U N I V E R S I T Y OF S P L I T
FACULTY OF ELECTRICAL ENGINEERING, MECHANICAL ENGINEERING
AND NAVAL ARCHITECTURE

S V E U Č I L I Š T E U S P L I T U
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Ivo Marinić-Kragić

EFFICIENT PARAMETERIZATION METHODS FOR GENERIC SHAPE OPTIMIZATION WITH ENGINEERING APPLICATIONS

METODE UČINKOVITE PARAMETRIZACIJE ZA GENERIČKO OPTIMIRANJE OBLIKA S INŽENJERSKIM PRIMJENAMA

DOCTORAL QUALIFYING EXAM
DOKTORSKI KVALIFIKACIJSKI ISPIT

Split, 2016.

Doctoral qualifying exam was prepared at Department of mechanical engineering and naval architecture,
Faculty of electrical engineering, mechanical engineering and naval architecture

Mentor: dr.sc. Damir Vučina, red. prof.

Kvalifikacijski doktorski ispit je izrađen na Zavodu za strojarstvo i brodogranju,
Fakulteta elektrotehnike, strojarstva i brodogradnje u Splitu

Mentor: dr.sc. Damir Vučina, red. prof.

EFFICIENT PARAMETERIZATION METHODS FOR GENERIC SHAPE OPTIMIZATION WITH ENGINEERING APPLICATIONS

Abstract

Engineering optimization very often includes complex and computationally expensive numerical simulations with time requirements ranging from few seconds to months per single simulation, depending on the simulated physics and available computational resources. Such optimization tasks should therefore contain no more than a small yet sufficient number of optimization variables. Additional difficulties arise when the underlying physics has nonlinear character modeled by nonlinear partial differential equations. The most common cases of engineering optimization involving computationally very expensive nonlinear simulation include problems in computational fluid dynamics. Furthermore, technical objects considered in fluid dynamics (ship hulls, wind-turbines or fan blades,…) are almost always complex three-dimensional shapes that cannot adequately be described with only a few shape variables.

The most commonly used tools for generic shape representation are non-uniform rational basis spline (NURBS) curves and surfaces. NURBS surfaces can be used as mutually connected patches when the shape is too complex to be represented by a single NURBS surface. In recent years, a generalization of NURBS called T-splines is becoming increasingly more used. The objective of this doctoral qualifying exam is to review these and other currently available methods of generic shape parameterization and evaluate possibilities of integration with engineering optimization tasks related to computational fluid dynamic.

METODE UČINKOVITE PARAMETRIZACIJE ZA GENERIČKO OPTIMIRANJE OBLIKA S INŽENJERSKIM PRIMJENAMA

Kratki sažetak

Inženjerska optimizacija često sadržava složene i računalno skupe numeričke simulacije s računalnim vremenskim zahtjevima od nekoliko sekundi do nekoliko mjeseci po simulaciji, ovisno o simuliranoj fizici i dostupnim računalnim resursima. Ovakvi optimizacijski zadaci stoga ne mogu sadržavati nego mali broj optimizacijskih varijabli. Dodatne teškoće nastaju kada ju problem opisan nelinearnim parcijalnim diferencijalnim jednadžbma. Primjer inženjerske optimizacije s računski zahtjevnim simulacijama i nelinearnim modelima vrlo često uključuju računalnu dinamiku fluida. Uz sve navedeno, tehnički objekti u dinamici fluida (trup broda, lopatice vjetro-turbine ili ventilatora,…) često su kompleksni trodimenzionalni oblici koji se ne mogu adekvatno opisati s malim brojem varijabli oblika.

Najčešće korišteni alati za zapis generičkih krivulja i ploha su ne-uniformni racionalni bazni spline (NURBS). Plohe se mogu zapisati pomoću više NURBS ploha spojenih po dijelovima kada je oblik presložen da bi se opisao s pojedinačnom NURBS plohom, a posljednjih godina se u te svrhe koristi i generalizacija NURBSa takozvani T-spline. Cilj ovog kvalifikacijskog doktorskog ispita je napraviti pregled metoda generičke parametrizacije oblika u sklopu njihove integracije u probleme inženjerske optimizacije na primjeru računalne dinamike fluida.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVATIONS

CFD – computational fluid dynamics
CV – control volume
IGA – isogeometric analysis
PDE – partial differential equation
SO – shape optimization

1. INTRODUCTION

Shape optimization (SO) is a rather complex undertaking which involves many challenges but nevertheless becomes a necessity in several industries [1]. SO is a part of computational mechanics and can be subdivided in three branches: sizing optimization (for example thickness distribution), shape optimization itself and topology optimization [2]. This qualifying doctoral exam will focus on engineering application of shape optimization itself, but variable shape topology will also be considered.

In SO, the system subject to optimization is usually described by partial differential equations (PDEs). In recent years, the field of optimization of systems based on PDEs has received a large impulse with a variety of research projects being funded by national and international agencies [3], [4]. In engineering application of SO, the optimization problems usually contain multiple mutually exclusive objectives functions, modeled by very different computational methods. This causes difficulties in engineering applications of methods developed specially for solving a specific type of PDEs. This doctoral qualifying exam gives a review of methods required for generic engineering SO of complex engineering systems.

A common example of complex engineering optimization tasks includes computational fluid dynamics (CFD), a system modeled by (nonlinear) PDEs. Concerning CFD, the first applications of optimization are found for aeronautical problems, in particular to improve wing profile and flight properties (typically, reduce drag) [5]. CFD models are a major concern as appropriate turbulence models and domain discretization exhibit a major influence on the simulation results and require substantial computing resources. These tasks also include complex 3D shapes such as ship hulls, wind-turbine blades, fan vanes, etc. Computational modeling of the respective geometry is correspondingly difficult as modeling of both global and local variations of shape is required. Furthermore, only a modest-size data-set of shape parameters has to be used, since otherwise the dimensionality of the subsequent optimization space is very high. The most often solution to solving these problems is to apply local SO while keeping most of the 3D shape fixed i.e. ignoring the global character. In any case, an integrated optimization workflow must be constructed to contain (at least) a geometry modeler and an engineering simulation node controlled by the optimizer. A typical engineering optimization workflow is illustrated in Figure 1.

This doctoral qualifying exam gives a review of methods that can increase the efficiency of engineering SO problems. Particularly, consideration is given to the application of CFD within an integrated optimization workflow since these tasks present are the most computationally demanding, consequently there exists a large potential of improvement exists. Methods for both global and local variations of shape will be investigated. First a layout of various parameterization methods and an overview with comparison between the methods is given. Parametric shape fitting is considered important for testing of parameterizations on existing shapes (point-cloud) as this way a parameterization can be tested geometrically without including computationally expensive numerical simulation. For multi-patch

parameterization methods, feature detection methods are applied to 3D point cloud to improve fitting performance. The major part of the doctoral qualifying exam is concerned with geometry modeling since much research is currently directed towards geometry modeling methods and universal approach does not exist. After all important aspects of geometry modeling are considered a layout of CFD is given. It is important to understand the physical principles and engineering models used in the engineering simulation node of the workflow. Without an efficient numerical simulation, and parameterization method (no matter how well selected) will not be able to show its full potential. The optimizer controlling the shape variations within the numerical workflow is also a major concern. Depending on optimization problem, a different optimization algorithm (genetic algorithm, gradient method,…) ,pre appropriate. For example a gradient method will not be able to not improve the solution in the case of a "noisy" objective function. An opposing case would be the case of local optimization with a smooth objective function where a genetic algorithm will eventually obtain a solution but the application of a gradient method can save orders of magnitude of computational time.

Numerical methods in combination with powerful computers have enabled the designers to create various design tools based on optimization algorithms. Numerical integration of the above elements in the form of a workflow is not trivial as it needs to encapsulate both process flows with synchronization of processes and the necessary data-mining. The data transfers within the workflow include results for all candidate designs. On the input side of the engineering simulations (CFD,…), the current shapes of the candidate designs need to be communicated to the simulators ('data-burying'). As the engineering object generally functions across a range of operating regimes, a good definition of the excellence criteria is also crucial. Moreover, the design variables may include additional parameters beyond those which control the shape, including discrete variables. Some of the variables can be given only as statistical making the problem even more difficult. A schematic of a general workflow s is illustrated, Figure 1. Corresponding data mining is represented with vertical direction while the coordination of process is represented by horizontal direction.

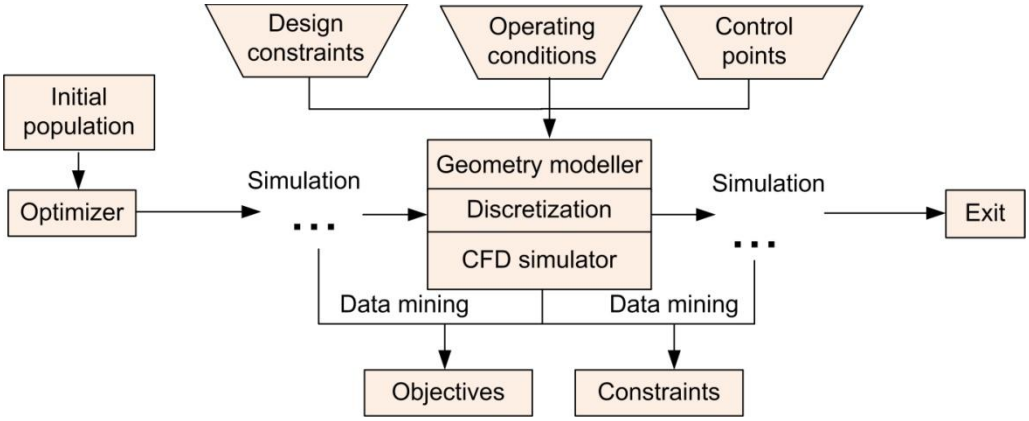


Figure 1. Typical multidisciplinary shape optimization workflow with CFD simulator.

A generic engineering SO task has the following elements:

1. For an engineering object and corresponding operating conditions the objectives and constraints have to be defined;
2. Numerical simulations are conducted (and developed) to confirm that the computer model is able to replicate reality for the required operating conditions. Typically multiple test cases are conducted and compared with experimental results;
3. Fine tuning of the numerical simulation to improve the computational efficiency and keep the required accuracy;
4. Real-life solutions of a similar problem already exist, but they are not optimal for a particular optimization objective. The shape of the pre-existing solutions can be used as a starting point of SO task. If the geometry file is not available, 3D optical scanning can be used to provide the point cloud representing the initial shape;
5. Parameterization of the 3D point cloud into computational geometry entities (NURBS, T-spline,…) provides for optimization parameters. Fitting procedures can be used to evaluate the performance of parameterization and in this way the most appropriate shape parameterization procedure can be selected;
6. Definition of the excellence criteria and objective functions for the optimizer such that the fitness functions for the candidate designs can be evaluated. In the case of multi-regime operation, numerical sample of operating regimes has to be generated in order to evaluate excellence;
7. Definition of the optimization constraints. Common example is setting bounds for the selected parameterization control point mobility. Further constraints can be related to non-shape variables (mass, stress, temperature, revolution velocity, etc.).
8. Initial solution or a set of initial solutions based on the selected parameterization is generated;
9. Launching the optimizer with corresponding operators and parameter values. Most often a genetic algorithms- based optimizer is selected. This requires a setup of selection operators, cross-over operators and probabilities, mutation, elitism, fitness scaling, etc;
10. Linking an array of engineering simulators to provide values of excellence and constraints for all candidate designs represented by corresponding shape parameterization.
11. Iteratively shape-optimizing the numerical object within the numerical cycle embedding the optimizer, shape modeler and engineering simulators (CFD, etc.).

Elements of the engineering SO tasks (3), (5) and (9) have the most potential to be improved and a review of the existing solutions was given in this doctoral qualifying exam. The focus was mainly put on step (5) since new emerging shape parameterization and fitting procedures are not yet sufficiently tested in the context of generic shape engineering optimization hence leaving room for improvement.

2. NUMERICAL OPTIMIZATION

All of the components of the optimization task can be integrated in a single numerical workflow controlled by the optimizer based on the selected optimization methods. After the initial solution is selected (based on earlier designs or randomly generated), the direction of the shape modification is controlled by the selected optimization algorithm. The result of the engineering optimization task is of course primarily dependent on the selected optimization method. This chapter will briefly describe important optimization methods and when is it appropriate to apply one or the other since no single superior approach exists. Further improvements that could improve the convergence speed of numerical optimization are also briefly described, for example a surrogate model can be introduced in the optimization workflow.

Engineering SO problems usually belong to class of continuous nonlinear multi-objective optimization (MOO) although discrete variables can be present. Methods for solving continuous MOO problems can be roughly divided in methods relying on standard optimization engines (single-objective optimization methods) such as gradient methods, and other approaches such as genetic algorithms [6]. When discrete variables are present in a MOO problem, genetic algorithms represent an appropriate and robust but computationally more demanding method. Since this doctoral qualifying exam will focus mainly on shape parameterization methods, this chapter will conduct a basic review of optimization methods.

2.1. Classical single-objective optimization

In classical optimization, the problem with which this review is concerned is that of determining the values of a set of parameters $x_1, x_2, \ldots x_n$, called the independent variables of the problem, which correspond to the minimum of a given objective function $f(\mathbf{x})$. This problem can be written as:

$$\underset{\mathbf{x}\in\Re^n}{\text{Minimize}}\ f(\mathbf{x})$$
$$\text{Subject to}\begin{Bmatrix} g_j(\mathbf{x}) \le 0, & j = 1,2,\ldots m \\ h_l(\mathbf{x}) = 0, & l = 1,2,\ldots,e \end{Bmatrix} \tag{1}$$

where m is the number of inequality constraints, and e is the number of equality constraints. $\mathbf{x} \in \mathbb{R}^n$ is a vector of design variables, where n is the number of independent variables x_i. The point $\mathbf{x}_i^*$ minimizes the objective function $f(\mathbf{x})$.

2.1.1. Unconstrained optimization

Many of the methods used for constrained optimization deal with the constraints by converting the problem in some way into an unconstrained one. In the classical approach, analytically, a stationary point of a function $f(\mathbf{x})$ is defined to be the one where all of the first partial derivatives of the function with respect to the independent variables are zero:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 0, \qquad i = 1, 2, ..., n \tag{2}$$

Hence the problem could be solved by differentiating the objective function with respect to each of the variables in turn and setting them to zero, which would yield *n* equations and *n* unknowns to be solved for the stationary points. Since this is usually not possible, various numerical optimization techniques were developed as described in chapter 2.3.

2.1.2. Constrained optimization

The classical method of solving the constrained optimization problem uses Lagrangian multipliers to convert the problem into an unconstrained one. To incorporate the constraints in one equation, inequality constraints are first converted to equality. The Lagrange function $\mathcal{L}$ can be written as:

$$\mathcal{L}(\mathbf{x}, \lambda_1, \lambda_2, ..., \lambda_m) = f(\mathbf{x}) - \sum_{k=1}^{m} \lambda_k g_k(\mathbf{x}) \tag{3}$$

where $\lambda_1, \lambda_2, ..., \lambda_m$ are the Lagrange multipliers. The optimal solution is the stationary point (saddle-point) of the Lagrange function:

$$\nabla \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{4}$$

The problem with Lagrange method is that finding a saddle-point can be even more difficult to solve than the original constrained problem. So other methods (such as penalty method) are usually used for practical optimization problems when constrained optimization is converted to unconstrained.

2.2. Multi-objective optimization

Most engineering optimization problems involve multiple excellence criteria. Single-objective optimization is a special case with number of objective functions equal to one. By generalization of (1), a general multi-objective problem can be posed:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{Minimize}}\ \mathbf{F}(\mathbf{x}) = \left[F_1(\mathbf{x}),\ F_2(\mathbf{x}),\ ..., F_k(\mathbf{x})\right]^T \\ &\text{Subject to} \begin{Bmatrix} g_j(\mathbf{x}) \le 0, & j = 1, 2, ...m \\ h_l(\mathbf{x}) = 0, & l = 1, 2, ..., e \end{Bmatrix} \end{aligned} \tag{5}$$

where *k* is the number of objective functions, $\mathbf{F}(\mathbf{x}) \in \mathbb{R}^k$ is a vector of objective functions $F_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^1$. $F_i(\mathbf{x})$ are the objective functions, whereby individual function is a mapping from the design space to single objective space. The gradient of a function $F_i(\mathbf{x})$ with respect to $\mathbf{x}$ is written as $\nabla_{\mathbf{x}} F_i(\mathbf{x}) \in \mathbb{R}^n$. The point $\mathbf{x}_i$ * minimizes the objective function $F_i(\mathbf{x})$. The feasible design space $\mathbf{X}$ is defined as the set $\{\mathbf{x} \mid g_i(\mathbf{x}) \le 0, i = 1, 2, ...m;$ and $h_i(\mathbf{x}) = 0,\ i = 1, 2, ..., e\ \}$. Each point in the design space maps to a point in criterion space, but every point in the criterion space does not necessarily correspond to a single point in design space.

For single objective functions, k=1 and the minimum of $F_1(\mathbf{x}_i^*)$ where $\mathbf{x}_i^*$ is subset of $\mathbf{X}$ and the solution of the optimization problem. In comparison to single-objective optimization, with multi-objective problems solution cannot be defined uniquely. The optimal solution can be regarded more as a concept than as a definition. The principal concept in defining an optimal point is that of Pareto optimality, which is defined as follows:

A point, $\mathbf{x}^* \in \mathbf{X}$, is Pareto optimal if there does not exist another point, $\mathbf{x} \in \mathbf{X}$, such that $\mathbf{F}(\mathbf{x}) \leq \mathbf{F}(\mathbf{x}^*)$ and $F_i(\mathbf{x}) < F_i(\mathbf{x}^*)$ for at least one function.

In case of maximization problems, the Pareto optimal designs are illustrated in the objective space in Figure 2. In this case, the objective is usually multiplied by minus one so that the earlier definitions do not need to be modified.

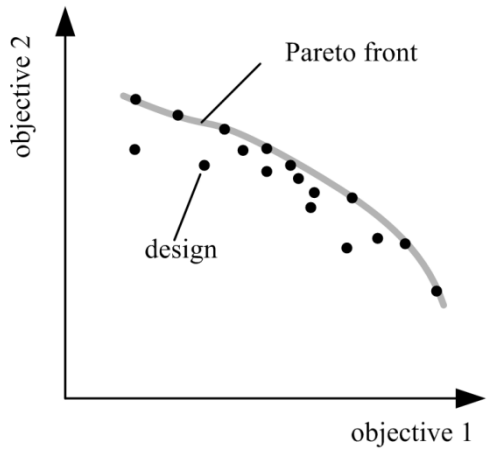


Figure 2. Pareto front.

2.3. Gradient and direct search methods

Gradient based methods are efficient methods for continuous nonlinear single-objective problems; they require both the optimization function and its derivative. To extend them to multi-objective optimization several approached exist, but all require a model of decisionmaker's preferences (ordering or relative importance of objectives and goals) [6]. Gradient methods can also be used for final fine-optimization after the relatively slow but robust genetic algorithm reaches a near-optimal solution. Gradient methods can also be used for efficient shape fitting in order to test the shape parameterization on existing designs.

Since the derivative of the function is usually not available, it can be numerically evaluated or alternatively direct search (value-based) methods could be used. In the direct search methods, only values of the function to be minimized are used. If the solution is constrained, i.e. if certain limits of values (or relations between the values) of the parameters must be obeyed, the problem is more difficult. One approach is representing the amount of constraint violation by a penalty function and adding it to original function. For equality constraints, the penalty function is usually selected as $PF_l = c \cdot h_l(\mathbf{x})^2$ while for inequality

constraints $PF_j = c \cdot \max(0, g_j(\mathbf{x}))^2$ where c is arbitrarily selected constant. Now we need to solve the minimum of the expanded objective function $F'(\mathbf{x})$:

$$F'(\mathbf{x}) = F(\mathbf{x}) + \sum_l c \cdot h_l(\mathbf{x})^2 + \sum_j \max(0, g_j(\mathbf{x}))^2 \tag{6}$$

Both the gradient based and the direct search methods are iterative and starting from an initial approximation $\mathbf{x}^0$ to the minimum they proceed by defining a sequence of points $\{\mathbf{x}^i\}$, i= 1,2, . .., in such a way that:

$$F(\mathbf{x}^{i+1}) < F(\mathbf{x}^i) \tag{7}$$

This series of improved approximations $\{\mathbf{x}^i\}$ may be considered to be generated by the general iterative equation

$$\mathbf{x}^{i+1} = \mathbf{x}^i + h^i \mathbf{d}^i \tag{8}$$

where h^i is a positive constant and $\mathbf{d}^i$ is an n-dimensional direction vector evaluated at the ith iteration. The vector $\mathbf{d}^i$ determines the direction to be taken from the ith point $\mathbf{x}^i$ and the magnitude of h^i $\mathbf{d}^i$ determines the size of the step in that direction. There are many methods in the literature for determining the vector $\mathbf{d}^i$ and they can be divided into two natural classifications - direct search methods and gradient methods. Direct search methods use only the values of the objective function. On the other hand, gradient methods in addition to function values require values of the first and (for some algorithms) second order partial derivatives of the function. Few methods will be summarized here.

The first intuitive attempt of linear direct, value based search procedure consist of subsequently minimizing along each coordinate direction. This optimization method is known as the alternating variable method. Starting from the given guess, moving parallel to each axis in turn and changing direction when a minimum in the direction being searched is reached is illustrated in Figure 3a.

Obviously this method not efficient, but by a simple modification a practical method could be obtained – Hooke-Jeeves method. Alternating changing of directions is implemented as the first step of the method, but now only one discrete move Δx is conducted for each positive and/or negative direction, and the new point is kept if smaller function value is obtained. This is called the exploratory move. The method consists of a combination of exploratory moves and pattern moves: the former seek to locate the direction of any valleys in the surface and the latter attempts to progress down any such valleys. This procedure illustrated in Figure 3b continues until all n variables in exploratory move cannot find a better solution. The procedure can than be continued with smaller Δx until Δx is larger than some pre-set tolerance value.

A frequently used robust optimization method belongs to a group of simplex methods, of which Nelder-Mead and regular simplex method are widely used. A regular simplex in *n* dimensions is *n*+1 mutually equidistant point. In two dimensional plane the simplex is an equilateral triangle that is, its vertices. Useful property of regular simplexes is that a new simplex can be easily set up on any face of a given simplex by the addition of only one new point. This makes a basis for the Simplex Method of optimization, illustrated in Figure 3c. The search begins by setting up an arbitrarily regular simplex. The vertex corresponding to the greatest function value is then replaced by its reflection in the hyperplane of the remaining points, forming a new simplex. The function is evaluated at the new vertex and the process continues. Exception must be made if the vertex with the greatest function value is the one most recently introduced. This would cause oscillation that can be avoided by simply using second largest function value instead of the largest.

Three earlier methods were value-based methods which are robust but computational inefficient. A more efficient method can be obtained by computing gradients to aid us in search for optimal solution. Gradient methods h use the values of the partial derivatives of the function with respect to free variables in addition to the values of the function itself. Simplest gradient based method is the steepest descent method, At any given point, the search direction is the direction whose components are proportional to the first partial derivatives of the function, Method is illustrated in Figure 3d. Many variations of this method direction have subsequently been proposed [7].

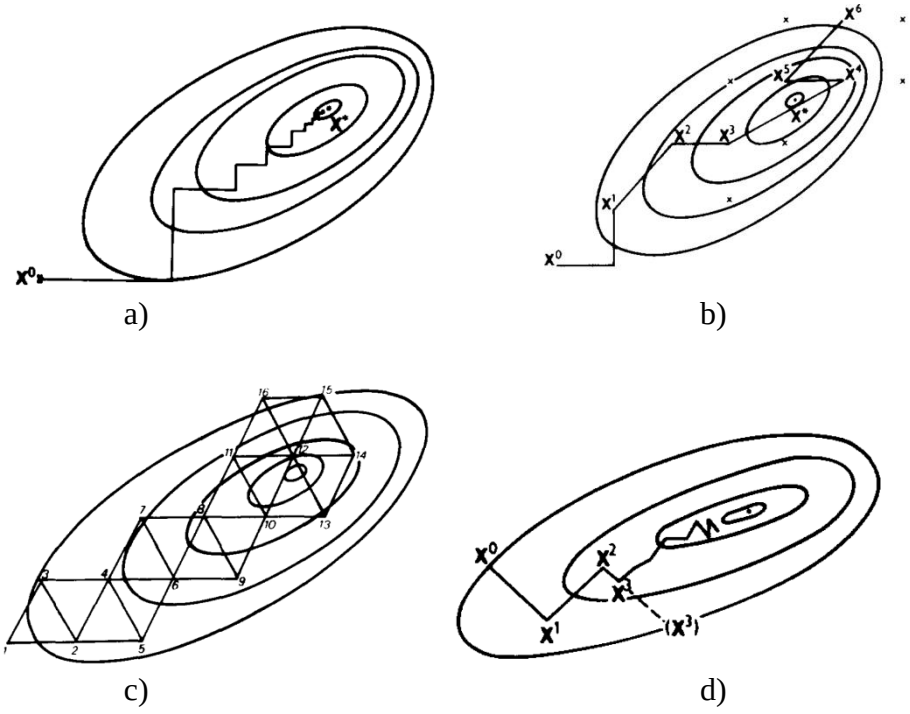


Figure 3 Illustration of iterative optimization procedures: a) Alternating Variable Method b) Hooke- Jeeves c) Simplex method and d) steepest decent method, [7].

2.4. Genetic algorithms

Several modern heuristic tools exist for solving optimization problems that are difficult (or even imposible) to solve using classical methods. These tools include simulated annealing, tabu search, particle swarm, evolutionary computation, etc. These techniques are finding popularity within research community as design tools and problem solvers because of their flexibility and ability to optimize in complex highly non-linear search spaces with multiple local minima. GA can be viewed as a general-purpose search method. It is based loosely on Darwinian principles of biological evolution, reproduction and "the survival of the fittest". GA maintains a set of candidate solutions called the population and repeatedly modifies them by application of genetic operators.

Initial population is usually generated randomly and its size is selected based on experience on similar optimization problems. From the initial (and subsequent) generations, next generation is generated in several steps by application of selection, reproduction and mutation operators. In first step, selection operator creates temporary clones of the selected individuals. By preferring more fit individuals the selection operators implements "the survival of the fittest" principle. Children are obtained by combining features of randomly selected clones, from two parents two children are created by reproduction operator. Third step is applying small amount of random modifications to children i.e. mutation. In some cases it is desirable to keep selected individuals so they are cloned without modifications. Over successive generations, the population evolves toward the optimal solution since the individuals with lower function values have higher fitness value and are more likely to be selected for reproduction (survival of the fittest). The GA is well suited to and has been widely applied to solve complex engineering optimization problems. GA can handle both continuous and discrete variables, nonlinear objective and constraint functions without requiring gradient information. Genetic algorithms are global optimization techniques, which means that they generally converge to the global solution rather than to a local solution. However, this distinction becomes unclear when working with multi-objective optimization where Pareto optimal solutions are obtained. The defining feature of GA multiobjective optimization methods is that not just local but global Pareto solutions are determined by the procedure [6]. The flowchart of genetic algorithm is illustrated in Figure 4.

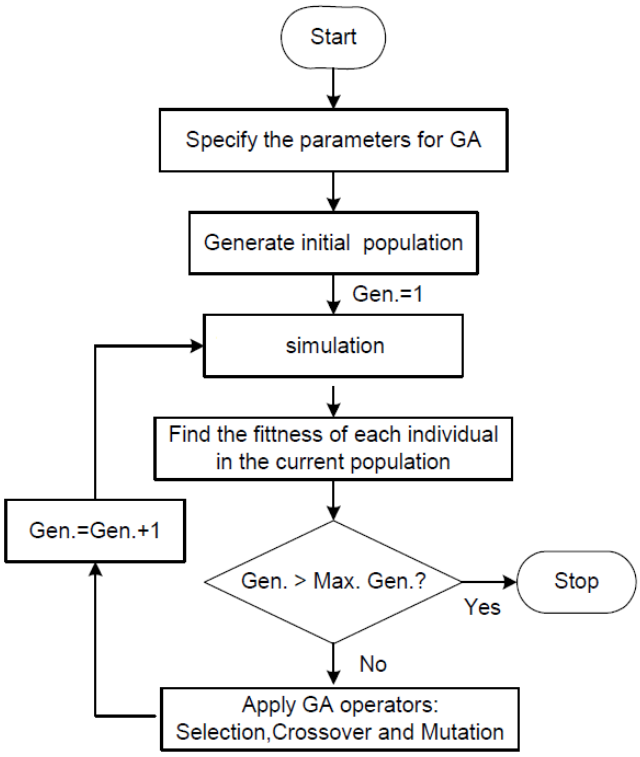


Figure 4. Flowchart of genetic algorithm [8].

2.4.1. Variable chromosome

In optimization which involves simultaneous topology and shape optimization, various approaches exist. An often applied method is the level-set method [9] when applied, the same parameterization can be used while allowing the topology variations. In the cases when the level-set method is not applied, a different approach to topology optimization is required. One of the approaches could by the application of variable length chromosomes [10]. For such problems, the number of variables for which a solution is searched does not need to be known in advance. This type of variable length chromosomes could contain information of multiple set of objects, their shapes, locations, etc. The chromosomes would grow in size adding more genes if more objects are added during the optimization procedure. Figure below shows examples of structure (phenotype) and chromosomes of different length (genotype) for two hypothetical designs. In [10], the changes of the chromosome length are implementing from coarse to fine. After the near optimal solution is found by small number of variables (short chromosome), the number of variables is increased (longer chromosome) for the whole population and the procedure continues to the next stage.

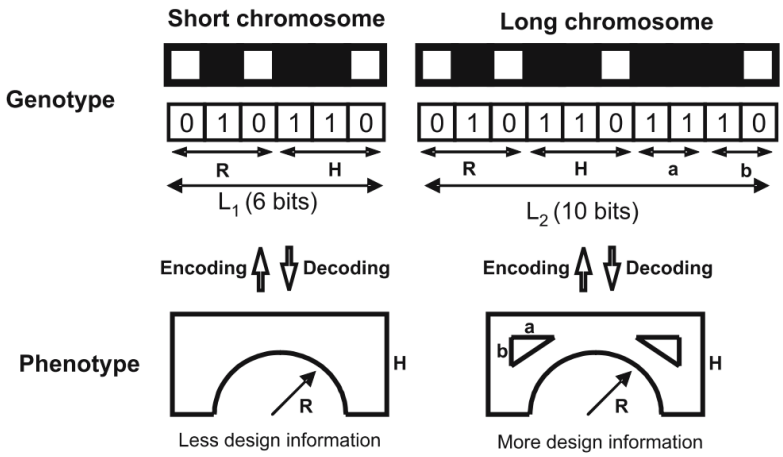


Figure 5. Examples of designs in the phenotype and corresponding chromosomes in the genotype domain[10].

Various problems and solutions exist and a good review of the methods is given in [11]. Genetic algorithm with variable length chromosome could be applied for multi-patch parameterization since the topology of the optimal solution is not known in advance.

2.5. Sensitivity analysis

Design sensitivity analysis plays an important role in several areas including numerical optimization. Sensitivity information is also valuable on its own for estimation of robustness of the obtained solution with respect to the optimization variables. It is possible to determine how the value of a given response function (optimization objective), changes with respect to a given change in the model parameters. For small perturbations $|\Delta x|$, this information is obtained through a first-order Taylor series expansion. If the response function is the stress at a point of a structural system, aerodynamic performance measure, or some other quantity evaluated through a computationally expensive numerical analysis, then it can be predicted how the response function value varies for small perturbations in the model parameters without performing a re-analysis. Much work has been performed in the field of design sensitivity analysis. Most notably, perhaps, is the work in structural mechanics with applications to structural optimization [12].

The simplest method to obtain sensitivity information is the finite difference method. In the finite difference method, a simple Taylor series expansion is used to approximate to the derivative:

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \sum_i \frac{\partial f(\mathbf{x})}{\partial x_i}\Delta x_i + o(\Delta x_i^{\,2}) \tag{9}$$

where the derivatives are approximated with finite differences for every design variable independently. This method thus suffers from computational inefficiency and possible errors. More often used is the adjoint method whose basics are explained briefly in continuation. The adjoint method originates in the theory of Lagrange multipliers in optimization [15].

As an example, consider the minimization of a function J from $\Re^n$ into $\Re^1$ (single-objective), under the equality constraints h from $\Re^n$ to $\Re^m$.

$$\text{Minimize } J(\mathbf{x}) \text{ such that } h(\mathbf{x}) = 0 \tag{10}$$

Introducing the Lagrange multilplier λ in $\Re^m$ and the Lagrangian $\mathcal{L}$:

$$\mathcal{L}(\mathbf{x}, \lambda_1, \lambda_2, ..., \lambda_m) = J(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k h_k(\mathbf{x})$$

the optimality condition for (10) (under some qualification conditions) is the stationarity of the Lagragian, namely

$$\nabla \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0$$

The adjoint method is an extension of this approach in the framework of optimal control theory. In this context, the variable **x** is the union of a state variable **y** and a control (shape) variable **u**, while the constraint $h(\mathbf{y},\mathbf{u}) = 0$ is the state equation (governing equations) which gives **y** in terms of **u**. In such a case the Lagrange multiplier λ is called the adjoint state. Denote by **u** in $\mathbb{R}^k$ a control variable. The state of the system is denoted by **y** in $\mathbb{R}^n$ and is defined as (for finite dimensional case) the solution of the following state equation:

$$\boldsymbol{A}(\mathbf{u})\mathbf{y} = \mathbf{b} \tag{11}$$

where **b** is a given right hand side in $\mathbb{R}^n$ and $\boldsymbol{A}(\mathbf{u})$ is an invertible *n*x*n* matrix. Since the matrix depends on **u**, so does the solution **y**. The goal is to minimize, over all admissible controls **u**, an objective function $J(\mathbf{u},\mathbf{y})$ under the constraint (11). The difficulty of the problem is that **y** depends nonlinearly on **u**. We assume here that **u** is a continuous variable.

To numerically minimize the objective function *J*, the most efficient algorithms are those based on derivative informations. Therefore, a key issue is to compute the gradient of $J(\mathbf{u},\mathbf{y}(\mathbf{u}))$. Several method exist for computing the gradient of the objective function $J(\mathbf{u},\mathbf{y}(\mathbf{u}))$ There are at least two ways for introducing an adjoint in a computer code. Either, a so-called analytic adjoint, is implemented. Or a program for the adjoint is obtained by automatic differentiation of the code solving the state equation (11). The drawback of the whole adjoint approach it requires the development of an adjoint solver (no general purpose adjoint solver exist for CFD). This can be a labor intensive task which requires good knowledge of the implemented numerical tools [15].

Optimization with aid of the solution to the adjoint system of equations is an active area of research with applications in both structural optimization and computational fluid dynamics, particularly for aeronautical applications [13]. In fluid dynamics, the first use of adjoint equations for design was by Pironneau [14] for minimization of drag in stokes flow. Application of the adjoint approach is popular in the optimal design of structures for topology optimization problems. In those cases, the number of optimization variables describing the system is so large that it is the only viable approach. In the context of topology optimization for mechanical structures, the number of design variables is even larger since any cell of a "hold-all" computational domain is potentially a design variable: either it is full of material or

empty. Example of solution from [16] is illustrated in Figure 6, where the shape derivative is computed by an adjoint method and level-set method is used for front propagation.

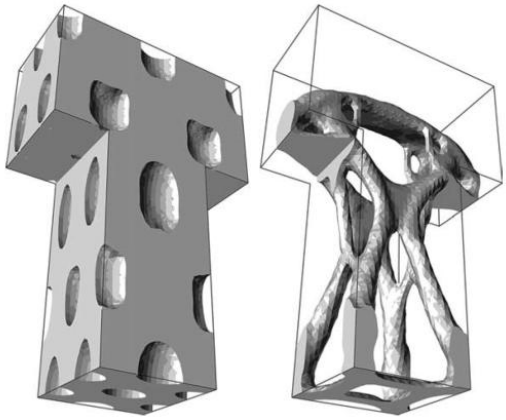


Figure 6. Initial guess and optimal shape for the three-dimensional electrical mast [16].

The adjoint approach is only helpful in the context of gradient-based optimization and such optimization has its own limitations. Firstly, it is only appropriate when the design variables are continuous. For design variables which can take only integer values (e.g. the number of vane of the fan, or number of blades of wind turbine) stochastic procedures such as GA are usually implemented since gradient does not exist in those cases. Secondly, if the objective function contains multiple minima, the gradient approach will generally converge only to the nearest local minimum. This is usually solved by multi-start from various locations in search space if one wants to find the global minimum using gradient methods. Still. if the objective function is known to have multiple local minima, and possibly discontinuities, a stochastic search method such as GA may be more appropriate [13].

2.6. Surrogate models

Engineering simulations such as CFD tend to be computationally very expensive. The idea to improve the optimization speed is to partialy use an approximation of the simulation result with computationally efficient methods during the optimization run.

Surrogate-based optimization represents a class of optimization methodologies that make use of surrogate modeling techniques to quickly find the local or global optima. It provides an optimization framework in which the conventional optimization algorithms, e.g. gradient-based or evolutionary algorithms are used for sub-optimization(s). Surrogate modeling techniques are of special interest when using computationally expensive CFD simulations. They can be used to greatly improve the design efficiency and be very helpful in finding global optima, filtering numerical noise, realizing parallel design optimization and integrating simulation codes of different disciplines into a process chain. The term “surrogate model” has the same meaning as “response surface model”, “metamodel”, “approximation model”, “emulator” etc. For optimization problems, surrogate models can be regarded as approximation models for the cost function (s) and state function (s), which are built from sampled data obtained by randomly probing the design space (called sampling via Design of

Experiment (DoE)). Once the surrogate models are built, an optimization algorithm such as Genetic Algorithms (GA) can be used to search the new design (based on the surrogate models) that is most likely to be the optimum. Since the prediction with a surrogate model is generally much more efficient than that with a numerical analysis code, the computational cost associated with the search based on the surrogate models is generally negligible. Surrogate modeling is referred to as a technique that makes use of the sampled data (observed by running the computer code) to build surrogate models, which are sufficient to predict the output of an expensive computer code at untried points in the design space. Thus, how to choose sample points, how to build surrogate models, and how to evaluate the accuracy of surrogate models are key issues for surrogate modeling [17].

2.7. Example of optimization workflow

The overall numerical optimization problem with all included components is usually integrated in an optimization workflow. An example of a workflow is presented here. This is important since the workflow represents the final product of integrating all of the required engineering simulation, parameterization methods and optimization algorithms.

Numerical workflow developed in [18] will be demonstrated as an example. The developed workflow is capable of fully generic 3D SO of a centrifugal roof fan vane by manipulating the control points of parametric surfaces as illustrated in Figure 7. This workflow is an upgrade version of workflow developed in [19] for 2D optimization.

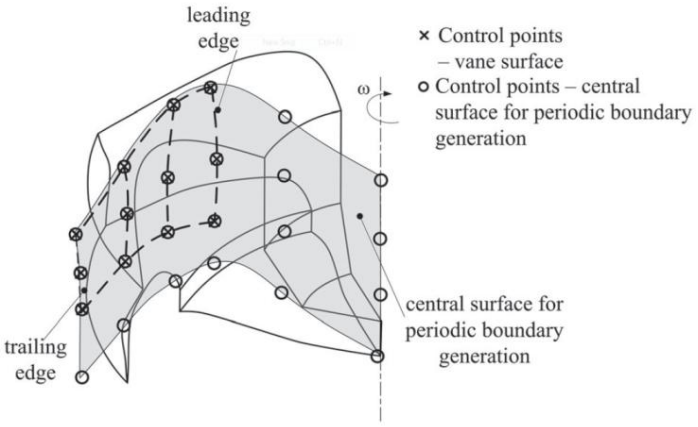


Figure 7. Shape parameterization using generic B-spline surface, used for both vane and domain parameterization [18].

Additional variables used are rotation speed and the discrete number of vanes. The excellence formulation is based on design flow efficiency, multi-regime operational conditions and noise criteria for various cases including multi-objective optimization. Multiple cases of optimization demonstrate the ability of customized and individualized fan design for the specific working environment according to the selected excellence criteria. Noise analysis is also considered in an multi-objective optimization workflow as an additional decision making tool. The workflow for the multi-regime operational conditions is illustrated in Figure 8. In the figure, the horizontal direction represents the process flow while the vertical direction

represents simultaneous data flow. Corresponding data mining and coordination of process flow were implemented using commercial software modeFRONTIER [20]. The computational workflow needed to implement the procedure encapsulates geometric modeling (computer-aided design, CAD), simulation software (computational fluid dynamics, CFD) and calculators (statistical mean calculation) coupled with the evolutionary numerical optimization. The numerical coupling needs to include both the process executions and their mutual synchronization as well as data flows between the individual applications. The workflow is in this example controlled by MOGA-II, a version of multi-objective genetic algorithm. Detailed description of input and output quantities is available in [18]. Here the workflow is illustrated just to present that it is indeed possible to construct a working numerical workflow that integrates all the necessary applications and procedures.

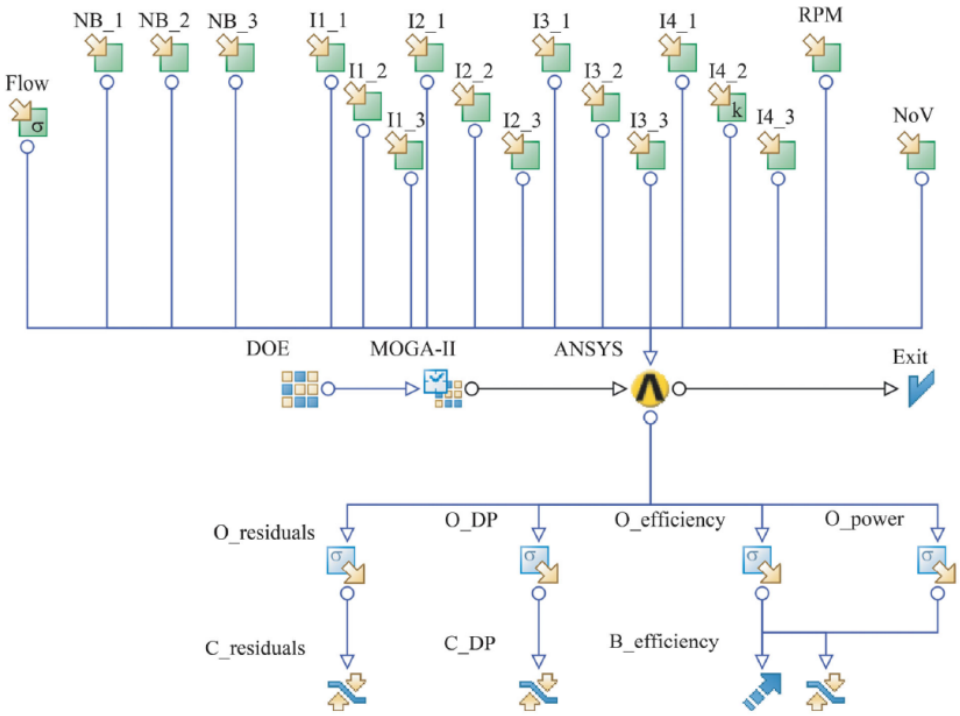


Figure 8 Example of optimization workflow [18].

3. SHAPE PARAMETERIZATION

This doctoral qualification exam considers engineering SO as a combination of fully generic 3D shape modeling, complex engineering numerical simulation simulation (3D CFD), and global optimization combined. Hence a heavy computational effort of the respective numerical workflow can be expected. This requires an efficient shape parameterization with as much geometric modeling capability with a small number of optimization variables. Surveys of parameterization methods for engineering applications have been made but they are usually limited to a specific object, for example [21], [22] give a survey of parameterization methods on 2D airfoil. So this chapter will give a more general overview of currently available efficient generic shape parameterization methods, focusing on genetic surfaces.

Wide variety of shape parameterization methods exist, as there is no single representation of surfaces that satisfies all of the needs for engineering SO problems. The simplest shape representation method is the mesh points method, which requires a large number of design variables but there are no restrictions on attainable geometry. Since it requires a large number of variables, it is not very useful for generic SO. While CAD models are often used in engineering, they consist of many interconnected partitioned geometric entities with corresponding parameters, relationships and constraints. This makes them not appropriate for SO and therefore other parameterization methods need to be considered. Complex 3D shape might in the most elementary approach be represented by a single polynomial surface, requiring high-degree polynomials. High degree polynomials have issues of lacking local control and they exhibit oscillatory behavior likely to introduce numerical difficulties. The first shape parameterization method that will be presented in the doctoral qualification exam is the Bezier surface (single-patch) parameterization. The Bezier single-patch surface offers a more intuitive shape parameterization method, explained in more detail in continuation of this chapter. A generalization of Bezier patches is B-spline surface, one of the most often used surfaces for shape parameterization. Furthermore, their generalizations are the non rational uniform B-splines (NURBS) and T-splines. Radial basis functions (RBF) are also reviewed since their different definition allows for some possible advantages in comparison to Bezier surface family. All of the mentioned generalizations of the Bezier surface can be regarded as single-patch parameterizations. In order to achieve further generalizations, multi-patch surfaces are required. Often used multi-patch surfaces in CAD are subdivision surfaces, but any single patch surface can be connected (with varying degree of smoothness) to create a multi-patch surface. When connecting multiple shapes, aspects of continuity between the surfaces are very important, thus a brief chapter about prescribing boundary conditions and continuity between the surfaces is also presented. If the prescription of boundary conditions is not an option, a smooth transition between the patches or at the surface-intersection could be achieved with blending functions. Finally and interesting possible method that could integrate the shape parameterization method with numerical analysis (CFD, FEM,…) is isogeometric analysis. Regarding the application in engineering optimization, still, no universal solutions exist. So an appropriate selection amongst a wide variety of existing methods and development of new parameterization methods offers a lot space for improvement.

3.1. Parametric, explicit and implicit shape

Before the actual shape parameterization methods are presented, three types of shape representation will be discussed. Descriptions will focus mostly on surfaces since they are important in engineering SO application although the extension to more dimensions is usually trivial. Two commonly used surfaces in shape modeling are parametric and implicit surfaces but sometimes explicit surfaces are also appropriate.

The explicit surface can be defined with single expression $z = \mathrm{P}(x, y)$. The parametric surface required three expressions similar to the explicit surface expression, what can be written as:

$$\begin{bmatrix} x(\boldsymbol{u}) \\ y(\boldsymbol{u}) \\ z(\boldsymbol{u}) \end{bmatrix} = \mathbf{P}(\boldsymbol{u}), \quad \boldsymbol{u} \in \mathbb{R}^2 \tag{12}$$

where:

- $\mathbf{P}$ is a mathematical function, mapping $\mathbb{R}^2 \to \mathbb{R}^3$, i.e. from parametric space to model space. For two-dimensional shapes that exist in three dimensional space, but in general case $\mathbb{R}^n \to \mathbb{R}^m$;
- $\boldsymbol{u}$ is a point in 2D parametric space.

The implicit surface is defined by:

$$\mathrm{I}(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \mathbb{R}^3 \tag{13}$$

where:

- I is a mathematical function, mapping $\mathbb{R}^3 \to \mathbb{R}^1$, but in general case $\Re^n \to \Re^1$. For example n=2 gives a implicit definition of curve. Algebraic, trigonometric, exponential, logarithmic etc. functions can be used, as well as any function described in the continuation of this chapter (most often with RBF-s);
- $\boldsymbol{x}$ is a point in 3D space;
- n is the dimensionality of space in which surface is located, for real objects n=3.

Comparably, explicit surfaces have the least shape generality as they cannot represent infinite slopes (if polynomials are used) or closed and multi-valued surfaces. The advantage is that they are easy to construct and display. The explicit surface can easily be converted to implicit $\mathrm{P}(x, y) - z = 0$ but reverse can be achieved only for simple cases. Implicit surfaces are difficult and non-intuitive to manipulate and not easy to construct (display). Curves can be constructed only in 2D. The implicit surfaces compared to parametric surfaces offer some advantages such as: any topology can be represented by a single mathematical function; easy representation of intersection; easy point classification for internal $(\mathrm{P}(\boldsymbol{x}) < 0)$ and external points $(\mathrm{P}(\boldsymbol{x}) > 0)$. But parametric surfaces are easy to construct and enable intuitive manipulation what is very important and makes parametric surfaces the most used type in engineering SO applications. The following chapters will chiefly presume that parametric surfaces are used although with little modification the same mathematical functions can be used to define implicit surfaces.

3.2. Bezier Patches

Bezier curves and surfaces (12) are convenient in shape parameterization since they pass through the end control points and the end slopes are defined by the respective control points at the ends, and analogously for higher-order derivatives. These properties are used in this paper to impose inter-segment continuity for piecewise chained curves. Parametric curve can be defined by the n-th degree Bezier curve P(u) for (n+1 control points) as [23]:

$$\begin{bmatrix} x(u) \\ y(u) \end{bmatrix} = \mathbf{P}(u) = \sum_{i=0}^{n} B_{i,n}(u) \cdot \boldsymbol{Q}_i \quad , \quad 0 \le u \le 1 \tag{14}$$

where *B* (Bernstein polynomial) is defined recursively (non-recursive and more efficient definition also exists) as:

$$\begin{gathered} B_{i,n}(u) = (1-u) \cdot B_{i,n-1}(u) + u \cdot B_{i-1,n-1}(u) \\ with: B_{0,0}(u) = 1,\ B_{i,n}(u) = 0 \quad for\ i < 0,\ i > n \end{gathered} \tag{15}$$

Vector $\boldsymbol{Q}_i$ is the vector of control points (Figure 9) coordinates and $u \in [0,1]$ is the curvilinear coordinate parameter.

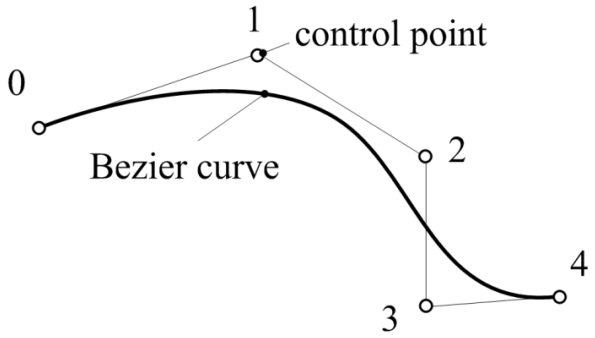


Figure 9. Bezier curve of degree 4 with 5 control points

Bezier curves can be extended to 3D Bezier surfaces (and bodies) for describing parametric surfaces (and bodies). This is achieved by combining Bezier curves, whereby control points of a Bezier curve are replaced by Bezier curves in the orthogonal direction,

$$\mathbf{P}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(u) \cdot B_{j,m}(v) \cdot \boldsymbol{Q}_{i,j} \quad , \quad u,v \in (0,1) \tag{16}$$

where $\boldsymbol{Q}_{i,j}$ are the control points of the control polyhedron. Bezier curves do not possess the property of locality and directly link the number of control nodes with the degree of the respective curve.

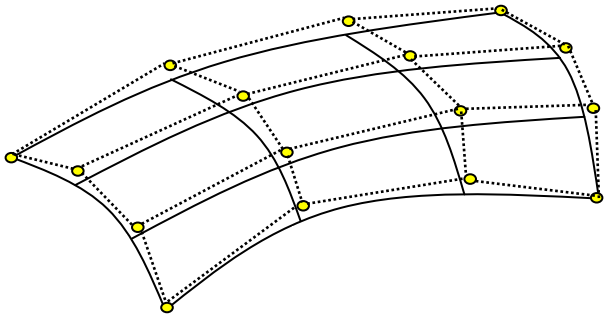


Figure 10. Single patch Bezier surface and control points and of 3rd degree [23].

3.3. B-spline and NURBS

B-splines are a generalization of Bezier curves (and a special case of NURBS) where the degree of the curve is independent of the number of control points and where the change of one of the control points only affects *k* segments. Parametric surface described by B-splines can be defined as:

$$\mathbf{P}(u,v)=\sum_{i0=0}^{n0}\sum_{i1=0}^{n1} N_{i0,d0}(u)\cdot N_{i1,d1}(v)\cdot \mathbf{Q}_{i0i1} \quad , \quad u,v\in[0,1] \tag{17}$$

with blending functions defined recursively as

$$\begin{aligned} N_{i,0}(t)&=\begin{Bmatrix}1, & t_i\le t<t_{i+1}\\ 0\,, & otherwise\end{Bmatrix} \quad , 0\le i\le n+d \\ N_{i,j}(t)&=\frac{t-t_i}{t_{i+j}-t_i}N_{i,j-1}(t)+\frac{t_{i+j+1}-t}{t_{i+j+1}-t_{i+1}}N_{i+1,j-1}(t) \quad , 1\le j\le d\,, 0\le i\le n+d-j \end{aligned} \tag{18}$$

$$0\le i\le n+d+1 \tag{19}$$

The equations (17)-(19) define the B-spline. As the individual shape functions $N_{i,j}$(t) are non-zaero just for the [t_i, t_{i+j+1}) interval, while amounting to zero for $t<t_i$ and $t\ge t_{i+j+1}$, the property of local control is ensured. As a result, the surface is locally formed exclusively by a small number of adjacent control points, $\boldsymbol{Q}_{ij}$. The first and the last blending function in both *u* and *v* directions are equal to unity at the ends while the rest of the blending functions equal to zero making the surface pass coincidently through the end curves. The distribution of knots t_i along the parametric coordinate influences the shape of the basis functions as illustrated in the following figure.

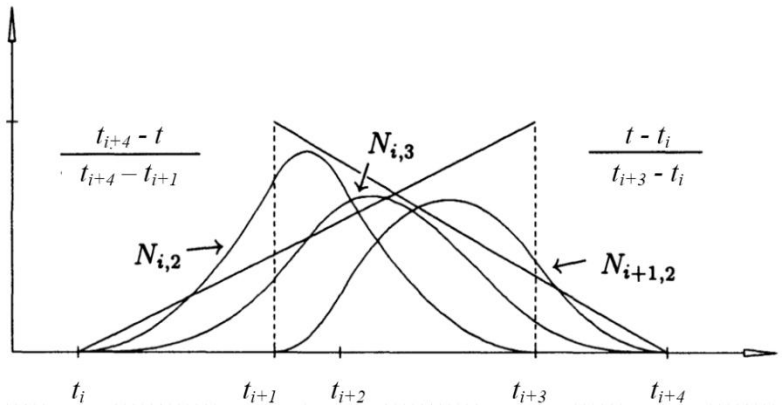


Figure 11. The recursive definition of B-spline basis functions [24].

The shape of the surface is controlled by modifying the control points and the knot vectors. The properties of local support, partition of unity and non-negativity add to the numerical stability of the subsequent optimization procedure. B-spline and NURBS surfaces are flexible enough and provide sufficient degrees of freedom (DOF) to represent the necessary shape for ship hull representation. Those integral shape parameterizations are also scalable as the number of control points and the degrees of the basic polynomials can be varied. The NURBS curves and surfaces are a generalization of the B-spline. NURBS use the same blending functions as B-splines but in n+1 dimensional space where n is the actual space dimensionality (n=2 for plane, n=3 for 3D space). The additional coordinate adds the ability to increase and decrease the impact of an individual control point on the resulting shape thus enabling better control. Furthermore, with additional coordinate analytical shapes (conics) can exactly be represented. For NURBS curve, the fourth coordinate is:

$$h = \sum_{i=0}^{n} h_i \cdot N_{i,d}(u) \tag{20}$$

where h_i denotes respective weights. The shape is projected to the original space by dividing with the fourth coordinate. Thus the NURBS for parametric curve:

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{n} h_i \cdot N_{i,d}(u) \cdot \mathbf{Q}_i}{\sum_{i=0}^{n} h_i \cdot N_{i,d}(u)} \tag{21}$$

For parametric surfaces, extension of NURBS curves to surfaces is defined by:

$$\mathbf{P}(u,v) = \frac{\sum_{i0=0}^{n0}\sum_{i1=0}^{n1} h_{i,j} \cdot N_{i0,d0}(u) \cdot N_{i1,d1}(v) \cdot \mathbf{Q}_{i0i1}}{\sum_{i0=0}^{n0}\sum_{i1=0}^{n1} h_{i,j} \cdot N_{i0,d0}(u) \cdot N_{i1,d1}(v)} \tag{22}$$

3.4. T-Splines

B-splines and NURBS belong to a type of tensor-product surfaces that use a rectangular grid of control points. A goal of T-splines is to generalize B-spline surfaces to

allow the control points rows or columns to be partial i.e. the row or column can start and be terminated before reaching the first of the last row or column position. T-spline control grids permit T junctions, so lines of control points need not traverse the entire control grid as with B-splines or NURBS. T-splines support many operations not possible with standard B-spline, such as local refinement, and the merging of several B-spline surfaces that have different knot vectors into a single gap-free model as illustrated in Figure 12. Merging B-splines into a T-spline can be achieved with C^0, C^1 or C^2 continuity[25].

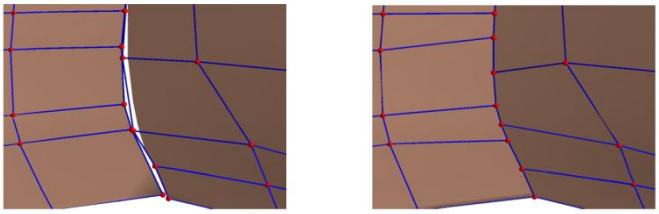


Figure 12. A gap between two B-spline surfaces, fixed with a T-spline [25].

To define a T-spline, it is necessary first to describe a surface whose control points have no topological relationship with each other whatsoever (as opposed to regular grids). This surface is called a PB-spline, because it is point based instead of grid based. The equation for parametric surface using a PB-spline is:

$$\mathbf{P}(u,v) = \frac{\sum_{i=0}^{n} B_i(u,v) \cdot \mathbf{Q}_i}{\sum_{i=0}^{n} B_i(u,v)} \tag{23}$$

where $\mathbf{Q}_i$ are control points and $B_i(u,v)$ are basis functions given by:

$$B_i(u,v) = N_{i0}^3(u) N_{i0}^3(v) \tag{24}$$

where $N_{i0}^3(u)$ and $N_{i0}^3(v)$ are the cubic B-spline basis function associated with the knot vectors:

$$\begin{aligned} u_i &= \left[u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}\right] \\ v_i &= \left[v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}\right] \end{aligned} \tag{25}$$

as illustrated in Figure 13. Thus, to specify a PB-spline, one must provide a set of knot vectors for each control point.

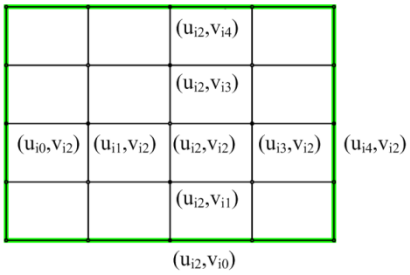
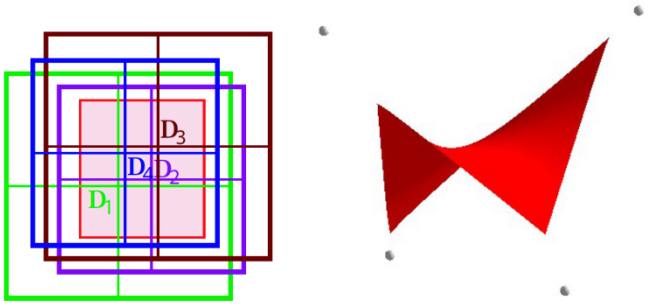


Figure 13. Knot lines for basis function $B_i(u,v)$ [25].

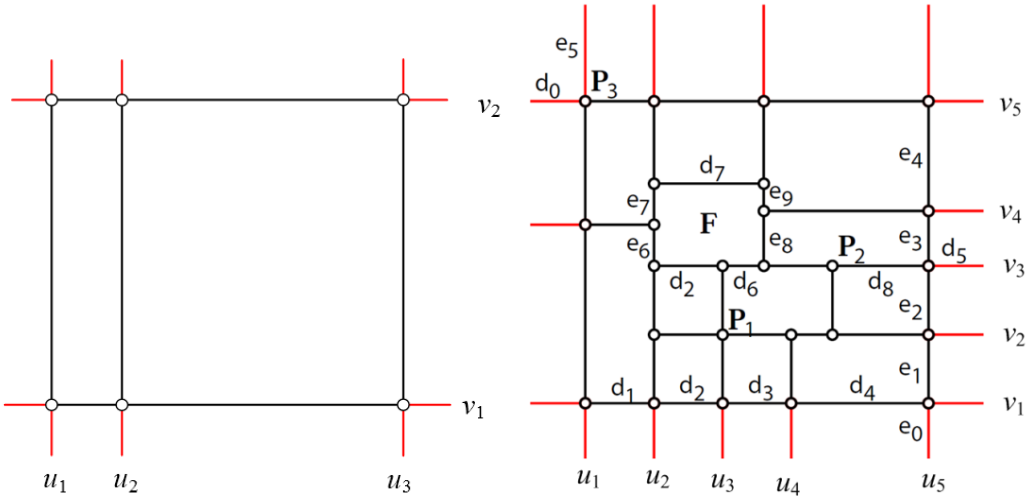
The PB-spline domain is usually selected such as there are at least three influence domains D_i , an example where four influence domains are defining a PB-spline domain is illustrated in Figure 14 together with the resulting shape. It can be noticed that a PB-spline is mesh-free method; the knot vectors for each basis function are completely independent of the knot vectors for any other basis function.



a) b)

Figure 14.A PB-spline with four control points: a) cubic PB-spline domain b) PB-spline control points in space and the resulting PB-spline shape [25].

A T-spline is a PB-spline for which some order has been imposed on the control points. The control points of T-spline are obtained by modifying standard B-spline mesh by following certain rules as described below, thus creating T-mesh. Figure 15a illustrates parametric location of PB-spline control points $[u_2, v_2]$; when PB-splines are set in an ordered grid, the result is equivalent to B-spline. Example of application of T-spline rules to an initial B-spline control point mesh results a T-mesh as illustrated in Figure 15b. A T-mesh serves two purposes. First, it provides easier and more intuitive control over the surface than does the completely arbitrary PB-spline control points. Second, the knot vectors u_i and v_i for each basis function are deduced from the T-mesh. When a rectangular grid (the same grid as for B-spline) is used for T-mesh, the T-spline reduces to a B-spline.



a) b)

Figure 15. Creating a T-mesh: a) initial B-spline mesh b) T-mesh created by following T-spline rules [25].

So called standard and non-standard T-splines exist. Both T-splines require that the following rules are met:

- The sum of knot intervals (distance in parametric u-v domain) on opposing edges of any face must be equal. Thus, for face F in Figure 15, $d_2 + d_6 = d_7$ and $e_6 + e_7 = e_8 + e_9$.
- If a T-junction on one edge of a face can be connected to a T-junction on an opposing edge of the face (thereby splitting the face into two faces) without violating previous rule, that edge must be included in the T-mesh.

To each P_i corresponds a basis function $B_i(u,v)$ defined in terms of knot vectors (25). The knot coordinates of P_i are (u_{i2}; v_{i2}). The knots u_{i3} and v_{i4} are found by considering a ray in parameter space $R(\alpha) = (u_{i2}+ \alpha; v_{i2})$. Then u_{i3} and v_{i4} are the s coordinates of the first two u-edges intersected by the ray (not including the initial (u_{i2}; v_{i2})). The other knots in u and v are found in similar manner. P_3 is a boundary control point. In this case, $u_{3,0}$ and $v_{3,4}$ do not matter, so any displacement can be taken. This is valid for both standard and non-standard T-splines. A standard T-spline is defined as one whose basis functions $B_i(u,v)$ in equation (23) sum to one for every (u, v) in the domain. A standard T-spline can be constructed by starting from a regular B-spline mesh, and using the knot insertion rule:

- P3' can only be inserted if $v_1 = v_2 = v_4 = v_5$ (see Figure 10). v_i is the v-knot vector for basis function B_i. If the control point were being inserted on a vertical edge, the four neighboring u_i knot vectors would need to be equal [25].

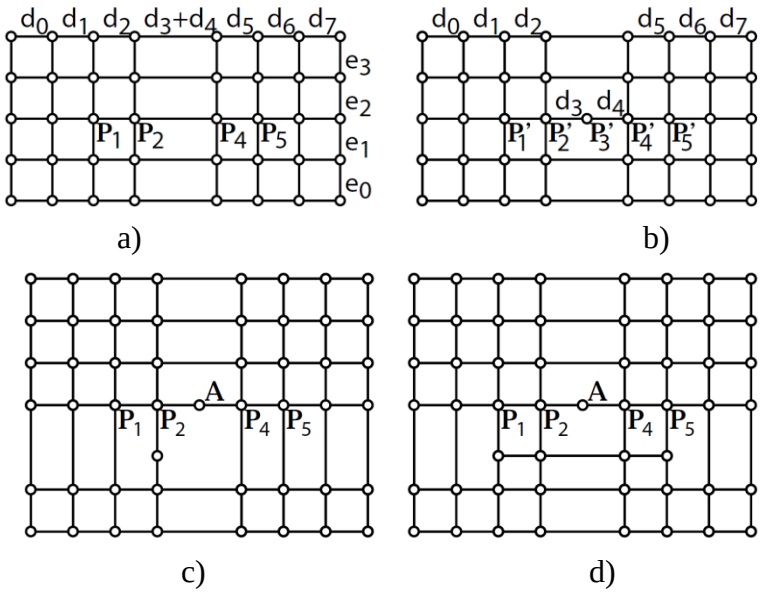


Figure 16. T-mesh knot insertion: a) before insertion, b) after insertion c) A cannot be inserted for a standard T-spline d) A can be inserted keeping a standard T-spline [25].

3.5. Subdivision Surfaces

The past few decades witnessed a lots of research activities in the area of CAD especially with relation to subdivision surfaces [26]. A recursive subdivision surface can be defined as a limit in application of a set of subdivision rules R to an initial polyhedral network P as illustrated in Figure 17. Subdivision rule from the set R is repeatedly selected and applied to the intermediate results of the subdivision process. Thus the initial polyhedron P is driven though a sequence of transformations which in proper case lead to a smooth limit surface.

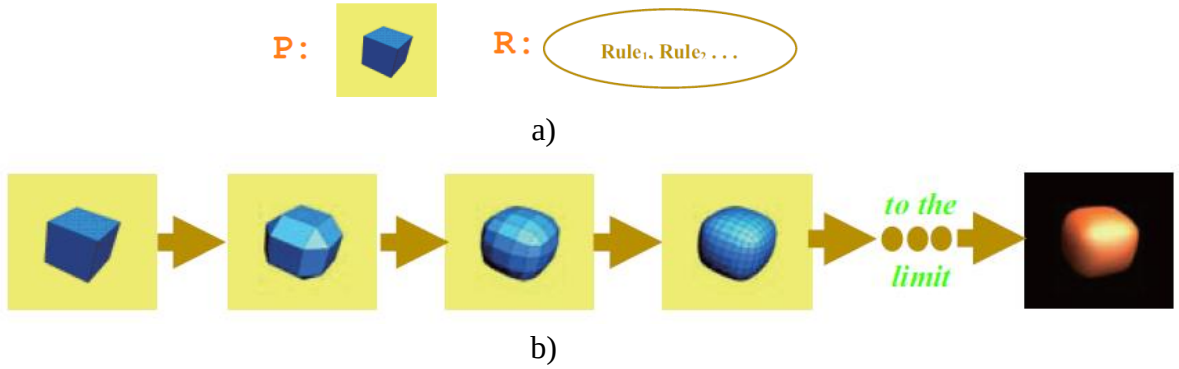


Figure 17. Recursive subdivision scheme: a) The elements of a recursive subdivision scheme b) The recursive subdivision process[26].

The main attraction of recursive subdivision is, first, the unified character of the underlying theory and, second, its flexibility in view of its ability to handle shapes of arbitrary topology. However, there are multiple concerns when applying subdivision surfaces in engineering optimization problems. Conceptual issues exist such as continuity and differentiability of the limit surface. Another concern is accuracy of representation in relation to parameters such as normal vectors, curvatures. These limitations are a major concern for

application in engineering optimization, especially if integration of CAD and analysis in isogeometric analysis solution is sought. Additional problem is the efficiency of subdivision algorithms and their heavy memory consumption in the case of large applications.

3.6. Radial basis functions

Radial Basis Functions (RBF) are often used for interpolating scattered points, appropriate for example on 3d point cloud obtained by 3D scaning. They are usually used with large number of basis functions, an example of interpolation to 438,000 point-cloud is illustrated in Figure 18.

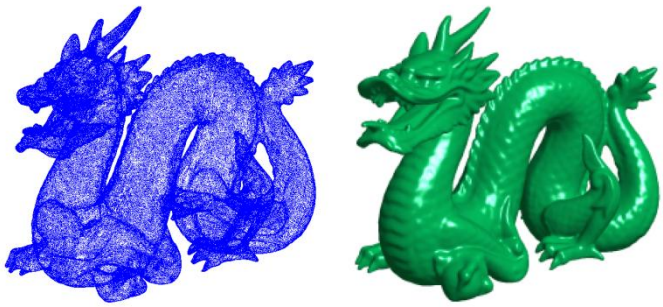


Figure 18. Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud [27].

Generally, parametric shape can be described by RBF (surfaces, curves,…) that are build up by summing the individual RBFs multiplied by weighting coefficients:

$$\mathbf{P}(\boldsymbol{u}) = \sum_{i=0}^{N} R_i(\boldsymbol{u}) \cdot \boldsymbol{w}_i \tag{26}$$

where:

- N is the number of used RBFs,
- $\boldsymbol{w}_i$ is weighting coefficient, for surface it is 2-dimensional, in general case n-dimensional. It could be regarded as a control mechanism since it represents a free variable that influences the shape of the resulting surface,
- $R_i(\boldsymbol{u})$ is the RBF. Various different RBFs exist, the most simple and often used is the Gaussian RBF:

$$R_i = e^{-(c\|\boldsymbol{u}-\boldsymbol{u}_i\|)^2} \tag{27}$$

where:

- c is constant;
- $\boldsymbol{u}$ is parametric coordinate, in general m-dimensional;
- $\|\ \|$ is Euclidian distance;
- $\boldsymbol{u}_i$ is the individual RBF center.

Because RBFs usually require large amount of points to accurately represent shape, they might not be appropriate for engineering SO. Nevertheless RBFs may potentially be useful since they can be used with unstructured (control) points. Gradients and higher

derivatives are easily determined analytically and are continuous and smooth. Additionally RBFs can be used for smoothing and re-meshing existing surfaces [27].

3.7. Boundary conditions

In order to describe a complex geometry it is often required to join multiple surfaces with positional, tangential or curvature continuity. In earlier discussion, most of the parameterization methods can be regarded as a single patch parameterizations. While the subdivision surfaces are a sort of multi-patch parameterization able to represent arbitrary topology, they exhibit problems with continuity and differentiability. To convert single patch parameterization to multi-patch and enable representation of arbitrary topology, the single patch parameterizations need to be connected mutually. This requires imposing the boundary conditions on the connected parts of single patch parameterizations. Two types of continuity exist, geometric continuity and parametric continuity. G^k and C^k are used to denote geometric and parametric continuity of order k respectively. G^0 and C^0 continuity means that the surface is continuous regarding its value both in model space and in parametric space (notice that both value continuities have the same definition). If the shape is continuous in first derivative G^1, this does not mean it is continuous in value. Difference between geometric and parametric continuity can be explained on the example of two connecting curves in a point $\boldsymbol{p}(u)$. Both curves have tangents at the point, $\boldsymbol{p}_1^u(u)$ and $\boldsymbol{p}_2^u(u)$ (derivative with respect to u). If $\boldsymbol{p}_1^u(u) = c\boldsymbol{p}_2^u(u)$ and $c \neq 1$, the curves are G^1 continuous, meaning they are only continuous in first derivative in model space but not in parametric space. When connecting two patches, depending on application both geometric and parametric continuity are important. Single patch can be connected both on edges of the local support domain of the respective single-patch parameterization (boundary conditions) or by intersecting two (or more) single patch surfaces and imposing the continuity conditions at the intersection (imposing conditions within the domain).

When using RBF surfaces, the domain is usually the entire $\Re^n$ where n is the dimensionality of parametric space, so the "boundary" conditions are imposed within the domain if one wants to connect two different surfaces. Since RBFs are not often used for geometry modeling (more often for interpolation); this chapter will focus on NURBS surfaces and how to impose conditions both on the boundary and within the domain.

3.7.1. Imposing conditions on domain boundary

Constraints can be imposed by a direct method or by basis modification method in which the B-spline basis is replaced by a modified basis [28]. For a simple coincidence constraint, imposing constraint is trivial when using B-spline or NURBS curve. A NURBS curve $\Re^1 \to \Re^1$ having a knot sequence $u_i \in [a,b]$ with k coincident knots at the ends:

$$u_i = \{a = u_1 = \ldots = u_k \leq u_{k+1} \leq \ldots \leq u_n \leq u_{n+1} = \ldots = u_{n+k} = b\} \tag{28}$$

interpolates its end control points. This means that, e.g., at the end $u=a$:

$$\mathrm{P}(u)\big|_{u=a} = \frac{\sum_{i=1}^{n} N_i(a)\cdot h_i(u)\cdot \mathrm{Q}_i}{\sum_{i=1}^{n} N_i(a)\cdot \mathrm{Q}_i} = \frac{h_1\cdot \mathrm{Q}_1}{h_1} = \mathrm{Q}_1 \tag{29}$$

So, a NURBS curve by default interpolates its end control point i.e. if two points are given, they are interpolated by setting the same coordinate values to endpoints. Positional continuity between two NURBS curves is obtained by setting the end control points to be identical. Furthermore, a NURBS curve is tangent to the lines connecting the first two or, respectively, the last two control points at its ends. At the end $u=a$:

$$\begin{aligned} & h_1\cdot \mathrm{Q}_1\cdot N_1'(u)\big|_{u=a} + h_2\cdot \mathrm{Q}_2\cdot N_2'(u)\big|_{u=a} = \mathrm{P}'(u)\big|_{u=a}\, h_1 + \mathrm{Q}_1\left(h_1\cdot N_1'(u)\big|_{u=a} + h_2 N_2'(u)\big|_{u=a}\right) \\ & \textit{or} \\ & \mathrm{P}'(u)\big|_{u=a} = N_2'(u)\big|_{u=a}\,\frac{h_2}{h_1}(\mathrm{Q}_2 - \mathrm{Q}_1) \end{aligned} \tag{30}$$

Where N_i' denotes derivation of the respective basis function. So, if a tangential conditions needs to be imposed on a NURBS curve end, the end-point and the nearest control point have to be set collinear with this direction. Tangential continuity between two NURBS curves is obtained by setting the end control points of both curves to be collinear. For higher order continuity general formulas can also be derived but it becomes more and more complex. There also exist sufficient conditions with which two curves can match smoothly, based on the theory of subdivision of curves.

The conditions described for curve can be generalized to NURBS surfaces by applying them on a row by row basis, the rows taken perpendicular to the side on which the boundary condition is imposed, see Figure 19. This implies that the NURBS surfaces at the end-to-end location need to be of the same length, the same order and the same knot vector along the connecting location. If the bounding entity is a NURBS curve or a NURBS surface, this condition will not be fulfilled automatically and some conversions may be necessary.

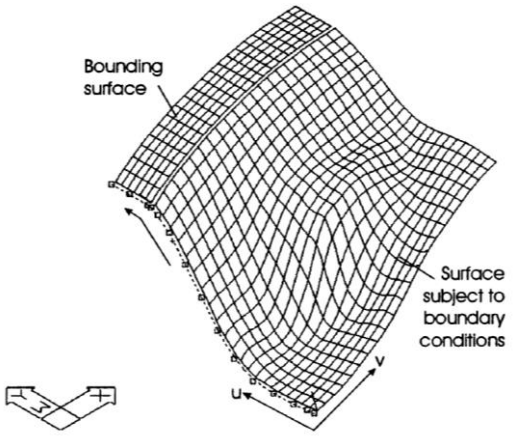


Figure 19. Imposing boundary conditions on a NURBS surface can be done on a row by row basis [29].

Boundary conditions can be imposed on a NURBS curve or surface at two levels, i.e. by incorporating boundary conditions into the fitting process or by imposing boundary conditions on a fitted surface and thereby modifying the surface locally. Some commercial CAD systems support the latter method. The first method has the advantage that the least squares fitting, in which the distances to the digitized points are minimized, is performed taking into account the imposed conditions so as to obtain the best fit [29].

In [30], two cases of imposing boundary conditions between two surfaces are considered. The first case illustrated in Figure 20a is fitting of two surfaces to two sets of measurement data with a G2 continuity between the surfaces. The second case (Figure 20b) is the case for one known surface and one data set for fitting. B-spline surfaces are used. In was shown that when two surfaces share the same boundary and are G2 continuous, the three arrays of the control points nearest to the common boundary for both surfaces can be related to each other. When one surface is known, the first three arrays of the control points of the second surface can therefore be determined, in the case where two surfaces are G2 continuous. The problem is solved by including the G2 continuity conditions in fitting procedure.

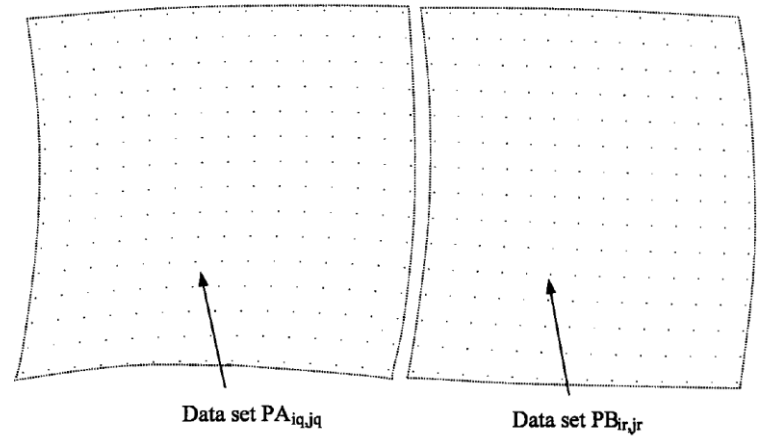


Figure 20. Imposing boundary conditions between two surfaces: a) Two sets of measurement data; b) A set of measurement data and a known surface [30].

Chui et.at. [31] introduced an approach for constructing effective algorithms for removing gaps between parametric bicubic NURBS surfaces, while maintaining G1 smoothness for the combined surface. The smoothness was accomplished by manipulating the control points and weights, but without disturbing the interpolation data, while minimizing the modification of the NURBS surfaces patches. The technique was developed for multiple surfaces but in the paper was applied to up to four NURBS surfaces as illustrated in Figure 21.

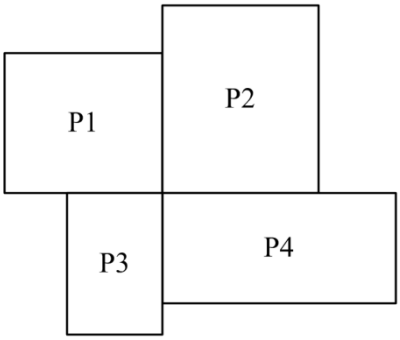


Figure 21. Schematic diagram of parametric domains of four parametric NURBS surfaces P1, P2, P3and P4.

Another common problem imposed on boundaries is construction of a NURBS (or B-spline) surface satisfying prescribed angle distribution along its boundary curve. In [32], the problem is (among other constraints) prescribing a given angle distributions along a splitter for a Pelton turbine bucket as illustrated in Figure 22. Geometry is described with uniform bicubic B-spline surface composed of 7 × 4 patches which is determined by the control net of 12 × 7 control points and knot vectors. Conditions for C2 continuity across the inner curves are prescribed on the control points in the control net, what can be accomplished as presented earlier. The most difficult part of imposing given angles and distances on the Pelton turbine bucket B-spline model is to satisfy prescribed angle along a splitter and not between the individual B-spline patches. It was proven that for a given B-spline curve $\mathbf{P}(\boldsymbol{u})$ the exact solution exists only in very special cases, for a special form of an angle function $f(u)$, so algorithms for finding approximate solutions have been developed [32].

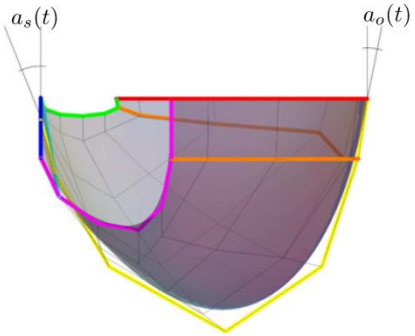


Figure 22. Angle distribution along a splitter $a_s(u)$ [32].

3.7.2. Imposing user defined constraints

Many current geometric modelling systems allow constraints to be specified on models constructed by a user. In parametric systems, a generic model is defined using a set of parameters. A particular realization of the model is then instantiated by allocating particular values to the parameters. It may be possible in such systems to specify some parameters in terms of others, but there must be a unique ordering allowing the geometry to be constructed sequentially from previously determined elements. In contrast, variational systems allow the user to input the geometry in a schematic way, and for the actual geometry to be determined by constraints linking the geometric elements. Various methods of solving such constraint systems exist. These may be broadly summarized as (i) analytical solvers, which rely on numerical methods (such as Newton-Raphson methods), or symbolic or both, (ii) graph-theoretical based methods, and (iii) rule-based methods. Selecting the desired solution when the non-linear constraint system has many multiple solutions is a tricky problem; it persists even when the user provides an approximately correct initial geometry, and heuristics are often used. Generally, the geometric problem is translated into an algebraic equation system, which is possibly reduced by algebraic simplification [33]. The standard numerical method for solving a minimisation problem subject to a set of equality constraints is to use the Lagrangian multipliers method.

3.8. Surface intersections

Intersections are a necessary part of CAD, geometric design and modeling, analysis, and especially important in CAM and manufacturing applications. The most important intersection problem involves intersections of surfaces to surfaces. In order to solve general surface to surface (S/S) intersection problems, the following auxiliary intersection problems need to be considered [34]:

1. point/point (P/P)
2. point/curve (P/C)
3. point/surface (P/S)
4. curve/curve (C/C)
5. curve/surface (C/S)

All of the above intersection problems are especially useful also in robotics, manufacturing simulation, collision avoidance, etc. When the geometric elements involved in intersections problem are nonlinear (curved), the problem normally reduces to solving set of nonlinear equations. Solving the nonlinear systems is a complex process in general, requiring application of numerical methods. Furthermore, the problems that come about in geometric modeling applications pose severe accuracy, robustness, and efficiency requirements on solvers of nonlinear systems. Therefore, specialized solvers have been developed to address these requirements explicitly using geometric formulations [34].

The first - point/point intersection problems reduces to checking the Euclidean distance between two points usually by checking if the distance is smaller than some tolerance. Next case is point / parametric curve intersection . In general case of parametric curve, there is no

known and easily computable convex box decreasing in size arbitrarily with subdivision for a procedural parametric curve. An approximate solution method may involve minimization of distance with parametric coordinate *u* as minimization variable. Initial solution for the possible minima may be found by using linear approximation of the curve. Convergence of the minimization processes is not guaranteed in general and there more than one local minimum could exist. Furthermore convergence to local and not global minimum (distance note equal to zero) is possible. The same applies for point/surface intersection. Since even with the simplest cases, the solution is not guaranteed to yield a solution, the same will apply to the following cases. General curve/curve intersection is usually solved by minimizing the squared distance of function with two variables – u_1 and u_2 – parametric coordinates of the respective parametric curve, and an initial solution can be obtained by linear approximation as illustrated in Figure 23. Curve to surface intersections are useful and often used for solving the more general surface to surface intersection problems. The general problem is solved as earlier by minimizing the squared distance of function with now three variables; two parametric variables of the surface and one for the curve. Although solutions of previous problems exist for specialized cases, in general the problem is solved by numerical minimization of the non linear objective function.

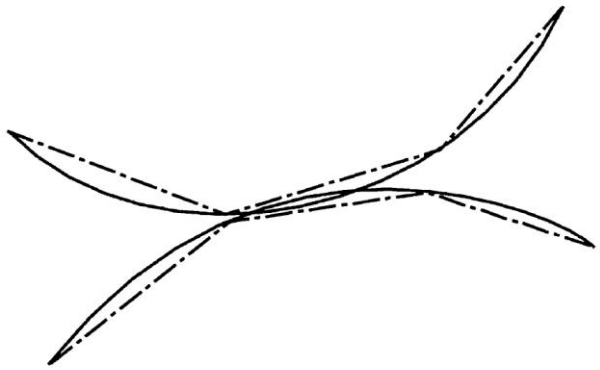


Figure 23. Linear approximation of parametric curve by polygon [34].

General surface/surface intersection algorithms that could handle a broad class of surfaces, and obtain a closed form intersections are difficult, if not impossible. Barnhill and Kersy [35] proposed a general marching method for surface/surface intersection problem. The marching algorithm includes three processing steps for obtaining intersection approximations: obtaining start points on intersection curves; marching along intersection curves; and sorting disjoint intersection segments into ordered curve approximations. The procedure is the extension of the procedure by Barnhill [36]. The algorithm proceeds in two stages:

- Find an initial point by solving curve/surface intersection problem between isoparametric curve of the first surface, and the second surface.
- Follow this intersection curve by producing more points in a sequential fashion along the intersection curve.

3.9. Multi-patch Parameterizations

Any of the mentioned single patch shape parameterization methods can be used to generate a multi-patch parameterization by imposing the required continuity conditions on the boundaries. If the single patch shapes are not adjacent in parametric domain, first the intersection has to be calculated. When using T-splines or RBFs, multi-patch parameterization may even not be necessary i.e. are only necessary if topological changes are required.

Application of multi-patch parameterization can be illustrated on chained Bezier surfaces. In the paper [23], chained piecewise Bezier curves and surfaces are developed to provide for locality and low-degree curve. Bezier patches are piecewise by definition, with each defined only within a partial segment of the entire domain. However, while reducing the problems of locality and possible oscillations, this raises an additional request of imposing and ensuring adequate continuity between the piecewise segments. In the framework of design optimization, chaining of piecewise curves should also not increase the parameterization complexity, as increasing the number of control points would drastically increase the numerical effort due to higher dimensionality of the search space. The proposed procedure in [23] is based on subdividing the domain into patches and chaining piecewise low-degree Bezier curves and surfaces into complex shapes. The procedure starts by subdividing the problem domain into patches, for each of which an individual approximation curve or surface with corresponding control points is assigned. In segments (between adjacent original control points) where chained surfaces are to be joined with C1 continuity, additional control points are interpolated as illustrated in Figure 24 using the local original points. The number of optimum design variables (original control points) is thereby not increased since the generated points depend on the original ones.

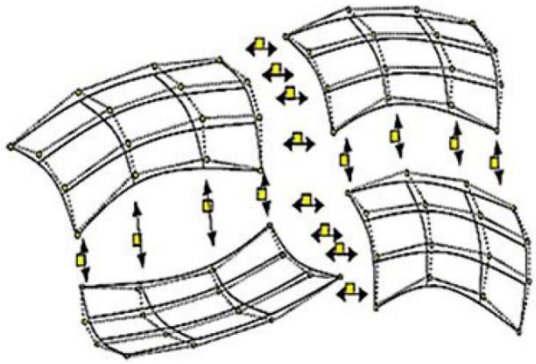


Figure 24. Chaining 3rd degree Bezier surfaces with C1 continuity [23].

3.9.1. Blending Functions

Blends are, in-between surfaces which smoothly join multiple other surfaces. Often, some of the original surfaces of the object would have intersected in sharp edges (or would not intersect at all) and, for a variety of reasons it may be desired to replace some or all of the

sharp edges by the corresponding smooth faces. The blending surfaces are usually required to have at least G1 continuity [37]. In general, smoothing is applied not just to the edges, but to the whole region of the original surface of the object(s). Blending can be considered as a technique for design of new surfaces on basis of existing ones. Various methods for blending of parametric surfaces exist, of which Coon patches [38] are commonly used. They are applied for "filling in" between curves (surfaces). If four arbitrary curves $\mathbf{c}_1(u)$, $\mathbf{c}_2(u)$ and $\mathbf{d}_1(v)$, $\mathbf{d}_2(v)$, are defined over $u \in [0,1]$ and $v \in [0,1]$ respectively, Coon patch is method of finding a surface **P** that has these four curves as boundary curves:

$$\begin{aligned} &\mathbf{P}(u,0) = \mathbf{c}_1(u), \quad \mathbf{P}(u,1)\mathbf{c}_2(u), \\ &\mathbf{P}(0,v) = \mathbf{d}_1(v) \ , \quad \mathbf{P}(1,v) = \mathbf{d}_2(v). \end{aligned} \tag{31}$$

The four boundary curves define two ruled surfaces:

$$\begin{aligned} &\mathbf{r}_c(u,v) = (1-v)\mathbf{P}(u,0) + v\mathbf{P}(u,1) \text{ and} \\ &\mathbf{r}_d(u,v) = (1-u)\mathbf{P}(0,v) + u\mathbf{P}(1,v) \end{aligned} \tag{32}$$

As illustrated in figure Figure 25., the ruled surface $\mathbf{r}_c$ fails to reproduce d-curve, while $\mathbf{r}_d$ fails to reproduce c-curve. Coon strategy is to try to retain what each ruled surface interpolates correctly, and eliminate what each fails to interpolate. This can be achieved by bilinear interpolant $\mathbf{r}_{cd}$:

$$\mathbf{r}_{cd}(u,v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} \mathbf{P}(0,0) & \mathbf{P}(0,1) \\ \mathbf{P}(1,0) & \mathbf{P}(1,1) \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \tag{33}$$

The Coons patch can now be defined as:

$$\mathbf{P} = \mathbf{r}_c + \mathbf{r}_d - \mathbf{r}_{cd} \tag{34}$$

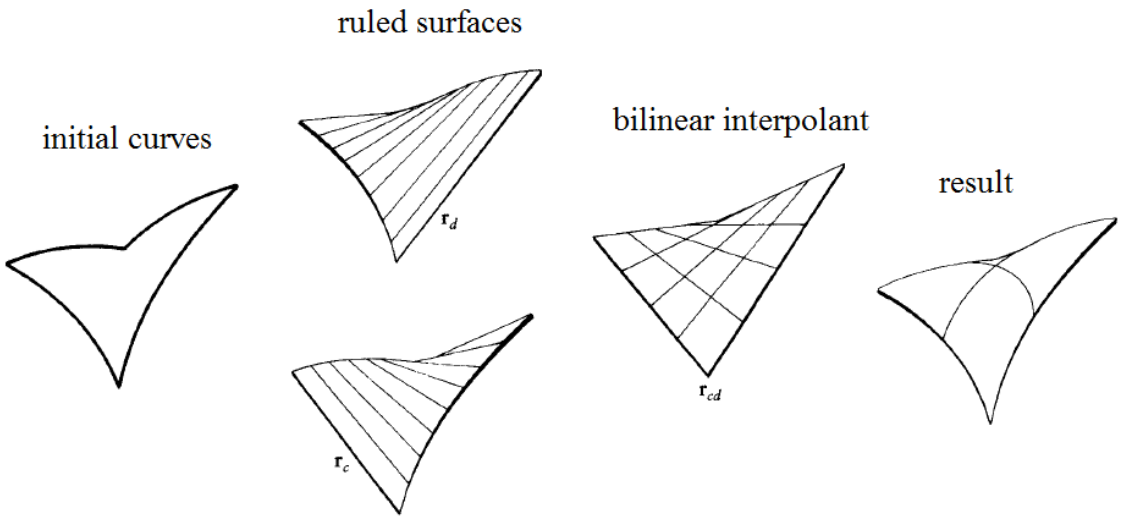


Figure 25. Coons patches: bilinearly blended Coons patch comprises two lofted surfaces and a bilinear surface [38].

In [39] a method similar to Coon patches is developed. The developed method can be used to create a blending surface between two arbitrary surfaces. This method requires that a nonzero tangent vector at every point on the boundary curve exists and is unique. The method works with both parametric surfaces and implicit surfaces.

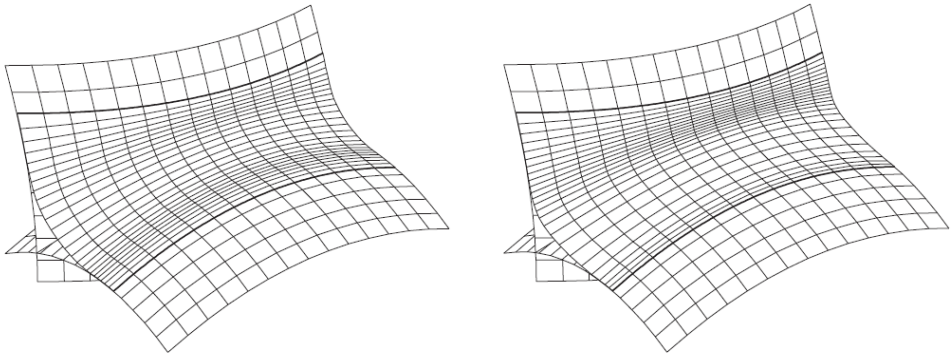
In [40], blending method that produces G^n–continuous parametric blend curves and surfaces, given two design parameters with which users can manipulate the solution. The blending curves are just linear combinations of the given curves (base curves) with suitably chosen coefficients (blending functions). The blending surfaces are linear combinations of the base surfaces with blending functions depending on one of the common parameters. The C^n continuous blending function that "mixes" two curves (surfaces $\mathbf{P}_1(u,v)$ and $\mathbf{P}_2(u,v)$) is selected with the following properties:

$$\begin{aligned} &f(0)=1, \quad f(1)=0, \\ &f^{(k)}(0)=f^{(k)}, \quad f^{(k)}(1)=0 \quad \text{for } k=1,...,n \end{aligned} \tag{35}$$

Then the surface patch (blending surface), $\mathbf{P}_B(u,v)$:

$$\mathbf{P}_B(u,v)=f(u)\mathbf{P}_1(u,v)+(1-f(u))\mathbf{P}_2(u,v), \quad u\in[u_1,u_2], \quad v\in[0,1] \tag{36}$$

Where u_1 and u_2 are arbitrarily selected values representing the "mixing" part of the original function for $\mathbf{P}_1(u,v)$ and $\mathbf{P}_2(u,v)$ respectively. Various blending functions are considered, and they generally allow for modification of sharpness of the blending surface. Example of surface generated by this approach is illustrated in Figure 26, where the same blending function is used but with modified coefficient that influences the sharpness of the blending transition.



a) b)

Figure 26. Example of blended surfaces with variable blending functions. [40].

In [41], semi-structured B-spline surface is generated by skinning a series of B-spline curves with different knot vectors. The B-spline surface blending problem is approached as an optimization problem with continuity constraints. The developed semi-structured B-spline surface has shown an ability to blend two base B-spline surfaces with mismatched knot vectors, and keep G^2 and C^2 continuity. In [42] s a method of G^n blending of multiple parametric surfaces in polar coordinates is presented with mechanism of converting a Cartesian parametric surface into its polar coordinate form. The method is applied directly to

filling N-sided holes without compatibility restrictions on the boundary. The model obtains G^n continuity and NURBS compatibility.

3.10. Isogeometric analysis

Isogeometric analysis (IGA) has large potential of improving the efficiency of the overall optimization procedure by integrating shape parameterization and finite element method. The root idea behind IGA is that the basis used to exactly model the geometry will also serve as the basis for the solution space of the numerical method. Using the same basis for geometry and analysis is quite common in classical finite element analysis, and it is called the isoparametric analysis. The fundamental difference between this new concept of IGA and the old concept of isoparametric finite element analysis is that, in classical FEA, the basis chosen to approximate the unknown solution fields is then used to approximate known geometry. IGA turns this idea around and selects a basis capable of exactly representing the known geometry and uses it as a basis for the fields we wish to approximate. In a sense, we are reversing the isoparametric arrow such that it points from the geometry toward the solution space, rather than vice versa [43]. NURBS are most often used for this application.

4. PARAMETRIC SHAPE FITTING

The previous chapter described how to generate surfaces from a known control point (CP) net. The inverse problem is also of interest; i.e., given a known set of data on a surface, determine the control net that best interpolates that data. This is known as shape fitting. By using shape fitting of parametric surfaces to the already existing solutions of the optimization problem at hand, a preliminary result of a parameterization method efficiency can be obtained. This makes a valuable tool since numerically expensive full optimization is not required in order to test a parameterization method. Also, adaptive parameterization method (switching of parameterization methods during the optimization procedure) could be developed with the aid of parametric shape fitting.

4.1. Linear fitting

The problem of fitting is shown for illustration in Figure 27. The problem is determining a control polygon that generates a B-spline curve for a set of known data points.

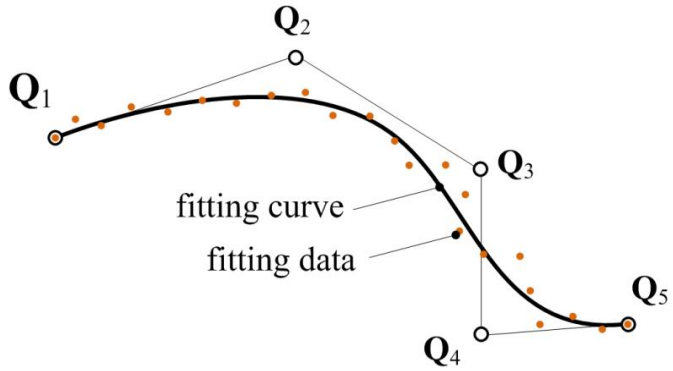


Figure 27. Determining control points polygon for a known data set.

If objective is that the curve passes through all data points, the following equations must be satisfied:

$$\sum_{i=0}^{n} B_i(u_j) \cdot Q_i = P_j \quad \text{where } i = 1,...,n \text{ and } j = 1,...,m \tag{37}$$

where $B_i(u)$ are the basis functions (Bezier, B-spline,NURBS,…) P_j are the data points, Q_i are the control points. The system of equations is more compactly written in matrix form as:

$$[\boldsymbol{B}][\boldsymbol{Q}] = [\boldsymbol{P}] \tag{38}$$

If *n*=*m*, the problem is interpolation and simple inversion $[\boldsymbol{Q}] = [\boldsymbol{B}]^{-1}[\boldsymbol{P}]$ can be used to obtain the control points. For *n*<*m*, the following can be conducted by using the pseudoinverse:

$$\begin{aligned} [\boldsymbol{B}]^T [\boldsymbol{B}][\boldsymbol{Q}] &= [\boldsymbol{B}]^T [\boldsymbol{P}] \\ [\boldsymbol{Q}] &= \left([\boldsymbol{B}]^T [\boldsymbol{B}]\right)^{-1} [\boldsymbol{B}]^T [\boldsymbol{P}] \end{aligned} \tag{39}$$

In general case, the problem is solved by minimization of the following least square error function:

$$E(\boldsymbol{Q}) = \left| \sum_{i=0}^{n} B_i(u_j) \cdot \boldsymbol{Q}_i - \boldsymbol{P}_j \right|^2 \tag{40}$$

For surfaces, the least square error function can be written as:

$$E(\boldsymbol{Q}) = \left| \sum_{i=0}^{n} B_i(u_j, v_j) \cdot \boldsymbol{Q}_i - \boldsymbol{P}_j \right|^2 \tag{41}$$

4.2. Enhanced fitting methods

The advanced method of non-linear fitting procedure allows for B-spline control points crowding and aggregation at locations of certain geometric features thus enabling good shape fitting while keeping low numerical complexity and high stability [44][45]. The advanced fitting method is crucial since the linear fitting cannot replicate the mentioned control point aggregation that would accrue in the case of real optimization. The fitting method is composed of a projection of point cloud ***P*** to a rectangular domain and obtaining an initial solution by linear fitting a B-spline to the projected point cloud. After the linear fitting, a gradient method and genetic algorithm are used for improving the solution. As opposed to classical B-spline fitting, the parametric coordinates (*u* and *v*) of the (point cloud) individual points are additional fitting variables that allow for the spline control points movement towards locations of certain geometric features such as the sharp edges. In that case the error function subjected to minimization is:

$$E(\mathbf{U},\mathbf{V},\mathbf{Q}) = \frac{1}{2}\sum_{j0=0}^{m0}\sum_{j1=0}^{m1}\left|\sum_{i0=0}^{n0}\sum_{i1=0}^{n1} N_{i0,d0}(u_{j0\,j1})\cdot N_{i1,d1}(v_{j0\,j1})\cdot \mathbf{Q}_{i0i1} - \mathbf{P}_{j0,\,j1}\right|^2 \tag{42}$$

where **P** is point cloud matrix representing the hull shape; **U** and **V** are matrices of parametric values:

$$\mathbf{U} = \begin{bmatrix} u_{00} & \dots & u_{0m_1} \\ \dots & \dots & \dots \\ u_{m_0 0} & \dots & u_{m_0 m_1} \end{bmatrix}$$
$$\mathbf{V} = \begin{bmatrix} v_{00} & \dots & v_{0m_1} \\ \dots & \dots & \dots \\ v_{m_0 0} & \dots & v_{m_0 m_1} \end{bmatrix} \tag{43}$$

For the initial solution, the **U** and **V** values are fixed and only the control points coordinates are obtained. Before obtaining the initial solution, an ordered structured point cloud is required such that it can be written in matrix form. To obtain the required ordered distribution for **U** and **V** in the case of the full 3D spline fitting, a projection of shape from physical space to parametric space is required. The most- simple method is to use parallel sections which can individually be projected to the respective location in the parametric domain. However this is not appropriate for a complex hull shapes such as the DTMB hull illustrated in Figure 28.

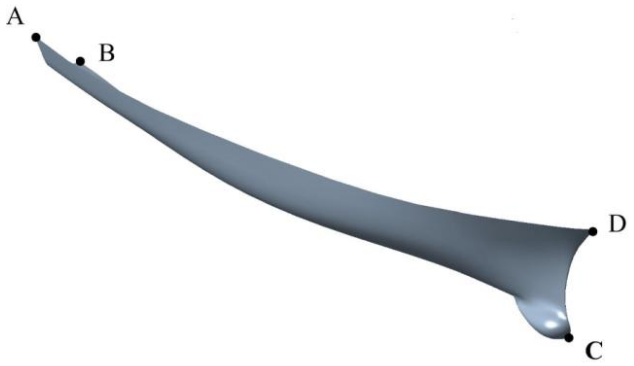


Figure 28 DTMB half of ship hull.

Figure 29 illustrated the necessity of projection to rectangular domain. The line j-j which is the straight deck line with constant height can easily be projected to *u*-*v* domain by projection in *y*-axis direction. Similarly, the line k-k also appears that it can be easily be projected but in a general case, line k-k is a 3D curve. The line designated l-l is obviously a 3D curve and it cannot be projected to the parameter space in a simple matter as the line j-j. Various methods of projection of 3D surfaces to rectangular domain exist. Another possibility is applying the spring analogy to the meshed point cloud in order to achieve a regular square

shape in parametric *u* and *v* coordinates [46] and applying linear interpolation to generate the required ordered distribution of points. Here, a procedure developed in [44] is implemented to achieve the required matrix topology since it provides for much better initial guess.

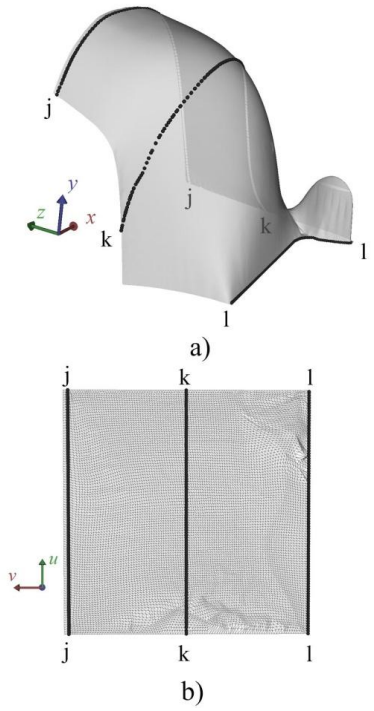


Figure 29. Lines j-j, k-k and l-l illustrated in: a) physical space b) parametric space.

The procedure will be illustrated on a half of the DTMB, scaled to the unit cube. The starting point is a triangulated surface with an unordered point cloud $\mathbf{p}^U$ with physical coordinates, scaled to the unit cube as illustrated in Figure 30. The unordered points $\mathbf{p}^U$ can be divided into two sets of points, the first set being the boundary points $\mathbf{p}^B$ and the second set of points the remaining internal points $\mathbf{p}^I$.

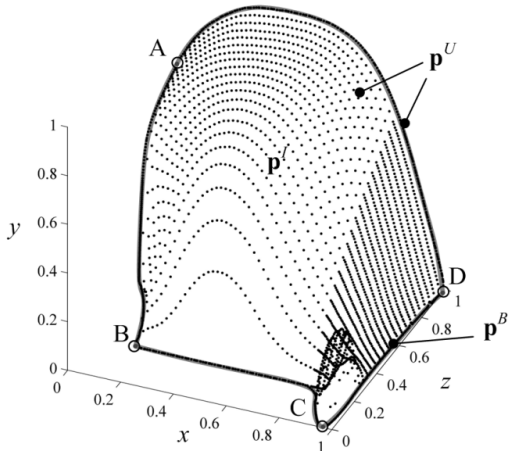


Figure 30 Point cloud illustration: internal points, boundary points and corner points.

The first step is selecting four points on the geometry boundary $\mathbf{p}^B$, illustrated as small circles A,B,C and D in Figure 30.These points will be pre-set in the corners of the square *u*-*v* parametric domain as illustrated in Figure 31. The second step is moving the rest of the $\mathbf{p}^B$ in-between the pre-set corner points, thus making a subset of projected boundary points $\mathbf{p'}^B$ as illustrated with shaded line in Figure 31. Spacing of the $\mathbf{p'}^B$ points in the *u*-*v* domain between the respective corner points is linearly correlated to the physical distances between the points.

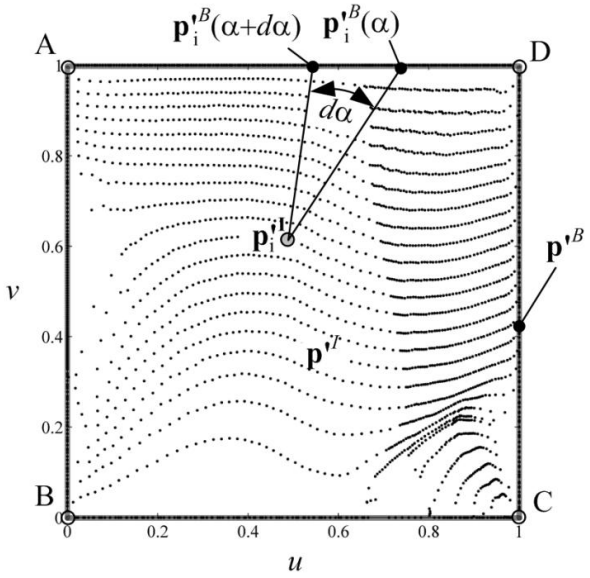


Figure 31. Point cloud projected to rectangular parametric domain.

Before the projection of the remaining (internal) points, designated $\mathbf{p'}^I$,(Figure 31) additional values and functions have to be defined as followed. First a pre-projection of the point cloud has to be conducted, for the case of a ship hull a simple *y*-axis projection is sufficient as illustrated in Figure 32. Now the parameter of angle α can be defined as the angle between the vector connecting the boundary point $\mathbf{p}_i^B$ with the internal point $\mathbf{p}_i^I$ and *u0* axis in the pre-projection domain. Now the angle values $\alpha p_{i,j}$ and distance values $dp_{i,j}$ for each combination of boundary-internal point has to be calculated, pseudo-code as follows:

for *i*=1:*nI*

for j=1:*nB*

$\alpha p_{i,j}$ = calculate_angle($\mathbf{p}_j^B, \mathbf{p}_i^I$)

$dp_{i,j}$ = calculate_distance ($\mathbf{p}_j^B, \mathbf{p}_i^I$)

end

end

where *nI* is the number of internal points and *nB* is the number of boundary points. The distances $dp_{i,j}$ are calculated on the original triangulated surface by Dijkstra's method.

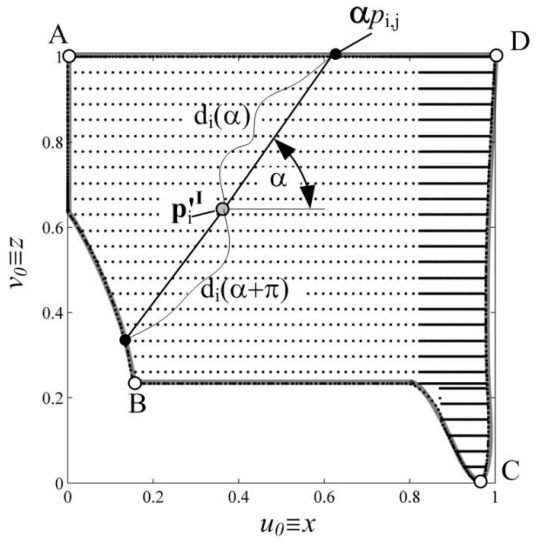


Figure 32. Definition of angle α and angle-related distance in the pre-projection domain.

The functions $\mathbf{p'}_i^B(\alpha)$ representing a function of projected boundary coordinates with respect to angle can now be created from $\alpha p_{i,j}$ and the coordinates of points $\mathbf{p'}_i^B$ by interpolation. The function $\mathbf{p'}_i^B(\alpha)$ is different for each internal point since the boundary points are at different angular positions with respect to the internal point as illustrated in Figure 31. From $\alpha p_{i,j}$ and $dp_{i,j}$, function $d_i(\alpha)$ for each internal point can also be created by interpolation. If two or more boundary points have the same angle value α, points with larger distance *d* are ignored.

The projection of each internal point $\mathbf{p'}_i^I$ is finally obtained by:

$$\mathbf{p}_i^I = \frac{\int_{-\pi}^{\pi} f_P(d_i(\alpha),\alpha)\cdot \boldsymbol{p'}_i^B(\alpha)\cdot d\alpha}{\int_{-\pi}^{\pi} f_P(d_i(\alpha),\alpha)\cdot d\alpha} \tag{44}$$

where f_P is the projection operator, selected as:

$$f_P = 1 - \frac{d_i(\alpha)}{d_i(\alpha)+d_i(\alpha+\pi)} \tag{45}$$

Other projection operators can be used, but the above one has shown the best results. The value of f_P varies from 0 to 1 depending on the ratio of closest point an angle α and angle $\alpha+\pi$ (Figure 32). If the projection is calculated only by using two angles α and $\alpha+\pi$, the projected point would be located on the line between $\boldsymbol{p'}_i^B(\alpha)$ and $\boldsymbol{p'}_i^B(\alpha+\pi)$. The point would divide the line in ratio $d_i(\alpha)/d_i(\alpha+\pi)$. When integrated over all angles α, a smooth

projection can be obtained. The denominator in (45) represents sort of normalization, an equivalent is the number of points in the case when the projection is achieved with finite number of angles α with spacing $\Delta\alpha$. After projection of unstructured points $\mathbf{p}^U$ into the square parametric domain, the required matrix topology of point cloud is obtained by simple interpolation. The fitting procedure is continued according to (42).

4.3. Feature Detection

Feature detection on unstructured point clouds is important for recognition of possible areas in which single patch (fitting) can be used when using multi-patch parameterization. For a known previous design, initial multi patch parameterization could be created to speed up the solution. During the subsequent optimization procedure, the initial multi-patch parameterization could be adopted to new features that appear during the optimization. Feature detection methods could then be applied to detect new features that are emerging during the optimization procedure to aid in re-patching.

4.3.1. Point Based Methods

Very few feature detection methods are dedicated to point-sampled geometry only [47]. The major problem of these point based methods is the lack of knowledge concerning normal and connectivity information. This means that point based methods for feature detection will invlove more challenges than mesh based methods. In [48], a method is presented that uses the computationally efficient method based on a neighbor graph connecting nearby points. The algorithm first analyzes the neighborhood of each point via a principal component analysis (PCA). The eigenvalues of the correlation matrix are then used to determine a probability of a point belonging to a feature. The analysis of the ellipsoid formed by the three eigenvectors and their eigenvalues allows further conclusions about the underlying feature type. This way the algorithm can differentiate between line-type features, border and corner points. The result is a quite dense set of points covering all kinds of features independent if the feature is sharp or not. This set of points is then reduced by computing a minimal spanning tree followed by a branch cutting. The method is applied to several examples, as illustrated in figure below, shows the extracted crease patterns of the torso, the fandisk and the bunny model.

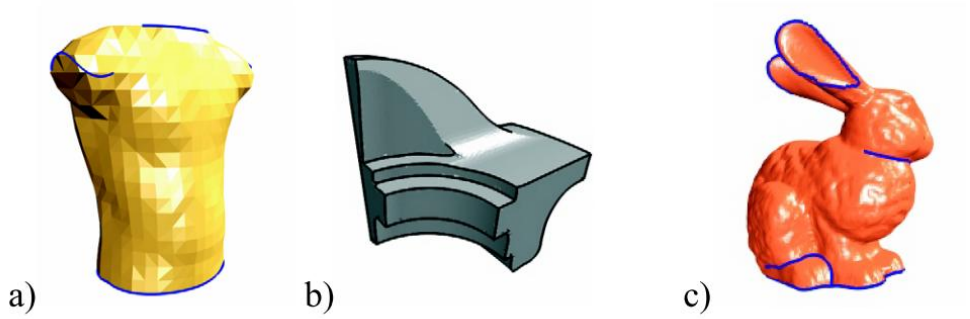


Figure 33. Crease patterns on torso, fandisk and bunny model [48].

In [49], the previous PCA approach is extended by the with a multi scale analysis of the neighborhoods. To obtain more information, a multi scaling approach that varies the size of these neighborhoods is used. That means they feature detection operator is applied to multiple neighborhood sizes, which allows to measure the persistence of the feature. This approach has shown to be superior to the single scale analysis as illustrated in the following figure but better results come with an additional computational cost.

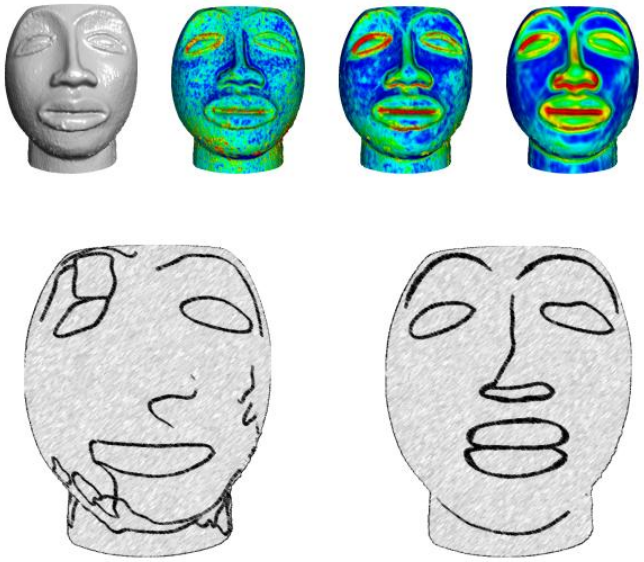


Figure 34. Multi-scale feature extraction (bottom right) is superior to single scale extraction (bottom left) on a noisy range scan. The top row shows the original point cloud and variation estimates for different scales.

4.3.2. Polygonal methods

There exist multiple techniques for feature extraction relying on polygonal mesh [47]. The following methods belong to groups of different approaches.

Hubeli and Gross [50] use a normal based multi-resolution framework and generate a set of edges with a normal-based classification operator. In a classification phase they assign a weight to every edge in the input mesh, proportional to the probability of belonging to a feature. The authors provide different types of operators for different mesh types like a "second order difference" operator for very coarse data sets, an "extended second order difference" operator for finer meshes, or a computational more expensive "best for polynomial" operator which performs well on noisy points sets. After this, in a detection phase they reconstruct the features from the information gained in the classification phase.

Hildebrand et al. [51] use anisotropic filtering on third order derivatives of the surface mesh. Approximation of the derivatives are calculated by discrete differential geometric approximations.

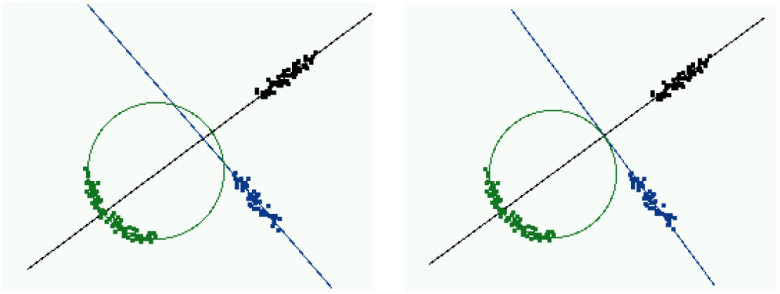
Watanabe and Belyaev [52] use the so called focal surfaces to detect curvature extrema on dense triangle meshes.

All mesh-based techniques take advantage of the connectivity information and corresponding normals associated with the underlying mesh. Often, 3D scanning devices do not directly produce a mesh as raw data, but an unsorted set of point data representing the original surface. In this case, a mesh-based method has to rely on the proper reconstruction of the features during the mesh generation.

4.4. Constrained fitting

The constraints to be satisfied may be determined manually, or by an automatic process for example feature extraction methods from previous chapter. A wide range of work considers constraints in the context of fitting a single (possibly piecewise) free-form surface. In that context the constraints are typically global requirements for the surface fitted to be positive, monotonic or convex. Applying constraints when fairing data is a similar application. Local modifications of single surfaces may use constraints on control (or other) points during user modification of the shape of the surface. Such approaches are often referred to as physics-based modelling: changes in shape of a dynamically deformable surface are controlled by a set of virtual springs attached to it, constraining its shape [53].

The following example illustrates some problems that can appear in constrained fitting. A method of prioritized constraints is used. Consider a planar example that uses four constraints, one of which is inconsistent with the other three. The highest priority constraint requires two straight lines to be orthogonal, the next highest priority constraint, in contradiction requires them to be parallel, and two further constraints require one line to pass through the center of a circle, and the other line to be tangent to it. The initial unconstrained approximation to the solution and the final state after directly enforcing constraints can be seen in Figure 35a and Figure 35b. As was expected, the contradictory second constraint was rejected, and the other three satisfied.



a) b)

Figure 35. Constrained fitting: a) unconstrained fitting b) constrained fitting after several iterations [53].

5. COMPUTATIONAL FLUID DYNAMICS

Computational fluid dynamics (CFD), is undergoing significant expansion regarding its engineering applications, number of researchers active in the field, and also the number of courses offered at universities. For solving fluid flow problems, numerous software packages exist but still the market is not quite as large as the one for structural mechanics codes. While for structural problems, the finite element methods is well established, CFD problems are, in general, more difficult to solve. However, CFD codes are slowly being accepted as design tools by industrial [54]. Even if accurate CFD modeling is possible, the issue when using CFD as part of optimization workflow is the respective computational time required for simulation.

The most simple optimization problem with CFD would be single variable, single objective optimization. In such a case, simple gradient method could be used for the optimization, or even it could be carried out "by hand", computing the full functional behavior, since this will be quite trivial. In the case when CFD simulation is ideally smooth, still multiple minima are possible, so when starting from different initial points, two different solutions would be the result of optimization as illustrated Figure 36. But often, the CFD simulation will not yield in ideally smooth curve but will have some "noise" added on the smooth function. This makes the optimization even more difficult since even when starting from near global minimum, the optimization procedure will soon end trapped in local noise-induced optimum (example illustrated by small rectangles). This illustrated that even in the simplest cases single objective case, the optimization is not as trivial as it seems. The gradient methods that are by far most efficient in optimization of smooth functions are in this case not directly applicable. Thus when incorporating engineering simulations in an optimization workflow, all components of the workflow (optimization method, shape parameterization and engineering simulation - CFD) need to be considered

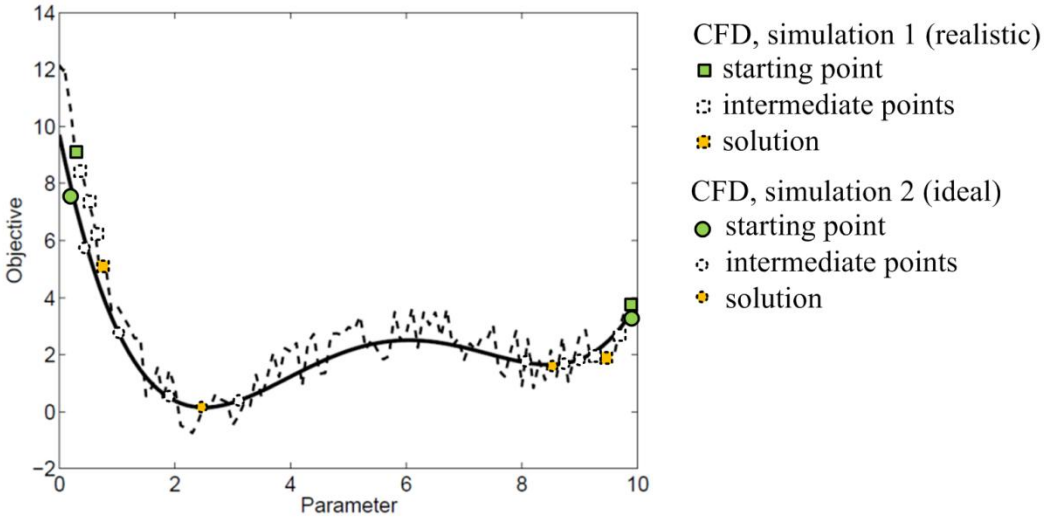


Figure 36. Schematic representation of a simple optimization problem involving a single parameter and a single objective. The objective function shows two minima. The exact

objective function is represented by the solid line while the inaccurate and more realistic case of CFD-based evaluation of the objective function is shown as a dashed line.

Real optimization problems found in engineering as well as in fundamental research will generally involve several (often too many) shape variables and several objectives. It is important know the practical limits of optimization based on CFD. The most limiting factor is the computing time requirement for single evaluation. Admittedly, each CFD evaluation with shape modification will usually lead to slightly different numerical costs but the variations are normally negligible relative to the overall computational cost. This means that computational cost depends mostly the type of simulation that is conducted, for example, on modern PC the following CFD simulations could be conducted:

- one simulation of the burner considering multispecies transport and combustion takes typically about 20 hours of computing time and 1.8 GB of computer memory;
- one simplified simulation of the turbulent channel flows requires less than 5 seconds of computing time and 10 Megabytes (MB) of computer memory;
- one transient 3D simulation of Darrieus wind turbine requires 16 Gigabytes (GB) of computer memory and about 30 days of computing time;

By looking at the computing requirement for these few examples, it is evident that computational time can vary by 5 orders of magnitude and the computer memory by 3 orders of magnitude. This is typical of what will be found in practice: simple CFD problems can be solved within seconds on a standard PC; high-end CFD problems can require weeks of computing times and Terabytes (1 TB=1000 GB) of memory on supercomputers. In such cases, the limits of optimization when using CFD are clear in principle: CFD based optimization is possible for simple CFD problems and almost impossible for high-end CFD configurations. To somewhat quantify the possibilities, considering again the earlier examples:

- only 64 evaluations corresponding to realizable configurations have been conducted in [5]. for the burner involving multispecies transport, and this already at an extremely high computational cost;
- on the other hand, more than 5000 evaluations have been easily carried out for the simplified turbulent channel flow;
- for the transient 3D simulation of Darrieus wind turbine lasting 30 days, no practical optimization could be conducted.

As a rule of thumb for engineering practice, it seems therefore appropriate to state: CFD based optimization is possible when the duration of a single CFD computation does not exceed a few hours at most [5]. Practical limits of CFD optimization are illustrated in

Table 1 for a modern PC.

Table 1. Practical limits of CFD optimization for a modern PC.

Evaluation cost	Optimization cost for ≤2 parameters	Optimization cost for ≥3 parameters
low cost (2 min CPU, 100 MB memory)	very easy	easy up to 20 parameters
medium cost (1h CPU, 1000 MB memory)	very easy	still possible
high cost (20h CPU, 10 GB memory)	still possible	almost imposible

In order to achieve improvements in optimization speed, adjoint methods explained in chapter 2.5. Adjoint system must first be identified for the considered system of equation. While this is easily done (and well documented in the literature), e.g. for the Euler equations, the task will become much more difficult when considering complex, multiphysics problems. Furthermore, the adjoint approach requires a full knowledge of the intermediate approximations on a way to the full solution of the system of equations solved by CFD. In simple words, this means that the adjoint approach, while greatly reducing the number of evaluations (and computational time), will lead to a huge increase of the requested computer memory which will again become a major problem for complex, three-dimensional flows involving many unknowns at each discretization point [5].

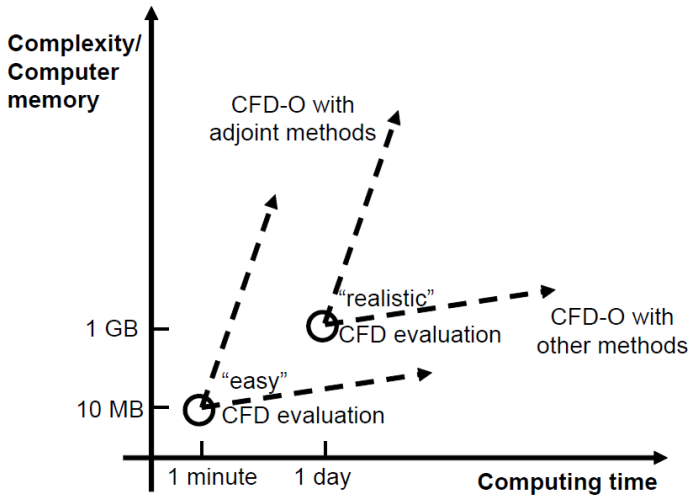


Figure 37. Principle requirements of CFD based optimization (CFD-O) in terms of computing time and computer memory. Again, the figures listed here are just orders of magnitude, and should by no means be considered as exact limits[5].

Adjoint approach (by default) used gradient optimization methods and is accompanied with all of the disadvantages of gradient methods such as finding only the nearest local minimum. For a generic optimization task with ambition to find a global solution this approach can be helpful but it cannot be used exclusively. This doctoral qualifying exam aims mostly at investigating efficient shape parameterization techniques that could be used in any engineering simulation (structural, multiphysics…) involving SO. But CFD is used since it is

widely applied in combination with SO, and it illustrates many problems accruing when using engineering simulations. The following chapters will briefly layout the basics of CFD.

5.1. Fluid flow equations

Fluids are substances whose molecular structure offers no resistance to external shear forces: even the smallest force causes deformation of a fluid particle. For most engineering applications, a fluid is regarded as a continuous substance (continuum). Flow equations can be given by multiple conservation law. For mass, which cannot be created or destroyed, the conservation equation can be written (for a control mass, CM):

$$\frac{\mathrm{D}m}{\mathrm{D}t} = 0 \tag{46}$$

where m stands for mass and t for time, $\frac{\mathrm{D}}{\mathrm{D}t}$ is the material derivative (later D will be used for rate of strain tensor). As opposed to mass the amount of change momentum is not equal to zero. Newton's second law of motion states that:

$$\frac{\mathrm{d}(m\boldsymbol{v})}{\mathrm{d}t} = \sum \boldsymbol{f} \tag{47}$$

where $\boldsymbol{v}$ is the velocity, and $\boldsymbol{f}$ is the forces acting on the control mass. Left hand side of equations (46) and (47) can be generalized for conservation of an intensive property ϕ. This generalization written as integral equation:

$$\frac{\mathrm{D}}{\mathrm{D}t}\int_{\Omega_{CM}} \rho\phi \mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_{CV}} \rho\phi \mathrm{d}\Omega + \int_{S_{CV}} \rho\phi(\boldsymbol{v}-\boldsymbol{v}_b)\cdot\boldsymbol{n}\mathrm{d}S \tag{48}$$

where Ω_{CM} stands for volume occupied by the control mass ρ is density , Ω_{CV} is the control volume (CV) volume, S_{CV} is the surface enclosing CV, n is the unit vector orthogonal to S_{CV} and directed outwards, $\boldsymbol{v}$ is the fluid velocity and $\boldsymbol{v}_b$ is the velocity with which the CV surface is moving. For mass conservation $\phi = 1$, for momentum conservation $\phi = \boldsymbol{v}$. For common case of a CV fixed in space, $\boldsymbol{v}_b = 0$ and the first derivative on the right hand side becomes a local (partial) derivative. This equation states that the change of the amount of property in the control mass Ω_{CM} is the rate of change of the property within the control volume plus the flux of it through the CV boundary caused by the fluid motion relative to CV boundary. The flux is called the convective flux of $\rho\phi$ through the CV boundary. The integral form of the mass conservation (continuity) equation in case of fixed control volume follows directly from (48) by setting $\phi = 1$:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \mathrm{d}\Omega + \int_{S} \rho\boldsymbol{v}\cdot\boldsymbol{n}\mathrm{d}S = 0 \tag{49}$$

Again for a fixed control volume, the momentum conservation equation is obtained by setting $\phi = \boldsymbol{v}$ and combining with (47). The resulting equation is:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \boldsymbol{v} \mathrm{d}\Omega + \int_{S} \rho \boldsymbol{v}\boldsymbol{v} \cdot \boldsymbol{n} \mathrm{d}S = \sum \boldsymbol{f} \tag{50}$$

Continuity equations can also be written in coordinate-free differential form by application of Gauss divergence theorem to the second term and reducing the volume to infinitesimally small. To express the right hand side in terms of intensive properties, surface forces and body forces have to be considered.

The surface forces are from the molecular point of view, the microscopic momentum fluxes across a surface, macroscopically represented by pressure and stresses. If these fluxes cannot be expressed by density and velocity, the system of equations is not closed since there would be fewer equations than dependent variables. By making assumptions that the fluid is Newtonian (it is in most practical engineering applications), a closed system of equations can be assured. For Newtonian fluids, the stress tensor T, which is the molecular rate of transport of momentum, can be written (using index notation):

$$\mathrm{T}_{ij} = -\left(p + \frac{2}{3}\mu \frac{\partial u_j}{\partial x_j} \right) \delta \mathrm{I}_{ij} + 2\mu \mathrm{D}_{ij} \tag{51}$$

where, μ is the dynamic viscosity, I is the unit tensor, p is the static pressure and D is the rate of strain (deformation) tensor:

$$\mathrm{D}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \tag{52}$$

The body forces per unit mass can be represented by force vector $\boldsymbol{b}$, so the integral form of the momentum conservation equation now becomes:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho \boldsymbol{v} \mathrm{d}\Omega + \int_{S} \rho \boldsymbol{v}\boldsymbol{v} \cdot \boldsymbol{n} \mathrm{d}S = \int_{S} \mathrm{T} \cdot \boldsymbol{n} \mathrm{d}S + \int_{\Omega} \rho \boldsymbol{b} \mathrm{d}\Omega \tag{53}$$

The equation (53) with (51) and (52) included is the most general form of momentum conservation equation for Newtonian fluids. All fluid properties (density,…) can vary both in space and in time. However in many applications the fluid density is practically constant. Constant density may be assumed for most liquid flows, but also for gases in case of low Mach number (<0.3). Such flows are called to be incompressible. However this simplification is generally not of a great value, as the equations are similar to the original ones regarding difficulty of obtaining a solution. However, assuming constant density and viscosity does help in numerical solution and it should be implemented if the problem allows it. In addition to simplification to incompressible flow, various other simplifications are used depending on the conducted simulation. It is important to know what simplifications can reasonably be used to lower the computational time but still keep sufficient simulation physics so that the correct optimal solution can be obtained in a reasonable computational time.

5.2. Turbulence models

Most flows encountered in engineering application are turbulent. Turbulent flow main characteristics are the following:

- Turbulent flows are very unsteady. Velocity magnitude plotted as a function of time appears almost random;
- Turbulence is a three-dimensional phenomenon. While the time-averaged velocity at a point may be a function of only two coordinates, turbulent flow has fluctuations in all three spatial dimensions;
- Turbulence increases the "mixing" rate of the conserved quantities. This means that fluid areas of differing momentum content are also mixed. While the actual mixing is achieved by (viscous) diffusion, this process is called turbulent diffusion.
- Turbulent flows require simulation of a broad range of length and time scales. The wide range of turbulent flows length and time scales is numerically very demanding.

The most accurate approach to turbulence simulation is to directly solve the governing equations (49) and (53) (Navier-Stokes equations). This is the simplest approach from the conceptual point of view. In such simulations, all of the motions contained in the flow are resolved. The obtained flow field is equivalent to a single laboratory experiment, this approach is called direct numerical simulation (DNS). In engineering applications, usually just a few quantitative properties of a turbulent flow, such as the forces on a body (or torque) are of practical interest. Furthermore, only the time average of the quantitative properties is sufficient in most cases. Using the DNS to compute these quantities is not practical and even not possible with today's computers. Even a single simulation of the simplest practical flows such as flow about airfoil (thin) section requires hours of computational time on modern supercomputers [55]. Application of DNS for practical optimization is thus not an option.

Based on ideas proposed by Osborne Reynolds over a century ago, so called Reynolds-averaging method was developed. In a statistically steady flow, every variable can be written as time-averaged value plus a fluctuation about that value:

$$\phi(x_i,t) = \bar{\phi}(x_i) + \phi'(x_i,t) \tag{54}$$

where

$$\bar{\phi}(x_i) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \phi(x_i,t)dt \tag{55}$$

Here t is the time and T is the averaging interval. This interval must be large compared to the typical time scale of the fluctuations i.e. $T \to \infty$ If T is large enough, $\bar{\phi}$ does not depend on time at which the averaging is started. If the flow is unsteady, time averaging cannot be used and it must be replaced by ensemble averaging (Figure 38):

$$\bar{\phi}(x_i) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \phi(x_i,t) \tag{56}$$

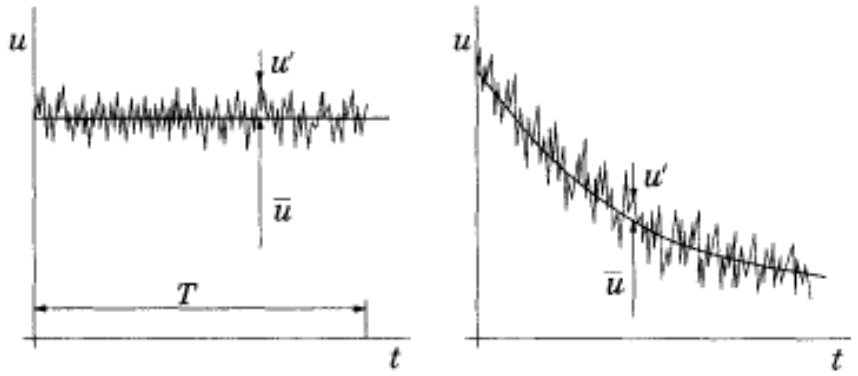


Figure 38. Time averaging for statistically steady flow and ensamble averaging for unsteady flow [54].

Since $\overline{\phi'} = 0$, averaging any linear term in the conservation equations simply gives the identical term for the averaged quantity. From a quadratic nonlinear term two terms are obtained, the product of the average and a covariance:

$$\overline{u_i\phi} = \overline{(\bar{u}_i + u_i')(\bar{\phi} + \phi')} = \bar{u}_i\bar{\phi} + \overline{u_i'\phi'} \tag{57}$$

The last term is zero only if the two quantities are uncorrelated but this is rarely the case in turbulent flows. So the conservation equations contain terms such as $\rho\overline{u_i'u_j'}$ called the *Reynolds stresses*, and $\rho\overline{u_i'\phi'}$, known as the *turbulent scalar flux*, among others. These cannot be represented uniquely in terms of the mean quantities. The averaged continuity and momentum equations in the case of incompressible flows (and body forces excluded), can be written in index notation and Cartesian coordinates as:

$$\frac{\partial\rho\bar{u}_i}{\partial x_i} = 0 \tag{58}$$

$$\frac{\partial\rho\bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\bar{u}_i\bar{u}_j + \rho\overline{u_i'u_j'}\right) = -\frac{\partial\bar{p}}{\partial x_i} + \frac{\partial\bar{\tau}_{ij}}{\partial x_j} \tag{59}$$

where the $\bar{\tau}_{ij}$ are the time-averaged (mean) viscous stress tensor components:

$$\bar{\tau}_{ij} = \mu\left(\frac{\partial\bar{u}_i}{\partial x_j} + \frac{\partial\bar{u}_j}{\partial x_j}\right) \tag{60}$$

Finally the equation for the mean of a scalar quantity ϕ can be written:

$$\frac{\partial\rho\bar{\phi}}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\bar{u}_i\bar{\phi} + \rho\overline{u_j'\phi}\right) = \frac{\partial}{\partial x_j}\left(\Gamma\frac{\partial\bar{\phi}}{\partial x_j}\right) \tag{61}$$

The presence of new terms in the conservation equations namely Reynolds stresses and turbulent scalar flux, means that the equations are not closed i.e. they contain more variables than there are equations. Closure requires use of some approximations, which usually take the form of prescribing the Reynolds stress tensor and turbulent scalar fluxes in terms of the mean flow quantities. Equations for the higher order correlations have also been derived, for example for the Reynolds stress tensor, but these contain still more (and higher-order) unknown correlations that require modeling approximations. In any case it is impossible to derive a closed set of exact equations. To close the system of equations turbulence models are introduced. A reasonable model is obtained by noting that in laminar flows, energy dissipation and transport of mass, momentum, and energy normal to the streamlines are mediated by the viscosity. So it is natural to assume that the effect of turbulence can be represented as an increased viscosity [54]. This leads to the model for the Reynolds stress according to Boussinesq eddy-viscosity assumption, R_{ij}:

$$R_{ij} = -\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial\overline{u_i}}{\partial x_j} + \frac{\partial\overline{u_j}}{\partial x_i} - \frac{2}{3}\frac{\partial\overline{u_k}}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho\delta_{ij}k \tag{62}$$

and the eddy-diffusion model for a scalar:

$$-\rho\overline{u_j'\phi'} = \Gamma_t\frac{\partial\overline{\phi}}{\partial x_j} \tag{63}$$

In (62), *k* is the turbulence kinetic energy:

$$k = \frac{1}{2}\overline{u_i'u_i'} = \frac{1}{2}\left(\overline{u_x'u_x'} + \overline{u_y'u_y'} + \overline{u_z'u_z'}\right) \tag{64}$$

Although the eddy-viscosity hypothesis is not correct exactly, it is widely used, easy to implement and its application has shown reasonably good results for many flows. In the simplest description, turbulence can be characterized by two parameters: its kinetic energy, *k*, or a velocity, $q = \sqrt{2k}$, and a length scale, *L*. Dimensional analysis shows that:

$$\mu_t = C_\mu\rho qL \tag{65}$$

where C_μ, is a dimensionless constant whose value is determined by the turbulence model. In the simplest practical models, mixing-length models, *k* is determined from the mean velocity field using the approximation $q = L\partial u/\partial y$ and *L* is a prescribed function of the coordinates (usually dependent on distance to the nearest wall). Problem is that accurate prescription of *L* is not possible for complex 3D flows. Thus mixing-length models can be applied only to relatively simple flows. These models are also known as zero-equation models.

Two-equation turbulence models are often used for numerical modeling of turbulent flow since they are robust, have shown good results and in numerical terms only modestly expand the system of governing equations.

5.2.1. RANS equations

The continuity and momentum equations in Reynolds averaged differential form can be summarized, in tensor notation and Cartesian coordinates as:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho \bar{u}_j)}{\partial x_j} = 0 \tag{66}$$

$$\frac{\partial \rho \bar{u}_i}{\partial t} + \frac{\partial \rho \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j} \mu \left[\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial \bar{u}_k}{\partial x_k} \delta_{ij} \right] + \frac{\partial}{\partial x_j} \left[R_{ij} \right] + \rho f_i \tag{67}$$

$$R_{ij} = -\rho \overline{u_i u_j} = \mu_t \left[\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \frac{2}{3} \frac{\partial \bar{u}_k}{\partial x_k} \delta_{ij} \right] - \frac{2}{3} \rho k \delta_{ij} \tag{68}$$

where ρ, p and $\bar{u}_i$ are the mean flow density, pressure and the Cartesian velocity components respectively: x_i is the Cartesian coordinate; μ is the molecular viscosity; The specific body force f_i includes external body forces such as gravity in this case. Turbulent viscosity μ_t and turbulence kinetic energy k are obtained from a selected turbulence model. The continuity and momentum equations in some cases have to be solved together with energy conservation equation:

$$\frac{\partial \rho e}{\partial t} + \frac{\partial}{\partial x_j} \left[(\rho e + p) \overline{u_j} \right] = \frac{\partial}{\partial x_j} \left[(k + k_t) \frac{\partial T}{\partial x_j} \right] + S_h \tag{69}$$

Where h is enthalpy; k thermal conductivity; k_t is the thermal conductivity due to turbulence and is defined by turbulence model; T is the temperature ; S_h is volumetric heat source; and e (specific total internal energy) can defined by:

$$e = h - \frac{p}{\rho} + \frac{\bar{u}^2}{2} \tag{70}$$

5.3. Computational domain discretization

Most often in CFD, the finite volume method is implemented to discretize the computational domain. It uses the integral form of the conservation equation as the starting point:

$$\int_S \rho \phi \boldsymbol{v} \cdot \boldsymbol{n} \mathrm{dS} = \int_S \Gamma grad(\phi) \cdot \boldsymbol{n} \mathrm{dS} + \int_\Omega q_\phi \mathrm{d}\Omega \tag{71}$$

The usual approach is to define control volumes (CVs) by a suitable grid and assign the computational node to the CV center, see Figure 39. Nodes on which boundary conditions are applied are shown as full circles in this figure. The derivatives that appear in conservation equations are approximated by finite differences. Domain discretization methods are very important since computational time can be reduced by an order of magnitude while keeping

good computational physics if appropriate methods are used. Also, mesh generation can take a considerable amount of time of overall evaluation of a shape generated by the optimizer.

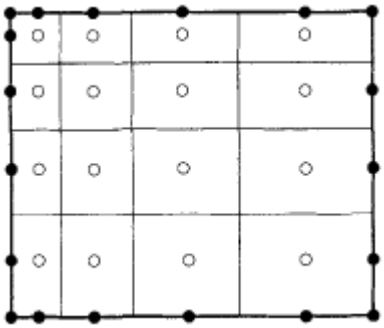


Figure 39. Finite volume schematic representing nodes and faces.

5.3.1. Isogeometric analysis in CFD

While the isogeometric methods (IGA) are more and more applied in structural problems, their application in CFD has shown to be more challenging [43]. This is in part due to the wide range of scales present in such problems, and also to the fact that these scales frequently interact with each other in complex ways. Failure to properly represent these interactions can result in inaccurate and/or unstable calculations. The keys to success when performing computational fluids analysis are accuracy and robustness. These attributes may or may not be possessed by the methods and functions used to approximate solutions. NURBS are functions that satisfy both of these criteria and seem to be an ideal basis for fluid mechanical applications. Incompressible turbulence is a highly nonlinear problem, and it requires solving of extremely wide range of scales. Success in capturing the character of the solution relies on two key components: a basis capable of accurately representing both large and small scales and a formulation that encapsulates the effect of the scales that are simply beyond reach. NURBS based isogeometric analysis, paired with the variational multiscale method explained in shortly in [43] provides both. Still, IGA is far from widespread application as FV methods are very well developed and still show superior performance. Nevertheless IGA is of great importance for this doctoral qualifying exam since large improvement capacity exists in integrating the overall procedure of shape parameterization, numerical simulation and optimization methods.

IGA is successfully used for estimating ship resistance problem by boundary element momentum method (Neumann-Kelvin problem) and it is considered that it is applicable for systematic calculations involved in hull optimization problems [56]. It was shown as illustrated in Figure 40. that a good agreement with experimental data is obtained.

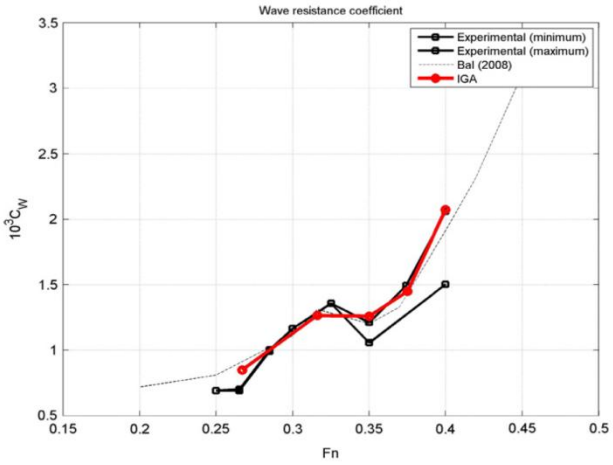


Figure 40. Wave resistance coefficient C_W of the Wigley hull forvarious Froude numbers, as calculated by the IGA tmethod (redbullets). Comparison with experimental data (blacksquares) and an other panel method (thin dashed curve) [56].

Also IGA is appropriate for fluid structure interaction problems, for example used on windturbines for 2D problem [57]. Another interesting example is application of IGA (in structural applications) in combination with gradient based optimization with application of sensitivity analysis [58].

5.3.2. Solution of discretized equations

In the previous chapters IGA and FV methods were considered as means of computational domain discretization. In either case, the result of the discretization process is a system of algebraic equations, which are linear or non-linear according to the nature of the partial differential equations from which they are derived. In the non-linear case, the discretized equations must be solved by an iterative technique that involves guessing a solution, linearizing the equations about that solution, and improving the solution; the process is repeated until a converged result is obtained. So, whether the equations are linear or not, efficient methods for solving the linear systems of algebraic equations are needed. The selection solving method is an important step, since order magnitude in computational time can be lost if inappropriate method is chosen. Various guidelines (for example ANSYS Fluent help [59]) exist but the appropriate method is problem dependent.

After the solution is obtained the last step is post-processing of the flow field. This usually includes integrating surface forces (or forces*distance to obtain torque) on boundaries that represent the object of optimization. This value is finally sent to the optimizer and the whole process is repeated till the convergence criteria are met.

6. CHALLENGES AND FUTURE WORK

Engineering shape optimization with integrated numerical simulations is a demanding multidisciplinary problem and various methods for conducting the optimization procedure exist. While various different numerical simulations could be required, this doctoral

qualification exam considers CFD because of its wide usage in combination with shape optimization. Also, CFD involves problems such as time-consuming simulations and non-linear governing equations, making it a representative of common engineering numerical simulation. To solve such problems, computational efficiency and global optimization methods are required.

After a review of optimization methods, two approaches could be highlighted: gradient-based adjoint method, appealing because of its computational efficiency; and an integrated numerical workflow based on genetic-algorithm (GA), attractive since it offers a generic approach to wide array of optimization problems. While the adjoint method offers a promising solution with fast optimization times, it has several disadvantages. First, it is not suitable for global optimization since it is by nature gradient-based. Since the objective function(s) and constraints are highly nonlinear, it is unlikely that a global optimum will be reached from an arbitrarily selected initial solution. Engineering optimization can also contain discrete variables which cannot be solved by classical gradient methods. Furthermore, engineering optimization problems usually involve several objectives, making them suitable for GA based methods. A generic engineering shape optimization task can be solved by linking GA with shape parameterization methods, and (multiple) engineering simulation nodes in an integrated numerical workflow. Nevertheless, fast gradient based methods could eventually be implemented when the GA obtains a solution close to the global optimum.

When using GA, each generated shape needs to be subject to time consuming CFD simulation. One of the major concerns in this kind of optimization is that large number of simulations requires great computational effort. So in order to setup numerical workflow efficiently, one of the most important parts of the optimization problem is selecting a suitable parameterization method. If parameterization is selected such that the respective geometry can be described with small number of parameters, number of necessary simulations can be reduced to lower computational effort towards practical realization of optimization procedures.

This doctoral qualification exam gives an overview of shape parameterization method that could be implemented in the numerical optimization workflow. The most promising are the Bezier family of parametric surfaces, B-spline, NURBS and T-splines. They have favorable properties of local control, abilities for reproducing sharp edges. Additionally, possibility of multi-patch parameterization enables reduction of control points in “smooth” areas of shape and increasing the shape resolution in areas of large geometry changes. This gives a possibility for reduction in the number of variables. A review of current research papers does not provide for a universal method and leaves room for further improvements. Parametric shape fitting methods applied on existing shapes can be used as a method for evaluating individual shape parameterization methods. This can also be possible conducted even during the optimization which allows for switching between different parameterization methods. Finally an introduction to CFD and review of optimization methods applied with CFD is given. Another possible improvement can be obtained by unifying parameterization methods with simulation by isogeometric analysis.

Expected scientific contributions of doctoral thesis are:

- Reduction of the number of shape variables while keeping the shape generality in an engineering optimization task i.e. development of a more efficient parameterization methods;
- Development of adaptive parameterization methods for switching between different shape parameterizations during the optimization procedure;
- Development of variable length chromosomes for adaptive multi-patch parameterization to enable smooth transitions between shapes during the optimization procedure,

REFERENCES

[1] O. P. Bijan Mohammadi, *Applied shape optimization for fluids*, 2ed. ed. OUP, 2010.

[2] J. Haslinger and R. a E. Mäkinen, *Introduction to Shape Optimization*, vol. 7. Society for Industrial and Applied Mathematics, 2003.

[3] J. C. De los Reyes, *Numerical PDE-Constrained Optimization*. Cham: Springer International Publishing, 2015.

[4] L. Blank, M. H. Farshbaf-shaker, H. Garcke, C. Rupprecht, and V. Styles, *Trends in PDE Constrained Optimization*, vol. 165. 2014.

[5] D. Thévenin and G. Janiga, *Optimization and Computational Fluid Dynamics*, vol. XVI. 2008.

[6] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidiscip. Optim.*, vol. 26, no. 6, pp. 369–395, 2004.

[7] W. Swann, "A survey of non-linear optimization techniques," *FEBS Lett.*, vol. 2, no. March, pp. S39–S55, 1969.

[8] S. Panda and N. P. Padhy, "Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 1418–1427, 2008.

[9] M. Y. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Comput. Methods Appl. Mech. Eng.*, vol. 192, no. 1–2, pp. 227–246, 2003.

[10] I. Y. Kim and O. L. de Weck, "Variable chromosome length genetic algorithm for progressive refinement in topology optimization," *Struct. Multidiscip. Optim.*, vol. 29, no. 6, pp. 445–456, Jun. 2005.

[11] H. Stringer, "Behavior of variable-length genetic algorithms under random selection," University of Florida, 2007.

[12] D. a. Tortorelli and P. Michaleris, "Design sensitivity analysis: Overview and review," *Inverse Probl. Sci. Eng.*, vol. 1, no. 1, pp. 71–105, 1994.

[13] M. B. Giles and N. A. Pierce, "An introduction to the adjoint approach to design," *Flow, Turbul. Combust.*, vol. 65, no. 3–4, pp. 393–415, 2000.

[14] O. Pironneau, "On optimum design in fluid mechanics," *J. Fluid Mech.*, vol. 64, no. 01, p. 97, Jun. 1974.

[15] G. Allaire, "A review of adjoint methods for sensitivity analysis , uncertainty quantification and optimization in numerical codes To cite this version : A review of adjoint methods for sensitivity analysis , uncertainty quantification and optimization in numerical co," 2015.

[16] G. Allaire, F. Jouve, and A.-M. Toader, "Structural optimization using sensitivity analysis and a level-set method," *J. Comput. Phys.*, vol. 194, no. 1, pp. 363–393, Feb. 2004.

[17] Z.-H. Han and K.-S. Zhang, "Surrogate-Based Optimization," in *Real-World Applications of Genetic Algorithms*, InTech, 2012.

[18] I. Marinić-Kragić, D. Vučina, and Z. Milas, "3D shape optimization of fan vanes for multiple operating regimes subject to efficiency and noise-related excellence criteria and constraints," *Eng. Appl. Comput. Fluid Mech.*, vol. 10, no. 1, pp. 210–228, 2016.

[19] Z. Milas, D. Vučina, and I. Marinić-Kragić, "Multi-regime shape optimization of fan vanes for energy conversion efficiency using CFD, 3D optical scanning and

parameterization," *Eng. Appl. Comput. Fluid Mech.*, vol. 8, no. 3, pp. 407–421, 2014.
[20] Esteco s.p.a, "modeFRONTIER® (computer program)." Esteco s.p.a, 2014.
[21] S. Nadarajah, P. Castonguay, and A. Mousavi, "Survey of Shape Parameterization Techniques and its Effect on Three-Dimensional Aerodynamic Shape Optimization," in *18th AIAA Computational Fluid Dynamics Conference*, 2007, no. June, pp. 1–23.
[22] J. A. Samareh, "Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA J.*, vol. 39, no. 5, pp. 877–884, May 2001.
[23] D. Vucina, Z. Lozina, and I. Pehnec, "Computational procedure for optimum shape design based on chained Bezier surfaces parameterization," *Eng. Appl. Artif. Intell.*, vol. 25, no. 3, pp. 648–667, Apr. 2012.
[24] L. Piegl and W. Tiller, *The NURBS Book*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.
[25] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and T-NURCCs," *ACM Trans. Graph.*, vol. 22, no. 3, p. 477, Jul. 2003.
[26] A. M. Abbas, "Survey of Subdivision Controls for Interpolating Curves and Surfaces," *Int. J. Softw. Informatics*, vol. 7, no. 1, pp. 113–132, 2013.
[27] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," *Proc. 28th Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 67–76, 2001.
[28] M. G. Cox, "The incorporation of boundary conditions in spline approximation problems," in *Numerical Analysis*, G. A. Watson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 51–63.
[29] J.-P. Kruth and a. Kerstens, "Reverse engineering modelling of free-form surfaces from point clouds subject to boundary conditions," *J. Mater. Process. Technol.*, vol. 76, pp. 120–127, 1998.
[30] J. Y. Lai and W. D. Ueng, "G2 continuity for multiple surfaces fitting," *Int. J. Adv. Manuf. Technol.*, vol. 17, no. 8, pp. 575–585, 2001.
[31] C. K. Chui, M. Lai, and J. Lian, "Algorithms for G 1 connection of multiple parametric bicubic NURBS surfaces ," vol. 23, pp. 285–313, 2000.
[32] K. Michálková and B. Bastl, "Imposing angle boundary conditions on B-spline/NURBS surfaces," *Comput. Des.*, vol. 62, pp. 1–9, May 2015.
[33] P. Benko, G. Kos, and T. Varady, "Constrained Fitting in Reverse Engineering," *Comput. Aided Geom. Des.*, vol. 19, no. 3, pp. 173–205, 2002.
[34] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, vol. 1. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
[35] R. E. Barnhill and S. N. Kersey, "A marching method for parametric surface/surface intersection," *Comput. Aided Geom. Des.*, vol. 7, no. 1–4, pp. 257–280, 1990.
[36] R. BARNHILL, "Geometry processing," in *Aircraft Design, Systems and Operations Meeting*, 1987.
[37] J. Vida, R. R. Martin, and T. Varady, "A survey of blending methods that use parametric surfaces," *Comput. Des.*, vol. 26, no. 5, pp. 341–365, May 1994.
[38] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.
[39] J. Cheng and X. S. Gao, "Constructing Blending Surfaces for Two Arbitrary Surfaces," *J. Eng. Graph.*, vol. 1, no. 22, p. 8, 2005.
[40] E. Hartmann, "Parametric Gn blending of curves and surfaces," *Vis. Comput.*, vol. 17, no. 1, pp. 1–13, 2001.

[41] H. Lin, Y. Xiong, and H. Liao, “Semi-structured B-spline for blending two B-spline surfaces,” *Comput. Math. with Appl.*, vol. 68, no. 7, pp. 706–718, 2014.
[42] K. Le Shi, J. H. Yong, J. G. Sun, and J. C. Paul, “Gn blending multiple surfaces in polar coordinates,” *CAD Comput. Aided Des.*, vol. 42, no. 6, pp. 479–494, 2010.
[43] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs, *Isogeometric Analysis*. Chichester, UK, UK: John Wiley & Sons, Ltd, 2009.
[44] M. Ćurković, “Parameterization of 3D Objects for Numerical Analysis and Optimization of Shape and Topology,” University of Split, 2014.
[45] M. Ćurković and D. Vučina, “3D shape acquisition and integral compact representation using optical scanning and enhanced shape parameterization,” *Adv. Eng. Informatics*, vol. 28, no. 2, pp. 111–126, 2014.
[46] G. Becker, M. Schäfer, and A. Jameson, “An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations,” *49th AIAA Aerosp. Sci. Meet. Incl. New Horizons Forum Aerosp. Expo.*, no. January, 2011.
[47] C. Weber, S. Hahmann, and H. Hagen, “Methods for Feature Detection in Point Clouds,” *Vis. Large Unstructured Data Sets IRTG Work.*, vol. 19, pp. 90–99, 2010.
[48] S. Gumhold, X. Wang, and R. Macleod, “Feature Extraction from Point Clouds,” in *In Proceedings of the 10 th International Meshing Roundtable*, 2001, pp. 293–305.
[49] M. Pauly, R. Keiser, M. Gross, and E. Zürich, “Multi-scale Feature Extraction on Point-sampled Surfaces.” 2003.
[50] A. Hubeli and M. Gross, “Multiresolution Feature Extraction for Unstructured Meshes,” in *Proceedings of the Conference on Visualization ’01*, 2001, pp. 287–294.
[51] K. Hildebrandt, K. Polthier, and M. Wardetzky, “Smooth Feature Lines on Surface Meshes,” in *Proceedings of the Third Eurographics Symposium on Geometry Processing*, 2005.
[52] K. Watanabe and A. G. Belyaev, “Detection of Salient Curvature Features on Polygonal Surfaces,” *Comput. Graph. Forum*, vol. 20, no. 3, pp. 385–392, 2001.
[53] P. Benko, G. Kos, and T. Varady, “Constrained Fitting in Reverse Engineering,” *Comput. Aided Geom. Des.*, vol. 19, no. 3, pp. 173–205, 2002.
[54] J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*. Springer, 2002.
[55] H. Shan, L. Jiang, and C. Liu, “Direct numerical simulation of flow separation around a NACA 0012 airfoil,” *Comput. Fluids*, vol. 34, no. 9, pp. 1096–1114, 2005.
[56] K. A. Belibassakis, T. P. Gerostathis, K. V. Kostas, C. G. Politis, P. D. Kaklis, A. I. Ginnis, and C. Feurer, “A BEM-isogeometric method for the ship wave-resistance problem,” *Ocean Eng.*, vol. 60, pp. 53–67, 2013.
[57] T. Van Opstal, E. Fonn, R. Holdahl, T. Kvamsdal, and A. Morten, *Isogeometric methods for CFD and FSI-simulation of flow around turbine blades*, vol. 80, no. 1876. Elsevier B.V., 2015.
[58] X. Qian, “Full analytical sensitivities in NURBS based isogeometric shape optimization,” *Comput. Methods Appl. Mech. Eng.*, vol. 199, no. 29–32, pp. 2059–2071, 2010.
[59] ANSYS © Inc., “ANSYS help.” Ansys Inc., 2016.