be classified into different categories.

Single-objective and Multiobjective

If there is only one objective function, the problem is a single-objective optimization problem (SOOP); if there are more than one objective function, then it is a multiobjective optimization problem (MOOP). The objective functions form the objective space \mathcal{Z} . Each solution in the design space corresponds to one point in the objective space. For multiobjective optimization problems, there is usually no solution that is optimal for all objective functions at the same time. The properties and solution methods of MOOPs will be further explained in Section 3.1.

Constrained and Unconstrained

Most of the real-life optimization problems have one or more restrictions, i.e., equal and unequal as well as the bound constraints. Absolutely unconstrained optimization problems occur often in theoretical and mathematical models, or they are simply the reformulation of the constrained problems, in which the constraints are neglected or replaced by penalization terms added to the objective functions.

Linear and Nonlinear

In linear optimization problem, all objective functions and constraint functions are linear; a non-linear optimization problem is the one that has at least one nonlinear objective function or constraint function. The special difficulties that occur in nonlinear optimization problems include numerical instability, convergence to spurious minima, and slow convergence.

Discrete and Continuous

Discrete or continuous optimization problems are determined by the type of design variables. Discrete problems only contain integers, binary design variables or an ordered set. The set of design variables is finite. However, the design variables of the continuous optimization problem are real numbers and the set of design variables is usually infinite. Another type is the mixed integer programming problem that have both the discrete and continuous design variables.

Convex and Non-convex

The convex function and convex optimization problems are defined in Definition 2.1 and Definition 2.2, respectively.

Definition 2.1 (Convex Function) A function $f : \mathbb{R}^{N_{\text{dv}}} \rightarrow \mathbb{R}$ is a convex function if for any two pair of solutions $\mathbf{x}^a, \mathbf{x}^b \in \mathbb{R}^{N_{\text{dv}}}$, the following condition is true:

$$f(\lambda \mathbf{x}^a + (1 - \lambda) \mathbf{x}^b) \leq \lambda f(\mathbf{x}^a) + (1 - \lambda) f(\mathbf{x}^b), \quad (2.18)$$

for all $0 \leq \lambda \leq 1$.

Definition 2.2 (Convex Optimization Problem) A multiobjective optimization problem is convex if all objective functions are convex and the feasible region is convex (or all the inequality constraints in equation (2.17) are convex and the equality constraints are linear).

A convex function has a positive definite Hessian matrix for all design variables and the local optimum is always the global optimum. A function f that doesn't satisfy equation (2.18) is called a non-convex function. Many optimization algorithms can handle convex MOOPs well, but face difficulties when solving non-convex MOOPs.

Generally, engineering optimization tasks are mostly multiobjective, constrained, nonlinear problems with continuous or discrete design variables.

2.3.2 Optimization Methods

The optimization methods provide an iterative process to improve the solutions by generating new design variables based on the evaluations of the objectives and constraints of one or more previous design variables. A considerable number of optimization algorithms have been developed, which can be divided into two basic categories according to the nature of search process namely deterministic method and stochastic method.

Deterministic methods generate new designs based completely on the previous results by interpolation, extrapolation or gradient information. It can be further divided into derivative-based methods and derivative-free method. Derivative-based methods such as the Newton method or sequential quadratic programming (SQP) require the objective and constraint functions to be continuously differentiable. The first and sometimes the second derivative are employed to determine the search direction. The newly obtained solution is usually better than the previous one, but this kind of methods cannot be applied for solving discrete or combinatorial problems. A main class of derivative-free deterministic methods consisting in modelling the objective functions is embedded in a trust-region framework. It constructs a linear or quadratic model of the objective functions minimize this model inside a trust-region. Other derivative-free methods include the simplex-reflection method of Nelder and Mead, pattern-search method, conjugate-direction method, etc. A brief introduction can be found in [89]. Most of the deterministic methods cannot guarantee a global optima except for the Branch and bound method [72] which is especially used in discrete and combinatorial optimization.

In contrast to deterministic methods that determine the new designs basically from previous results, stochastic methods introduce the randomness into the searching mechanism, which can register as a complete random search process or a random influence on the selected parameters for the applied heuristic strategies. Only after the evaluation of new solutions can it be assessed if the new solutions are the improvement over the old ones. The stochastic methods that are employed most often can be found in [94], which include random search, simulated annealing (SA), stochastic hill climbing, particle swarm optimization (PSO) and evolutionary algorithms (EAs), differential evolution, graduated optimization. A common property of these methods is the global search ability since the randomness provides the necessary impetus to move away from a local solution. Consequently a relatively large number of function evaluations are necessary. But as population-based optimization strategies, they are particularly appropriate for solving multiobjective optimization problems.

The performance of the optimization method can be evaluated by three main properties as follows:

- **Robustness.** Robustness means the capability of reaching a global optimum without

trapping into a local optimum when starting from any initial design. Robustness can be computationally expensive to achieve.

- **Efficiency.** Efficiency is measured by the number of iterations, the number of function evaluations inside one iteration, the computational time as well as the storage required before the optimal solutions are achieved. An efficient method has a faster convergence rate.
- **Accuracy.** Accuracy is the ability to converge to the precise mathematical optimum.

There is no optimization method that is superior in all these three aspects. Deterministic methods converge much faster than stochastic methods but they face the risk of trapping in a local optimum when solving non-convex problems, whereas the stochastic methods are more robust but consequently they need higher computational cost. One always needs to make a trade-off between the robustness and efficiency or between the efficiency and accuracy. Furthermore, the properties of the optimization problem itself should also be considered when selecting an optimization approach, such as if the problem is a linear or nonlinear problem, with single or multiobjective, discrete or continuous design space, convex or non-convex regions, with or without constraints as well as if the derivative information is available, etc..

2.4 Automated Shape Optimization Loop

Solving a shape optimization problem includes three main aspects: flow simulation, shape variation and optimization algorithm. As mentioned above, FASTEST is chosen as the flow solver in this work. FFD is coupled to directly modify the grid data required by FASTEST. Regarding the optimization method, since for flow shape optimization problems the derivative information is usually expensive to calculate and the accuracy is hard to verify, the derivative-free optimization method is preferred. The whole procedure includes grid preparation and optimization loop as illustrated in Figure 2.3.

The grid generation only needs to be conducted once before the optimization process starts. After the FFD set-up, the data including grid information, number and position of the control points as well as the SBVs are prepared for the later shape variation during the iterative optimization process. An automated optimization loop can be accomplished using the optimizer to manage the entire process and integrating the shape deformation and the flow simulation process.

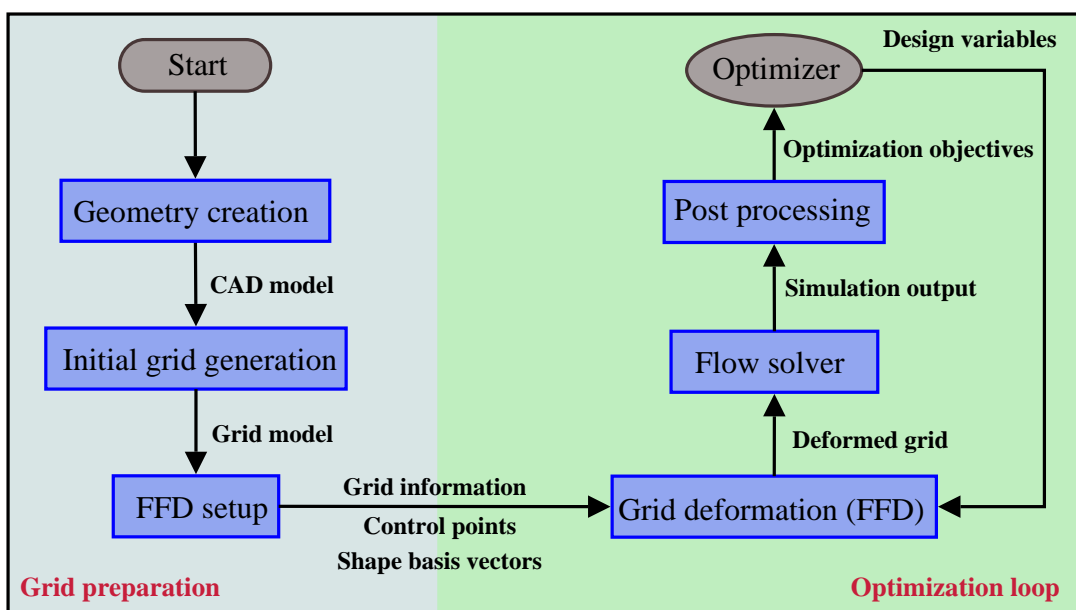


Figure 2.3: A general flowchart of CFD-based shape optimization

Chapter 3

Multiobjective Optimization Methods

Real optimization problems in engineering as well as in fundamental research usually involve more than one objective. The special features of MOOPs, several traditional solution approaches as well as some general aspects of the evolutionary method are given first in Section 3.1. For the consideration that the evolutionary optimization method is particular advantage in solving MOOPs, it is adopted in this work as the global search method in the proposed optimization strategy. Section 3.2 details the applied evolutionary method.

3.1 Multiobjective Optimization Problem

3.1.1 Pareto-optimal Concepts

In this work, only the MOOPs with conflicting objectives are considered, which is also true in most of practical cases. For these kinds of problems, there does not exist one solution which is the optimum of all objectives simultaneously. A MOOP always has a set of optimal solutions, for which there is no way to improve one objective value without deterioration of at least one of the other objective values. Definition 3.1 - 3.4 give a series of general concepts varying from the criterion to measure the solutions to the Pareto front.

Definition 3.1 (Dominance) A design vector $\mathbf{x}^a \in \mathcal{S}$ is said to dominate a design vector $\mathbf{x}^b \in \mathcal{S}$ (denoted $\mathbf{x}^a \leq \mathbf{x}^b$) if:

1. The design vector \mathbf{x}^a is not worse than \mathbf{x}^b in all objectives, i.e., $f_i(\mathbf{x}^a) \leq f_i(\mathbf{x}^b), \forall i = 1, \dots, N_{\text{obj}}$.
2. The design vector \mathbf{x}^a is strictly better than \mathbf{x}^b in at least one objective, i.e., $f_i(\mathbf{x}^a) < f_i(\mathbf{x}^b)$ for at least one $i = 1, \dots, N_{\text{obj}}$.

A design vector $\mathbf{x}^a \in \mathcal{S}$ strongly dominates $\mathbf{x}^b \in \mathcal{S}$ (denoted $\mathbf{x}^a < \mathbf{x}^b$) if the design vector \mathbf{x}^a is strictly better than \mathbf{x}^b in all objectives, i.e., $f_i(\mathbf{x}^a) < f_i(\mathbf{x}^b), \forall i = 1, \dots, N_{\text{obj}}$.

A design vector $\mathbf{x}^a \in \mathcal{S}$ is different to $\mathbf{x}^b \in \mathcal{S}$ (denoted $\mathbf{x}^a \sim \mathbf{x}^b$) if $\mathbf{x}^a \not\leq \mathbf{x}^b \wedge \mathbf{x}^b \not\leq \mathbf{x}^a$.

Definition 3.2 (Nondominated Set) Among a set of solutions \mathcal{P} , the nondominated set of solutions \mathcal{P}' contains those solutions that are not dominated by any member of the set \mathcal{P} .

Definition 3.3 (Pareto-optimal Solution) A design vector $\mathbf{x}^a \in \mathcal{S}$ is called Pareto-optimal if there is no other $\mathbf{x}^b \in \mathcal{S}$ that dominates it. An objective vector \mathbf{f}^a is called Pareto-optimal if the corresponding design vector \mathbf{x}^a is Pareto-optimal.

Definition 3.4 (Pareto-optimal Set) The nondominated set of the entire feasible search space \mathcal{S} , is called the Pareto-optimal set. The Pareto-optimal set in the objective space \mathcal{Z} is called Pareto-optimal front or simply Pareto front, denoted by \mathcal{F} .

For a better understanding of these definitions, Figure 3.1 illustrates a bi-objective minimization problem in the objective space \mathcal{Z} . In this figure, the solution C dominates both solutions D and E and strongly dominates solution E since it has smaller objective values than E for both objectives. Besides, the solution F , G and C are different to each other. The solutions in set \mathcal{P}' construct the nondominated set of solution set \mathcal{P} . Furthermore, C is strongly dominated by A and B , which are not dominated by any other solutions in the entire feasible search space, therefore they are Pareto-optimal solutions. It is clear to see that actually all the solutions lie on the red curves are Pareto-optimal solutions and they constitute the Pareto front. As shown in the figure the Pareto front can be non-convex and also noncontinuous.

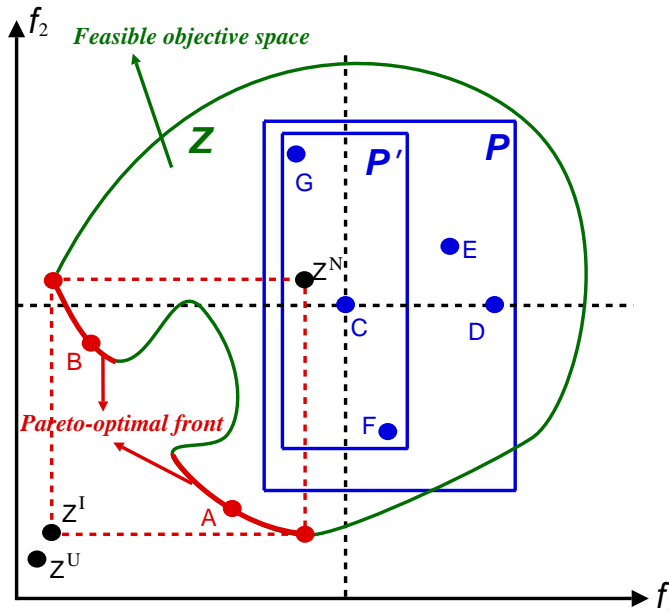


Figure 3.1: Illustration of dominance concept, Pareto-optimal and reference vectors

To solve a MOOP, some special solution vectors relative to the range of Pareto front may be required as reference solutions. As indicated in Figure 3.1, they are ideal vector \mathbf{z}^I , utopian vector \mathbf{z}^U and nadir vector \mathbf{z}^N , respectively. The ideal solution vector is constructed by the best values of each objective in the entire search space. Only if all of the optimization objectives are not conflicting, does a feasible ideal solution that optimizes all the objectives at the same time exist.

Definition 3.5 (Ideal Objective Vector) The components z_i^I of the ideal objective vector $\mathbf{z}^I \in \mathbb{R}^{N_{\text{obj}}}$ are obtained by minimizing each of the objective functions individually subject to the

constraints, i.e., by solving the following problem:

$$\begin{aligned} \min \quad & f_i(\mathbf{x}) \quad i = 1, \dots, N_{\text{obj}}, \\ \text{with} \quad & \mathbf{x} \in \mathcal{S}. \end{aligned} \quad (3.1)$$

On the contrary, the nadir vector \mathbf{z}^N is composed of the upper bound of each objective in the entire Pareto front. Whether it is a feasible solution or not depends on the convexity and continuity of the Pareto front. The exact nadir point is difficult to obtain therefore in practice it is usually approximated using the *payoff table* method [87]. It selects the components of the nadir vector from the vectors $\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^{N_{\text{obj}}}$, whose components are used to construct the ideal vector, by taking the maximum value of each corresponding component, i.e.,

$$z_i^N = \max_{j=1, \dots, N_{\text{obj}}} (f_i^j), \quad i = 1, \dots, N_{\text{obj}}. \quad (3.2)$$

Another useful reference vector, the utopian vector, is defined to strictly dominate every solution in Pareto front.

Definition 3.6 (Utopia Objective Vector) A utopian objective vector $\mathbf{z}^U \in \mathbb{R}^{N_{\text{obj}}}$ is an infeasible objective vector whose components are formulated by

$$z_i^U = z_i^I - \beta_i, \quad i = 1, \dots, N_{\text{obj}}, \quad (3.3)$$

where β_i is a positive parameter.

3.1.2 Classical Methods

An overview of usually used classical methods can be found in [22]. Most algorithms convert the MOOP into a SOOP by using different assumptions. The main restriction of classical methods in solving MOOP is that in each optimization process only one optimal solution can be found and even different initial solutions cannot guarantee different optimal solutions. Furthermore, all the methods depend highly on the selected parameters and require the prior information. Besides, some of the methods have limitations in finding the non-convex solutions. Despite these drawbacks, the key advantage of these classical methods is the fast convergence rate and high efficiency. For this reason, classical methods like the weighting method and ε -constraint method are employed in this work in conjunction with evolutionary optimization process.

Weighted Sum Method

The weighted sum method is a direct way to convert multiple objectives into a single one. When using the weighted sum method, it is advisable to normalize the objectives first so that the objective values are approximately the same magnitude. The normalization is performed in a region defined by the ideal solution and the nadir solution:

$$\frac{f_j - z_j^I}{z_j^N - z_j^I}, \quad j = 1, \dots, N_{\text{obj}}. \quad (3.4)$$

The optimization objective z is then simply a linear combination of all scaled objectives f_j , which is formulated as

$$\min z = \sum_{j=1}^{N_{\text{obj}}} \omega_j f_j, \quad (3.5)$$

where $\omega_j \in [0, 1]$ is the weighting factor of the j -th objective, and it is usually chosen such that $\sum_{j=1}^{N_{\text{obj}}} \omega_j = 1$. An example employing weighted sum method is shown in Figure 3.2 (a), where the isolines of objective values are plotted, whose gradient is the ratio of the weighting factors ω_1 and ω_2 . Obviously, the obtained optimal values are highly dependent on the choice of weighting factors. As illustrated in Figure 3.2 (b), for non-convex problems the real Pareto optimal solution C will never be found by using the weighted sum method.

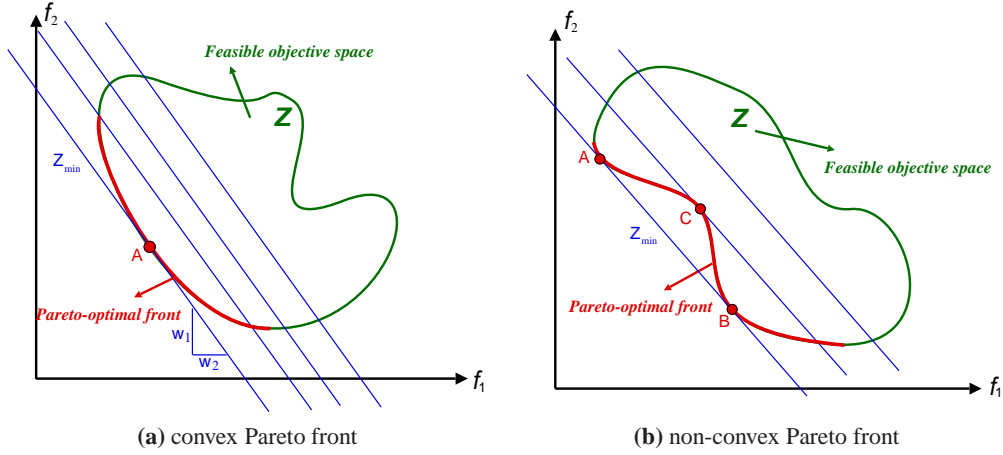


Figure 3.2: Illustration of weighted sum method

Weighted Metric Method

The weighted sum method can be easily extended to higher-order methods, i.e., the weighted metric method by minimizing l_p metrics:

$$\min l_p = \left(\sum_{j=1}^{N_{\text{obj}}} \omega_j (f_j)^p \right)^{1/p}, \quad (3.6)$$

here p can be any value between 1 and ∞ . The weighted metric problem is called a *weighted Tchebycheff* problem if p is ∞ , which is formulated as

$$\min l_\infty = \max_{j \in 1, \dots, N_{\text{obj}}} (\omega_j f_j). \quad (3.7)$$

The increase of value p also increases the difficulty to calculate the gradient, which brings the difficulty to the application of the gradient-based method. But the great benefit is the enlarged explore region, e.g., Figure 3.3 (a) and (b) compare the unreachable Pareto front that lies between solution B and C . Obviously more solutions are obtained by the case $p = 2$. Moreover, the *weighted Tchebycheff* method is supposed to be able to find any Pareto solution

even the non-convex solutions [87].

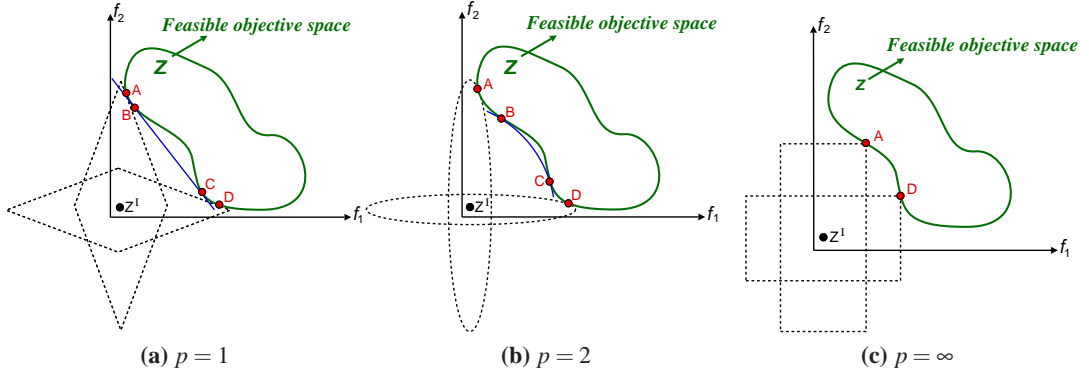


Figure 3.3: Illustration of weighted metric method

ε -constraint Method

The ε -constraint method is first introduced by Haimes *et al* [49]. It selects one objective function to optimize and all the other objective functions are converted into constraints by setting an upper bound to each of them. The optimization problem is formulated as

$$\begin{aligned} \min \quad & f_i(\mathbf{x}), \\ \text{subject to} \quad & f_j(\mathbf{x}) \leq \varepsilon_j, \quad j = 1, \dots, N_{\text{obj}}, \quad j \neq i. \end{aligned} \quad (3.8)$$

Using ε -constraint method, the non-convex solutions can also be found. An illustration is given in Figure 3.4, where A, C, B are three optimal solutions which were achieved when the objective function f_1 is under the constraint ε_1 , ε_2 and ε_3 , respectively. It has been proved in [87] that the unique solution of the ε -constraint method is Pareto-optimal for any given upper bound vector $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_{i-1}, \varepsilon_{i+1}, \dots, \varepsilon_{N_{\text{obj}}}]^T$.

Furthermore, Miettinen summarized the work of Wendell and Lee [111] and Corley [19] in presenting a hybrid method that combines the advantages of both the weighting method and ε -constraint method. Instead of one selected objective, the hybrid method optimizes a weighted sum objective and all the objective functions are employed as constraints. The hybrid problem to be solved is formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{N_{\text{obj}}} \omega_i f_i(\mathbf{x}), \\ \text{subject to} \quad & f_j(\mathbf{x}) \leq \varepsilon_j, \quad j = 1, \dots, N_{\text{obj}}. \end{aligned} \quad (3.9)$$

Using the hybrid method, the achieved Pareto solution is not restricted by the problem convexity and one does not need to solve several problems or think about uniqueness to guarantee the Pareto optimality of the solutions.

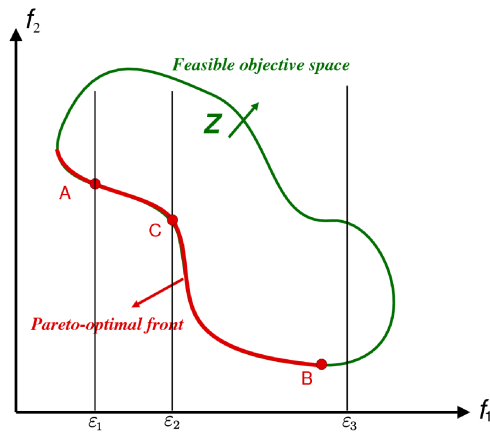


Figure 3.4: Illustration of ε -constraint method

3.1.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) cover a group of search and optimization algorithms based upon the natural evolution. The following details the working principle and properties of EA using genetic algorithm (GA) as an example.

Working Principles

The working principle of GA is to spread a set of solutions in the potential design space in a randomized manner. Each solution, also known as the individual, can be represented by a vector of binary or real-coded parameters. It is then assigned a fitness value related to the objective function in the optimization problem. Thereafter, the solutions are varied iteratively by the selection, recombination and mutation process, which are formulated by mimicking the evolution phenomena in nature, towards an optimal state. The selection operator decides which solutions are maintained and used as parents to produce new solutions for the following generation. A solution with a high fitness value has more chance to be selected as one of the parents than a solution with a low fitness value. New solutions are the combination of existing good solutions with some occasional variations. They are created by recombination and mutation operators. Recombination operator defines the way to create new solutions by combining and varying the selected parent operators. Whereas the mutation operator adds perturbations to the individuals with a probability. It ensures sufficient population to be spread in the decision space, therefore GAs have a global search ability after an infinite computation time. The recombination and mutation operator are actually the exploitation and exploration operator, respectively. The purpose of the recombination operator is to pass the best information of the parent solutions to the child solutions. While the purpose of the mutation operator is to find a better solution which owns completely different characteristics from its parents.

Advantages and Disadvantages

Although conceptually simple, EAs are sufficiently complex to provide robust and powerful search mechanisms for solving MOOPs. The main advantages of EAs are listed as follows:

- EAs work on a set of candidate solutions, therefore unlike the classical methods, they are able to approximate the Pareto solutions in one complete optimization run.
- As stochastic methods, EAs are designed to have the ability to find both global and local optima.
- They are derivative-free optimization methods, which makes them attractive for solving fluid flow optimization problems, for which it is usually not possible to get easy access to the derivative information. Consequently, EAs can be applied to solving the problems with either discrete or discontinuous design spaces.
- Another key advantage of EAs is that they are easy to implement with parallel computing, which greatly improves the optimization efficiency by utilizing the fast progress in computer technology and hardware.
- EAs work as black-box optimizers and are not problem-dependent.

Considering these advantages, the fact that EA as a population-based method require a large number of function evaluations is less important and can be overcome by employing the parallel scheme or surrogated models instead of expensive exact function evaluations.

3.2 Modified NSGA-II

The applied evolutionary method is based on the nondominated sorting genetic algorithm II (NSGA-II) with several modifications with regard to the initialization method, an external population, an additional selection process as well as the parallel structure, which are detailed in the following sections. NSGA-II is proposed by Deb *et al.* and has demonstrated its performance on a number of multiobjective optimization problems [27, 116, 118]. It belongs to the catalog of elitist strategy, which means the best solutions can be directly passed on to the next generation without having to survive the stochastic operators. The population size is set to be N_{pop} . In each generation g , parent population P_g produces the child population Q_g with size N_{pop} by using appropriate selection, recombination and mutation operators. Selection is performed by crowded tournament selection operator. It compares two randomly selection solutions and returns the one that dominates the other one or the one that has a larger crowding distance than the other one if these two solutions are different to each other. The calculation of the crowding distance will be detailed later. This process is repeated as often as solutions must be chosen. The selected parents then generate the child population Q_g by using recombination and mutation operators. Since the algorithm is designed to be able to deal with continuous and discrete design spaces, both binary and real-coded genetic operators are included, where binary operators can be used to handle the discrete design parameters, and the real-coded operators are used for their superior ability to handle continuous search space optimization problems. The recombination and mutation operators in NSGA-II are the binary operators like two-point binary crossover, bit-wise mutation and real-coded operators like simulated binary crossover (SBX) and polynomial mutation. The NSGA-II works on the combined parent-child population R_g with the size $2N_{\text{pop}}$. The N_{pop} best solutions are chosen from R_g as the parent population of the next generation with the help of Pareto front identification and crowding sort procedures.

Pareto Front Identification

Using Pareto front identification, the solutions are ordered using the Pareto dominance concept. All the Pareto nondominated solutions are denoted as the first Pareto front \mathcal{F}_1 and copied into the next parent population P_{g+1} . The remainder solutions then undergo the nondominated sort again and the Pareto front is denoted as \mathcal{F}_j , $j = 2, 3, \dots$, and filled into P_{g+1} successively as long as the number of solutions in P_{g+1} is less than N_{pop} .

Crowding Sort

The Pareto front identification is repeated until the Pareto front \mathcal{F}_j which cannot be fully accommodated by P_{g+1} . Crowding sort is performed by using a crowding distance to determine which solution in \mathcal{F}_j should be selected to complete the population P_{g+1} . For each solution, it first calculates the distances between this solution and its neighboring solutions in each objective dimension, then adds them together to get a sum distance. As illustrated in Figure 3.5, the crowding distance of k -th solution is the average side-length of the cuboid. The boundary solutions are assigned an infinite distance. These crowding distances are then sorted in descending order. The solution corresponding to a largest crowding distance is least crowded and goes into the population P_{g+1} first. The crowding sort algorithm is given below, where N_j is the number of solutions in Pareto front \mathcal{F}_j , the index I_k^m denotes the solution index of the k -th member in the list sorted according to the m -th objective. Besides, $f_{m,\max}$ and $f_{m,\min}$ are used to express the maximum and minimum values of the m -th objective, respectively.

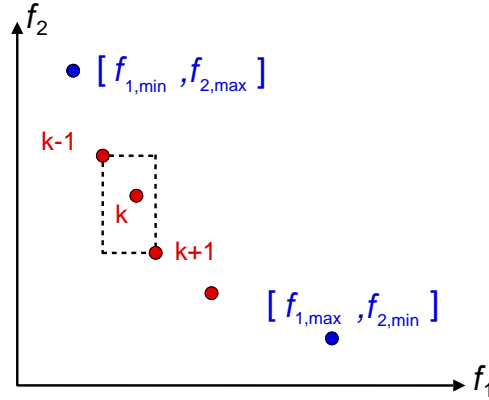


Figure 3.5: Illustration of crowding distance

Algorithm 3.1 Crowding Sort

1. Set crowding distance $d_k = 0$ for each $k \in \{1, \dots, N_j\}$.
2. for $m = 1, \dots, N_{\text{obj}}$ do
 sort the Pareto set \mathcal{F}_j to obtain the sorted indices vector $\mathbf{I}^m = \text{sort}(f_m, >)$.
3. for $m = 1, \dots, N_{\text{obj}}$ do
 $d_{I_1^m} = d_{I_{N_j}^m} = \infty$

$$\text{for } k = 2, \dots, N_{J-1} \text{ do}$$

$$d_k^m = d_{k-1}^m + \left(f_m^{I_{k+1}^m} - f_m^{I_{k-1}^m} \right) / (f_{m,\max} - f_{m,\min}).$$

After this process is completed, the parent population is completely updated and the whole procedure is repeated until the stopping criterion is satisfied.

3.2.1 Initialization

Two options are provided in the modified NSGA-II to generate the initial population. One is random initialization method, where the population is produced at random. This is also the employed initialization method which is used more often. The goal is to include as much information on the design spaces as possible in the initial population using same number of individuals. To this end, Latin hypercube sampling (LHS) [63], an efficient multidimensional sampling method for design of experiments (DOE), has been employed as the other option to explore the initial design spaces.

Using LHS, each design variable x_i in a N_{dv} -dimensional design space has an individual distribution function D_i in the given design region. The first step of the sampling points generation is to divide the range of each design variable into N_{pop} intervals on the basis of equal probability. Then in each interval, one value on the distribution function will be determined randomly and these variables are combined in a random manner without repetition to generate a sampling point. This process continues until all sampling points are generated. LHS offers an efficient method of exploring the design region with a flexible sample size. In Figure 3.6, the generation of four sampling points by LHS is illustrated. In this example, the sampling points are generated in a 2D design space. x_1 and x_2 are associated with uniform and normal distribution, respectively. The range of x_1 and x_2 are subdivided into 4 intervals of equal probability. The subdivisions are represented by the lines that originate at 0.25, 0.5, and 0.75 on the ordinates of Figure 3.6 (a) and 3.6 (b). Then the lines are extended horizontally to the cumulative distribution functions, and dropped vertically to produce the four indicated intervals of design region. Figure 3.6 (c) and Figure 3.6 (d) show two possible sampling results by pairing the randomly chosen values from each interval.

3.2.2 External Population and Final Selection

Although the crowding sort provides a way to select the solutions for next generation according to the diversity, there is still a chance of losing the real Pareto solutions, which happens especially in the latter generations when the combined population contains more than N_{pop} non-dominated solutions. In that case, the real Pareto-optimal solutions may be removed and the dominated solutions are preserved according to their crowding distances. If the optimization runs for infinity generations eventually all the final solutions will be Pareto optimal solutions. However in practice, especially in engineering application EAs usually run a limited number of generations based on the available computer resource and engineering expectations. Therefore, the goal is to keep as many nondominated solutions as possible. Out of this reason, an external archive is suggested for the storage of all the newly generated Pareto solutions obtained in every generation and it is updated during the whole process. Figure 3.7 explains this idea

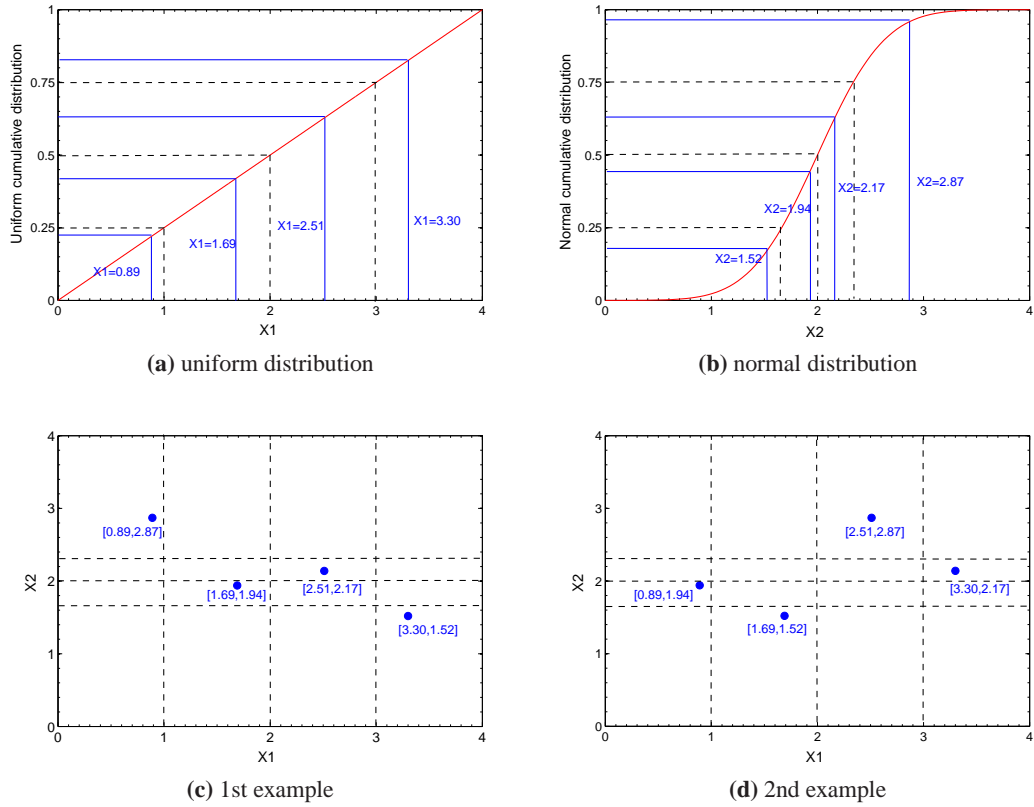


Figure 3.6: Generation of four sampling points in 2D design space using LHS

well, where the red points, black points denote the solutions in the parent and child population, respectively. In the combined population R_g the Pareto front identification is performed and the nondominated solutions are determined, which are represented by the blue squares in the figure. Apparently in this generation the number of nondominated solutions is larger than the population size. N_{pop} of them are selected to go into the next parent population P_{g+1} and meanwhile, the four newly generated nondominated solutions including the ones that are not accommodated in P_{g+1} are stored into the archive population A_g . At last a nondominated search is performed on the whole archive population to get the final Pareto front.

3.2.3 Parallel Structure

The employed NSGA-II is implemented in a parallel scheme to reduce the computational time. A summary of the available parallel evolutionary optimization models is given in [64]. For example, the island model divides the population into several subpopulations and runs a parallel EA, while using the master-slave model only the function evaluations are run in parallel. Considering that for most of the shape optimization problems the cost of function evaluations is computationally much expensive compared to the communication time, and all the function evaluations inside one generation are independent, a master-slave model is employed. In

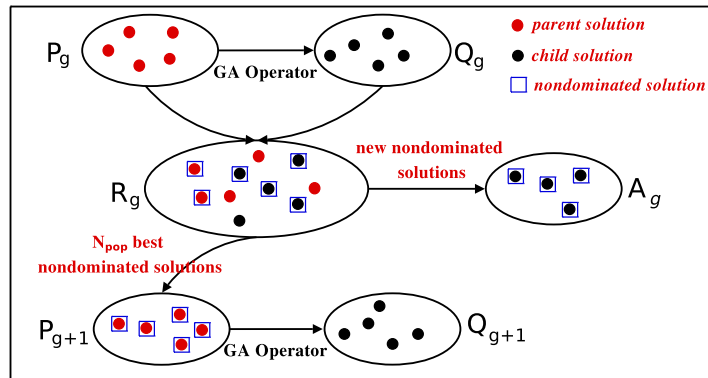


Figure 3.7: Archive population A_g

a master-slave model, one master processor controls the whole optimization process by conducting the genetic operators such as selection, recombination and mutation operators. All the function evaluations are executed on several parallel slave processors. In each evolutionary generation, all function evaluations are submitted to the slave processors at once. It is a synchronized process, which means that only after the function evaluations of all the individuals on slave processors have finished, will the master processor continue conducting genetic operations and the optimization process proceeds onto the next generation. Figure 3.8 shows this working principle.

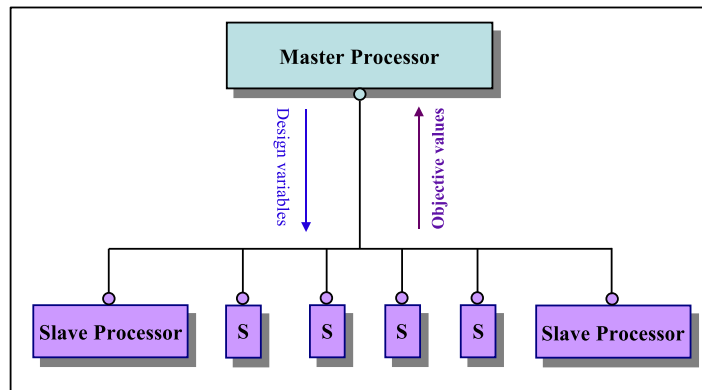


Figure 3.8: Master-slave model for parallel function evaluations

In the test cases of this work, the parallel algorithm works on the Hessian high performance computer (HHLR) that is a cluster of 15 symmetric multiprocessor (SMP)-nodes with a total number of 452 processors. Loadleveler is the employed parallel job scheduling system that allows users to run more jobs in less time by matching each job's processing needs and priority with the available resources, thereby maximizing resource utilization. It is important to emphasize that the master-slave parallelization model does not affect the behavior of the algorithm; the same optimal results can be expected when using a serial scheme. Sending n jobs at one time, the computational time t_{parallel} can ideally be reduced to $1/n$ of t_{serial} , where it is assumed that there are enough free processors so that all the jobs run immediately after being sent and

the communication time is also neglected.

3.2.4 Optimization Procedure

The complete optimization procedure of the modified NSGA-II can be summarized in Figure 3.9.

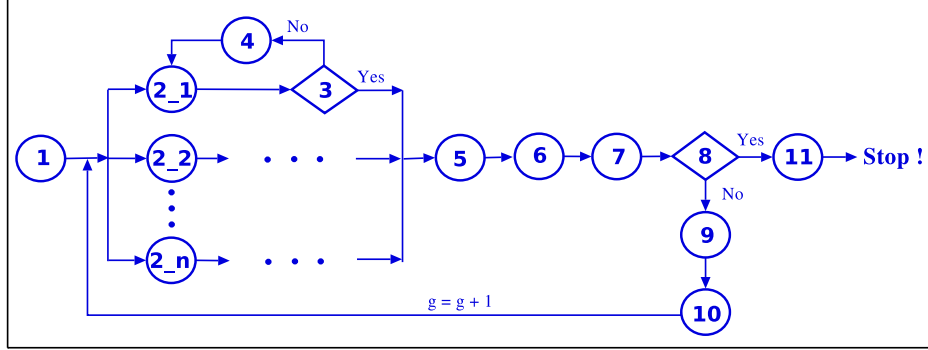


Figure 3.9: Flowchart of modified NSGA-II

1. Initialize parent population P_0 and child population Q_0 .
2. Evaluate the population Q_g at generation g by submitting function evaluations to n parallel processors.
3. Check if the function evaluation (flow simulation) is converged. If it is not converged, then check if the maximum allowed times of design vector regenerations N_{\max} is exceeded.
4. Regenerate the design vector using the mutation operator.
5. Merge population P_g and Q_g into a combined generation R_g .
6. Conduct Pareto front identification to R_g and apply crowding sort on Pareto front \mathcal{F}_J to generate P_{g+1} .
7. Copy all the newly generated Pareto solutions in R_g to the external archive A_g and update A_g .
8. Determine if the stopping criterion is satisfied.
9. Perform crowded tournament selection on P_{g+1} and recombination operator on the selected parent solutions.
10. Perform mutation operator to generate child population Q_{g+1} .
11. Perform Pareto front identification on the current external archive A_g to determine the final Pareto-optimal solutions and terminate.

Chapter 4

RBFN-Based Approximation Model

When solving a CFD-based optimization problem, the function evaluations can sometimes be quite time consuming. Thus, one usually needs to make a trade-off between the optimization efficiency and evaluation accuracy, i.e., using the approximation models to substitute parts or all of the computational expensive CFD simulations required by the optimizer. These approximation models are mostly constructed based on the prior information relative to the relationship between the design variables and objective values stored in a database. In this chapter the approximation model based on radial basis function network (RBFN) is introduced, which is incorporated in the proposed efficient optimization strategy. Section 4.1 gives a general introduction about the network structure and the commonly used radial basis functions (RBFs). The emphasis lies on the network training and the methods of determining the important parameters which are presented in Section 4.2. The last part of this chapter is a summary of the applied models as well as their properties.

4.1 Introduction

4.1.1 Network Structure

Radial basis function approximations were firstly introduced by Hardy in 1971 to represent topographical surfaces given by sets of sparse scattered measurements [50]. As a technique for multivariate interpolation in a high-dimensional space, it has been found to be an effective tool for function approximations. A well-trained RBFN model based on a given input and output data set can be used for predicting the unknown values.

A fully connected and feed-forward network RBFN is composed of three layers: input layer $\mathbf{x} \in \mathbb{R}^{N_{dv}}$, output layer $f(\mathbf{x}) \in \mathbb{R}$ and intermediate hidden layer $\mathbf{h} \in \mathbb{R}^M$, as illustrated in Figure 4.1. Each neuron of the hidden layer corresponds to a point \mathbf{c}^m in the space $\mathbb{R}^{N_{dv}}$, $m = 1, \dots, M$. These points are referred to as network centers and M denotes the number of selected RBF centers. The output layer defined in this work is a scalar and only corresponds to one objective. RBFN performs a non-linear mapping ϕ from the input layer to the hidden layer, followed by a linear mapping ω to the output layer. The term 'feed-forward' refers to the forward direction of computing the output for a given input. The value of the hidden neuron

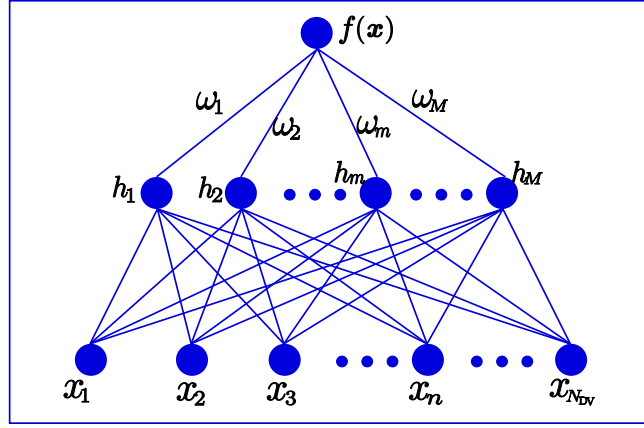


Figure 4.1: RBFN architecture

$h_m, m = 1, \dots, M$ is calculated by

$$h_m = \phi(z_m(\mathbf{x})), \quad (4.1)$$

where ϕ are usually nonlinear functions, known as RBFs, z_m denotes the distance between the input \mathbf{x} and the network center \mathbf{c}^m , which is calculated by

$$z_m(\mathbf{x}) = \sqrt{\sum_{n=1}^{N_{dv}} \frac{(x_n - c_n^m)^2}{(r_n^m)^2}}, \quad (4.2)$$

where the r_n^m is the scaling factor and also called RBF radius. The scaling factors are employed for the consideration that if the components of the input variable \mathbf{x} have widely different scales, without scaling some components may have little influence on the network. Furthermore, the scaling factor combined with a coefficient α_m also determines the sharpness of the function. For each dimension of the input vector and each RBF there is a separate scaling factor r_n^m . It is usually calculated by

$$r_n^m = \alpha_m \left| \max_{k \in \{1, \dots, K\}} (x_n^k) - \min_{k \in \{1, \dots, K\}} (x_n^k) \right|, \quad (4.3)$$

where x_n^k denotes the n -th design variable of the k -th point in the training set and K is the size of the training set. Once the network centers and the coefficient ω are determined, the output $f(\mathbf{x})$ of an arbitrary input can be calculated through the following linear relation:

$$f(\mathbf{x}) = \sum_{m=1}^M \omega_m h_m = \sum_{m=1}^M \omega_m \phi(z_m(\mathbf{x})). \quad (4.4)$$

4.1.2 Radial Basis Functions

A RBF is defined as a real-value function whose value depends only on the distance from the network center. Some of the most commonly used RBFs are listed in Table 4.1. Each of these

functions has its own properties. The Gaussian function and inverse multiquadric function are locally response functions. They only give significant response in the near-center points; the function values decrease with the distance from the center, i.e., $\phi(z) \rightarrow 0$ as $z \rightarrow \infty$. All the other functions in Table 4.1 are global response functions; the function values increase with the distance from the center, i.e., $\phi(z) \rightarrow \infty$ as $z \rightarrow \infty$.

Table 4.1: Radial Basis Functions

	Name	$\phi(z)$
Piecewise smooth	Thin plate spline function	$z^2 \ln(z)$
	Linear function	z
	Cubic function	z^3
Infinity smooth	Multiquadric function	$(1+z^2)^\beta, 0 < \beta < 1$
	Inverse multiquadric function	$(1+z^2)^{-\gamma}, \gamma > 0$
	Gaussian function	$\exp(-z^2)$
	Cauchy function	$1/\sqrt{1+z^2}$

4.2 Network Training

The purpose of network training is to determine the coefficient $\boldsymbol{\omega} = [\omega_1, \omega_2, \dots, \omega_M]$ and network centers $\boldsymbol{c}^m, m = 1, \dots, M$ according to the selected set of points $\{\boldsymbol{x}^k, y_k\}_{k=1}^K$, which are already known as the database. For an exact interpolation, the network centers are defined to be all the points in the training set. Thus, there are as many functions as data points. The coefficient $\boldsymbol{\omega}$ is easily obtained by the network training:

$$\sum_{k=1}^K \omega_k \phi(z_k(\boldsymbol{x}^i)) = y_i, \quad i = 1, \dots, K. \quad (4.5)$$

It can be written in a matrix form as

$$\boldsymbol{\Phi} \boldsymbol{\omega} = \boldsymbol{y} \quad (4.6)$$

with

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi(z_1(\boldsymbol{x}^1)) & \phi(z_2(\boldsymbol{x}^1)) & \cdots & \phi(z_K(\boldsymbol{x}^1)) \\ \phi(z_1(\boldsymbol{x}^2)) & \phi(z_2(\boldsymbol{x}^2)) & \cdots & \phi(z_K(\boldsymbol{x}^2)) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(z_1(\boldsymbol{x}^K)) & \phi(z_2(\boldsymbol{x}^K)) & \cdots & \phi(z_K(\boldsymbol{x}^K)) \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_K \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}, \quad (4.7)$$

where $\boldsymbol{\Phi}$ is the interpolation matrix. According to Micchelli's theorem [86], the infinitely smooth RBFs listed in Table 4.1 will give a nonsingular coefficient matrix $\boldsymbol{\Phi}$, i.e., there is a unique interpolation no matter how the data points are scattered in any number of space dimensions. While for the piecewise smooth functions, interpolation can become singular in the multi-dimensional case. The proof and further discussions can be found in [12, 112]. This work will restrict itself with infinitely smooth RBFs. Besides, Gaussian function and inverse

multiquadric function have special properties, i.e., they are both strictly positive definite functions [37], which ensures a positive definite interpolation matrix Φ . Using the above infinity smooth RBFs, the matrix Φ is invertible and the coefficients ω can be simply calculated by

$$\omega = \Phi^{-1}y. \quad (4.8)$$

However, using all points in the training set as network centers is computationally inefficient to calculate the inverse matrix. On the other side, when the training data is noisy, the function passing through all the data points will be oscillating and unsmooth (overfitting). It means that the network is too sensitive to the details of a particular training set and loses the generalization ability. There are basically two ways to avoid overfitting. One is to use the regularization technique to control the network smoothness. It introduces an extra term in the network training with the purpose to penalize the unsmooth mappings [92]. The other possibility to avoid overfitting is to reduce the number of network centers. Besides, the network centers must not coincide with the training points.

There are mainly two kinds of training procedures [61]. The first approach is a two-step training, which determines the number and position of the network centers in the first step and obtains the output coefficients based on the previously fixed network centers in the second. The other approach performs a gradient-descent optimization on the sum-squared-error (SSE) of the approximated output values and updates the network centers and output coefficients simultaneously using the gradient information. Since the latter approach requires a large computational cost, it is not considered in this work. The following sections presents the methods applied to determine the training set, network centers as well as the output coefficients.

4.2.1 Determination of Training Size

The approximation performance of a RBFN model is dependent on the training set, which should be carefully selected because either a too large or a too small training set can lead to bad approximations. When RBFN is employed as a local approximation model, for each unknown input, not all the members in the database but only several adjacent points are selected for model training. The first advantage of a local model is the lower computational cost for training. The second is the alleged ability to produce better approximations than a globally built model, especially in the case of high-dimensional input space. Algorithm 4.1 defines a way to select the training set for a to be approximated input vector \mathbf{x}^* [43], where S denotes the size of the database.

Algorithm 4.1 Training Set Selection

1. Define the upper and lower bounds of the size of training set K_{\max} and K_{\min} .
2. Calculate the distance z_i , $i = 1, \dots, S$, between \mathbf{x}^* and all the points in the database individually using equation 4.2, where r_n^i is selected by

$$r_n^i = \left| \max_{i \in \{1, \dots, S\}} (x_n^i) - \min_{i \in \{1, \dots, S\}} (x_n^i) \right|. \quad (4.9)$$

3. Sort the distance z_i , $i = 1, \dots, S$ and obtain the sorted indices vector $\mathbf{I} = \text{sort}(z_i, <)$.

4. Calculate the average distance z_{ave} by

$$z_{\text{ave}} = \sum_{j=1}^{K_{\min}} z_{I_j} / K_{\min}. \quad (4.10)$$

5. for $K_{\min} \leq j \leq K_{\max}$ do

if $z_{I_j} / z_{\text{ave}} < 1.5$ then

(a) Add the j -th number into the current training set, set $j = j + 1$.

(b) Recalculate the average distance z_{ave} by

$$z_{\text{ave}} = z_{\text{ave}} \frac{j-1}{j} + \frac{z_{I_j}}{j}. \quad (4.11)$$

else The training set collection stops and the training size K equals to j .

4.2.2 Determination of Output Coefficients

Using the two-step training approach, after the network centers are fixed, the coefficients $\boldsymbol{\omega}$ will be obtained by minimizing the cost function C using the regularization technique:

$$C = \sum_{k=1}^K (y_k - f(\mathbf{x}^k))^2 + \sum_{m=1}^M \lambda_m \omega_m^2, \quad (4.12)$$

where the first part on the right hand side is the SSE between the approximated output of the network and the exact value in the database, the second part is an added penalty term. The fundamental idea of using the penalty term is to stabilize the solution in order to prevent overfitting. λ_m is a positive number that is called the regularization parameter, which controls the trade-off between reducing error and increasing the smoothing. A larger value of λ_m leads to a smoother function. The minimization of C is achieved by differentiating the cost function C and set the derivative to be zero:

$$\frac{\partial C}{\partial \omega_m} = 2 \sum_{k=1}^K (f(\mathbf{x}^k) - y_k) \frac{\partial f(\mathbf{x}^k)}{\partial \omega_m} + 2\lambda_m \omega_m = 0, \quad m = 1, \dots, M, \quad (4.13)$$

where the derivative of the approximation function $f(\mathbf{x}^k)$ can be obtained from equation (4.4):

$$\frac{\partial f(\mathbf{x}^k)}{\partial \omega_m} = \phi(z_m(\mathbf{x}^k)). \quad (4.14)$$

Substituting equation (4.14) into equation (4.13) leads to

$$\sum_{k=1}^K f(\mathbf{x}^k) \phi(z_m(\mathbf{x}^k)) + \lambda_m \omega_m = \sum_{k=1}^K y_k \phi(z_m(\mathbf{x}^k)). \quad (4.15)$$

Thus, minimization of the cost function C equals to solve a linear equation system:

$$\boldsymbol{\Phi}^T \mathbf{f} + \boldsymbol{\Lambda} \boldsymbol{\omega} = \boldsymbol{\Phi}^T \mathbf{y}, \quad (4.16)$$

where

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_M \end{bmatrix}, \quad \mathbf{\Phi} = \begin{bmatrix} \phi(z_1(\mathbf{x}^1)) & \phi(z_2(\mathbf{x}^1)) & \cdots & \phi(z_M(\mathbf{x}^1)) \\ \phi(z_1(\mathbf{x}^2)) & \phi(z_2(\mathbf{x}^2)) & \cdots & \phi(z_M(\mathbf{x}^2)) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(z_1(\mathbf{x}^K)) & \phi(z_2(\mathbf{x}^K)) & \cdots & \phi(z_M(\mathbf{x}^K)) \end{bmatrix} \quad (4.17)$$

and

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_K \end{bmatrix}. \quad (4.18)$$

Substituting equation 4.4 into 4.16, the coefficient $\boldsymbol{\omega}$ then can be obtained by

$$\boldsymbol{\omega} = (\mathbf{\Phi}^T \mathbf{\Phi} + \mathbf{\Lambda})^{-1} \mathbf{\Phi}^T \mathbf{y}. \quad (4.19)$$

4.2.3 Determination of Network Centers

As previously mentioned, using all the points as centers will lead to a minimum SSE. However, it pays too much attention to the specifics of the training set. Therefore when the data is noisy, an overfitting of the data occurs that may not provide a good generalization. The best model is not the one with the least training error but the one with the least estimated prediction error. To measure the prediction error, different model selection criteria (MSC) involving various adjustments to the SSE have been proposed. Two of them are adopted in this work: generalized cross-validation criterion (GCV) [47] and Bayesian information criterion (BIC) [100]. They are defined as

$$GCV = \frac{K \mathbf{y}^T \mathbf{P}^2 \mathbf{y}}{(K - \gamma)^2}, \quad (4.20)$$

$$BIC = \frac{K(\ln(K) - 1) \gamma \mathbf{y}^T \mathbf{P}^2 \mathbf{y}}{(K - \gamma) K},$$

with the projection matrix \mathbf{P} and effective parameter γ , which are defined by

$$\mathbf{P} = \mathbf{I}_K - \mathbf{\Phi}(\mathbf{\Phi}^T \mathbf{\Phi} + \mathbf{\Lambda})^{-1} \mathbf{\Phi}^T, \quad (4.21)$$

and

$$\gamma = K - \text{trace}(\mathbf{P}), \quad (4.22)$$

respectively. \mathbf{I}_K is the identity matrix of size K .

The center determination is accomplished in this work using two representative methods, i.e., regularized forward selection [2] and regression tree method [3] developed by Orr. Instead of fixing the number of network centers in advance, both methods select the centers iteratively from the training set according to the given rules. The algorithms are detailed in the following paragraphs.

Regularized Forward Selection Method

Forward selection has been used by Chen *et al.* to choose RBFN centers [16]. Afterwards, Orr combined the regularization technique with forward selection to improve its generalization performance, i.e., to prevent overfitting. The selection procedure starts from an empty center set $\{\mathbf{c}^m\}_{m=1}^M$. It goes through the whole training set and recursively adds one point into the center set. The chosen point should be the one that mostly minimizes the cost function C . The matrix form of the cost function in equation (4.12) is written as

$$C = (\Phi\omega - \mathbf{y})^T (\Phi\omega - \mathbf{y}) + \omega^T \Lambda \omega. \quad (4.23)$$

Equation (4.23) it can be further transformed as follows by defining $\mathbf{A} = \Phi^T \Phi + \Lambda$ and substituting equation (4.19) and (4.21) into it:

$$\begin{aligned} C &= \mathbf{y}^T (\Phi \mathbf{A}^{-1} \Phi^T - \mathbf{I}_K) (\Phi \mathbf{A}^{-1} \Phi - \mathbf{I}_K) \mathbf{y} + \mathbf{y}^T \Phi \mathbf{A}^{-1} \Lambda \mathbf{A}^{-1} \Phi^T \mathbf{y} \\ &= \mathbf{y}^T \mathbf{P}^2 \mathbf{y} + \mathbf{y}^T (\Phi \mathbf{A}^{-1} \Lambda \mathbf{A}^{-1} \Phi^T) \mathbf{y} \\ &= \mathbf{y}^T \mathbf{P}^2 \mathbf{y} + \mathbf{y}^T \left(\Phi \mathbf{A}^{-1} (\mathbf{A} - \Phi^T \Phi) \mathbf{A}^{-1} \Phi^T \right) \mathbf{y} \\ &= \mathbf{y}^T \mathbf{P}^2 \mathbf{y} + \mathbf{y}^T \left(\Phi \mathbf{A}^{-1} \Phi^T - (\Phi \mathbf{A}^{-1} \Phi^T)^2 \right) \mathbf{y} \\ &= \mathbf{y}^T \mathbf{P}^2 \mathbf{y} + \mathbf{y}^T (\mathbf{P} - \mathbf{P}^2) \mathbf{y} \\ &= \mathbf{y}^T \mathbf{P} \mathbf{y} \end{aligned} \quad (4.24)$$

According to Orr, in each iteration, as a new point \mathbf{c}^i is added into the center set, the projection matrix is updated by

$$\mathbf{P}_{m+1} = \mathbf{P}_m - \frac{\mathbf{P}_m \phi^i (\phi^i)^T \mathbf{P}_m}{\lambda_i + (\phi^i)^T \mathbf{P}_m \phi^i}, \quad (4.25)$$

where ϕ^i denotes the i -th column of the matrix Φ in equation (4.17) and \mathbf{P}_m is the projection matrix obtained by projecting the training set on m network centers. Thus, the search for the next center point is equal to maximizing the difference of the cost functions C_{m+1} and C_m after adding one more point \mathbf{c}^i into the center set $\{\mathbf{c}^j\}_{j=1}^m$:

$$\max C_m - C_{m+1} = \frac{(\mathbf{y}^T \mathbf{P}_m \phi^i)^2}{\lambda_i + (\phi^i)^T \mathbf{P}_m \phi^i}. \quad (4.26)$$

Besides, the corresponding MSC is also calculated in each iteration. The whole selection procedure will stop when the MSC starts increasing due to the network overfitting.

Regression Tree Method

Combining regression trees and RBFN is first proposed by Kubat [70] and improved by Orr. The basic idea is to recursively partition the input space into hyperrectangles that correspond to the nodes of a binary tree. Both the RBFN centers and radii are determined by the tree nodes. The advantage of using a regression tree is that the information provided in the split statistics about the relevance of each input variable is utilized. The procedure using the regression tree method to decide the network centers consists of three main steps: the generation of the

regression tree, the transformation from the tree nodes into network centers and the selection of network centers.

The process of generating the regression tree is illustrated in Figure 4.2. The root node of the tree is defined as the smallest hyperrectangle including all the training points $\{\mathbf{x}^k\}_{k=1}^K$. It is shown in the figure with the name N_0 . The root node is then split along a boundary B in direction n , $n = 1, \dots, N_{dv}$ into two subsets S_L and S_R , which is defined as

$$\begin{aligned} S_L &= \{k : x_n^k \leq B\}, \\ S_R &= \{k : x_n^k > B\}. \end{aligned} \quad (4.27)$$

The mean value of each subset is calculated by

$$\begin{aligned} \bar{y}_L &= \frac{1}{K_L} \sum_{k \in S_L} y_k, \\ \bar{y}_R &= \frac{1}{K_R} \sum_{k \in S_R} y_k, \end{aligned} \quad (4.28)$$

where K_L and K_R denote the number of data points in the subset S_L and S_R , respectively. The splitting is performed by varying the choice of boundary B and dimension n and the one which has the least residual square error between approximated output and exact value should be selected as the way to split the root node. The residual squared error is calculated by

$$E(n, B) = \frac{1}{K} \left(\sum_{k \in S_L} (y_k - \bar{y}_L)^2 + \sum_{k \in S_R} (y_k - \bar{y}_R)^2 \right). \quad (4.29)$$

As shown in Figure 4.2 the initial splitting boundary is determined as B_0 , which splits N_0 into two child nodes N_1 and N_2 . Each child node is split again along the boundary B_1 and B_2 . The splitting is repeated until the child nodes contain fewer points than a predefined number. The corresponding tree structure is given in Figure 4.3.

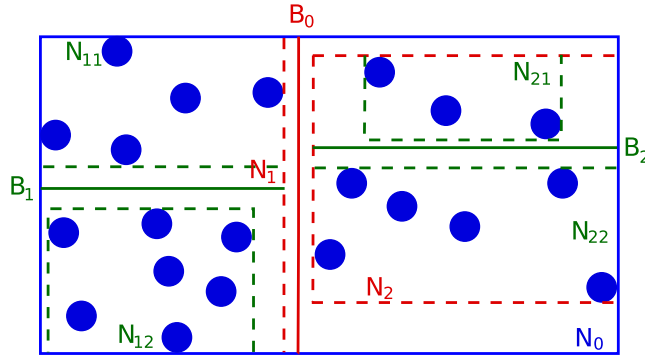


Figure 4.2: Generation of regression tree

After splitting, network centers and radii can be obtained from the child nodes, the trans-

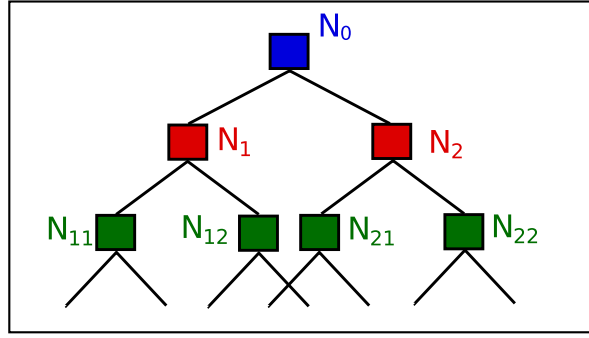


Figure 4.3: Structure of regression tree

formation from the i -th child node to the corresponding center and radius can be stated as

$$c_n^i = \frac{1}{2} \left(\max_{k \in S_i} (x_n^k) + \min_{k \in S_i} (x_n^k) \right) \quad (4.30)$$

and

$$r_n^i = \frac{1}{2} \left(\max_{k \in S_i} (x_n^k) - \min_{k \in S_i} (x_n^k) \right), \quad (4.31)$$

where S_i denotes the points set corresponding to the i -th child node.

The next step is to select the RBFN centers from all of the centers calculated using final child nodes. The selection is accomplished with the help of a search list and consists of the following steps:

1. Initialize the search list and network with a root node.
2. Modify the network by removing the current node, adding one or both of the child nodes.
3. Choose one of the modifications which mostly decreases the estimated prediction error using MSC, update the network accordingly and remove the current node from the search list and add its child nodes to the search list.
4. Repeat the process until the search list contains only the terminal nodes.

4.3 RBFN Summary

Obviously the accuracy of RBFN approximation models depends on the type of applied RBFs, the selected training set as well as the training approach. The reduced center RBFN model has the following properties:

- Not all the points in the database are used as the training set.
- Not all the points in the training set are selected as network centers.
- The network centers may be different from the points in the training (regression tree method).

- Each RBF has its own adjustable scaling factor.

In this work the models will be constructed using two different kinds of center determination methods, i.e., the regularized forward selection and regression tree method, and incorporated into the evolutionary optimization process to approximate the objective function. The methodology and application results will be presented in the next chapter. Also, a comparative study of the model accuracy using these two center selection methods combined with different RBFs and MSC is conducted. Furthermore, the exact RBF interpolation model is combined with proper orthogonal decomposition (POD) to approximate the whole flow region, which is shown in detail in Chapter 6.

Chapter 5

Hybrid Optimization Technique

Evolutionary algorithms have been receiving increasing interest in solving CFD-based shape optimization problems because of their derivative-free property and the capability of dealing with global and multiobjective optimization problems. An essential challenge for the application is the huge time consumption since EAs usually require a large number of function evaluations, which is especially true when the optimization problem has a high-dimensional design space, and the flow analysis may be computationally expensive. In this chapter, an efficient optimization methodology for solving flow shape optimization problems is proposed, which is based on EA for its attractive properties and meanwhile improves optimization efficiency of EA by adopting the following approaches:

- Combine a derivative-free deterministic optimization method with EA to accelerate the local convergence.
- Parallelize the function evaluations in EA to reduce the computational time.
- Construct approximation models to substitute the exact function evaluations to reduce the computational cost.
- Employ a promising control procedure during the evolutionary optimization to improve the approximation accuracy.

The proposed optimization methodology consists of two separate parts: the global search using EA and the local search using deterministic method. The approximation models are incorporated into the global part. The complete optimization procedure is detailed in Section 5.1 and 5.2 together with some special considerations on the approximation control, multiobjective local search and the employed deterministic methods. In Section 5.3, the efficiency of the optimization method, the accuracy of the approximation model as well as the quality of the optimization results are studied and presented by two analytical optimization problems and two flow design test cases.

5.1 Global Search

5.1.1 Global Search Procedure

The global search is the metamodel-assisted evolutionary optimization process. The modified parallel NSGA-II is carried out using both an exact function (high-fidelity flow solver) and RBFN models for function evaluations. This optimization procedure is illustrated in Figure 5.1. The initial population is generated either randomly or using LHS. In the first p_0 generations the objective functions are exactly evaluated. All the design vectors as well as the corresponding objective values are saved into the database for the later RBFN training. Thereafter, in every p generations there are q generations that are exactly evaluated and $(p - q)$ generations that are evaluated using RBFN models. In every p generations the last generation is defined as the control generation, which is used to supervise the following optimization process through the evaluation of the approximation quality. RBFN is employed here as a local approximation model, which means the training set is individually determined for each design vector and a local model is trained afterwards. It is more likely to obtain a better accuracy than the global models when the optimization problem has high-dimensional design spaces. Besides, RBFN works as an online approximation model. The database is consequently updated and the value of q in the next p generation is adaptively changed. Parallel optimization scheme is only activated for solving engineering optimization problems, and only the high-fidelity function evaluations are performed in parallel inside a single generation. The global search stops after a predefined number of generations.

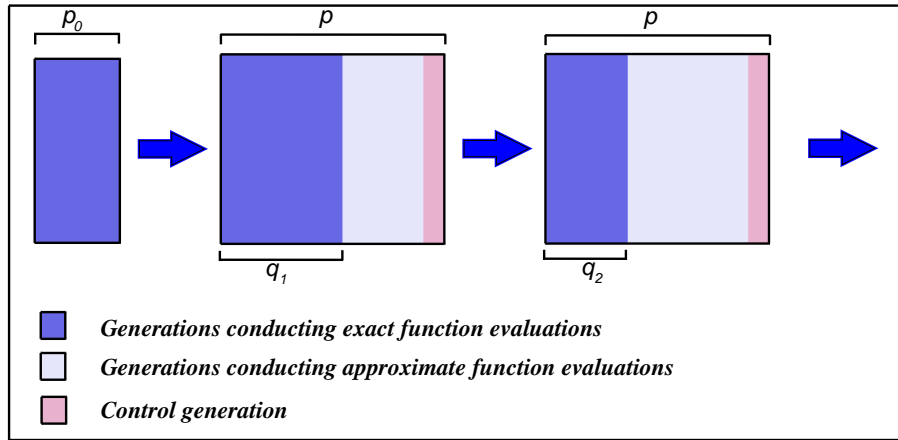


Figure 5.1: Evolutionary optimization procedure

5.1.2 Control Generation

The main function of the control generation is to improve the approximation quality during the optimization process and to ensure the accuracy of solutions staying in the optimization direction, i.e., a correct convergence. In the control generation, the Pareto-front identification and crowding sort selection are performed on the combined generation R , which is evaluated

using both high-fidelity and approximation models. The number of generations conducting high-fidelity function evaluations in the next p generation is determined by calculating the approximation errors. This approach combines the idea of individual control and generation control methods proposed by Jin [66]. Supposing the current parent population is P_g , the control procedure is shown in Figure 5.2 and summarized as follows:

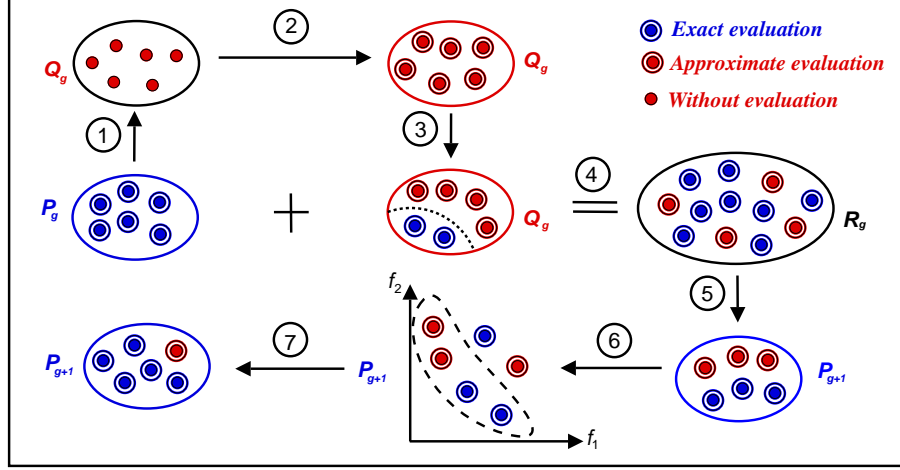


Figure 5.2: Working procedures in the control generation

1. Choose the solutions from parent population P_g using tournament crowded selection, then create offspring population Q_g by crossover and mutation operators.
2. Evaluate Q_g using RBFN models.
3. Conduct high-fidelity function evaluations on the selected N_e individuals, substitute the inexact function values, update the database and calculate the average percentage approximation error for each optimization objective by

$$e_{j,\text{ave}} = \sqrt{\frac{1}{N_e} \sum_{i=1}^{N_e} \left(\frac{\tilde{f}_j^i(\mathbf{x}) - f_j^i(\mathbf{x})}{f_j^i(\mathbf{x})} \right)^2} \times 100\%, \quad j = 1, \dots, N_{\text{obj}}, \quad (5.1)$$

where $f_j^i(\mathbf{x})$ and $\tilde{f}_j^i(\mathbf{x})$ are the exactly and approximately evaluated function values respectively. According to the approximation errors, the number of exactly evaluated generations q_{g+1} in the next p generations is determined by

$$q_{j,g+1} = q_{\min} + \left\lceil \max \left(\frac{e_{j,\text{ave}}}{e_{j,\text{max}}}, 1 \right) (q_{\text{ini}} - q_{\min}) \right\rceil, \quad (5.2)$$

$$q_{g+1} = \max_{j \in \{1, \dots, N_{\text{obj}}\}} (q_{j,g+1}),$$

where $q_{j,g+1}$ is the value of q_{g+1} calculated according to the j -th objective, q_{\min} and q_{ini} are the minimum and initial value of q , $e_{j,\text{max}}$ is the allowed maximum error of the j -th

objective given by the user, and $\lfloor \cdot \rfloor$ represents the maximum integer. It is defined that if one of the current average errors $e_{j,ave}$ is greater than the allowed maximum error, the number of exactly evaluated generations in the next p generations is equal to the initial one.

4. Combine P_g and Q_g into R_g .
5. Perform Pareto front identification and crowding sort to obtain P_{g+1} .
6. Select the Pareto solutions from P_{g+1} .
7. Evaluate the solutions in the current Pareto set exactly, which were obtained by approximation models, and include the newly generated ones into the archive population A_g . Then update parent population P_{g+1} as well as the database.

The global search is designed so that it runs fully automatically without any manual interventions. All the required parameters should be given once for all before the optimization starts. These parameters control the optimization process and approximation accuracy. They have significant influence on the optimization results and should be chosen carefully. Table 5.1 gives a list of the necessary parameters for the global search, which can be basically divided into four classes: optimization control parameters, genetic parameters, approximation control parameters and RBFN parameters.

5.2 Local Search

If the optimization problem has only one objective, the local search will start from the best solution obtained from the above global search. Application of the local optimization method is also straightforward. However, for MOOPs the issues such as how to select initial points for the local search from the current Pareto solutions and how to solve MOOPs using deterministic methods should be considered.

5.2.1 Starting Points of Local Search

In MOOPs, a large number of Pareto-optimal solutions will be obtained after the global optimization run. Since it is inefficient and also unnecessary to use all of them for local search in practice, it is recommended to divide the current Pareto set into several subregions and choose one solution from each subregion as the starting point for the local search. The diversity is a crucial issue for the selected points. Out of this consideration, the clustering method proposed by Zitzler [113] is applied. Actually the clustering method can be carried out on the complete Pareto set or only on a part of it according to the design preference. It divides the selected set into several clusters according to their distances between each other and chooses one solution that is closest to the cluster's centroid in each cluster as shown in Figure 5.3. The algorithm is detailed as follows:

Algorithm 5.1 Clustering Method

1. Initially, each Pareto solution belongs to a distinct cluster C_i , $i = 1, \dots, N_p$, where N_p denotes the number of Pareto solutions.

Table 5.1: Global optimization parameters

Optimization control parameters	Population size	N_{pop}
	Number of generations	N_{gen}
	Number of real DVs	N_{dvr}
	Number of binary DVs	N_{dvb}
	Number of objectives	N_{obj}
	Number of constraints	N_{con}
	Lower boundary of binary DVs	$x_i^{Lb} \ (i = 1, \dots, N_{\text{dvb}})$
	Upper boundary of binary DVs	$x_i^{Ub} \ (i = 1, \dots, N_{\text{dvb}})$
	Lower boundary of real DVs	$x_i^{Lr} \ (i = 1, \dots, N_{\text{dvr}})$
	Upper boundary of real DVs	$x_i^{Ur} \ (i = 1, \dots, N_{\text{dvr}})$
	Function type	0 - analytical problem 1 - engineering problem
	Number of parallel runs	N_p
	Maximum number of DV regenerations (if flow solver does not converge)	N_{max}
	Initialization type	0 - random 1 - LHS
Genetic parameters	Recombination probability	p_c
	Mutation probability	p_m
Approximation control parameters	Number of initial generations	p_0
	Generation control frequency	p
	Initial exactly evaluated generations (in p generations)	q_{ini}
	Minimum number of exactly evaluated generations (in p generations)	q_{min}
	Number of to be recalculated solutions in the control generation	N_e
	Maximum allowed approximation error	$e_{j,\text{max}} \ (j = 1, \dots, N_{\text{obj}})$
RBFN parameters	Type of RBFs	0 - Gaussian function 1 - multiquadric function 2 - inverse multiquadric function 3 - Cauchy function
	Type of MSC	0 - GCV 1 - BIC
	Center selection methods	0 - Regularized forward selection 1 - Regression tree method

2. If the current number of clusters is equal to the expected number of local search starting points N_{local} , go to step 6.

3. Calculate the cluster distances d_C for each pair of clusters C_a and C_b by

$$d_C(C_a, C_b) = \frac{1}{|C_a||C_b|} \sum_{i \in C_a, j \in C_b} d(i, j), \quad (5.3)$$

where d is the normalized Euclidean distance between the solution of \mathbf{x}^i in cluster C_a and \mathbf{x}^j in cluster C_b , and $|C|$ denotes the number of solutions in the cluster C .

4. Find the pair C_a and C_b that corresponds to the minimum cluster distance.

5. Merge these two clusters, go to step 2.

6. Choose one solution from each cluster and remove the others. The chosen solution is the one with the minimum average distance from the other solutions in the cluster.

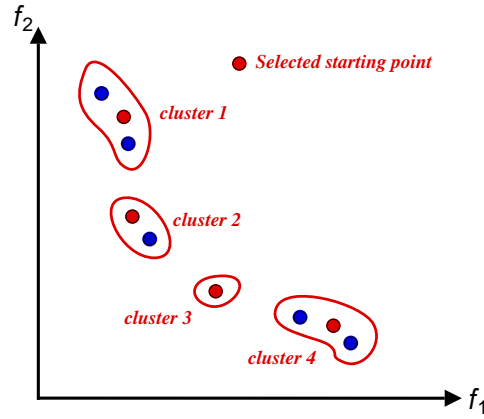


Figure 5.3: Clustering method

5.2.2 Multiobjective Problems

As deterministic methods are not supposed to be used for MOOPs directly, when performing local searches, the multiple objectives are converted into a single objective first. The methods utilizing weighting factors, which include weighted sum method, weighted metric method, and hybrid ε -constraint method, are provided in these optimization framework. To ensure a fast convergence of the local search, each starting point is assigned with a particular weight vector $\boldsymbol{\omega}$, which is called pseudo-weight vector [22]. The pseudo-weight assigned for the j -th objective of the n -th starting point is calculated by

$$\omega_i^n = \frac{(f_{i,\max} - f_i(\mathbf{x}^n)) / (f_{i,\max} - f_{i,\min})}{\sum_{j=1}^{N_{\text{obj}}} (f_{j,\max} - f_j(\mathbf{x}^n)) / (f_{j,\max} - f_{j,\min})}, \quad n = 1, \dots, N_{\text{local}}, \quad (5.4)$$

where $f_{i,\max}$ and $f_{i,\min}$ are the maximum and minimum values of objective f_i respectively. A bi-objective optimization example is illustrated in Figure 5.4, where three Pareto-optimal solutions, A' , B' , C' are obtained by local searches starting from three initial points A , B , C and employing three different weight vectors ω^A , ω^B and ω^C , respectively. For the function normalization, the ideal point and nadir point need to be determined first. In this case, they are achieved by conducting local searches on each initial point that has the optimal value in one of the dimensions in the objective space. It is assumed that the obtained optimal solutions after the local searches define the Pareto region. The point that has all of the best solutions in each dimension is considered as the ideal point. Similarly, the point that has all of the worst solutions in each dimension is treated as the nadir point. Figure 5.5 gives an example, where two objective functions vary in quite different ranges. Using local search, the utmost initial solutions A and E are first optimized to A' and E' with the weighting factors $\omega^A = [1, 0]^T$ and $\omega^E = [0, 1]^T$. The ideal point Z^I and nadir point Z^N are composed of the best and worst values of solutions A' and E' , i.e., $Z^I = [f_1^{A'}, f_2^{E'}]^T$ and $Z^N = [f_1^{E'}, f_2^{A'}]^T$. Afterwards the objectives of all the remaining starting solutions are scaled in the region inside the dotted lines.

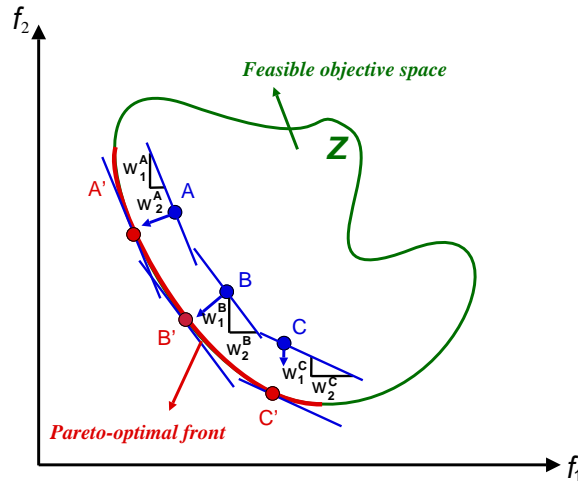


Figure 5.4: Local search using pseudo-weights

The selection of the appropriate method to convert a MOOP to a SOOP is dependent on the individual problem. It can be decided by observing the Pareto set after the global search. For example, Figure 5.6 illustrates a scenario where the arrows represent the calculated local search directions. Obviously the optimal solutions lying on the Pareto region $A'B'$ and $C'D'$ can hardly be found by using the weighted sum method. In this case, one can consider the hybrid ϵ -constraint method by choosing a proper constraint for each local search. If the current solutions show an apparent non-convex Pareto front, then solving the *weighted Tchebycheff* problem should be a good choice.

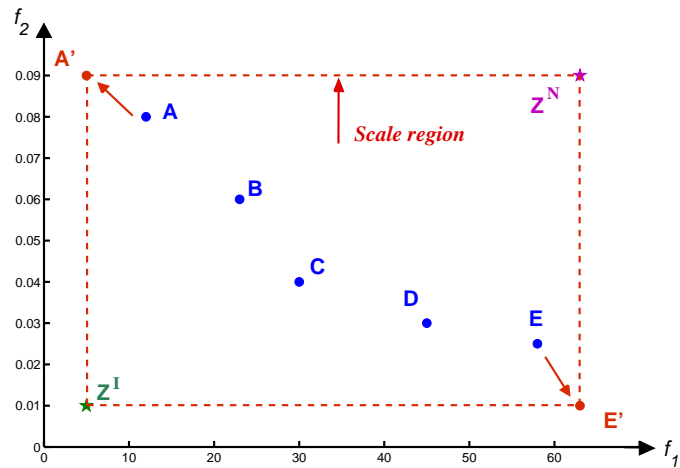


Figure 5.5: Determination of nadir point, ideal point and scale region

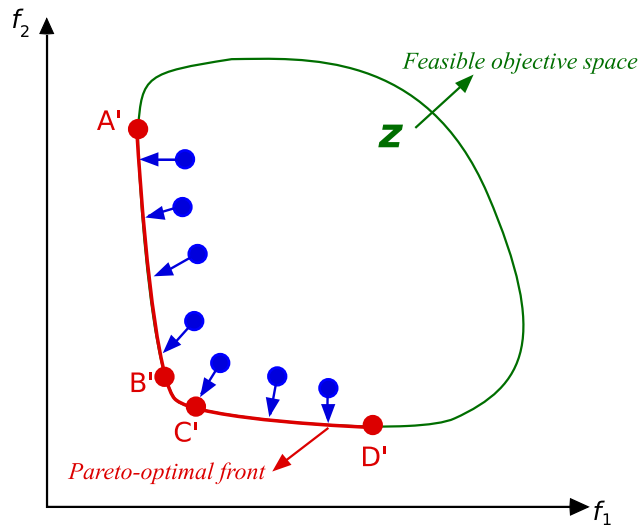


Figure 5.6: Illustration of a local search case

5.2.3 Deterministic Optimization Methods

Two optimization tools are suggested for the local search. One is DFO (Derivative Free Optimization) developed by IBM T. J. WATSON Research Center [17], and the other is CONDOR (CONstrained Non-linear, Direct, parallel, multiobjective Optimization using trust-region method for high-computing load, noisy objective functions) developed by Frank Vanden Berghen [9]. DFO and CONDOR have the similar working principle and both of them employ the derivative-free trust-region method with the idea of combining a trust-region framework with a local approximation model built via polynomial interpolation. In each iteration, instead of optimizing the objective function f , the interpolation model \tilde{f} is optimized within the 'trusted'

region, i.e.,

$$\min f(\mathbf{x}) \simeq \tilde{f}(\mathbf{x}) = \sum_{k=1}^K \alpha_k \phi_k(\mathbf{x}), \quad (5.5)$$

where ϕ_k is the fundamental polynomials and α_k is a nonzero coefficient. The trust radius Δ defines 'trusted' region around the current optimal solution and should be carefully selected. On one hand, it should be as large as possible to achieve the greatest improvement of the optimization objective. However it must be sufficiently local to provide good approximations. Whether an optimization iteration is successful or not is determined by evaluating the real function value f at the new obtained point and comparing the improvement calculated using the real function value and the approximate value. If it is successful, the new point is accepted and used as the center point in the next iteration. The trust radius in the next iteration is enlarged if the improvement is really good. Otherwise the new point is rejected and the size of trust radius should be reduced.

The decision of using DFO and CONDOR as a supplementary local search method is guided by the following considerations:

- They are derivative-free methods. Although in each iteration the optimization of the trust-region subproblem is derivative-dependent, the derivatives of the objective functions with respect to the design variables (DVs) must not be provided by the flow solver because they can be extracted from the interpolation polynomial inside the trust-region.
- As deterministic optimization methods, they have a faster converge rate to the local optimum than EAs. Also, several test cases have shown that DFO outperforms derivative-based methods like the Quasi-Newton method concerning the convergence and accuracy [109].
- Both DFO and CONDOR are designed for solving the optimization problems with box constraints, linear constraints as well as non-linear constraints.
- Both DFO and CONDOR are specially designed for problems with expensive objective function evaluations, which makes them suitable for solving flow shape optimization problems.
- Both DFO and CONDOR require a continuous design region and the dimensions of the design space should not more than 50 due to the applied interpolation technique. The optimization problems considered in this work satisfy these requirements.

Though they are implemented based on the same idea, they have minor differences in several aspects such as the way to conduct optimization inside the trust-region, the determination of the trust-region, and the criteria to update the trust radius. They are also different in the way they build interpolation models. CONDOR uses Lagrange polynomials as basis functions and always constructs a full quadratical model, which requires at least $(n+1)(n+2)/2$ points when the design space is n -dimensional. DFO employs Newton fundamental polynomials as the basis functions. Different to CONDOR, DFO does not necessarily require a full quadratic model. Especially at the beginning, DFO prefers performing optimization on the incomplete interpolation set than evaluating new random points to complete the interpolation set. Hence, DFO is actually oscillate between using a first and second order polynomial.

Comparisons between DFO and CONDOR have been carried out by Berghen concerning the number of function evaluations and the achieved optima on a set of benchmark problems [9]. According to Berghen, in most cases when the dimensions of the design space is larger than two, CONDOR outperforms DFO because CONDOR employs a more accurate interpolation model (full quadratic) to guide the search, which is especially advantage in higher dimensions. Meanwhile, it is also pointed out that if the dimension of the search space is greater than 25, DFO may becomes attractive because CONDOR needs more function evaluations at the beginning of the optimization process to build the full quadratic approximation model, and the cost of this part will increase quadratically when the number of design variables increases. The performance of these two optimization tools are also compared by Harth in [51], where the shape optimization of a pipe conjunction is considered with respect to the pressure drop. The comparisons are performed by using 6 and 14 design variables to define the shape variation, respectively. A remarkably similar optimal pressure drop is achieved by DFO and CONDOR. In case of 6 design variables, the result obtained by DFO is slightly better than CONDOR. The number of function evaluations required by DFO and CONDOR are 118 and 100, respectively. However when the design variables are 14, DFO requires about 30% less function evaluations than CONDOR to achieve the optimum. It is can also be observed that once the model is built, the convergence of CONDOR is very fast. The results by Harth are not completely consistent with those obtained by Berghen. However, both studies have verified that DFO has a shorter initialization phase but a relative long research phase compared to CONDOR. As to the optimization results, neither DFO nor CONDOR can be said to be absolutely better than the other. Regarding the required number of function evaluations, it is agreed that for a higher dimension design space CONDOR requires a larger number of function evaluations, but it seems possible that the definition of higher or lower dimensions is dependent on the optimization problems and there is not a uniform boundary.

5.2.4 Local Optimization Procedure

The local optimization consists of four main steps. A flowchart is given in Figure 5.7. In the first step, the algorithm reads all the Pareto-optimal solutions obtained by the global search, from which the starting points of the local search are selected using the clustering method. Then the maximum and minimum values of each objective are searched to calculate the pseudo-weight vectors for all starting points. In the second step, N_{obj} local searches are performed separately using the selected local optimization method, i.e., DFO or CONDOR. For each local optimization only one of the objectives is considered, and the starting point should be the point with the current optimal value of the corresponding objective. Step three sorts all the objective values of the solutions obtained by local optimizations in step two. The best and worst values of each objective are selected as the components of the ideal and nadir points. In the last step, the remaining $(N_{local} - N_{obj})$ points are locally optimized. Each process involves the normalization of the objective functions using the calculated nadir and ideal points, the formulation of the optimization objective, and the conduction of the local optimization. The local searches in the second and the fourth step are independent from each other and can be performed in parallel.

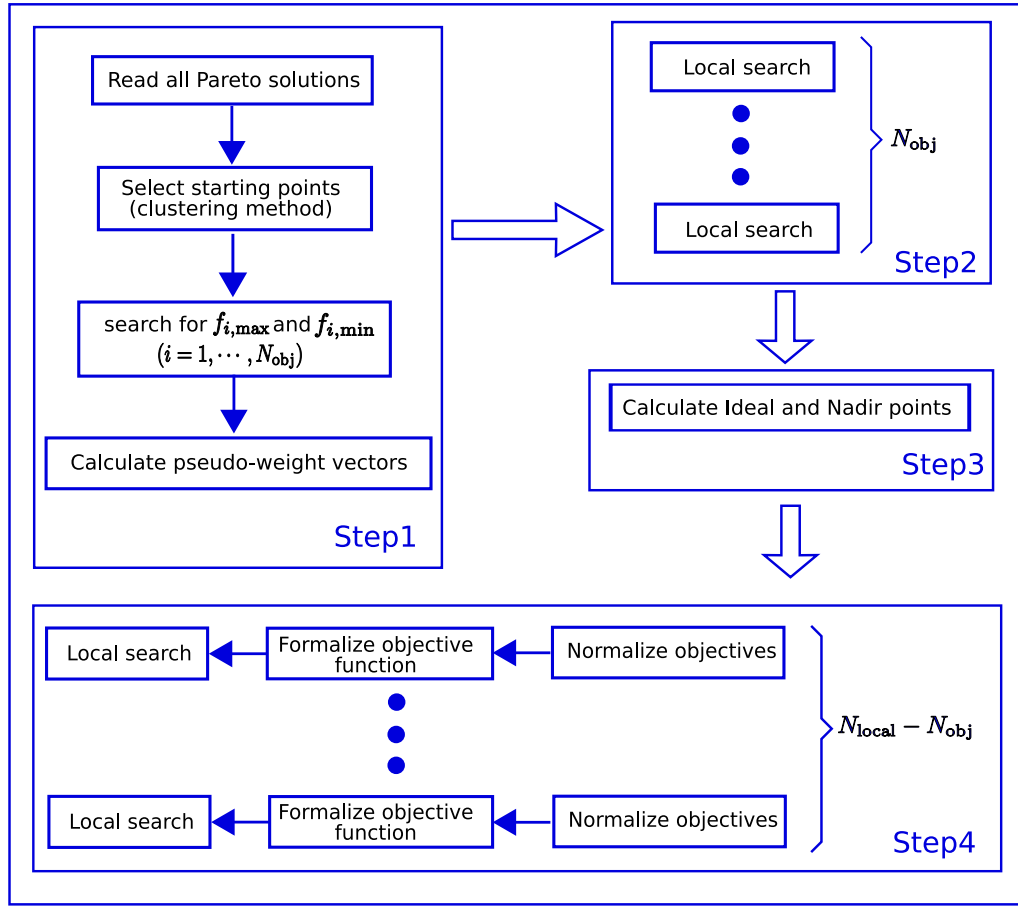


Figure 5.7: Local search procedure

5.3 Test Cases

In this section, two analytical optimization problems, which correspond to a convex and a non-convex Pareto front respectively, are solved. Besides, the first one investigates the influence of the RBFN parameters and approximation control parameters, and the second one verifies the ability of the hybrid optimization method for solving the optimization problem with a non-convex Pareto front. Afterwards, two numerical shape optimization problems are considered, which are a single-objective pipe shape optimization and a bi-objective heat exchanger shape optimization, to illustrate the performance of the hybrid optimization method in engineering applications. In all of the test cases, a set of reference solutions are calculated by the NSGA-II runs, which only employ the exact mathematical function or high-fidelity flow solver to calculate the objective values. The performance of the proposed hybrid optimization methodology is evaluated by comparing the obtained solutions with the reference solutions after both global and local optimizations. In the global part, the comparison is carried out based on the results using same number of exact function evaluations. In the local part, the final solutions using the hybrid method are compared with the reference solutions that are obtained after a relative large number of generations, which will be individually defined for each test case. Besides,

for all of the test cases in this chapter, crowded tournament selection, SBX crossover and real polynomial mutation are employed as the genetic operators.

The optimization performance is evaluated concerning the required computational cost and the quality of obtained optimal values. Because the proposed hybrid optimization method is specifically designed for the engineering shape optimization problems, where the computational cost required by optimization algorithm itself and the approximate function evaluations is negligible compared to the high-fidelity flow simulation, the computational cost is measured based on the number of evaluations conducted by using the flow solver. Also in analytical test cases, it is assumed that the exact function evaluations are much more expensive than the construction of RBFN models, although this is not exactly the case. As to the quality comparison between the optimal solutions, it is simple and straight forward for SOOPs. For MOOPs, since the results are usually a set of nondominated solutions, the quality evaluation considers both the distance of the obtained solutions to the true Pareto front and their distribution. According to [73], a comprehensive quality comparison is mainly based on three criteria: hypervolume [114], spacing [99] and set coverage metric [115]. They are defined in detail as follows:

Hypervolume

Hypervolume HV provides a way to measure the distance of the Pareto solutions to the true Pareto front. By introducing a reference point R , the hypervolume is defined as the union of all the hypercubes constructed using the Pareto points $S \in \mathcal{P}$ and the reference point R in the objective space. \mathcal{P} represents the current Pareto set, and the nadir point can be used for example as the reference point. Figure 5.8 illustrates the hypervolume in a bi-objective problem. It is believed that a Pareto front that is closer to the true Pareto front or more uniformly distributed more likely corresponds to a larger hypervolume.

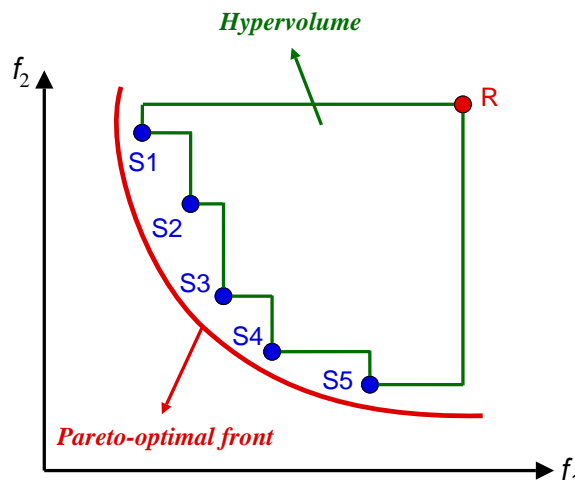


Figure 5.8: Illustration of hypervolume of a bi-objective optimization problem

Spacing

The spacing SP of a Pareto front is calculated by

$$SP = \sqrt{\frac{1}{N_P} \sum_{i=1}^{N_P} (d_{i,\min} - d_{\text{ave},\min})^2}, \quad (5.6)$$

where $d_{i,\min}$ and $d_{\text{ave},\min}$ are the minimum distance of the i -th Pareto solution to all the other solutions in the Pareto front and the average value of all $d_{i,\min}$, $i = 1, \dots, N_P$, respectively. The mathematical expressions are as follows:

$$d_{i,\min} = \min_{k \in \{1, \dots, N_P\} \wedge k \neq i} \left(\sum_{m=1}^{N_{\text{obj}}} |f_m^i - f_m^k| \right), \quad (5.7)$$
$$d_{\text{ave},\min} = \sum_{i=1}^{N_P} (d_{i,\min} / N_P).$$

Spacing SP is used to measure the Pareto front distribution. A smaller value corresponds to a more uniformly distributed Pareto front. It should be mentioned that the spacing is not a judgment of the spread extent of Pareto solutions. Both spacing and hypervolume are sensitive to the scaling and should be calculated after the normalization of the objective values in the objective space.

Set Coverage Metric

Set coverage metric compares two Pareto solutions directly by means of dominance concept. The percentage of solutions in the Pareto set \mathcal{P}^2 that are weakly dominated by the solutions in Pareto set \mathcal{P}^1 is defined as the set coverage metric $SCM(\mathcal{P}^1, \mathcal{P}^2)$. In the same way, $SCM(\mathcal{P}^2, \mathcal{P}^1)$ represents the percentage of solutions in the Pareto set \mathcal{P}^1 that are weakly dominated by the solutions in Pareto set \mathcal{P}^2 .

5.3.1 Analytical Test Case 1 - ZDT1

The first analytical test case ZDT1 is one of the ZDT test problems taken from a series of test problems designed by Deb *et al.* as benchmark tests of MOEAs [20]. They are scalable, algebraic and bi-objective optimization problems. The search complexity can be varied by changing the number of design variables. The problem statement is given by

$$\min f_1(\mathbf{x}), \quad (5.8)$$
$$f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})).$$

In ZDT1, $f_1(\mathbf{x})$, g , h and the design variable \mathbf{x} are defined as follows:

$$\begin{aligned}
f_1(\mathbf{x}) &= x_1 \\
g(\mathbf{x}) &= 1 + \frac{9}{N_{\text{dv}} - 1} \sum_{i=2}^{N_{\text{dv}}} x_i \\
h(f_1(\mathbf{x}), g(\mathbf{x})) &= 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} \\
0 \leq \mathbf{x} &= [x_1, \dots, x_{N_{\text{dv}}}]^T \leq 1
\end{aligned} \tag{5.9}$$

Global Optimization

In the global search, the modified NSGA-II is applied, and objective functions are evaluated by using both exact function calculation and RBFN models. The initial parent population with a size of 100 is generated using LHS. The recombination probability p_m is 0.9 and mutation probability p_c is defined as $1/N_{\text{dv}} = 0.033$.

The results of the global search obtained by employing different RBFN models and various size of initial database are compared in this test case. It is assumed that the mathematical formulations of the objective functions are not available. The approximation control parameters are listed in Table 5.2. To investigate the influence of RBFN parameters, the RBFN models are constructed using two center selection methods, i.e., regularized forward selection and regression tree method, combined with four different RBFs and two model selection criterion (MSC). The employed RBFs are Gaussian function, Cauchy function, multiquadric function and inverse multiquadric function, and the combined MSC are generalized cross-validation criterion (GCV) and Bayesian information criterion (BIC). All of the 16 RBFN models are used to evaluate 100 randomly selected solutions. Figure 5.9 and Figure 5.10 plot the comparison of average percentage approximation errors of these models for both optimization objectives. It can be observed that for the first objective, generally using regression tree leads to much better approximation results than using regularized forward selection. The minimum average approximation error is achieved when using Gaussian function and BIC as the RBF and MSC respectively, which is 0.14%. As to the second objective, the minimum average approximation error is 4.09%, which is achieved by using regularized forward selection to determine the centers, multiquadric function as RBF and BIC as MSC. Besides, the two comparison plots reveal that a good choice of RBFs is very important when using the regularized forward selection method since except for the multiquadric function, all the other RBFs fail to provide a good approximation; while, when using regression tree method, the selection RBFs and MSC has minor influence on the approximation accuracy. Based on the comparison, the best model for each objective is selected for the later approximation.

The global optimization run 100 generations, which requires 60 exactly evaluated generations and 6325 exactly evaluated functions. The average percentage approximation error in each control generation and the number of exactly evaluated generations q in the next round are shown in Figure 5.11. Because the approximation errors of the first objective are negligible compared to those of the second objective, the value of q is only dependent on the approxi-

Table 5.2: Approximation control parameters (ZDT1)

Parameter	Value
p_0	22
p	6
q_{ini}	5
q_{min}	1
N_e	25
e_{max}	6%

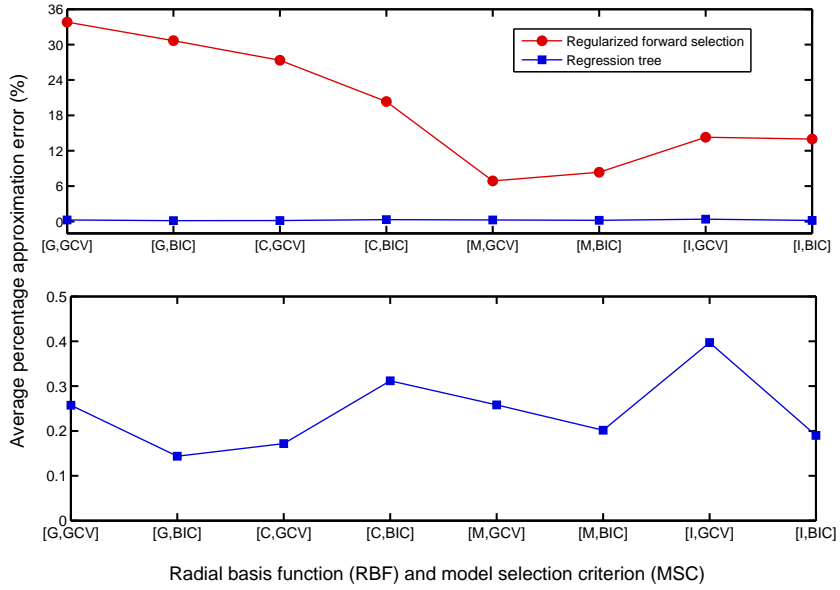


Figure 5.9: Approximation error of the 1st objective against RBFN models (ZDT1)

mation error of the second objective. An adaptive adjustment of q according to the error in the prior control generation can be observed. An increase of q will lead to a reduction of the approximation error in the following control generation. The approximation models tend to be more accurate as the optimization proceeds. In Table 5.3, the quality of obtained nondominated solutions is compared with that of the reference solutions obtained using also 6325 exact function evaluations. Meanwhile, both solution sets are plotted in Figure 5.12. Obviously, the solutions obtained by employing approximation models are closer to the true Pareto front and have a better distribution, which can also be verified by the higher hypervolume and the lower spacing value. Beside, Table 5.3 shows that 96.67% of the reference solutions are dominated by those obtained using both exact evaluations and RBFN models.

Furthermore, the influence of the approximation control parameters p_0 is investigated. Three optimization runs are conducted by varying the value of p_0 . Figure 5.13 plots the aver-

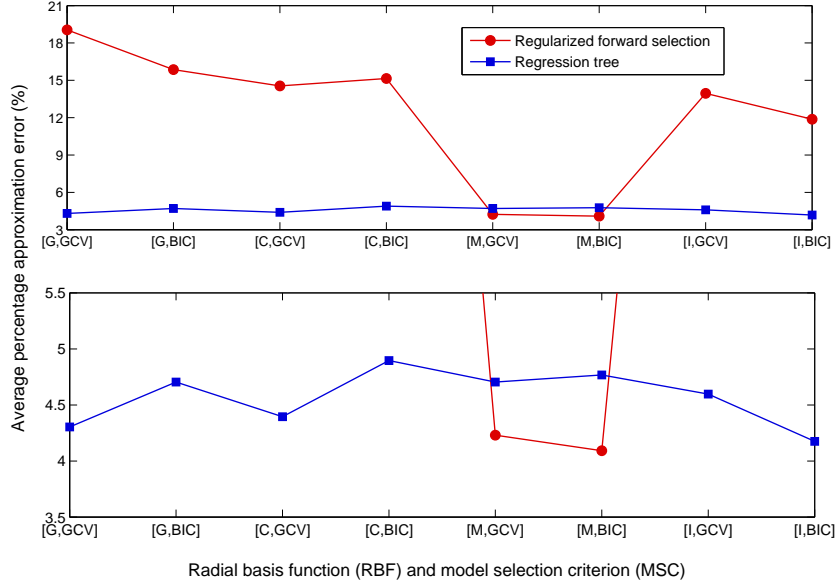


Figure 5.10: Approximation error of the 2nd objective against RBFN models (ZDT1)

Table 5.3: Performance comparison of the optimal solutions after global search (ZDT1)

No. of Run	1	2
Optimizer	NSGA-II (exact)	NSGA-II (exact + RBFN)
Hypervolume (HV)	0.8361	0.8808
Spacing (SP)	0.0089	0.0058
Set Coverage metric (SCM)	$(\mathcal{P}^2, \mathcal{P}^1) = 96.67\%$	$(\mathcal{P}^1, \mathcal{P}^2) = 0$

age approximation errors of both objectives in every control generation as well as the value of q in the next round. Table 5.4 compares the number of exactly evaluated generations $N_{\text{gen,e}}$ and number of required exact function evaluations $N_{\text{fun,e}}$ of three optimization runs as well as their performance with respect to the hypervolume HV , spacing SP and set coverage metric SCM . Besides, q_{ave} , the average value of q calculated in all the control generations is also listed for each case in Table 5.4. It can be observed from the figure that at the beginning of the optimization the approximation error corresponding to a larger p_0 is smaller. Consequently, when the approximation error is larger, a larger value of q is required to supplement the database in the subsequent generations. After about 6 control generations, since the database includes more and more points, there is no longer a distinct difference between the approximation errors of three optimization runs. In Table 5.4 the average value of q in the control generation indicates again that when the initial value of p_0 is smaller, more exactly evaluated generations will be required in the following generations. Regarding the optimization performance, when comparing the hypervolume and spacing, the solutions of the second run are the best and the

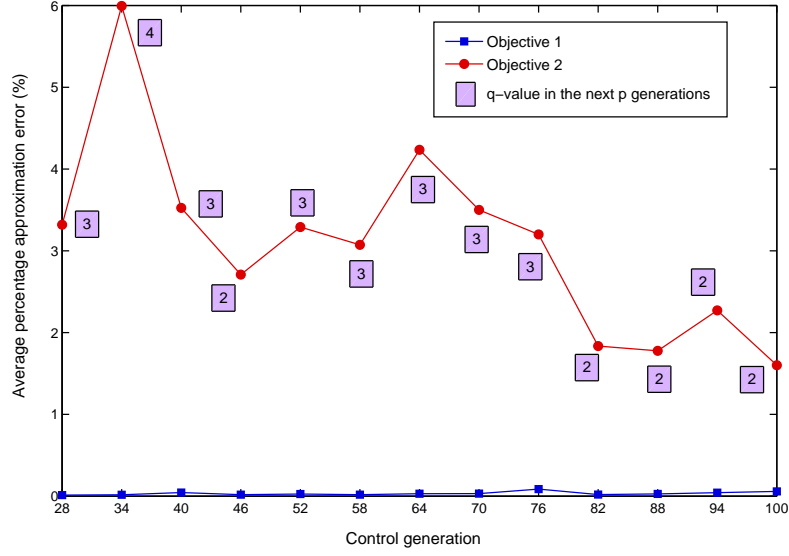


Figure 5.11: Average percentage approximation error and q -value in control generations (ZDT1)

solutions of the first run are better than those of the third. However, the largest percentage of the solutions from the second run is dominated by the solutions from other two runs. Also a larger percentage of solutions from the first run is dominated by the solutions from the third one than reversed one. A comparison of the results shows that a larger p_0 is not a prerequisite for the better solutions. Actually this exactly verified the advantage of utilizing control generations, i.e., an adaptively changed number of exactly evaluated generations q according to the approximation error.

Table 5.4: Performance comparison of Pareto solutions with different p_0 (ZDT1)

No. of run	p_0	NSGA-II (exact + RBFN)					
		$N_{\text{gen,e}}$	$N_{\text{fun,e}}$	HV	SP	SCM	q_{ave}
1	22	60	6325	0.9770	0.0056	$(\mathcal{P}^2, \mathcal{P}^1) = 29.10\%$ $(\mathcal{P}^3, \mathcal{P}^1) = 40.29\%$	2.75
2	16	58	6150	0.9819	0.0050	$(\mathcal{P}^1, \mathcal{P}^2) = 48.99\%$ $(\mathcal{P}^3, \mathcal{P}^2) = 48.32\%$	2.84
3	10	56	5975	0.9702	0.0058	$(\mathcal{P}^1, \mathcal{P}^3) = 38.19\%$ $(\mathcal{P}^2, \mathcal{P}^3) = 37.50\%$	2.92

Local Optimization

Using the clustering method, 14 solutions are selected from the nondominated solutions obtained by the global search as the starting points of the local search. They are plotted in Figure

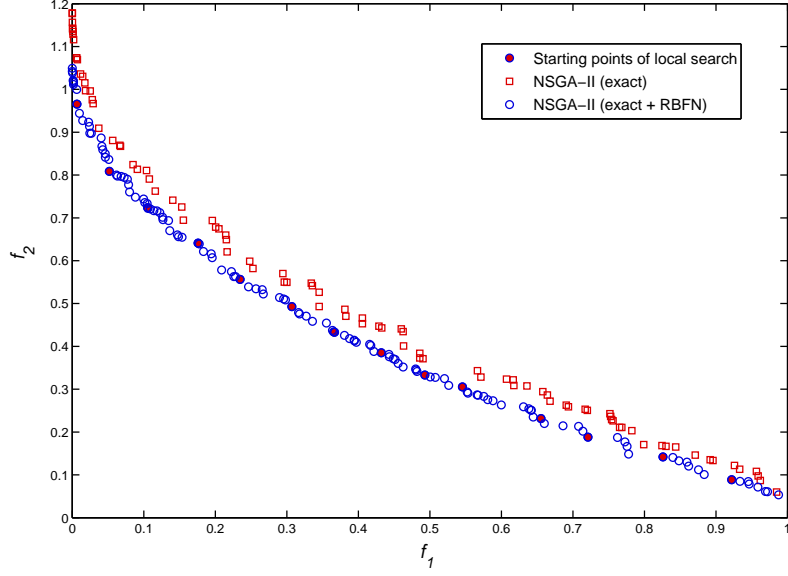


Figure 5.12: Optimization results comparison after global search (ZDT1)

5.14. The weight vectors are calculated and assigned individually to each local search. A hybrid method that combines the weighted sum and ε -constraint method is applied, which optimizes the weighted sum of both objectives and meanwhile uses both f_1 and f_2 as the constraint functions. For the i -th local search, ε_1^i and ε_2^i are set to be the objective values of the corresponding starting point, i.e., f_1^i and f_2^i , respectively. DFO is employed as the local optimizer. For these 14 local optimizations, only 1134 exact function evaluations are required, even when the number of design variables is 30. Including the evaluations in the global search, the hybrid method needs a total number of 7459 exact function evaluations. Figure 5.14 plots both the final solutions and the reference solutions that are obtained by running 200 NSGA-II generations (20000 exact function evaluations). Results show that the solutions obtained after local searches have a good diversity and 13 of them dominate some of the reference solutions.

5.3.2 Analytical Test Case 2 - FON

The second analytical test case FON is built by Fonseca and Fleming [40]. It is a bi-objective nonlinear minimization problem with a non-convex Pareto front, and the two objectives are symmetric and conflicting. The problem statement is given by

$$\begin{aligned}
 \min \quad & f_1(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{N_{\text{dv}}}\left(x_i - \frac{1}{\sqrt{N_{\text{dv}}}}\right)^2\right), \\
 & f_2(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{N_{\text{dv}}}\left(x_i + \frac{1}{\sqrt{N_{\text{dv}}}}\right)^2\right), \\
 & -4 \leq \mathbf{x} = [x_1, \dots, x_{N_{\text{dv}}}]^T \leq 4.
 \end{aligned} \tag{5.10}$$

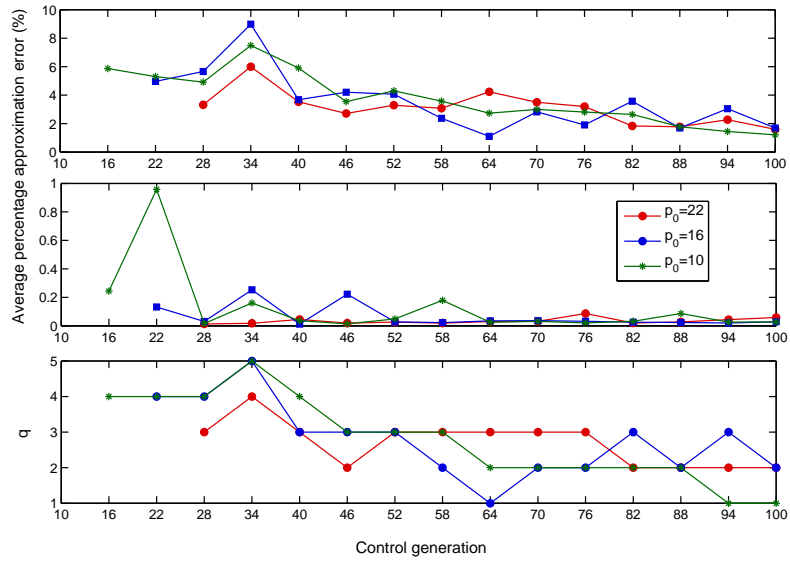


Figure 5.13: Comparison of average approximation error and q -value in control generations with different p_0 (ZDT1)

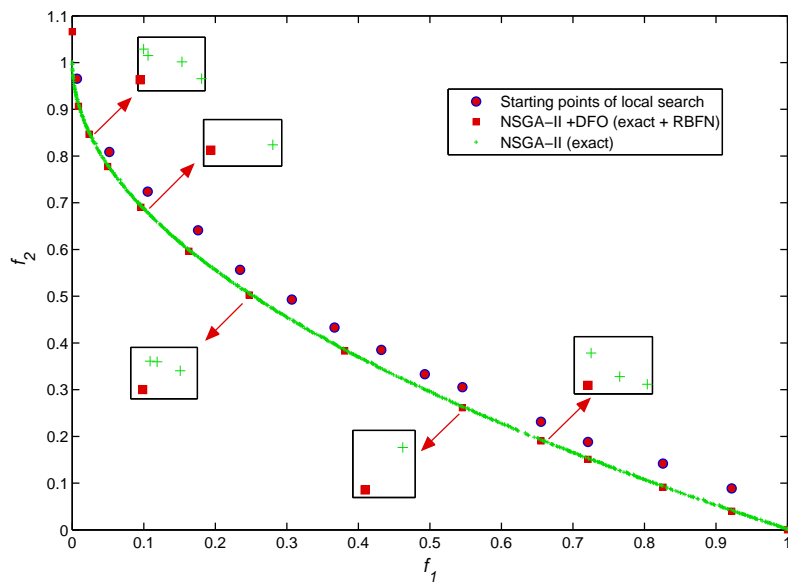


Figure 5.14: Comparison of final optimization results (ZDT1)

The search complexity varies with the number of design variables and N_{dv} is set to 10 in this test case.

Global Optimization

Using the hybrid optimization method, NSGA-II first runs 57 generations with a population size of 60. The recombination probability p_c is 0.9 and mutation probability p_m is 0.1. RBFN models are constructed using regularized forward selection to determine the network centers. Besides, multiquadric function is employed as the RBF and GCV is used as the MSC. The approximation control parameters are listed in Table 5.5. The average percentage approximation errors and the values of q in all control generations are plotted in Figure 5.15. A total number of 2210 exact function evaluations are performed in the global search. Like the prior test case, the Pareto solutions and the reference solutions obtained using same number of exact function evaluations are plotted in Figure 5.16. Table 5.6 gives a concrete quantitative comparison of both Pareto sets, which verifies again that employing approximation models leads to a faster converge and yields a Pareto front with larger extent.

Table 5.5: Approximation control parameters (FON)

Parameter	Value
p_0	15
p	6
q_{ini}	4
q_{min}	1
N_e	25
e_{max}	8%

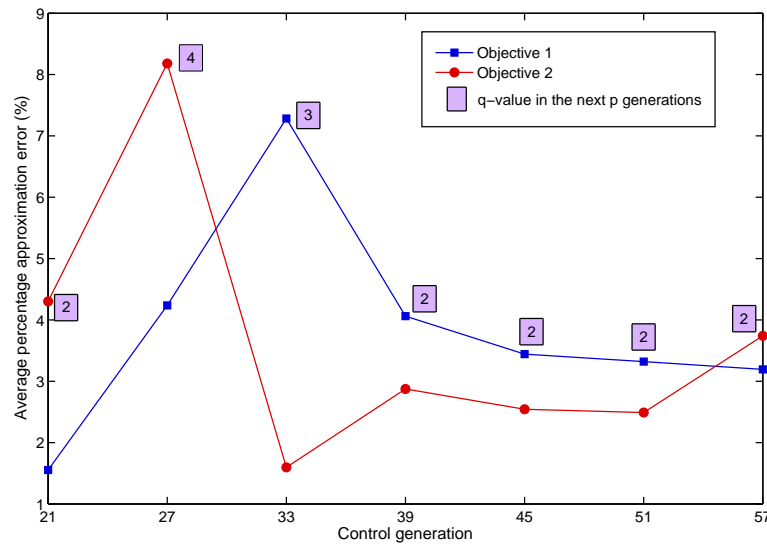


Figure 5.15: Average percentage approximation error and q -value in control generations (FON)

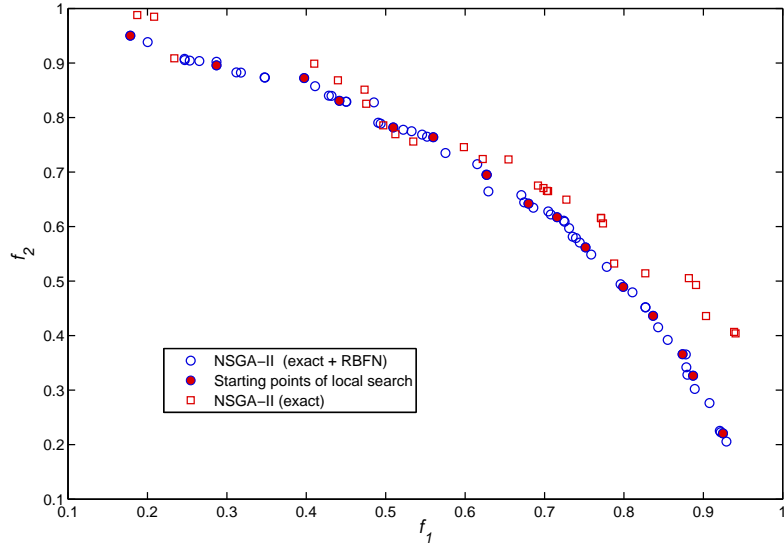


Figure 5.16: Optimization results comparison after global search (FON)

Table 5.6: Performance comparison of the optimal solutions after global search (FON)

No. of Run	1	2
Optimizer	NSGA-II (exact)	NSGA-II (exact + RBFN)
Hypervolume (HV)	0.2235	0.2731
Spacing (SP)	0.0235	0.0119
Set Coverage metric (SCM)	$(\mathcal{P}^2, \mathcal{P}^1) = 82.14\%$	$(\mathcal{P}^1, \mathcal{P}^2) = 8.82\%$

Local Optimization

The local search starts from 15 solutions selected from the current Pareto front, as shown in Figure 5.16. Since it is observed from the current solutions that the Pareto front is more likely to be non-convex, the *weighted Tchebycheff* method is applied for the local search. Pseudo-weights of the optimization objectives are calculated and assigned to different local searches. DFO is employed as the local optimizer and the 15 local runs need 3376 exact function evaluations, which means overall 5586 exact function evaluations are required by the hybrid method. The final solutions are well distributed as plotted in Figure 5.17. Compared to the reference solutions after 200 generations (12000 exact function evaluations), they spread in a larger range and 14 solutions dominate part of the reference solutions.

5.3.3 Numerical Test Case 1 - Pipe Junction

To illustrate the efficiency of the proposed optimization methodology in engineering applications, the shape optimization of a 3D pipe conjunction is first considered as a numerical test case. Comparisons using different optimization schemes are carried out with respect to the

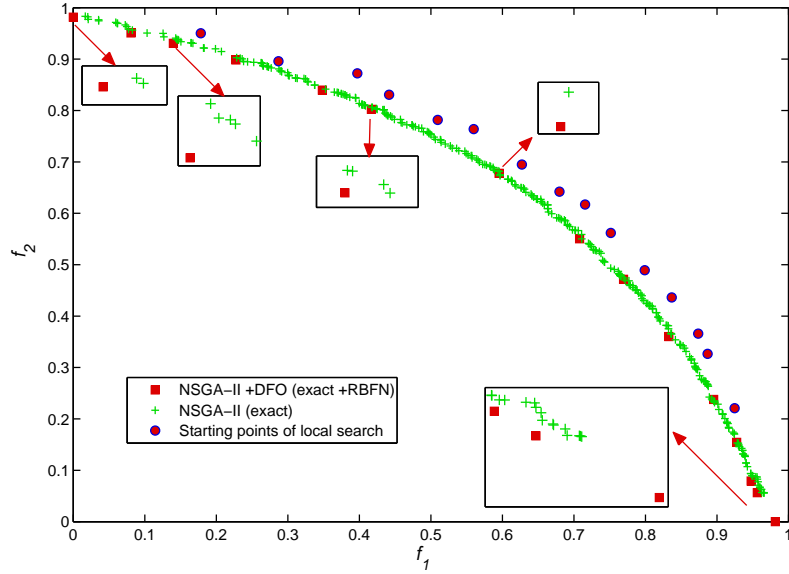


Figure 5.17: Comparison of final optimization results (FON)

quality of the achieved optimal values and the required computational cost.

Optimization Problem Definition

An initial sketch of the investigated geometry is shown in Figure 5.18. The optimization objective is to find the optimal shape of the conjunction part $B2$ in order to obtain a minimum pressure drop Δp between the inlet and the outlet of the pipe. The flow medium is assumed to be water with constant density and viscosity ($\rho = 1000\text{kg/m}^3, \mu = 10^{-3}\text{Pa}$). The characteristic Reynold number is 200 based on the diameter $2H$ and the inlet velocity. In the numerical design optimization process, the shape deformation is obtained by using free form deformation (FFD) on a shape box around $B2$. As shown in Figure 5.19, the shape box is discretized equidistantly using 4 points in x -direction, 3 points in y -direction and 3 points in z -direction, which yields 32 control points on the shape box surface. The control points on the corners can not be moved in order to assure the connection to the parts $B1$ and $B3$. 8 control points that directly intersect with the pipe surface, are selected to be moved perpendicularly to the pipe surface with an initial amount of $H/20$ and generate 8 shape basis vectors (SBVs) correspondingly. The deformation directions of the selected control points and the corresponding design variables are given in Figure 5.20. The deformations are bounded in y -direction between 0 and $20H$, which means that the maximum total amount of the control point displacement in the y -direction is $H/20 * 20H = H^2$. Different constraints are also given in the xz -plane.

The flow model is considered to be a steady, laminar, incompressible Newtonian fluid flow governed by the Navier-Stokes equations with necessary boundary conditions. The spatial discretization employs 32768 control volumes and FASTEST is employed as the flow solver. The pressure distribution of the initial configuration is given in Figure 5.21. The initial pressure

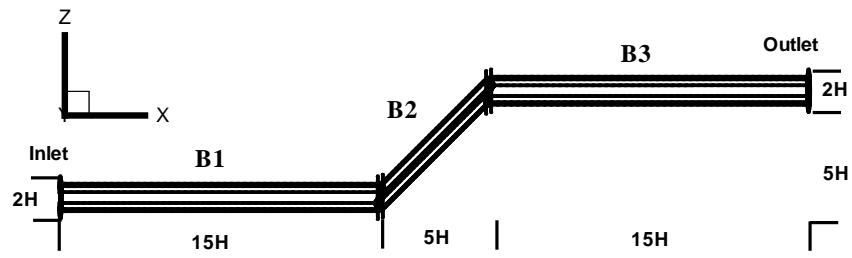


Figure 5.18: Sketch of the initial geometry configuration (pipe - 8 DVs)

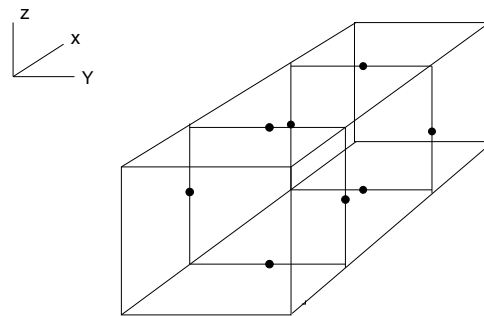


Figure 5.19: Shape box discretization and the selected control points (pipe - 8 DVs)

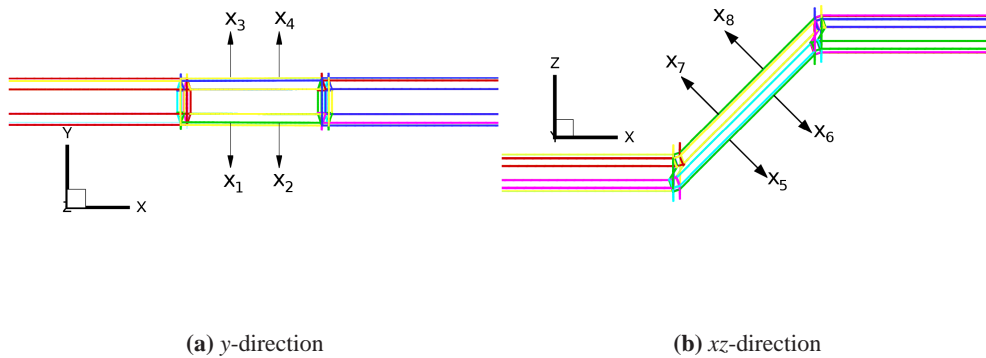


Figure 5.20: Deformation directions and the corresponding DVs (pipe - 8 DVs)

drop $\Delta p_{ini} = 0.5385\text{Pa}$. The optimization objective is to achieve the maximum efficiency of pressure drop reduction, which is defined to be the ratio of pressure drop reduction to the

pressure drop of the initial configuration:

$$\eta_p(\mathbf{x}) = \frac{(\Delta p_{\text{ini}} - \Delta p(\mathbf{x}))}{\Delta p_{\text{ini}}} \times 100\%, \quad (5.11)$$

where the pressure drop Δp is defined by

$$\Delta p(\mathbf{x}) = \bar{p}_{\text{in}} - \bar{p}_{\text{out}}. \quad (5.12)$$

The mean pressure of the inlet and outlet section is

$$\bar{p} = \frac{\int \int_{A_c} p dA_c}{\int \int_{A_c} dA_c}, \quad (5.13)$$

where A_c is the cross-sectional area of the inlet or outlet.

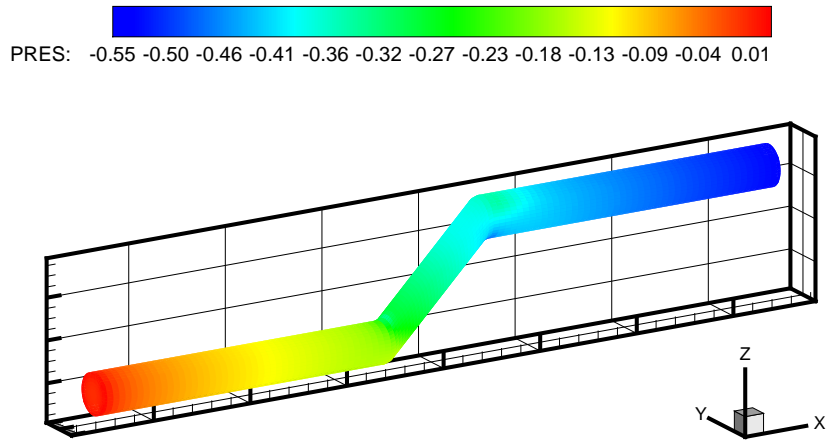


Figure 5.21: Pressure contour of the initial configuration (pipe - 8 DVs)

In summary, the optimization problem is formulated as

$$\begin{aligned} \max \quad & \eta_p(\mathbf{x}), \\ \text{with} \quad & \mathbf{x} = [x_1, \dots, x_8]^T, \\ \text{subject to} \quad & 0 \leq x_1, \dots, x_4 \leq 20H, \\ & 0 \leq x_5, x_8 \leq 8H, \\ & 0 \leq x_6, x_7 \leq 15H. \end{aligned} \quad (5.14)$$

Optimization Runs

To compare the optimization performance, five optimization runs are carried out with different optimization schemes.

The first optimization run only employs the modified NSGA-II as the optimizer. When solving the single optimization problem, the optimization procedures of Pareto front identification and crowding sort selection can be simplified by just choosing the best N_{pop} solutions from the combined population R into the next generation. Also, it is not necessary to use the external archive A and conduct final nondominated sort. In this optimization run, all the objective functions are evaluated by flow solver FASTEST. A parallel scheme is used so that in each generation, the function evaluations are sent simultaneously to different processors. The population size is 20. The initial population is generated randomly in the feasible region. The recombination probability and mutation probability are $p_c = 0.9$ and $p_m = 0.125$, respectively. The stopping criterion is defined so that the optimizer stops automatically if there is no improvement in the objective function for 10 continuous generations. The achieved optimal solution by this optimization run is used as the reference solution for the later comparison.

The second optimization run also only employs the modified NSGA-II as the optimizer. The difference is that both a high-fidelity flow solver and low-cost RBFN models are used for the function evaluations. The approximation control parameters are given in Table 5.7. The Gaussian function and GCV are chosen as RBF and MSC respectively for RBFN construction. The network centers are determined using the regularized forward selection method. The stopping criterion is defined in the same way as that in the first optimization run.

Table 5.7: Approximation control parameters (pipe - 8 DVs)

Parameter	Value
p_0	2
p	3
q_{ini}	1
q_{min}	0
N_e	4
e_{max}	0.15%

The third and fourth run combine both global and local search. The global search is conducted by choosing the same parameters as those in the second optimization run. The difference is that the optimization stops before the stopping criterion is satisfied. Then the local search starts from the current optimal solution and employs CONDOR in the third run and DFO in the fourth. For both DONDOR and DFO, the initial trust-region radii are equal to H and optimization process will be stopped automatically if the sampling distance of CONDOR or the trust radius of DFO falls below 0.0001.

The last run is only performed by CONDOR with the same optimization parameters as those in the hybrid optimization run. The initial shape is used as the starting point for this optimization run.

Results and Comparisons

The RBFN approximation error of each control generation is plotted in Figure 5.22. An adaptive adjust of the approximation accuracy can be observed, i.e., if the average percentage approximation error in the control generation is greater than the allowed maximum error, there is one generation conducting high-fidelity flow simulation in the next five generations, which leads to a reduction of approximation error in the next control generation.

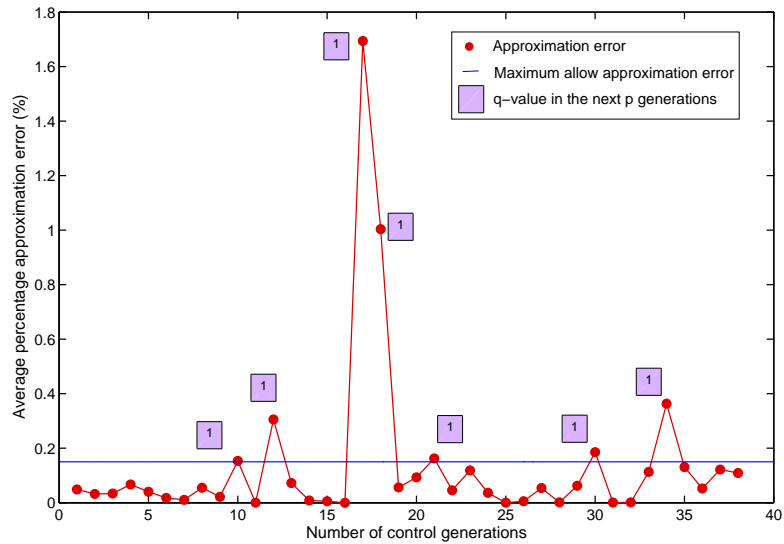


Figure 5.22: Average percentage approximation error and q -value in control generations (pipe - 8 DVs)

Since the computational cost of RBFN can be negligible compared to FASTEST evaluation, only the number of FASTEST evaluations is considered for the evaluate of optimization efficiency. The optimization history of the five optimization runs is plotted in Figure 5.23. Besides, the optimization performance concerning the maximum efficiency of pressure drop reduction, the number of function evaluations required to achieve the optimum as well as the total number of FASTEST evaluations required by the optimization run is compared and summarized in Table 5.8. It can be observed that the first optimization run obviously has a slow convergence rate in the near optimum region. It took 780 FASTEST evaluations to get the optimal solution and a total of 980 FASTEST evaluations to get convergence. It improves the pressure drop reduction efficiency to 24.09%. All the other runs involving the evolutionary global search find the optimal solutions with the same quality as the reference solution. In the second run, when the approximation model is employed during the optimization process, the required number of FASTEST evaluations to achieve the optimum is reduced drastically to 356. A further improvement of the optimization efficiency can be achieved by applying the hybrid optimization method. The local search starts from solution S , which corresponds to 100 FASTEST evaluations. The working principle of the two local search algorithms that were employed can be verified through the observation of their convergence behaviors. CONDOR

needs 45 FASTEST evaluations at the beginning to construct a full quadratic model. Thereafter, it converges quite fast to the optimum. While DFO starts by constructing only an incomplete model, but updates the model as well as the optimal solution as soon as it is available, and it takes longer than CONDOR to get convergence. Even though, both local searches accelerate the optimization process, which demonstrates the excellent local search ability of CONDOR and DFO. Furthermore, the comparison to the result obtained by the fifth run shows that starting CONDOR from the point in the near optimum region is more likely to find the global optimum. Apparently, CONDOR fell into a local optimum when starting from the initial point.

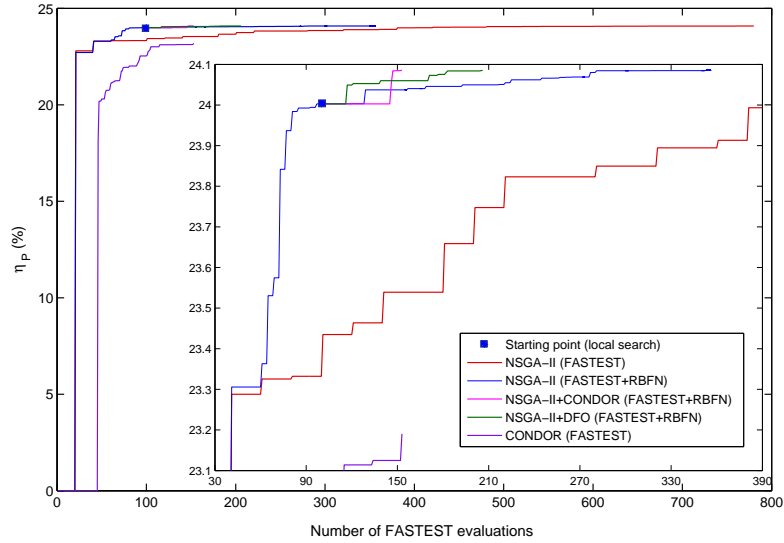


Figure 5.23: Convergence history of all optimization runs (pipe - 8 DVs)

Table 5.8: Optimization performance comparison (pipe - 8 DVs)

Optimizer	Optimal η_p	N_{FASTEST}	Total N_{FASTEST}
NSGA-II (FASTEST)	$\approx 24.09\%$	780	980
NSGA-II (FASTEST+RBFN)	$\approx 24.09\%$	356	392
NSGA-II+CONDOR (FASTEST+RBFN)	$\approx 24.09\%$	153	175
NSGA-II+DFO (FASTEST+RBFN)	$\approx 24.09\%$	206	246
CONDOR (FASTEST)	$\approx 23.19\%$	153	191

Quite similar design variables corresponding to all the optimal solutions are obtained. The optimal configuration achieved by the hybrid method (NSGA-II + CONDOR) is used here for an illustration, which is given in Table 5.9. In Figure 5.24 and 5.25 the pressure distributions of the initial configuration and optimal configurations are compared in the xy - and xz -plane, respectively. Symmetry deformations in both y - and xz -directions can be observed and there is almost no deformation caused by the shape basis vectors \mathbf{t}^6 and \mathbf{t}^7 . It is also shown in

Figure 5.26 that the recirculation regions before and after the conjunction in the initial pipe configuration disappear in the optimal configuration, and a new recirculation appears in the middle part of the pipe conjunction. On one hand, recirculation can bring an increase of the pressure drop due to the additional energy dissipation. On the other hand, recirculation can reduce the energy loss, which can be explained by the fact that the wall friction, another factor leading to energy loss, is proportional to the wall shear stress, which is correlated to the normal gradient of the tangential velocity component at the wall, and the recirculation reduces the gradient. Apparently the achieved best configuration utilizes the recirculation mostly to reduce the energy loss.

Table 5.9: Optimal solution obtained by NSGA-II+CONDOR (pipe - 8 DVs)

DV	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
Value	7.70H	8.89H	7.71H	8.86H	8.00H	0.68H	0	8.00H

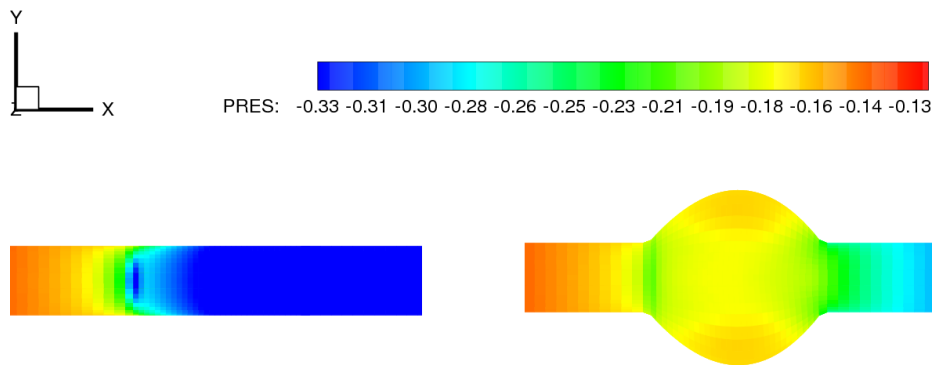


Figure 5.24: Pressure contour of initial configuration (left) and optimal configuration (right) in xy -plane (pipe - 8 DVs)

5.3.4 Numerical Test Case 2 - Heat Exchanger

Optimization Problem Definition

The optimization task is to seek a geometry design of a heat exchanger to maximize the Nusselt number Nu and minimize the pressure drop Δp , i.e., to achieve the maximum heat transfer and the minimum cooling power. The heat exchanger consists of four hot pipes passed by a fluid within a rectangular box with inlet and outlet channels. The air enters with an inlet temperature $T_{in} = 300K$ and a uniform velocity $u_{x,in} = 2m/s$. The pipes have a constant temperature of $800K$ and the walls are assumed to be adiabatic. There is a complex interaction between these two objectives. They are strongly dependent on the shapes of the pipe cross sections and the exchanger itself, which have direct influences on the amount of heat dissipation, the regions of recirculation as well as the wall friction. For example, a strong recirculation of the flow may

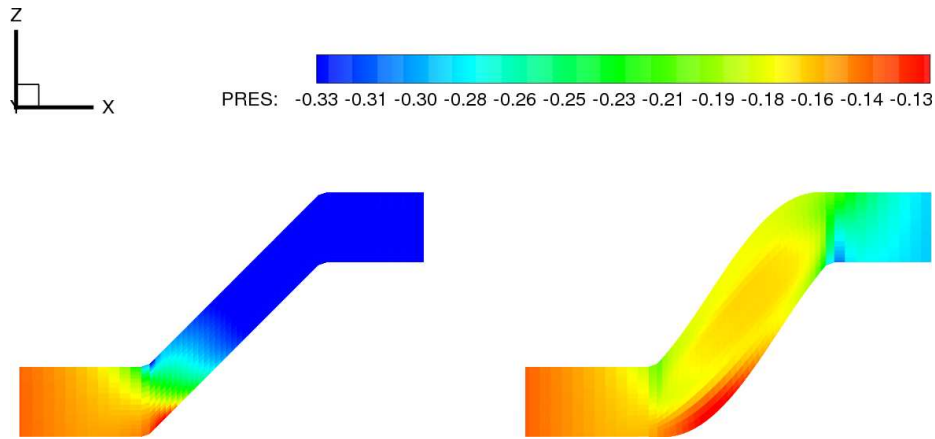


Figure 5.25: Pressure contour of initial configuration (left) and optimal configuration (right) in xz -plane (pipe - 8 DVs)

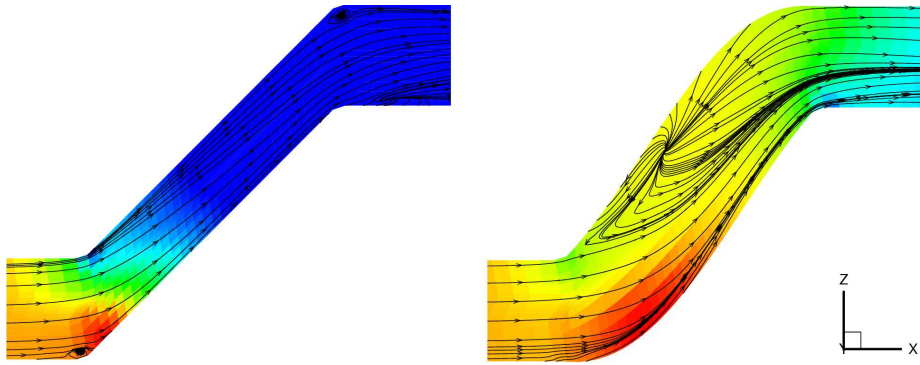


Figure 5.26: Recirculation of initial configuration (left) and optimal configuration (right) in xz -plane (pipe - 8 DVs)

cause a better heat transfer, but due to the large energy dissipation the pressure drop may also be large.

FFD technique is applied on 8 shape boxes B_1, \dots, B_8 , which are defined around four pipes and also at the four corners. $B_1 - B_4$ and $B_5 - B_8$ are discretized equidistantly by 4,4,2 points and 3,3,2 points in x, y, z -direction, respectively. The deformations are performed on the xy -plane with an initial amount of $H/5$ and generate a total number of 40 SBVs. Since the deformations in z -directions are defined to be symmetric, 20 design variables are used to control the magnitude of the deformations. For each design variable a box constraint is defined. The initial geometry sketch of the heat exchanger, the shape boxes, the deformation directions of the selected control points as well as the corresponding design variables are shown in Figure 5.27.

The fluid is assumed to be incompressible with constant properties. The characteristic Reynolds number is 3333 based on the inlet velocity and channel height. Non-slip conditions are specified on the walls and pipes. The flow solver is based on the numerical solu-

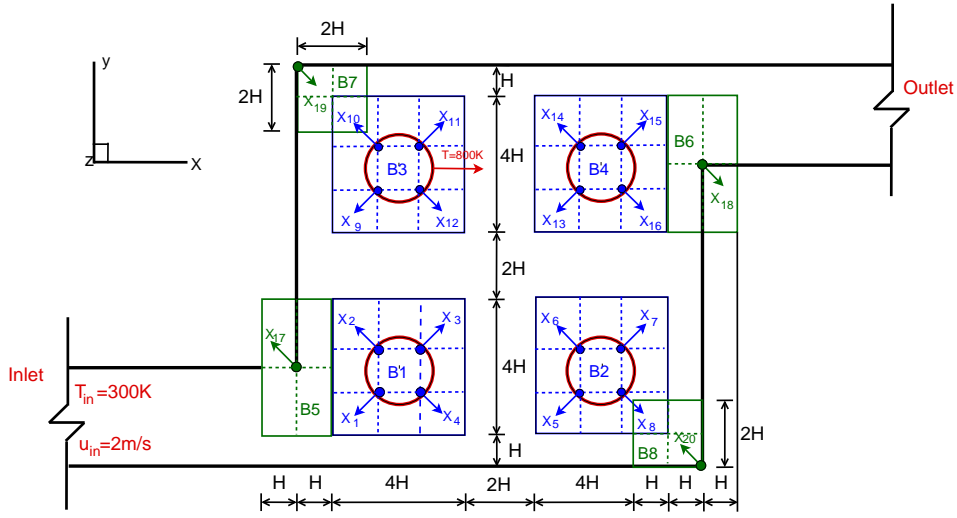


Figure 5.27: Sketch of the initial geometry with shape boxes, deformation direction of the control points and the corresponding DVs (heat exchanger - 4 pipes)

tion of Navier-Stokes equation for an incompressible Newtonian fluid. Assuming a turbulent, steady flow and neglecting buoyancy effects, the flow solver FASTEST is employed to solve the Reynolds averaged equations together with the $k - \varepsilon$ turbulence model. Three grid levels are used in the multigrid method. The spatial discretization employs 120832 hexahedral block-structured grids on the finest grid level.

In summary, the following optimization problem is solved:

$$\begin{aligned}
 \min \quad & \Delta p(\mathbf{x}) = \bar{p}_{\text{in}} - \bar{p}_{\text{out}}, \\
 \max \quad & Nu = \frac{QD_h}{\Delta T A_s \kappa}, \\
 \text{with} \quad & \mathbf{x} = [x_1, \dots, x_{20}]^T, \\
 \text{subject to} \quad & -2H \leq x_1, \dots, x_{16} \leq 2H, \\
 & -H \leq x_{17}, x_{18}, x_{19}, x_{20} \leq H,
 \end{aligned} \tag{5.15}$$

where D_h is the hydraulic diameter, κ is the thermal conductivity, A_s is the temperature surface area, \bar{p}_{in} and \bar{p}_{out} are the mean pressure at inlet and outlet, respectively, which are defined in Equation (5.13). The total heat transfer Q and the log-mean temperature difference ΔT are expressed as

$$\begin{aligned}
 Q &= \dot{m} C_p (\bar{T}_{\text{in}} - \bar{T}_{\text{out}}), \\
 \Delta T &= \frac{(T_{\text{wall}} - \bar{T}_{\text{in}}) - (T_{\text{wall}} - \bar{T}_{\text{out}})}{\ln[(T_{\text{wall}} - \bar{T}_{\text{in}})/(T_{\text{wall}} - \bar{T}_{\text{out}})]}.
 \end{aligned} \tag{5.16}$$

where the mean temperature of the inlet and outlet cross-section are defined by

$$\bar{T} = \frac{\int \int_{A_c} u_x T dA_c}{\int \int_{A_c} u_x dA_c}, \quad (5.17)$$

where u_x is the x -component of velocity \mathbf{u} and A_c is the cross-sectional area of the inlet or the outlet.

Optimization Runs

Two optimization runs are performed in this test case for comparison. The first one runs NSGA-II and only uses FASTEST for function evaluations. The results are used as reference solutions. The second one uses a hybrid method that combines the global optimizer NSGA-II and the local optimizer DFO; it also employs RBFN models during the global search. For both runs, the population size of NSGA-II is 60 and the initial population is generated using LHS. The recombination probability p_c is 0.9 and mutation probability p_m is 0.05. The first run is performed for 120 generations, the second one runs NSGA-II for 81 generations. The approximation control parameters are listed in Table 5.10. In the second run, after the global search, the optimizer selects 8 points from the current Pareto solutions and starts DFO for the local search. Both objectives are normalized then DFO is performed to optimize the weighted sum objective value. The initial trust radius is $0.4H$ and the stopping trust radius is defined to be 0.001.

Table 5.10: Approximation control parameters (heat exchanger - 4 pipes)

Parameter	Value
p_0	15
p	6
q_{ini}	5
q_{min}	1
N_e	25
e_{max}	1.6%

Results and Comparisons

Using the hybrid optimization method, parallel NSGA-II is first run for 15 generations using the flow solver FASTEST for function evaluations. After 15 generations, RBFN models substitute part of the evaluations using the flow solver. Approximation models that are constructed using different center selection methods, RBFs, and MSC are first employed to approximate the objective values of 60 different design vectors for an accuracy evaluation. The comparison of the average percentage approximation error of both objectives are plotted in Figure 5.28 and Figure 5.29, respectively. It can be observed that for both objectives the best approximations are achieved by using regularized forward selection together with the multiquadric function and the model selection criterion BIC. The errors are 0.32% and 0.13%, respectively. Also, when using the regularized forward selection, the choice of RBFs has a significant influence

on the approximation accuracy, especially for the pressure drop. When using the regression tree method, all the average approximation errors are less than 0.5%. Also, the RBF types and the MSC have a minor influence on the approximation results. These findings are consistent with those of the first analytical test case. Based on this comparison, the forward selection method is preferred to construct RBFN models with the multiquadric function and the model selection criterion BIC for later approximations.

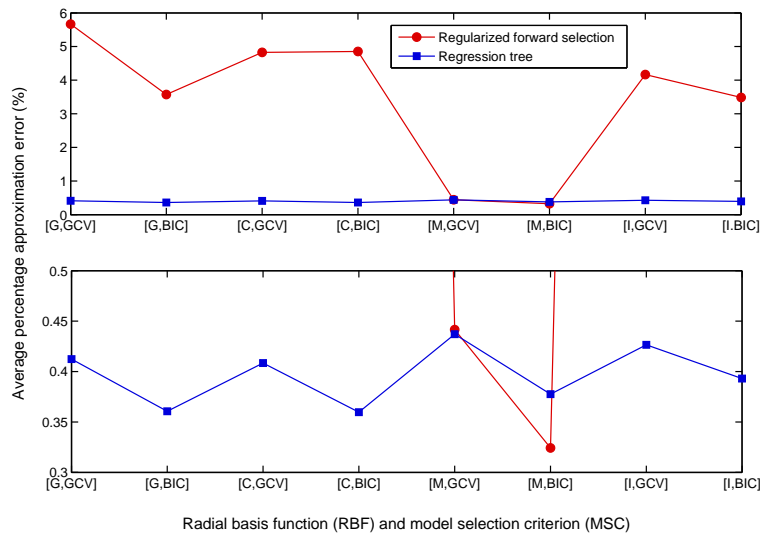


Figure 5.28: Average percentage approximation error of pressure drop against RBFN models (heat exchanger - 4 pipes)

The global search runs for 81 generations using FASTEST and RBFN evaluations. The average approximation error in each control generation and the consequently calculated q , i.e., the number of generations conducting FASTEST evaluations in the next 6 generation, are plotted in Figure 5.30. In total, 2607 FASTEST evaluations are required by the optimization run. In Figure 5.31, results of two optimization runs achieved using the same number of FASTEST evaluations are plotted since the computational cost of the RBFN evaluations is comparatively negligible. A quantitative comparison is also given in Table 5.11. Obviously, the Pareto solutions obtained by using FASTEST and RBFN evaluations should be closer to the true Pareto front. Most of them dominate the solutions obtained using only FASTEST. They also have a better diversity and spread a larger extent in the objective space. The comparisons verify again that, with the same computational cost, a better performance is achieved by the incorporation of approximation models.

In the local search, the selected starting points obtained by using the clustering method are plotted in Figure 5.31. In total, 694 FASTEST evaluations are required by the 8 DFO runs, i.e., 3301 FASTEST evaluations for the whole optimization process. After the local search, the final optimal solutions are given in Table 5.12. The values in bold font denote those that are better than the initial solution, which has a pressure drop of 6.262Pa and a Nusselt number of 28.820.

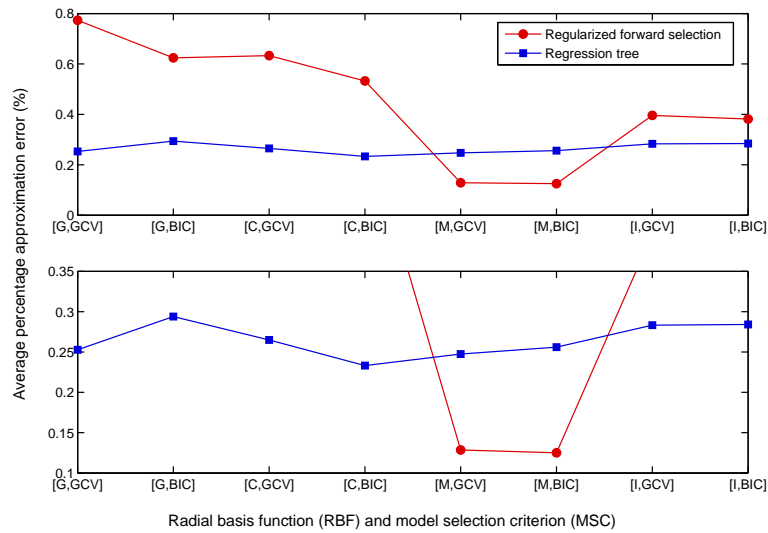


Figure 5.29: Average percentage approximation error of Nusselt number against RBFN models (heat exchanger - 4 pipes)

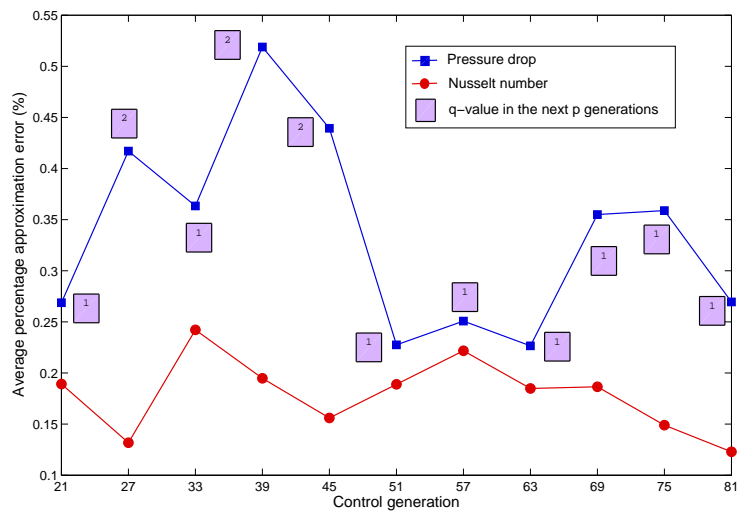


Figure 5.30: Average percentage approximation error and q -value in control generations (heat exchanger - 4 pipes)

Obviously the initial design is dominated by most of the Pareto solutions, which indicates that both optimization objectives, heat transfer and cooling power, can be improved by choosing appropriate solutions, e.g., the solution S_2 improves the initial Nusselt number by 8.95% and reduces the pressure drop by 19.39%. The Pareto solutions provide sufficient compromises to

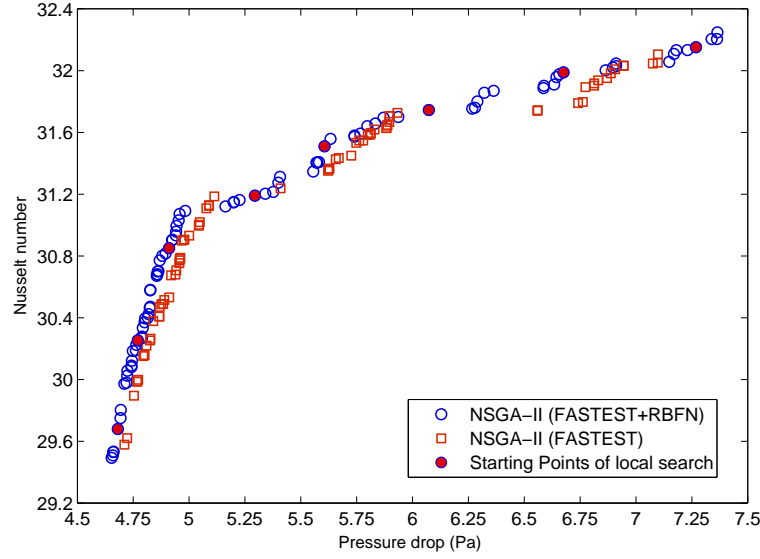


Figure 5.31: Optimization results comparison after global search (heat exchanger - 4 pipes)

Table 5.11: Performance comparison of the optimal solutions after global search (heat exchanger - 4 pipes)

No. of Run	1	2
Optimizer	NSGA-II (FASTEST)	NSGA-II (FASTEST + RBFN)
Hypervolume (HV)	0.7105	0.7417
Spacing (SP)	0.0148	0.0096
Set Coverage metric (SCM)	$(\mathcal{P}^2, \mathcal{P}^1) = 88.16\%$	$(\mathcal{P}^1, \mathcal{P}^2) = 6.52\%$

meet different design preferences. The maximal improvement of Nusselt number and pressure drop are achieved by $S8$ and $S3$, which are 12.64% and 26.49%, respectively. Figure 5.32 and Figure 5.33 illustrate the temperature and pressure contours together with stream lines of the original shape and three calculated optimal shapes.

Table 5.12: Final Pareto-optimal solutions (heat exchanger - 4 pipes)

Objectives	S1	S2	S3	S4	S5	S6	S7	S8
ΔP (Pa)	4.689	5.048	4.603	5.392	4.956	6.875	5.716	7.504
Nu	30.020	31.398	28.810	31.488	31.163	32.252	31.739	32.464

To verify the efficiency and performance of the hybrid optimization method, the results are compared with reference solutions obtained by the first run after 120 generations, which

corresponds to 7200 FASTEST evaluations. Both Pareto sets are plotted in Figure 5.34. It can be observed that most Pareto solutions obtained by using the hybrid method dominate the reference Pareto solutions. They also detect a better solution for each single objective. The results demonstrate that the proposed optimization method is able to provide a set of optimal solutions with a better convergence and a good diversity with much less computational cost.

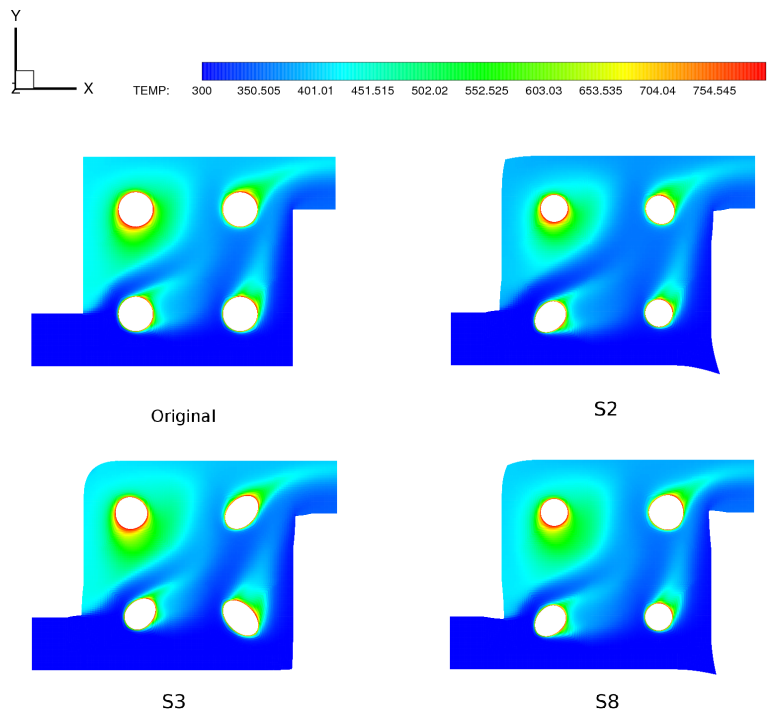


Figure 5.32: Temperature contour of the original and three optimal shapes (heat exchanger - 4 pipes)

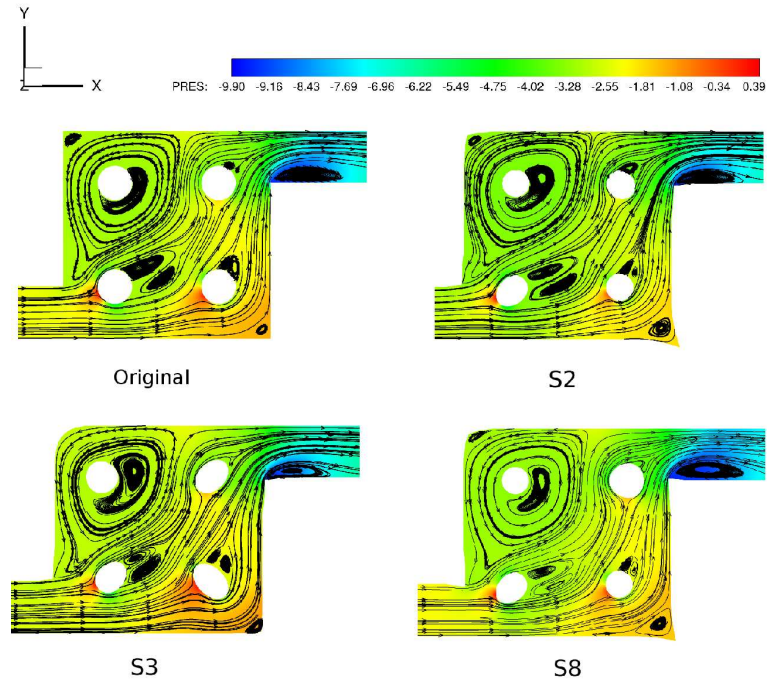


Figure 5.33: Pressure contour and recirculation of the original and three optimal shapes (heat exchanger - 4 pipes)

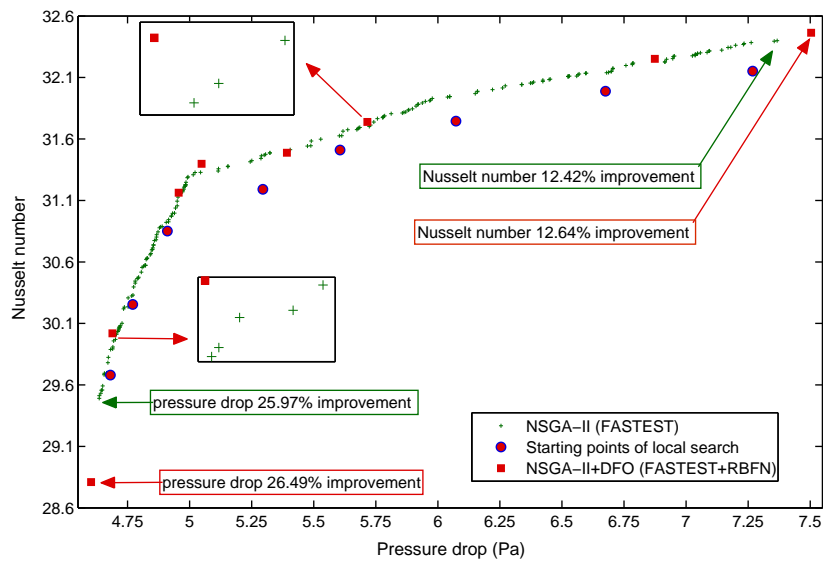


Figure 5.34: Comparison of final optimization results (heat exchanger - 4 pipes)

Chapter 6

Proper Orthogonal Decomposition (POD)-Based Reduced-Order Model

POD was originally introduced by Loeve in 1945 and by Karhunen in 1946. It is also known as the Karhunen-Loeve decomposition, principal component analysis, or single value decomposition. POD is a powerful technique to capture the optimal basis from a certain number of snapshots, which are collected from experiments or numerical simulations varying spatial or temporal parameters. Each basis represents a characteristic of the snapshots set. With these optimal bases, the solutions for the new parameters can be extrapolated. POD has been successfully utilized in a variety of applications. One important application area is fluid turbulence, where it has been used for the attraction of spatial scale organized motions [81], the identification of coherent structures [59, 79, 104], as well as for solving optimal control problems [4, 68, 80]. Besides, POD is also widely used in non-fluid fields such as signal processing, pattern recognition, and other industrial applications. More recently, POD has been used to develop reduced-order models to capture the parametric variations for the purpose of design optimization. LeGresley integrated POD models into a gradient-based optimization procedure for the inverse design of a 2D inviscid airfoil surface [74, 75]. In [13], Bui-Thanh proposed a gappy POD method for the reconstruction of flow field from incomplete aerodynamic data sets, then it was extended for inverse shape design.

In this work, POD is combined with the efficient interpolation technique to approximate the objective functions required by the evolutionary shape optimization. The key advantage of this POD-based reduced-order model is that instead of just a black box response, it carries out the model approximation on the entire flow field (pressure, velocity, temperature, etc.), and it is able to reflect the real physical behavior of the flow region under consideration. This work is restricted to steady fluid flow optimization and the POD models serve to capture the spatial information of the system. The snapshots are the flow fields that have been calculated with respect to various shape configurations by the high-fidelity computational flow solver. POD provides a methodology for extracting the basis vectors from the snapshots and reconstructing the snapshots approximately by the linear combination of a number of selected optimal basis vectors. In the same way, the arbitrary unknown configuration can be predicted by utilizing these optimal basis vectors combined with empirical coefficients extrapolated by an interpolation method.

The construction of the reduced-order model can be divided into two main parts: the calculation of the POD optimal basis vectors and the interpolation of the empirical coefficients, which will be detailed in Section 6.1 and Section 6.2. Section 6.3 introduces the evolutionary optimization procedure when utilizing POD models. Two exemplary shape optimization test cases are given in 6.4.1 and 6.4.2, which employ the models constructed by combining POD with the cubic spline interpolation and the RBF interpolation, respectively.

6.1 Proper Orthogonal Decomposition

The first step in the POD model construction is the collection of appropriate snapshots. Suppose that the POD model is constructed on n snapshots \mathbf{f}^i , $i = 1, 2, \dots, n$. Each snapshot \mathbf{f}^i is a vector and defined in this work as the CFD evaluation for the i -th shape design vector \mathbf{x}^i . Using a solution method such as FVM for the discretization of the governing equations, each snapshot is calculated on m nodes. The snapshot can represent the velocity, temperature or pressure field of the flow. The expression f_j^i denotes the value of the snapshot i calculated at node j , $j = 1, 2, \dots, m$. The whole snapshots ensemble is denoted as an $m \times n$ matrix \mathbf{F} :

$$\mathbf{F} = \begin{bmatrix} f_1^1 & f_1^2 & \cdots & f_1^n \\ f_2^1 & f_2^2 & \cdots & f_2^n \\ \vdots & \vdots & \vdots & \vdots \\ f_m^1 & f_m^2 & \cdots & f_m^n \end{bmatrix}_{m \times n}. \quad (6.1)$$

The aim of POD is to extract a set of characteristic vectors (POD optimal basis vectors) \mathbf{g}^k , $k = 1, \dots, q$, ($q \ll n$) from the snapshot matrix \mathbf{F} , so that the snapshot \mathbf{f}^i can be approximated in the best way as a finite sum of the vectors \mathbf{g}^k by

$$\mathbf{f}^i \simeq \sum_{k=1}^q \theta_k^i \mathbf{g}^k, \quad i = 1, \dots, n, \quad (6.2)$$

with the empirical coefficient vector $\boldsymbol{\theta}^i = [\theta_1^i, \dots, \theta_q^i]^T$ corresponding to the i -th snapshot. Since the representation of equation (6.2) is not unique, the POD is concerned with the selection of the vector \mathbf{g}^k . One criterion is that these basis vectors should be orthonormal, i.e.,

$$\mathbf{g}^i \cdot \mathbf{g}^j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}. \quad (6.3)$$

The notation \cdot denotes the inner product. For orthonormal basis vectors, the empirical coefficient θ_k^i can be obtained by projecting the solution field \mathbf{f}^i onto each POD basis vector \mathbf{g}^k :

$$\theta_k^i = \mathbf{f}^i \cdot \mathbf{g}^k. \quad (6.4)$$

Another objective is to find, a sequence of the orthonormal vectors \mathbf{g}^k , $k = 1, \dots, n$, such that the first k basis vectors give the best possible k -term approximation. This is done once and applied to the whole model construction. An efficient method called *the method of snapshots* [104] allows the solution of this problem be equal to solving an eigenvalue problem with the size n . This method assumes that the POD basis vectors can be written as a linear combination

of the snapshots:

$$\mathbf{g}^k = \sum_{i=1}^n \sigma_i^k \mathbf{f}^i(\mathbf{x}), \quad k = 1, \dots, n. \quad (6.5)$$

The coefficients σ_i^k are the components of eigenvectors $\boldsymbol{\sigma}^k$ of the eigenvalue problem:

$$\mathbf{A} \boldsymbol{\sigma}^k = \lambda_k \boldsymbol{\sigma}^k, \quad k = 1, \dots, n, \quad (6.6)$$

where λ_k is the k -th eigenvalue, and \mathbf{A} is an autocorrelation matrix which is defined by

$$\mathbf{A} = \frac{1}{n} \mathbf{F}^T \mathbf{F}. \quad (6.7)$$

Solving equation (6.6) yields the eigenvectors $\boldsymbol{\sigma}^k$, which should be normalized to satisfy the imposed condition so that $\mathbf{g}^k, k = 1, \dots, n$, are ensured to be orthonormal, i.e.,

$$\boldsymbol{\sigma}^i \cdot \boldsymbol{\sigma}^j = \begin{cases} \frac{1}{n\lambda_j} & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} \quad (6.8)$$

. Then the POD basis vectors \mathbf{g}^k is calculated by using equation (6.5) and the snapshots set can be reconstructed by the obtained POD basis vectors as follows:

$$\mathbf{f}^i = \sum_{k=1}^n \theta_k^i \mathbf{g}^k, \quad i = 1, \dots, n. \quad (6.9)$$

The k -th eigenvalue λ_k is supposed to be a measure of the system information transferred within the k -th POD basis vector \mathbf{g}^k . Therefore, by ordering these n POD basis vectors according to the magnitude of their corresponding eigenvalues in descending sequence, $\lambda_1 > \lambda_2 > \dots > \lambda_n$, most of the system information concentrates only on the first a few basis vectors. The snapshots reconstruction in equation (6.9) can be approximately truncated by just employing the first q POD basis vectors. These q basis vectors are called POD optimal basis vectors and the truncation degree M equals to q . They are later used to construct the POD approximation model for the prediction of an arbitrary function field \mathbf{f}^* within the design region by

$$\mathbf{f}^* \simeq \sum_{k=1}^q \theta_k^* \mathbf{g}^k. \quad (6.10)$$

The unknown empirical coefficient vector $\boldsymbol{\theta}^* = [\theta_1^*, \dots, \theta_q^*]^T$ can be determined by employing an appropriate interpolation method on the empirical coefficient vectors $\boldsymbol{\theta}^i$ of the snapshots with $i = 1, \dots, n$.

6.2 Combined Interpolation Approach

Obviously, the quality of the POD-based reduced-order model depends on the employed interpolation technique, which is used to obtain the unknown coefficient $\boldsymbol{\theta}^*$. Both cubic spline interpolation and radial basis function (RBF) interpolation are employed in this work. This

section concentrates on the procedure combining RBF interpolation and POD, which is very promising methodology since as presented before that RBF interpolation is an effective method in the interpolation of high-dimensional data with a small number of sampling data.

Applying RBF interpolation, the current design vector \mathbf{x}^* and the design vectors in the snapshots set $\mathbf{x}^1, \dots, \mathbf{x}^n$ are treated as the input layer and the hidden layer of a RBFN, respectively. The vector $\boldsymbol{\theta}^* = \boldsymbol{\theta}(\mathbf{x}^*)$ in equation (6.10) is calculated as the output nodes of the network. It is expressed as a linear combination of the predefined radial basis function $\phi(z_j(\mathbf{x}^*))$, $j = 1, \dots, n$:

$$\theta_k^* = \sum_{j=1}^n \omega_k^j \phi(z_j(\mathbf{x}^*)), \quad k = 1, \dots, q, \quad (6.11)$$

where $z_j(\mathbf{x}^*)$ denotes the distance between the input \mathbf{x}^* and network center \mathbf{x}^j , which was defined before in equation (4.2). The coefficients ω_k^j can be determined by the RBF network training of the snapshots set, which is

$$\theta_k^i = \theta_k(\mathbf{x}^i) = \sum_{j=1}^n \omega_k^j \phi(z_j(\mathbf{x}^i)), \quad i = 1, \dots, n, \quad k = 1, \dots, q. \quad (6.12)$$

Equation (6.11) and (6.12) can also be written in the matrix form as

$$\boldsymbol{\theta}_{q \times 1}^* = \boldsymbol{\Omega}_{q \times n} \begin{bmatrix} \phi(z_1(\mathbf{x}^*)) \\ \phi(z_2(\mathbf{x}^*)) \\ \vdots \\ \phi(z_n(\mathbf{x}^*)) \end{bmatrix}_{n \times 1} = \boldsymbol{\Omega}_{q \times n} \boldsymbol{\phi}_{n \times 1}^* \quad (6.13)$$

and

$$\boldsymbol{\Theta}_{q \times n} = \boldsymbol{\Omega}_{q \times n} \boldsymbol{\Phi}_{n \times n}. \quad (6.14)$$

In equation (6.14), the coefficient matrix $\boldsymbol{\Theta}_{q \times n}$ is already evaluated by equation (6.4) and the radial basis function matrix $\boldsymbol{\Phi}_{n \times n}$ is defined in the same way as in equation (??). Solving a linear equation system in equation (6.15), which is a transposition of equation (6.14), the unknown matrix $\boldsymbol{\Omega}_{q \times n}$ can be obtained.

$$\boldsymbol{\Phi}_{n \times n}^T \boldsymbol{\Omega}_{n \times q}^T = \boldsymbol{\Theta}_{n \times q}^T. \quad (6.15)$$

After the training process, the coefficient vector $\boldsymbol{\theta}^*$ is calculated by equation (6.11), and thereby the flow problem corresponding to an arbitrary design vector can be approximated by POD model using equation (6.10).

The above methodology is an illustration of combining exact RBF interpolation with POD, which is the most efficient way, but in case of a very large number of snapshots, the two-step network training introduced in Section 4.2 can be used to avoid overfitting.

6.3 Optimization Procedure

The POD-based model is tested as a global approximation model in this work, which means that it is constructed using all of the snapshots. The POD basis vectors are generated once for all at the beginning of the optimization process. When necessary, it can also be easily developed into a local and online learning approximation model. A good choice of snapshots directly influences the quality of the POD-based approximation model. Both uniform sampling and LHS are applied in this work. The uniform sampling generates snapshots by calculating all the possible combinations of parameters selected from each design space. Considering that it will lead to a quick increase of the required number of snapshots with the increasing of the dimension of the design space and the collection of snapshots is a computationally expensive process, for the problems with high-dimensional design space, a more efficient data sampling approach, LHS is suggested. An attractive aspect of LHS is that the number of sampling points is independent of the dimension of the design space, e.g., it is not necessary to be a multiple of powers of the dimension of the design space.

The complete working procedure of coupling a POD-based model and NSGA-II to solve a flow shape optimization problem is summarized as follows:

1. Generate snapshots using appropriate sampling method in the design region.
2. Calculate the flow region of the snapshots using high-fidelity CFD solver and save the results in POD matrices.
3. For each POD matrix, perform the POD procedure to obtain the POD optimal basis vectors $\mathbf{g}^k, k = 1, \dots, q$, as well as the empirical coefficient vector $\boldsymbol{\theta}^i, i = 1, \dots, n$.
4. Start optimization: Initialize the parent population P_0 and Q_0 .
5. Evaluate the objective functions of the solutions in Q_g at generation g , which includes:
 - For each solution \mathbf{x}^* calculate the empirical coefficient vector $\boldsymbol{\theta}^*$ using the chosen interpolation method.
 - Approximated the flow fields using POD models.
 - Calculate the objective functions based on the approximated solutions in the flow region.
6. Merge population P_g and Q_g into a combined generation R_g .
7. Conduct Pareto front identification to R_g and fill P_{g+1} with full Pareto front $\mathcal{F}_j, j = 1, \dots, J-1$, where \mathcal{F}_{J-1} is the last full Pareto front that can be accommodated in P_{g+1} .
8. Apply crowding sort on \mathcal{F}_J and fulfill P_{g+1} with the selected solutions.
9. Copy all the newly generated Pareto solutions in R_g to the external archive A_g and update A_g .
10. Termination: if the stopping criterion is satisfied, then perform Pareto front identification on the current external archive A_g to determine the final Pareto-optimal solutions and terminate.

11. Perform crowded tournament selection on P_{g+1} and recombination operator on the selected parent solutions.
12. Perform mutation operator to generate child population Q_{g+1} , go to step 5.

6.4 Test cases

6.4.1 Test Case 1 - Pipe Junction

The same pipe shape optimization problem as that in Section 5.3.3 is used to verify the POD-based approximation models. In this test case, the snapshots are generated using uniform sampling and cubic spline interpolation is employed to calculate the POD coefficients. The POD-based models are constructed using different numbers of optimal basis vectors and different number of snapshots. A comparative study is carried out and the approximation accuracy as well as optimization performance are investigated.

Optimization Problem Definition

The geometry of the pipe is given in Figure 5.18. The fluid model, the geometry discretization as well as the optimization objective, i.e., the maximum efficiency of pressure drop reduction, are all same as those defined in the previous test case. The difference is that the shape box around B2, on which the deformation is applied, is discretized in this case equidistantly by three points in each direction. The shape box discretization and four selected moving points are shown in Figure 6.1. The deformations are performed perpendicularly to the pipe surface with an initial amount of $H/20$, which are given in Figure 6.2 together with the corresponding design variables. The displacements of the control points are bounded between $9H$ and $15H$. The optimization problem is formulated as follows:

$$\begin{aligned}
 & \text{minimize} && \eta_p(\mathbf{x}), \\
 & \text{with} && \mathbf{x} = [x_1, \dots, x_4]^T, \\
 & \text{subject to} && 9H \leq x_1, \dots, x_4 \leq 15H,
 \end{aligned} \tag{6.16}$$

POD Model Construction and Validation

Considering this optimization problem has only four design variables, uniform sampling is employed for the snapshots generation. To investigate the performance of the POD-based approximation, the POD models are constructed by using 256 snapshots and 625 snapshots, respectively. Snapshot is defined as the pressure field and calculated by the flow solver FASTEST. The total number of the calculated grid points is 57352. The information of 256 snapshots is stored in a 57352×256 snapshot matrix \mathbf{F}_p . Another snapshot matrix with a size of 57352×625 saves the information of 625 snapshots. From these two snapshot matrices, 256 and 625 POD basis vectors are extracted and ordered according to the magnitude of their corresponding eigenvalues by the POD procedure, respectively.

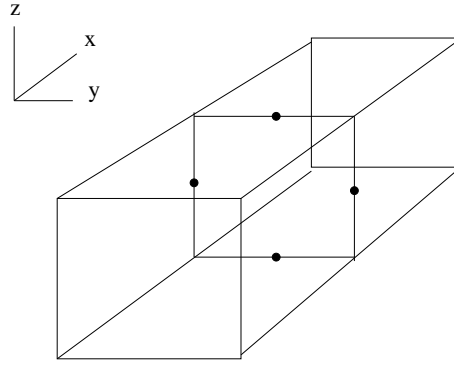


Figure 6.1: Shape box discretization and the selected control points (pipe - 4 DVs)

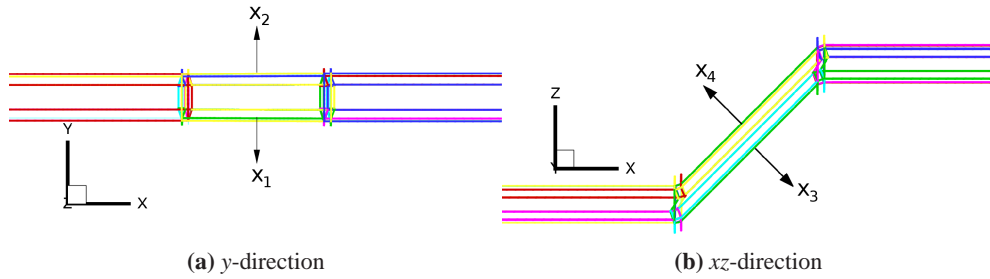


Figure 6.2: Deformation directions and the corresponding DVs (pipe - 4 DVs)

A general survey on how much information of the snapshots is captured by the POD basis vectors is first carried out. All of the snapshots are reconstructed using a different number of POD basis vectors (truncation degree M). The optimization objectives $\eta_{p,POD}$ of the snapshots that are reconstructed using POD models are calculated. Figure 6.3 plots the average percentage reconstruction error $e_{\eta,ave}$ of all of the snapshots against the number of employed POD basis vectors. This figure shows that the approximation error reduces when increasing the number of employed POD basis vectors for the reconstruction. In both cases, errors obtained using same truncation degree M are close to each other. The error of the models with 100 basis vectors can even reach about $10^{-5}\%$. From another point of view, it reflects that about 99.99999% of the system energy are captured by 100 basis vectors.

To determine the number of optimal POD basis vectors that should be involved in constructing the approximation models for the later prediction of the unknown solutions, both the model accuracy and the computational time should be taken into account. The model prediction accuracy is evaluated by approximating the pressure field of an arbitrary design vector using 10 and 100 POD basis vectors extracted from 256 and 625 snapshot matrices, respectively. The pressure contours obtained with $M = 10$ and $M = 100$ are compared to those obtained using FASTEST simulations in Figure 6.4 (256 snapshots) and Figure 6.5 (625 snapshots). Table 6.1 also gives the comparison of CPU time. Obviously, the POD models which were constructed using more basis vectors can provide more accurate results. Both figures show that in the

case of $M = 100$ the pressure distribution agrees very well with that obtained by FASTEST simulation, while comparable results cannot be achieved in the case of $M = 10$. Moreover, the calculation time when using 100 basis vectors is only about four times that required when using 10 basis vectors with respect to the time cost required by the flow solver FASTEST. Therefore, 100 basis vectors were decided to be employed for the POD model construction. Furthermore, to verify the prediction performance of the constructed POD models, 20 models of the pipes with arbitrarily deformed shapes are calculated using both FASTEST and the POD-based models constructed by 256 and 625 snapshots, respectively. The corresponding average percentage errors of 20 POD approximation are 0.0086% (256 snapshots) and 0.0038% (625 snapshots). This result shows that both POD models are able to provide satisfying approximation results, but obviously the model based on 625 snapshots behaves better.

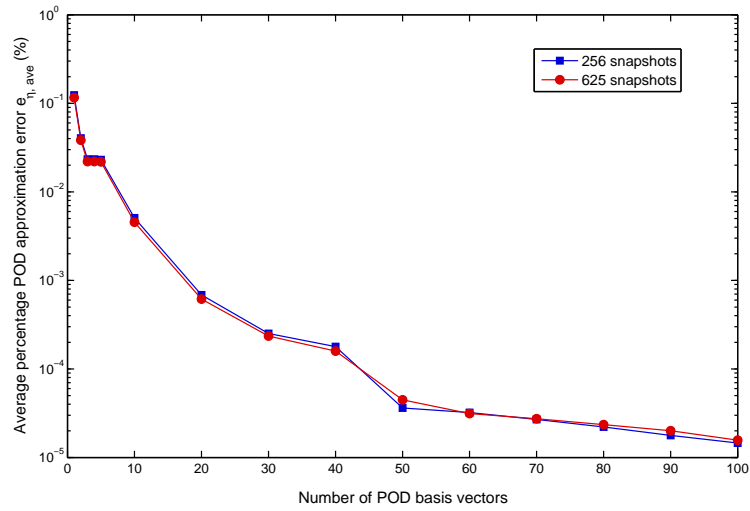


Figure 6.3: Average reconstruction errors of 256 and 625 snapshots (pipe - 4 DVs)

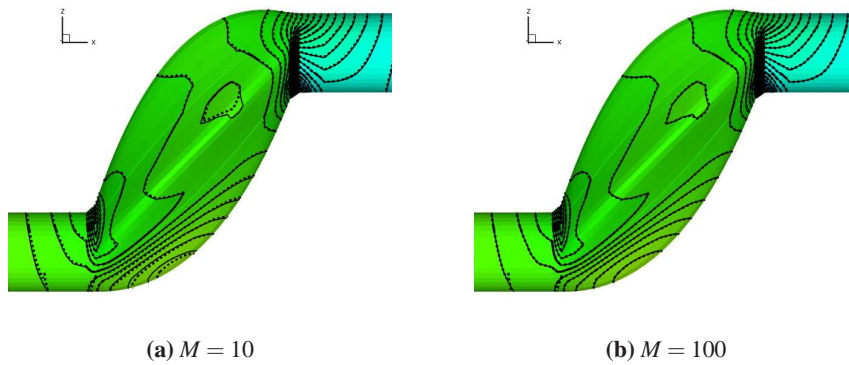


Figure 6.4: Comparison exact pressure contour (solid) and POD approximated pressure contour (dash) using 256 snapshots (pipe - 4 DVs)

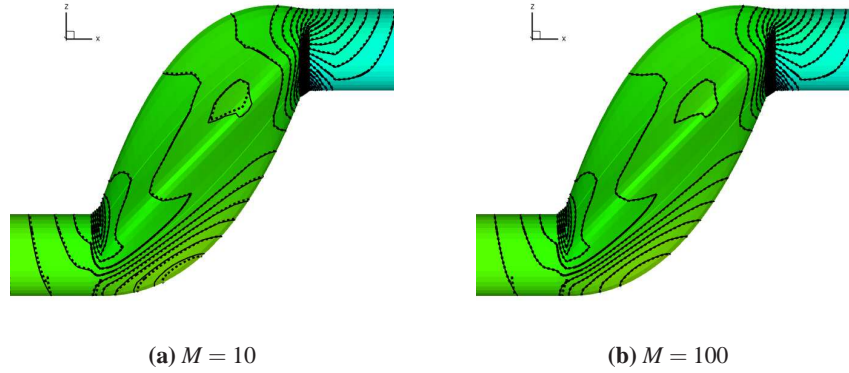


Figure 6.5: Comparison exact pressure contour (solid) and POD approximated pressure contour (dash) using 625 snapshots (pipe - 4 DVs)

Table 6.1: Comparison of CPU time (pipe - 4 DVs)

Evaluation method	CPU (s)
FASTEST	144
POD (256 snapshots, $M = 10$)	0.277
POD (625 snapshots ($M = 10$))	0.281
POD (256 snapshots, $M = 100$)	1.246
POD (625 snapshots, $M = 100$)	1.682

Optimization Results

Optimization is performed by employing the modified NSGA-II. The population size is 20 and initial population is generated randomly in the box constraint of the optimization problem. Crowded tournament selection, SBX crossover and real polynomial mutation are employed as the genetic operators. The recombination probability p_c and mutation probability p_m are 0.9 and 0.25, respectively. Stopping criterion is defined such that the optimizer stops automatically if the optimal pressure drop doesn't improve after 20 generations. Three optimization runs are carried out using the flow solver FASTEST, POD models constructed using 256 and 625 snapshots for function evaluations, and the optimal solutions are achieved at the generation 66, 49 and 55, respectively. The corresponding numbers of function evaluations are 1320, 980 and 1100. Figure 6.6 plots the optimization history. The objective values obtained by the POD approximations are recalculated using the flow solver FASTEST. Table 6.2 gives an overview of the optimal design variables and the achieved optimal solutions in three optimization runs. A symmetric deformation can be observed in y -direction, this result coincides with the results obtained using 8 design variables in the previous test case. Concerning the optimization performance, the same optimal efficiency of 23.2949% is obtained by employing FASTEST and the POD-based model (625 snapshots), which is slightly better than that obtained by using the POD-based model (256 snapshots). The results concerning the design variables are all similar, and on three control points the values obtained using FASTEST and using POD model (625 snapshots) are even the same. From the above comparison it can be concluded that the POD

models based on 625 snapshots perform better than those based on 256 snapshots in approximating the function values. However, constructing the POD models using 256 snapshots is computationally cheaper and the optimization result is also adequate.

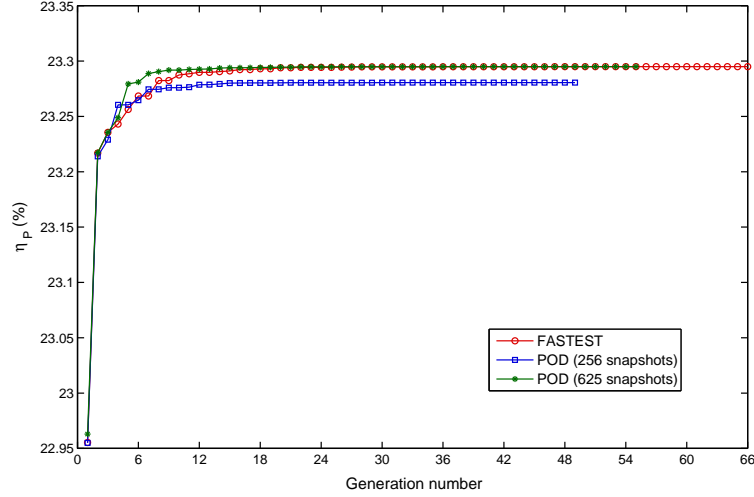


Figure 6.6: Comparison of optimization history (pipe - 4 DVs)

Table 6.2: Comparison of optimization results (pipe - 4 DVs)

	FASTEST	POD model (625)	POD model (256)
DVs	12.54H	12.53H	12.53H
	12.53H	15.53H	12.51H
	11.98H	11.98H	11.91H
	15H	15H	15H
η_p	23.2949%	23.2949%	23.2948%

6.4.2 Test Case 2 - Heat Exchanger

In the second test case, the optimization is performed on a 3D fin-tube heat exchanger to examine the effects of the fin shapes on both the heat transfer performance and cooling power. In this case, the snapshots for constructing the POD models are generated using LHS. RBF interpolation is combined with POD procedure to calculate the POD coefficients. The comparisons of POD-based models constructed using different numbers of POD optimal basis vectors are conducted. The approximation accuracy and optimization efficiency using POD models are investigated.

Problem Definition

The tube arrangement of the heat exchanger is staggered and the fins are of wave type. The flow model is laminar forced convection. The air enters at an inlet temperature $T_{in} = 300K$ and velocity $u_{x,in} = 0.3m/s$. The fluid is assumed to be incompressible with constant property. The Reynolds number based on the fin pitch is $Re = 285$ and the Prandtl number is $Pr = 0.71$. Non-slip conditions and a constant temperature of $700K$ are specified on the walls and tubes. Only the entrance region of the heat exchanger is considered for the optimization. The length of the hydrodynamic entrance region $x_{e,h}$ and thermal entrance region $x_{e,t}$ are $244mm$ and $173mm$, respectively, which are calculated by

$$\begin{aligned} x_{e,h} &= 0.05D_h Re, \\ x_{e,t} &= 0.05D_h Pr Re \end{aligned} \quad (6.17)$$

with the hydraulic diameter D_h . The optimization model is chosen between two fins in the entrance region with a length of $150mm$ including two half tubes. A top view of the fin-tube heat exchanger and the selected optimization domain are given in Figures 6.7 and 6.8, respectively.

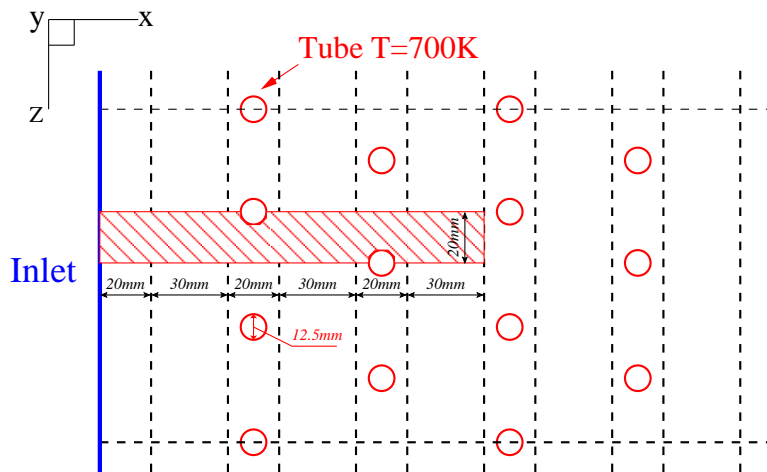


Figure 6.7: Top view of a fin-tube heat exchanger

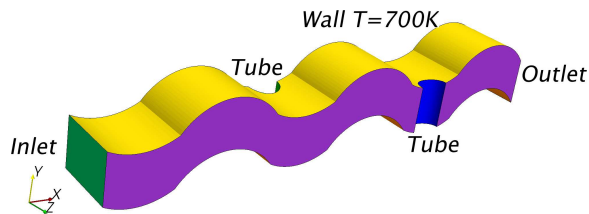


Figure 6.8: Selected optimization domain (fin-tube heat exchanger)

The shape deformation is applied on 6 shape boxes using FFD. Shape boxes $B1$ and $B2$

are used for illustration. As shown in Figure 6.9, they are discretized equidistantly by 4, 2, 2 points and 5, 2, 2 points in x , y and z -direction, respectively. Deformation will be accomplished by moving 20 selected control points in the xy -plane. Using the symmetry property of the problem, it is defined that displacements of the control points with same x coordinates are same. Therefore, the number of design variables is reduced to five. The deformation directions of the control points and the corresponding design variables are illustrated in xy -plane in Figure 6.10. The deformations have an initial amount of 0.5mm. The control points and the design variables on the other four shape boxes are chosen in the same way.

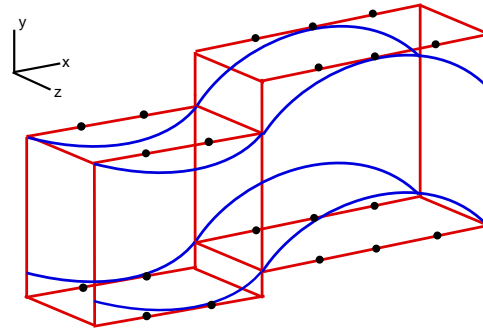


Figure 6.9: Shape boxes and selected control points (fin-tube heat exchanger)

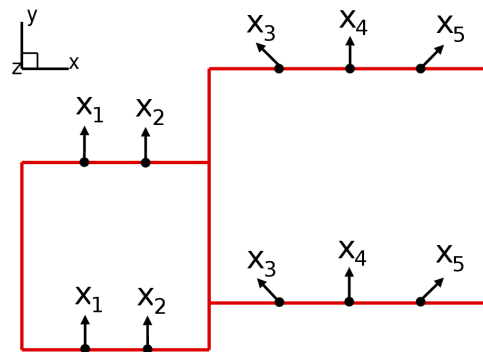


Figure 6.10: Deformation directions and the corresponding DVs (fin-tube heat exchanger)

The optimization task consists of achieving the maximum heat transfer coefficient h and

minimum pressure drop Δp between inlet and outlet:

$$\begin{aligned}
&\text{minimize} && \Delta p(\mathbf{x}) = \bar{p}_{\text{in}} - \bar{p}_{\text{out}}, \\
&\text{maximize} && h = \frac{Q}{\Delta T A_s}, \\
&\text{with} && \mathbf{x} = [x_1, \dots, x_5]^T, \\
&\text{subject to} && -10 \leq x_1, x_2, x_4 \leq 10, \\
&&& -20 \leq x_3, x_5 \leq 25,
\end{aligned} \tag{6.18}$$

where A_s is the surface area. The total heat transfer Q and the log-mean temperature difference ΔT are defined previously in equation (5.16). FASTEST is employed to calculate the snapshots. The flow model is assumed to be laminar, steady flow and the buoyancy effects are neglected.

POD Model Construction and Validation

The POD models are applied to the pressure, x -component of velocity and temperature fields. Using LHS, 200 snapshots are generated. In this case, the design region of each design variable is divided into 200 intervals using uniform distribution. From each interval one value is selected randomly, which means 200 values are prepared for each design variable. These values are then randomly permuted in order to maximize the minimum distance between the sampling points. The number of the calculated grid points is 138251. The simulation results of these 200 snapshots using FASTEST are stored in three 138251×200 snapshots matrices \mathbf{F}_p , \mathbf{F}_{u_x} and \mathbf{F}_T , respectively. Conducting the POD procedure, 200 POD basis vectors are extracted from the snapshots for each physical field and ordered according to the magnitude of their corresponding eigenvalues.

Like in the previous test case, the accuracy of the snapshots reconstruction and the unknown solution prediction with different truncation degree M are investigated. Figure 6.11 plots the average percentage reconstruction error of the 200 reconstructed snapshots for pressure drop $e_{\Delta p, \text{ave}}$ and heat transfer coefficient $e_{h, \text{ave}}$ against the number of employed POD optimal basis vectors. It can be observed that using 20 basis vectors, the reconstruction errors for both objectives are already less than 0.1%. In other words, more than 99.9% of the system information is captured by the first 20 POD basis vectors. When employing 100 basis vectors, the average error of pressure drop is below 0.001%. To quantify the POD prediction performance, the objectives of 30 deformed shapes that are arbitrarily selected in the design region but not included in the snapshots set, are calculated by FASTEST and POD models with $M = 20$, $M = 60$ and $M = 100$, respectively. The empirical coefficient vector $\boldsymbol{\phi}^*$ of each unknown shape is interpolated using RBF interpolation. The inverse multiquadric function is employed as RBF in this case. Table 6.3 shows the comparison of required CPU time and average percentage error of 30 POD approximations. One can observe that the CPU time depends almost linearly on the number of employed POD optimal basis vectors. Compared to the approximation results obtained by using 20 POD basis vectors, the results by using 60 POD basis vectors are apparently more accurate for both objectives. The accuracy doesn't improve much when using

100 POD basis vectors. Therefore, 60 POD basis vectors are chosen to be employed for the POD model construction for the later optimization. For the model validation, the contour plot of pressure, x-component of velocity and temperature distribution for an arbitrary deformed shape obtained by using FASTEST and POD models with $M=60$ are compared in Figure 6.12, 6.13 and 6.14. One can see that the contour plots obtained by POD models agree very closely with the FASTEST solutions.

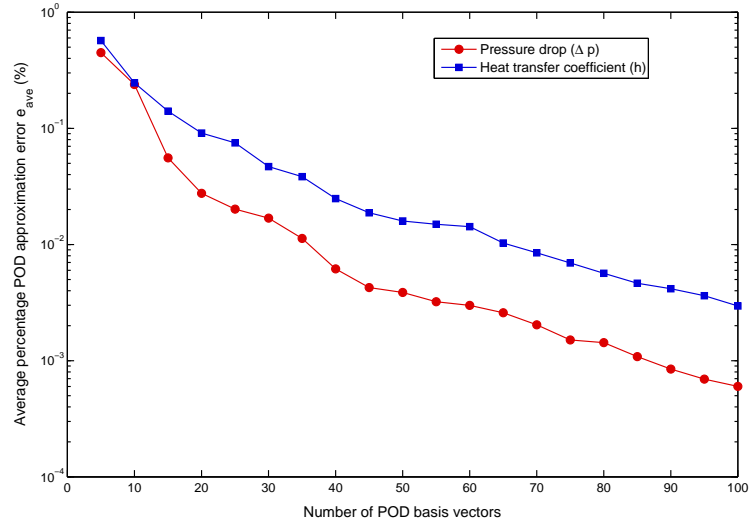


Figure 6.11: Average reconstruction error of 200 snapshots (fin-tube heat exchanger)

Table 6.3: Comparison of CPU time (fin-tube heat exchanger)

Evaluation method	CPU (s)	$e_{\Delta p,ave}(\%)$	$e_{h,ave}(\%)$
FASTEST	622	–	–
POD ($M = 20$)	1.99	0.0739	0.113
POD ($M = 60$)	5.82	0.0705	0.039
POD ($M = 100$)	9.85	0.0703	0.030

Optimization Results

The optimization runs 60 generations with a population size of 40, i.e., a total number of 2400 function evaluations using selected POD models are performed for both optimization objectives. Crowded tournament selection, SBX crossover and real polynomial mutation are employed as the genetic operators with a recombination probability $p_c = 0.9$ and a mutation probability $p_m = 0.2$. The objective values of the obtained optimal solutions are recalculated by FASTEST and plotted in Figure 6.15. Furthermore, two utmost solutions and two trade-off solutions are chosen from the Pareto front. Their corresponding fin shapes and the objective

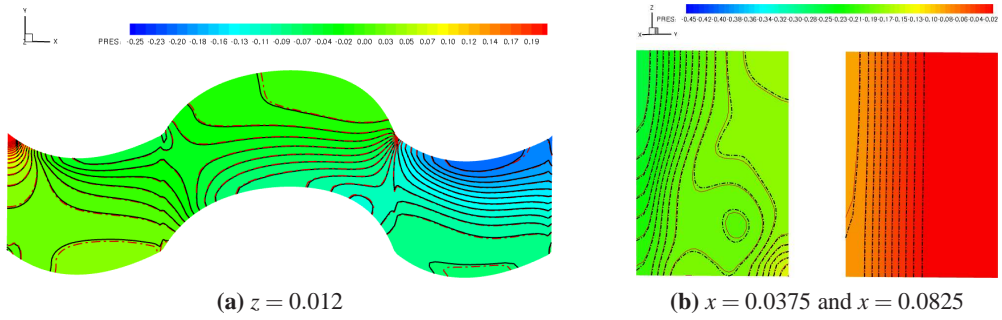


Figure 6.12: Comparison of exact (solid) and POD (dash) pressure contour (fin-tube heat exchanger)

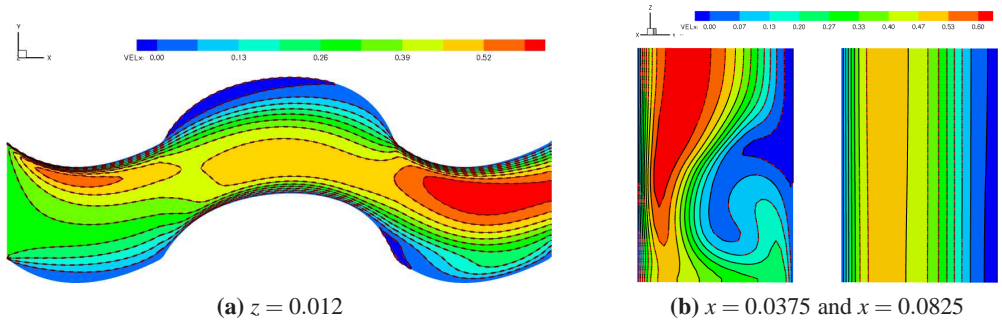


Figure 6.13: Comparison of exact (solid) and POD (dash) x -velocity contour (fin-tube heat exchanger)

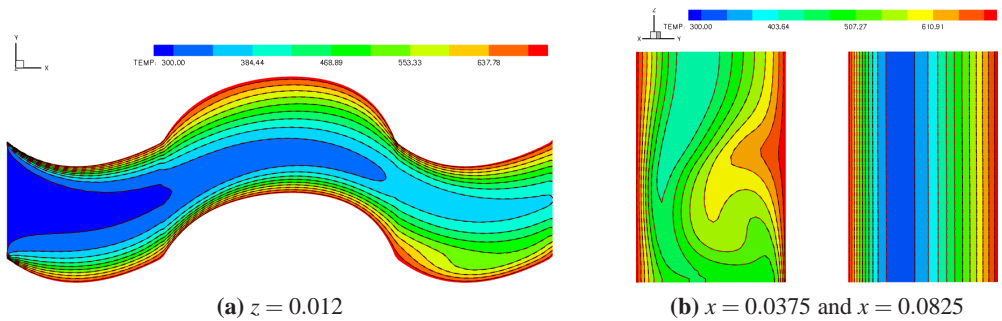


Figure 6.14: Comparison of exact (solid) and POD (dash) temperature contour (fin-tube heat exchanger)

values are illustrated in Figure 6.16 and Table 6.4, respectively. Both optimization objectives, the heat transfer and fan power, could be improved by choosing appropriate design variables, e.g., the solution S3 improves the initial heat transfer coefficient by 2.6% and reduces the pressure drop by 9.9%. The optimal Pareto front also provides sufficient compromise solutions to meet different design preferences. The greatest improvements for pressure drop and the heat transfer coefficient are 39.42% and 17.41%, that are achieved by the solution S2 and S1, respectively.

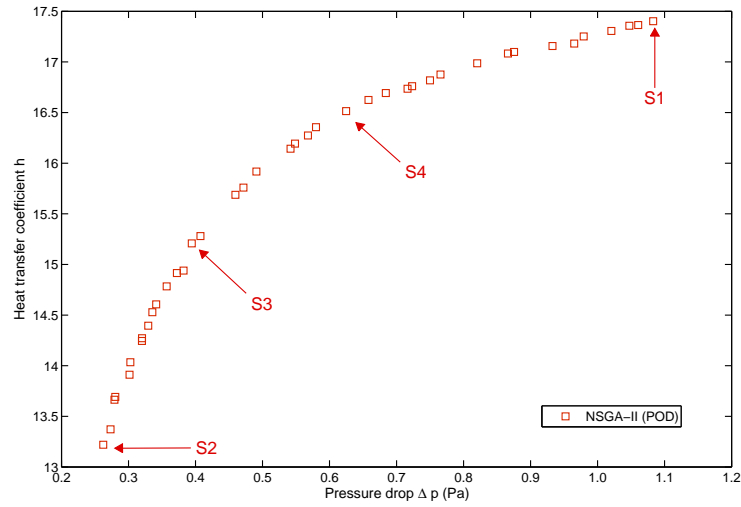


Figure 6.15: Pareto-optimal solutions achieved by POD function evaluations (fin-tube heat exchanger)

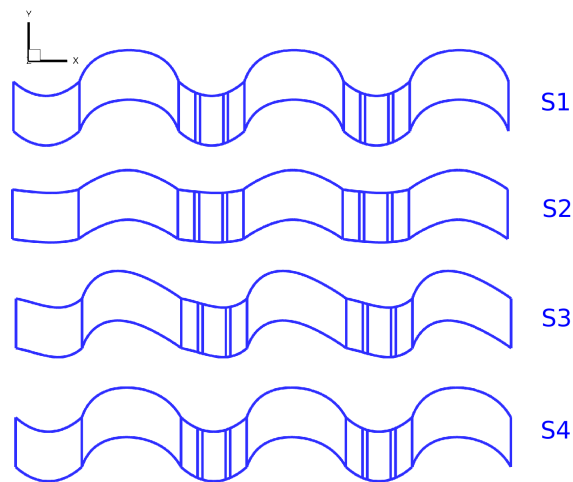


Figure 6.16: Four exemplary optimal shapes (fin-tube heat exchanger)

Only using FASTEST for function evaluations, the optimization is conducted again with the same genetic operators and optimization parameters. The achieved Pareto front is compared with that achieved by POD models in Figure 6.17. It can be observed that two runs yield a similar Pareto front. The Pareto solutions obtained by using POD models spread a larger extent than those obtained by FASTEST. A quantitative comparison of the performance of both optimization results and computational cost is given in Table 6.5. Regarding the computational cost, in the first run it means a sum of the cost required by the snapshots collection, POD basis vector construction, POD-based function approximations, and the recalculation of obtained Pareto-optimal solutions, in the second run only the cost for FASTEST evaluations are taken

Table 6.4: Objective values of 4 exemplary optimal solutions (fin-tube heat exchanger)

	$S1$	$S2$	$S3$	$S4$
Δp (Pa)	1.083	0.262	0.394	0.625
h	17.402	13.219	15.208	16.514

into account. One can find that the Pareto solutions obtained by POD models are actually even better than the solutions obtained using FASTEST. They are closer to the true Pareto front, have a better diversity, and also dominate 27.5% of the solutions obtained by using FASTEST, while only 12.5% of them are dominated by the solutions obtained by FASTEST. Moreover, the computational cost is dramatically reduced by using POD models (about 11% of the cost required by using FASTEST).

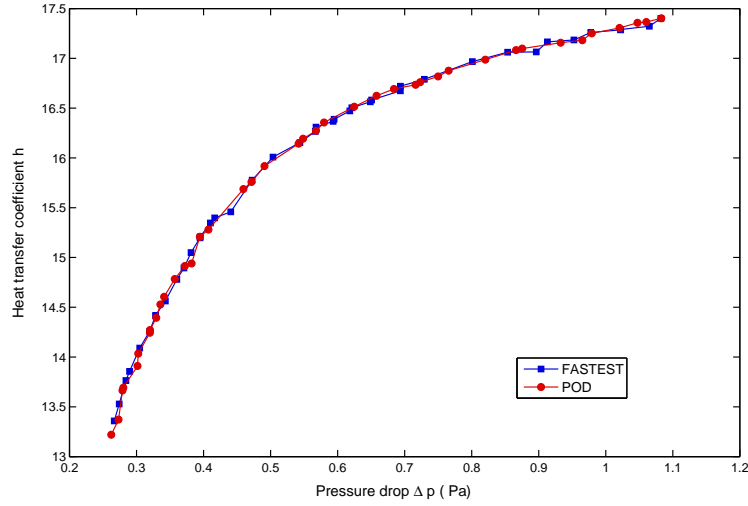


Figure 6.17: Comparison of Pareto fronts obtained using FASTEST and POD models (fin-tube heat exchanger)

Table 6.5: Performance comparison of two optimization runs (fin-tube heat exchanger)

No. of Run	1	2
Function evaluation	FASTEST	POD
Hypervolume (HV)	0.7061	0.7066
Spacing (SP)	0.0247	0.0207
Set Coverage metric (SCM)	$(\mathcal{P}^2, \mathcal{P}^1) = 27.50\%$	$(\mathcal{P}^1, \mathcal{P}^2) = 12.50\%$
Computational cost (h)	414.67	45.3

Chapter 7

Conclusions

This dissertation provides an efficient optimization methodology for the purpose of solving multiobjective flow shape optimization problems. Due to the facts that for the CFD-based optimization problems there is no easy access to the derivative information and the flow simulation is usually time consuming, gradient-independent and computationally efficient are two important factors that should be taken into account. The proposed optimization methodology is based on the evolutionary optimization method for its well-known derivative-free property as well as the advantages in dealing with MOOPs and providing global optimal solutions. Meanwhile, the approximation models and the deterministic optimization methods are combined with the evolutionary optimization to improve the optimization efficiency and local convergence. The optimization process consists of two parts: the design space exploration using EA (global search) and the convergence acceleration using the deterministic method (local search).

For the global search, a high performance, elitist evolutionary method NSGA-II is employed with several modifications that include a parallel optimization scheme based on the master-slave model to save the computational time, an efficient sampling method to explore the initial design space, an additional archive population as well as a final selection procedure to avoid the lost of true Pareto solutions. Moreover, the online and locally trained RBFN is used as an approximation model to replace the expensive flow solver for function evaluations in some generations. The adaptive exchange between the exactly and approximately evaluated generations is accomplished through an approximation control procedure. Once the global search is completed, the obtained result(s) is(are) supposed to near to the optima and treated as the starting point(s) for the local improvement. When dealing with MOOPs, the starting points are selected using the clustering from the entire or a part of the Pareto front method according to different design preferences. Several methods employing the weighting factors are provided to convert the MOOP into a SOOP, and the selection of an appropriate one is problem-dependent. Besides, to ensure a fast local convergence, different pseudo-weights are assigned to the corresponding starting points. Two optimization tools DFO and CONDOR are chosen to perform the local search. Both of them employ the trust-region framework in the derivative-free case. Inside the trust-region, instead of the objective functions, the constructed approximation models are optimized. Hence, no derivative information of the objective function with respect to the design variables is required.

For solving flow shape optimization problems a complete optimization framework is de-

veloped. The optimizer manages and controls the whole optimization process. The shape variation and flow simulation are incorporated for the evaluation of objective functions and the construction of the database. FFD is selected for the shape variation because it directly modifies the computational grids required by the flow solver and provides a flexible deformation through the movement of only a small number of points, which is especially advantage since the number of design variables is an important factor that determines the optimization cost and also influence the accuracy of the approximation model. The flow simulation is performed using the in-house developed finite-volume flow solver FASTEST.

In order to evaluate the performance of the proposed optimization methodology, it is applied to two analytical benchmark optimization problems and two numerical shape optimization problems. The influence of RBFN construction methods and the number of solutions in the initial database on the approximation accuracy is investigated. The comparisons of the optimal solution(s) achieved using different optimization schemes are carried out. The main conclusions are summarized as follows:

- The incorporation of approximation models overcomes the requirement of large numbers of computationally expensive function evaluations. The online and locally trained RBFN approximation model and the error control procedure makes it possible to keep the approximation error under a very small value even in the case of a high-dimensional design space.
- A larger initial database will not necessarily yield better Pareto solutions at last. This is because the required number of exact function evaluations is adaptively updated according to the approximation error calculated in the control generation. Even the approximation at beginning may be not good enough; it will be improved after several generations by adding more solutions into the database.
- When using the regularized forward selection method to determine the network centers, the choice of RBFs has a significant influence on the accuracy of the approximation model, and this selection method fails to provide a good approximation using all the tested RBFs except for the multiquadric function. When using regression tree method, the influence of RBF types on the approximation accuracy is minor, the model accuracy obtained by all RBFs is quite satisfying.
- For multiobjective optimization test cases, using the proposed optimization methodology, a set of optimal solutions with good diversity have been obtained with much less computational cost, and most of them dominate the reference solutions that achieved by only running evolutionary optimization for a large number of generations. Besides, it works well for both convex and nonconvex objective spaces.
- In the numerical test case with respect to the shape optimization of a pipe conjunction, two local search tools DFO and CONDOR are compared. Though CONDOR requires more function evaluations for the model construction at the previous stage, it converged much faster to the optimum afterwards than DFO. Furthermore, the comparison to another CONDOR run, which started from the initial point, verifies the fact that if the starting point is in the near optimal region it is more likely to find the global optimum, otherwise the optimization process may get stuck into a local optimum.

Furthermore, this work proposed an advanced metamodeling framework based on the POD technique. POD provides a way to represent the flow field efficiently using the linear combination of a set of basis vectors. Combining POD with an appropriate interpolation method, not only the objective functions but also the entire flow region can be approximated efficiently. This kind of approximation model is superior to other models since the physical behavior of the flow region as well as the other design objectives of the solutions (both intermediate and final solutions) can be easily accessed during or after the optimization process. The POD-based approximation models are applied to two shape optimization test cases with different sampling and interpolation techniques. The approximated flow fields include velocity, pressure and temperature. The comparison of POD models constructed with different truncation degree M as well as the numbers of snapshots are carried out. The investigation on the POD approximation accuracy and optimization performance leads to the following conclusions:

- The POD models constructed by larger number of snapshots are more accurate.
- The more optimal basis vectors are involved in the model construction, the more accurate POD models are obtained.
- The selection of appropriate number of snapshots and POD optimal basis vectors for the model construction is problem-dependent.
- With a drastic reduction of the computational cost, optimal solutions obtained by using POD models may achieve similar quality compared to those obtained by just using flow solver for function evaluations.

The efficiency of the proposed optimization strategy has been successfully validated by several test cases. It also should be applied to real engineering shape optimization problems in the future to prove its robustness. Also the construction of an accurate approximation model, especially in the high-dimensional space, is always a challenging task and needs further investigation. Although a lot of issues have been considered in this work, there is no guarantee that the best approximation is provided. Besides, this work has shown the promising properties of the POD-based reduced-order models. An especially interesting aspect of the ongoing research is to develop an online updated or(and) locally trained POD-based model. For an online trained POD-based model, the major difficulty consists in how to effectively and efficiently update the database because the POD procedure itself may involve the calculation of large-dimensional matrix. Furthermore, it is desired to extend the application of POD to the approximation of structure fields or the time-dependent models for solving shape optimization problems involving fluid-structure interaction or unsteady flow simulations.

Bibliography

- [1] FASTEST user manual. Available from: <http://www.fnb.tu-darmstadt.de/de/software/fastest/group/download/fastdoc/index.html>, 2004. → pages 8
- [2] Introduction to radial basis function networks. Available from: <http://www.anc.ed.ac.uk/rbf/rbf.html>, 1996. → pages 33
- [3] Recent advances in radial basis function networks. Available from: <http://www.anc.ed.ac.uk/rbf/rbf.html>, 1999. → pages 33
- [4] K. Afanasiev and M. Hinze. Adaptive control of a wake flow using proper orthogonal decomposition. *Lecture Notes Pure Applied Mathematics*, 216:317–332, 2001. → pages 74
- [5] L. Albert and D.E.Goldberg. Efficient discretization scheduling in multiple dimensions. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 271–278. Morgan Kaufmann, 2002. → pages 3
- [6] N. Ali and K. Behdinan. Optimal geometrical design of aircraft using genetic algorithms. *Transactions of the Canadian Society for Mechanical Engineering*, 26(4): 373–388, 2003. → pages 3
- [7] J. D. Anderson. *Computational Fluid Dynamics: An Introduction*. Springer, Berlin, 2 edition, 1996. → pages 7
- [8] G. K. Batchelor. *An Introduction to Fluid dynamics*. Cambridge University Press, 1 edition, 2000. → pages 7
- [9] F. V. Berghen. *A Constrained, Non-linear, Derivative-free Parallel Optimizer for Continuous, High Computing Load, Noisy Objective Functions*. PhD Dissertation, University of Brussels, Belgium, 2004. → pages 45, 47
- [10] S. Blum, R. Puisa, J. Riedel, and M. Wintermantel. Adaptive mutation strategies for evolutionary algorithms: a comparative benchmark study. In *Proceedings of EUROGEN2005*, 2005. → pages 2
- [11] D. Buche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with Gaussian process fitness function models. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(2):183–194, 2005. → pages 3

- [12] M. D. Buhmann. *Radial Basis Function: Theory and Implementations*. Cambridge University Press, 1 edition, 2003. → pages 30
- [13] T. Bui-Thanh, M. Damodaran, and K. Willcox. Aerodynamic data reconstruction and inverse design using proper orthogonal decomposition. *AIAA Journal*, 42(8): 1505–1516, 2004. → pages 74
- [14] T. Burczyński. The boundary element formulation for multiparameter structural shape optimization. *Applied Mathematical Modelling*, 9(4):195–200, 1985. → pages 1
- [15] J. Chen. Theoretical analysis of multi-objective genetic algorithms convergence time, population sizing, and disequilibrium. Technical report, Report for IEEE NNS Walter Karplus Research Grant, 2003. → pages 2
- [16] S. Chen, C. F. N. Cowan, and P. M. Grant. Orthogonal least squares learning for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991. → pages 34
- [17] A. R. Conn, K. Scheinberg, and P. L. Toint. A derivative free optimization algorithm in practice. In *Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998. → pages 45
- [18] S. Coquillart. Extended free-form deformation: a sculpturing tool for 3D geometric modeling. *Computer Graphics*, 24(4):187–196, 1990. → pages 9
- [19] H. Corley. A new scalar equivalence for Pareto optimization. *IEEE Transactions on Automatic Control*, 25(4):829–830, 1980. → pages 20
- [20] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation Journal*, 7(3):205–230, 1999. → pages 3, 50
- [21] K. Deb. A population-based algorithm-generator for real-parameter optimization. Technical Report 2003003, KanGAL, 2003. → pages 2
- [22] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, LTD, 1 edition, 2001. → pages 2, 18, 43
- [23] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex System*, 9(2):115–148, 1995. → pages 2
- [24] K. Deb, D. Joshi, and A. Anand. Real coded evolutionary algorithms with parent centric recombination. Technical Report 2001003, KanGAL, 2001. → pages 2
- [25] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. Technical Report 2002003, KanGAL, 2002. → pages 2
- [26] K. Deb, A. Pratep, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2): 181–197, 2002. → pages 2

- [27] K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread Pareto-optimal solutions. Technical Report 2003002, KanGAL, 2003. → pages 22
- [28] L. Dumas, V. Herbert, and F. Muyl. Comparison of global optimization methods for drag reduction in the automotive industry. *Lecture Notes in Computer Science*, 3483: 948–957, 2005. → pages 3
- [29] R. Duvigneau. Adaptive parameterization using free-form deformation for aerodynamic shape optimization. Technical Report N°5949, INRIA, 2006. → pages 11
- [30] R. Duvigneau and M. Visonneau. Hybrid genetic algorithms and artificial neural networks for complex design optimization in CFD. *International Journal for Numerical Methods in Fluids*, 44(11):1257–1278, 2004. → pages 3
- [31] R. Duvigneau, B. A. E. Majd, and J.-A. Desideri. Towards a self-adaptive parameterization for aerodynamic shape optimization. *European Series in Applied and Industrial Mathematics*, 22:169–174, 2007. → pages 3
- [32] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999. → pages 2
- [33] B. El Majd-Jean-Antoine, J.-A. Désidéri, and R. Duvigneau. Multilevel strategies for parametric shape optimization in aerodynamics. *European Journal of Computational Mechanics*, 17(1-2):149–168, 2008. → pages 3
- [34] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and L. Giannakoglou. *Metamodel assisted evolution strategies*, pages 361–370. Parallel Problem Solving from Nature - PPSN VII. Springer Berlin / Heidelberg, 2002. → pages 3
- [35] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006. → pages 3
- [36] G. Fabbri. Heat transfer optimization in corrugated wall channels. *International Journal of Heat and Mass Transfer*, 43(23):4299–4310, 2000. → pages 3
- [37] G. E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific Publishing, 1 edition, 2007. → pages 31
- [38] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, Berlin, 3 edition, 2002. → pages 7
- [39] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley, Chichester, UK, 1 edition, 1966. → pages 2
- [40] C. Fonseca and P. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. II. Application example. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):38–47, 1998. → pages 55

- [41] D. M. Fudge, D. W. Zingg, and R. Haimes. A CAD-free and a CAD-based geometry control system for aerodynamic shape optimization. In *AIAA Paper 2005-451, 43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005. → pages 9
- [42] J. Garnier and L. Kallel. Statistical distribution of the convergence time of evolutionary algorithms for long path problems. *IEEE Transactions on Evolutionary Computation*, 4(1):16–30, 2000. → pages 2
- [43] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Science*, 38(1):43–76, 2002. → pages 31
- [44] K. C. Giannakoglou, D. Papadimitriou, and I. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodel. *Computer Methods in Applied Mechanics and Engineering*, 195(44-47):6312–6329, 2006. → pages 3
- [45] D. E. Goldberg. *Genetic Algorithms for Search, Optimization and Machine Learning*. Addison-Wesley, 1 edition, 1989. → pages 2
- [46] D. E. Goldberg and K. Deb. *A comparison of selection schemes used in genetic algorithms*, pages 69–93. *Foundations of Genetic Algorithms 1*. Morgan Kaufmann, 1991. → pages 2
- [47] G. H. Golub, M. Heath, and G. Wahba. Generalised cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979. → pages 33
- [48] R. Haftka and R. Grandhi. Structural shape optimization - a survey. *Computer Methods in Applied Mechanics and Engineering*, 57(1):91–106, 1986. → pages 1
- [49] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971. → pages 20
- [50] R. L. Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research*, 76:1905–1915, 1971. → pages 28
- [51] Z. Harth. *Automated Numerical Shape Optimization of 3-dimensional Flow Geometry Configurations*. PhD Dissertation, Technische Universität Darmstadt, Germany, 2008. → pages 9, 47
- [52] Z. Harth and M. Schäfer. Investigation of derivative-free optimization tools for optimizing flow geometries. In *Proceedings of EUROGEN2005*, 2005. → pages 11
- [53] J. He and X. Yu. Conditions for the convergence of evolutionary algorithms. *Journal of Systems Architecture*, 47(7):601–612, 2001. → pages 2
- [54] J. Herskovits, G. Dias, and C. M. Soares. A full-stress technique for structural shape optimization. *International Journal of Applied Mathematics and Computer Science*, 6(2):303–319, 1996. → pages 1

- [55] K. Hirschen and M. Schäfer. A study on evolutionary multi-objective optimization for flow geometry design. *Computational Mechanics*, 37(2):131–141, 2004. → pages 3
- [56] K. Hirschen and M. Schäfer. Optimal design of heat exchanger configurations. In *Proceedings of the 4th International Conference on Computational Heat and Mass Transfer*, pages 834–838, 2005. → pages 3
- [57] K. Hirschen and M. Schäfer. Bayesian regularization neural networks for optimizing fluid flow processes. *Computer Methods in Applied Mechanics and Engineering*, 195(7-8):481–500, 2006. → pages 3
- [58] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. The University of Michigan Press, Ann Arbor, USA, 1 edition, 1975. → pages 2
- [59] P. Holmes, J. Lumley, and G. Berkooz. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge Monographs on Mechanics,. Cambridge University Press, 1996. → pages 74
- [60] M. Hortmann, M. Peric, and G. Scheuerer. Finite volume multigrid prediction of laminar natural convection: Benchmark solutions. *International Journal of Numerical Methods in Fluids*, 11(2):189–207, 1990. → pages 8
- [61] R. J. Howlett and L. C. Jain. *Radial Basis Function Networks 2 - New Advances in Design*. Springer, Heidelberg, 1 edition, 2001. → pages 31
- [62] X. Hu, Z. Huang, and Z. Wang. Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms. In *Proceedings of the 2003 Congress on Evolutionary Computation CEC 03*, volume 2, pages 870–877, 2003. → pages 3
- [63] R. L. Iman, J. C. Helton, and J. E. Campbell. An approach to sensitivity analysis of computer models, part 1. introduction, input variable selection and preliminary variable assessment. *Journal of Quality Technology*, 13(3):174–183, 1981. → pages 24
- [64] K. S. J. Mehnen, T. Michelitsch and T. Kohlen. pMOHypEA: parallel evolutionary multiobjective optimization using hypergraphs. Technical Report CI-189/04, University of Dortmund, 2004. → pages 25
- [65] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computation*, 9(1):3–12, 2005. → pages 3
- [66] Y. Jin, M. Olhofer, and B. Senhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5): 481–494, 2002. → pages 3, 40
- [67] M. K. Karakasis, A. P. Giotis, and K. C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43(10-11):1149–1166, 2003. → pages 3

- [68] G. Kepler, H. Tran, and H. Banks. Reduced order model compensator control of species transport in a CVD reactor. *Optimal Control Application Methods*, 21(4): 143–160, 2000. → pages 74
- [69] J. D. Knowles and D. W. Corne. Approximating the non-dominated front using the Pareto archived evolutionary strategy. *Evolutionary computation*, 8(2):149–172, 2000. → pages 2
- [70] M. Kubat. Decision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998. → pages 34
- [71] A. Kumar, D. Sharma, K. Deb, and K. Sindhya. A hybrid multi-objective optimization procedure using PCX based NSGA-II and sequential quadratic programming. Technical Report 2007008, KanGAL, 2007. → pages 3
- [72] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. → pages 13
- [73] H. Langer. *Extended Evolutionary Algorithms for Multiobjective and Discrete Design Optimization of Structures*. PhD Dissertation, Technische Universität München, 2005. → pages 49
- [74] P. A. LeGresley and J. J. Alonso. Airfoil design optimization using reduced order models based on proper orthogonal decomposition. In *Fluids 2000 Conference and Exhibit*, 2000. → pages 74
- [75] P. A. LeGresley and J. J. Alonso. Investigation of non-linear projection for POD based reduced order models for aerodynamics. In *AIAA Paper 2001-0926, 39th Aerospace Sciences Meeting and Exhibit*, 2001. → pages 74
- [76] T. Lehnäuser and M. Schäfer. Efficient discretization of pressure-correction equations on non-orthogonal grids. *International Journal of Numerical Methods in Fluids*, 42(2): 211–231, 2003. → pages 8
- [77] T. Lehnäuser and M. Schäfer. Improved linear interpolation practice for finite-volume schemes on complex grids. *International Journal of Numerical Methods in Fluids*, 38 (7):625–645, 2002. → pages 8
- [78] K. H. Liang, X. Yang, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000. → pages 3
- [79] C. Lopez and E. Hernandez-Garcia. Low-dimensional dynamical system model for observed coherent structures in ocean satellite data. *Physica A: Statistical Mechanics and its Applications*, 328(1-2):233–250, 2003. → pages 74
- [80] H. Ly and H. Tran. Proper orthogonal decomposition for flow calculations and optimal control in a horizontal CVD reactor. *Quarterly of Applied Mathematics*, 60(4): 656–631, 2002. → pages 74

- [81] X. Ma and G. Karniadaks. A low-dimensional model for simulating three-dimensional cylinder flow. *Journal of Fluid Mechanics*, 458:181–190, 2002. → pages 74
- [82] R. A. E. Mäkinen, J. Périaux, and J. Toivanen. Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms. *International Journal for Numerical Methods in Fluids*, 30(2):149–159, 1999. → pages 3
- [83] 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, 2008. → pages 3
- [84] S. Menzel, M. Olhofer, and B. Sendhoff. Application of free form deformation techniques in evolutionary design optimisation. In *Proceedings of 6th World Congress on Structural and Multidisciplinary Optimization*, 2005. → pages 11
- [85] A. Messac and A. A. Mullur. A computationally efficient metamodeling approach for expensive multiobjective optimization. *Optimization and Engineering*, 9(1):37–67, 2007. → pages 3
- [86] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2(1):11–22, 1986. → pages 30
- [87] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kruwer, Boston, 1 edition, 1999. → pages 18, 20
- [88] F. Muyl, L. Dumas, and V. Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computer and Fluids*, 33(5-6):849–858, 2004. → pages 3
- [89] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2 edition, 2006. → pages 13
- [90] I. Ono and S. Kobayashi. A real-coded genetic algorithm for functional optimization using unimodal normal distribution crossover. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 246–253, 1997. → pages 2
- [91] S. Patankar and D. Spalding. A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows. *International Journal of Heat Mass Transfer*, 15(10):1787–1806, 1972. → pages 8
- [92] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247(4945):978–982, 1990. → pages 31
- [93] M. M. Raghuwanshi and O. G. Kakde. Survey on multiobjective evolutionary and real coded genetic algorithms. In *Proceedings of the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pages 150–161, 2004. → pages 2
- [94] I. Rechenberg. *Evolutionsstrategie. Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Friedrich Frohmann Verlag, 1 edition, 1973. → pages 2, 13

- [95] G. Rudolph. Convergence rate of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics Journal*, 26(3):375–390, 1997. → pages 2
- [96] J. A. Samareh. Geometry and grid/mesh generation issues for CFD and CSM shape optimization. *Optimization and Engineering*, 6(1):21–32, 2005. → pages 9
- [97] M. Schäfer. *Computational Engineering: Introduction to Numerical Methods*. Springer, Berlin, 1 edition, 2006. → pages 7
- [98] J. D. Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD Dissertation, Vanderbilt University, USA, 1984. → pages 2
- [99] J. R. Schott. *Fault Tolerant Design Using Single and Multi-Criteria Genetic Algorithms*. Master Thesis, Massachusetts Institute of Technology, 1995. → pages 49
- [100] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978. → pages 33
- [101] T. W. Sederberg and S. R. Parry. Free form deformation of solid geometric models. In *Proceedings of SIGGRAPH'86*, pages 151–159, 1986. → pages 9
- [102] D. Sharma, A. Kumar, K. Deb, and K. Sindhya. Hybridization of SBX based NSGA-II and sequential quadratic programming for solving multi-objective optimization problems. Technical Report 2007007, KanGAL, 2007. → pages 3
- [103] T. W. Simpson, J. D. Peplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: survey and recommendations. *Engineering with Computers*, 17(2):129–150, 2001. → pages 3
- [104] L. Sirovich. Turbulence and the dynamics of coherent structures. I - coherent structures. II - symmetries and transformations. III - dynamics and scaling. *Quarterly of Applied Mathematics*, 45:561–571, 573–590, 1987. → pages 74, 75
- [105] W. B. Song. *Shape Optimisation of Turbine Blade Firtrees*. PhD Dissertation, University of Southampton, UK, 2002. → pages 3
- [106] H. Stone. Iterative solutions of implicit approximations of multidimensional partial differential equations. *SIAM Journal on Numerical Analysis*, 5:530–558, 1968. → pages 8
- [107] Y. Tahara, S. Tohyama, and T. Katsui. CFD-based multi-objective optimization method for ship design. *International Journal for Numerical Methods in Fluids*, 52(5): 499–527, 2006. → pages 3
- [108] S. Tsutsui, M. Yamamura, and T. Higuchi. Multi-parent recombination with simplex crossover in real-coded genetic algorithms. In *Proceedings of Genetic and Evolutionary Computing Conference*, pages 657–664, 1999. → pages 2

- [109] P. Van der Lee, T. Terlaky, and T. Woudstra. A new approach to optimizing energy systems. *Computer Methods in Applied Mechanics and Engineering*, 190(40-41): 5297–5310, 2001. → pages 46
- [110] G. G. Wang, Z. Dong, and P. Aitschison. Adaptive response surface method - a global optimization scheme for approximation-based design problems. *Engineering Optimization*, 33(6):707–733, 2001. → pages 3
- [111] R. E. Wendell and D. N. Lee. Efficiency in multiple objective optimization problems. *Mathematical Programming*, 12(1):406–414, 1977. → pages 20
- [112] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 1 edition, 2005. → pages 30
- [113] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD Dissertation, ETH Zürich, 1999. → pages 41
- [114] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *Proceedings of Conference on Parallel Problem Solving from Nature (PPSN V)*, 1998. → pages 49
- [115] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. → pages 49
- [116] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(2):173–195, 2000. → pages 3, 22
- [117] E. Zitzler, M. Laumanns, and L. Thiele. SPEA 2: Improving the strength Pareto evolutionary algorithm. Technical Report TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2001. → pages 2
- [118] E. Zitzler, K. Deb, and L. Thiele. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, 2002. → pages 22

Lebenslauf

Persönliche Daten

Name: Hongtao Sun
Geboren am: 02. Dezember 1978
Familienstand: verheiratet
Anschrift: Rugierstraße 8, D – 65929 Frankfurt am Main

Schulbildung

1985 – 1990
Grundschule
Die 1. Schule Xinghua der Stadt Shenyang, VR China

1990 – 1993
Unterstufe der allgemeinen Mittelschule
Die 88. Mittelschule der Stadt Shenyang, VR China

1993 – 1996
Oberstufe der allgemeinen Mittelschule
Die 4. Mittelschule der Stadt Shenyang, VR China

Studium

1996 – 2001
Hauptstudienfach: Bauingenieurwesen
Nebensstudienfach: Informatik
Abschluss: Bachelor of Engineer
Tongji Universität, Shanghai, VR China

2001 – 2004
Studienfach: Computational Science in Engineering
Abschluss: Master of Science
Technische Universität Braunschweig

Berufstätigkeit

seit 2005
wissenschaftliche Mitarbeiterin
Fachgebiet für Numerische Berechnungsverfahren im Maschinenbau
Technische Universität Darmstadt