Structural Optimization

Prof. Dr. Ing. Kai-Uwe Bletzinger

Lehrstuhl für Statik
Technische Universität München

Winter 2020/2021

Technische Universität München
Lehrstuhl für Statik
Arcisstr. 21
D-80333 München
October 29, 2020

Contents

1 Introduction

It is common practice that all text books on structural optimization start with a philosophical excursion about the nature of optimization and its relevance for human life. We will not do that here again. It is, however, accepted that the design procedure of any technical product is done in cycles which often may take years with the intention to improve the quality of the product. It is also a matter of philosophy to discuss whether by this procedure an everlasting optimal state may be reached or, if so, may be maintained. Common sense immediately advises to negate this question. Even more, the opposite of optimal seems to be a more realistic scenario of life: what can go wrong certainly goes wrong.

So much about philosophy. On the other hand, however, many years of active work in the field of structural optimization consolidates the experience that almost everybody has his or her own idea about what optimization is about. Even scientific discussions suffer from this effect, often ending up in a demagogic controversy about what personally can be accepted as optimal. This is frustrating and serves many people as major argument against optimization. It is, however, not scientific.

A first summary of what has been said so far states: an optimal state can usually be obtained only at a certain time and is defined within a personal context. The abstraction or modeling of the optimization problem decides about the result. This is trivial but cannot be emphasized strongly enough. It is important to realize that an optimal solution is found only within the chosen context. We can think of a design window which cuts out a little well defined scenario from the complex reality. To adjust this window with respect to the characteristics of the real problem whilst deciding about allowable simplifications is the challenging engineering task. That means that the results obtained within a chosen design window always have to be checked against other scenarios, Fig. 1.1.

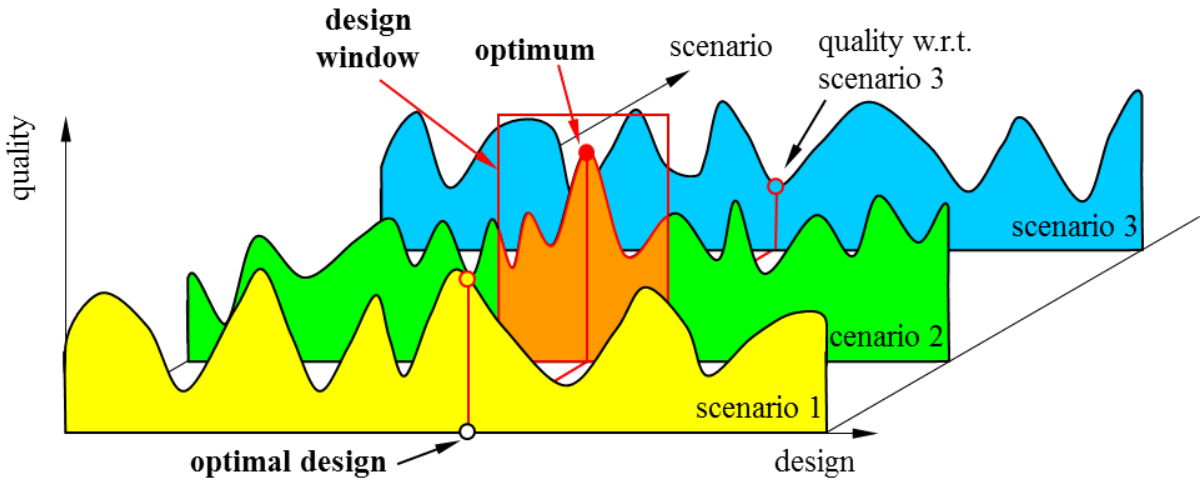


Fig.1.1: Choice of design window and “optimal design”.

To give an example, let us think of an optimal bridge design. *Design objectives* as means to measure the design quality can be e.g.: minimal construction cost, minimum life cycle cost, minimum weight, maximum stiffness or many others. Typically, the design is limited by *constraints*, e.g. the choice of material, feasible strength, displacements, eigen-frequencies, load cases, support conditions, and technical constraints like e.g. the kind and sizes of available structural members and cross sections etc. And, one has to decide about what is allowed to be modified during optimization. That is the selection of *optimization* or *design variables*.

Within the context of structural optimization we distinguish between *sizing*, *shape optimal design*, and *topology optimization* with respect to the increasing complexity of the optimization task, Fig. 1.2.

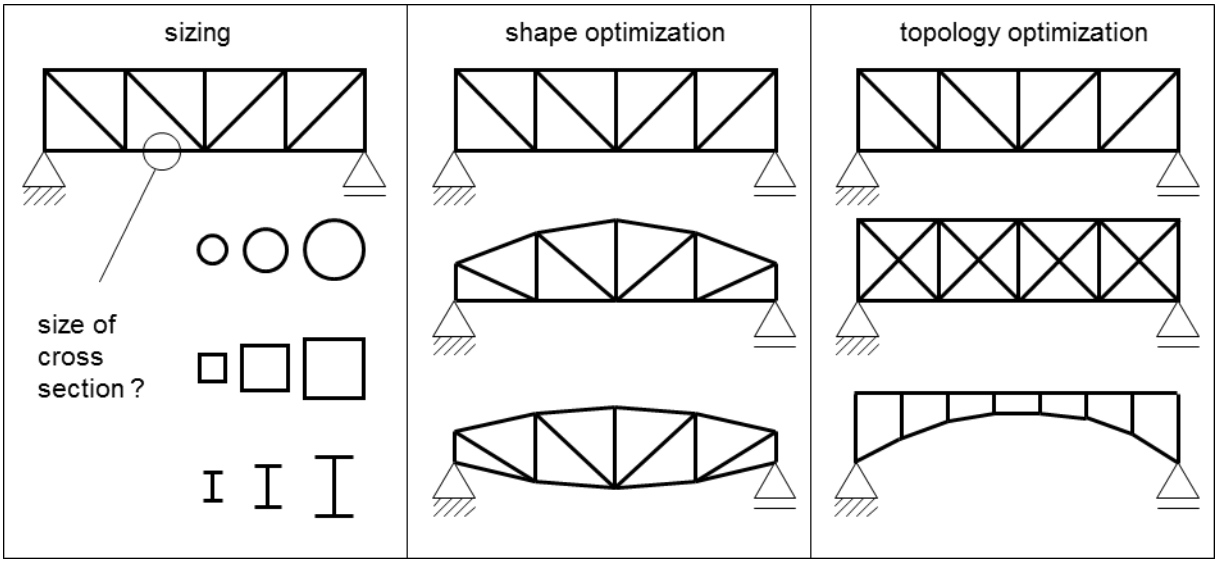


Fig. 1.2: Levels of structural optimization

While one has decided about the shape of the structure and the type of cross sections, sizing is concerned with the determination of cross section size. Additionally, in shape optimization the geometry of the structure has to be determined, whereas the connectivity of structural elements is given. In topology optimization a selection from various structural types can be made. In general, topology optimization is a question of combinatorial optimization. Since the effort to solve these questions is enormous, topology optimization is restricted in practice to problems which can be efficiently solved like the optimal member assembly in truss design or to the design of optimal numbers and positions of holes in continuous 2D and 3D structures. The latter has gained important industrial relevance during the last years.

An important issue of structural optimization is the questions of simulation, numerical modeling, and the model interactions of various disciplines which are concerned with the overall solution of the problem. Since optimization means quantitative comparison to choose the best solution amongst others, simulation is an important part of the game. Mathematical optimization theory is the backbone of structural optimization. Prerequisites of structural optimization are therefore:

- the whole problem must be formulated by mathematical means
  (i.e., objectives like aesthetic appearance have to be replaced by others as e.g. minimum weight or minimum energy consumption, etc.)
- appropriate algorithms for all concerned disciplines
- knowledge in numerical methods, mathematics and mechanics, optimization theory, FEM

Again, looking at the bridge example we realize that several different models are concerned. For construction and layout CAD or CAGD (Computer Aided Geometrical Design) models are used. To determine the structural response a finite element model is used and since we are

interested in the dynamical behavior of the bridge a further model is used to simulate the dynamics of the wind flow around the structure. The optimization itself is in general driven by nonlinear programming methods or for special cases by problem related procedures. As already stated, the optimum is determined within the chosen design window, this window is also affected by the chosen simulation models, their interactions, and their deficiencies.

Let us look at the general outline of an optimization procedure. It consists of two major steps:

- problem formulation
- problem solution

which can be subdivided as

	Project	Customers general definition of the problem.	problem formulation
	Design window	Set up of appropriate scenarios, selection of a design window, characterized by: - objective - constraints - variables	
↱	Optimization		problem solution
	Disciplines	selection of concerned disciplines,	
	Discretization and Models	setup of models and interactions	
	Solver	solution	
↳	Repeated analysis	by directive of mathematical optimization procedure (nonlinear programming)	

A more simplified representation focuses on the three main models or “columns” of structural optimization: design, analysis, and optimization model (the so called *three columns concept*).

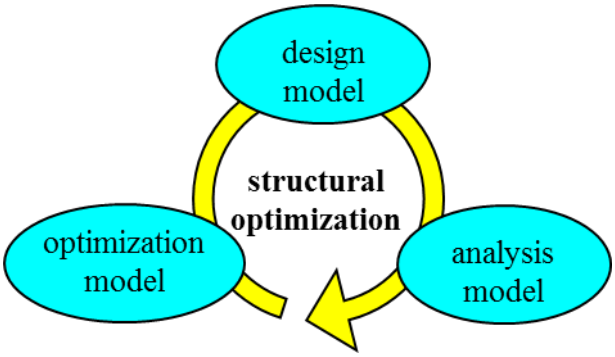


Fig. 1.3: The three columns of structural optimization.

If many different disciplines are involved the procedure is also called *multidisciplinary optimization*. Particularly in aerospace and automotive industries this branch of optimization is increasingly important as these structures are multifunctional and have to be determined e.g. with respect to strength, stiffness, dynamical behavior, safety, acoustics, economics, etc.

The intention of this manuscript is to give an introduction into the main ingredients of structural optimization. Since structural optimization is defined by the combination of several disciplines within the design loop we will concentrate on the procedural aspects, as there are mathematical

definition and background of the optimization problem, several popular algorithms, optimization related topics of the finite element technology, sensitivity analysis, and several special topics as e.g. topology optimization, hanging models, minimal surfaces, and form finding.

2 Basic Concepts

Within this chapter the most important terms of structural optimization will be introduced and demonstrated by examples. Although very simple these examples cover the main features of structural optimization.

2.1 Unconstrained vs. constrained optimization: A four bar truss

Consider a statically determinate four bar truss as shown in Fig. 2.1.

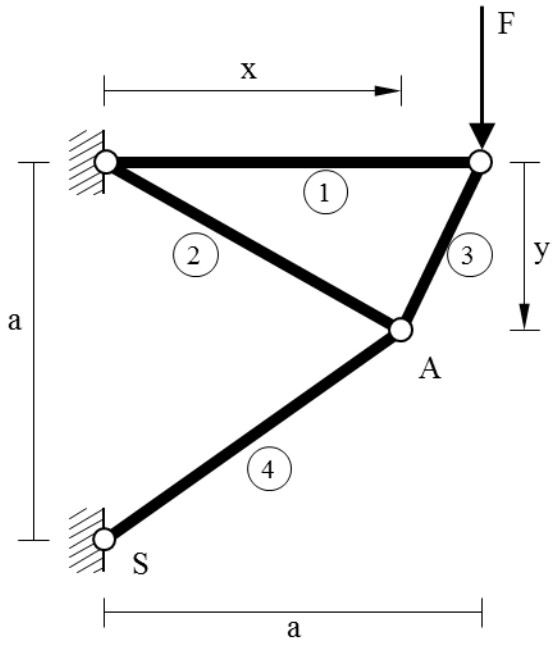


Fig. 2.1: Shape optimized, statically determinate truss.

Since the truss is statically determinate the member forces can be determined from equilibrium as functions of the geometrical *design variables* x and y. The dimension a and the force F are given. The member forces and lengths are evaluated as:

member	length	force
1	$L_1 = a$	$N_1 = \frac{a-x}{y} F$
2	$L_2 = \sqrt{x^2 + y^2}$	$N_2 = \frac{L_2}{x\,y}(x + y - a)F$
3	$L_3 = \sqrt{(a-x)^2 + y^2}$	$N_3 = -\frac{L_3}{y} F$
4	$L_4 = \sqrt{x^2 + (a-y)^2}$	$N_4 = -\frac{L_4}{x} F$

Obviously, the force in member 2 is depending on the location of node A. If A is right of the straight line S-F from the lower support S to the load F, as shown in the figure, member 2 is in tension, is A located left from that line, member 2 is in compression. The other member forces do not change their sign if A stays left and below of F, and right of the supports.

Neglecting buckling, we assume β_s as admissible material strength in tension and compression and the cross section area A_i of member i is determined by the relation $A_i = |N_i|/\beta_s$.

Finally, the weight of the truss is given as:

$$f(x,y) = \gamma \sum_{i=1}^{4} A_i L_i(x,y) = \frac{\gamma}{\beta_s} \sum_{i=1}^{4} |N_i(x,y)| L_i(x,y) \tag{2.1}$$

where γ is the specific weight of the material.

Looking for the minimum weight of the truss we have to minimize the *objective function* or simply *objective* f with respect to the two *optimization variables* x and y. Since there are only two variables for this example the objective can be shown in an isometric representation, or by contour lines over the unknowns x and y, Fig. 2.2. The problem is called *unconstrained* because no additional constraints are considered for the solution.

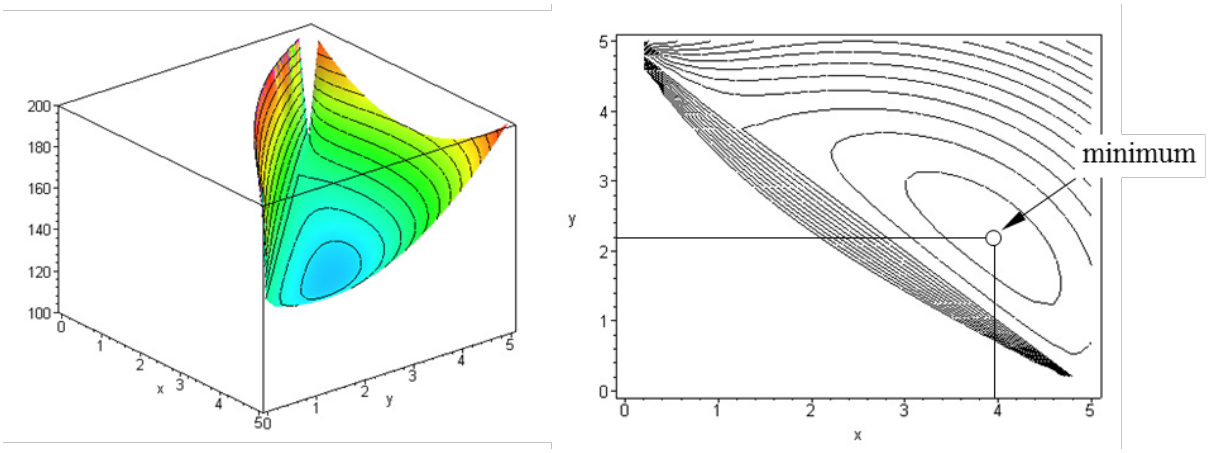


Fig. 2.2: Iso-metric representation (left) and top view with contour lines (right) of objective function (2.1)

Assuming a = 5.0 F = 10.0 β_s = 1.0 and γ = 1.0

the optimal values of can be identified from the figures to be

$$x^* = 3.943 \quad , \quad y^* = 2.224 \quad , \text{and} \quad f^* = 137.27 \tag{2.2}$$

The presented approach might be useful for hand calculations. It suffers, however, from discontinuous derivatives of the objective function as a consequence that the absolute values of the member forces have been used. For numerical solution procedures which are based on gradient information this deficiency might introduce severe convergence problems. Since these methods in many cases are very efficient it is advisable to reformulate the problem such as to omit discontinuities. Here, the absolute value of force N_2 can be resolved by assuming that node A is right from the line S-F and N_3 and N_4 are compression forces. The objective now rewrites as:

$$\tilde{f}(x,y) = \frac{\gamma}{\beta_s}(N_1 L_1 + N_2 L_2 - N_3 L_3 - N_4 L_4) \tag{2.3}$$

It is shown in Fig. 2.3. Now, the function is smooth with continuous derivatives. However, other problems appear now: an artificially low region of the objective which is separated from the minimum solution by a *saddle point* a location on the objective surface which is neither minimal nor maximal. If not started sufficiently close to the minimum solution, numerical algorithms will be trapped in the artificially low region of the problem where the modified $\tilde{f}$ does not represent the real objective f.

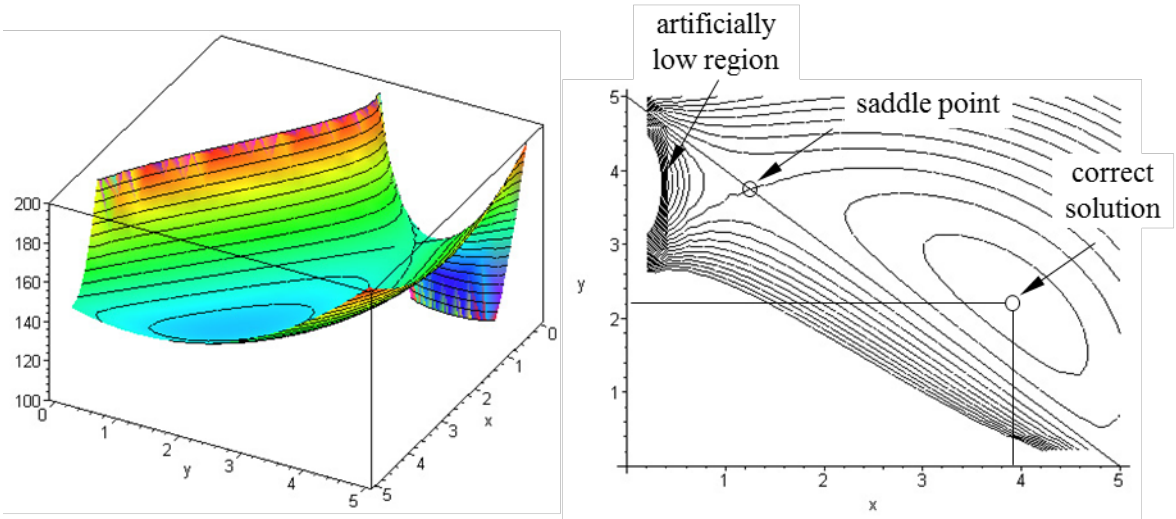


Fig. 2.3: Modified objective function 2.3.

To prevent the algorithms from converging to a wrong solution an additional *constraint* must be formulated. The problem now states as:

$$\begin{aligned} &\text{minimize} \quad \tilde{f}(x,y) = \frac{\gamma}{\beta_s}\left(N_1 L_1 + N_2 L_2 - N_3 L_3 - N_4 L_4\right) \\ &\text{such that} \quad y \geq 5 - x \end{aligned} \tag{2.4}$$

Usually, the constraint is reformulated in a standard representation where the following alternative exists:

$$g(x,y) = 5 - x - y \leq 0 \quad \text{or} \quad \tilde{g}(x,y) = x + y - 5 \geq 0$$

Throughout this manuscript we will use the “less than”-possibility and define *inequality constraints* to be satisfied or *feasible* if they are less or equal to zero. The “greater than”-choice affects the sign convention of several successive results. One should be aware of that comparing this text with others. Constraints can be represented as surfaces as well as the objective. However, only the contour line of value 0 is of interest because it divides the *design space* into a *feasible domain* and an *infeasible domain*. An admissible solution must lie inside the feasible domain. In Fig. 2.4 the modified problem is shown with the additional constraint. The constraint zero contour line is highlighted. For this example this line falls together with a contour line of the objective. A numerical algorithm is now prevented from descending into to the artificially low region of the modified objective function.

By far the most problems in structural optimization are constrained and non-linear. The effort to formulate unconstrained versions does not pay off. In general these modified problems ex-

hibit severe mathematical drawbacks, as e.g. discontinuous derivatives or complex non-linearity, which make them much more difficult to be solved. As a consequence, certain solution methods, as e.g. gradient based methods, may fail. It is advisable to choose a formulation which may have more unknowns and more constraints but is only moderately non-linear. Although large in size it is usually much easier to be solved. It is important to note that the feasible domain usually shrinks when more constraints are added and expands when some constraints are deleted. When the feasible domain shrinks, there are fewer feasible designs and the minimum value of the objective function is likely to increase. The effect is opposite when some constraints are dropped.

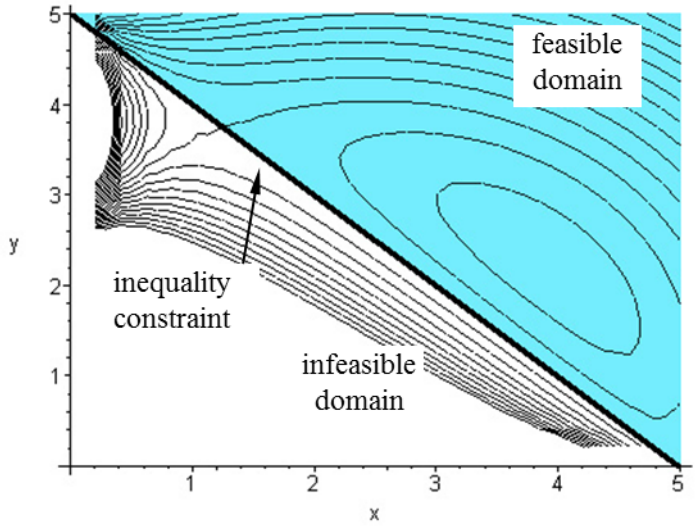


Fig. 2.4: Modified 4-bar truss problem with inequality constraint.

2.2 Sizing: The three bar truss

The following classical problem demonstrates an aspect which is intuitively surprising. Consider a three bar truss which is subjected to two load cases $F^{(1)}$ and $F^{(2)}$ which do not act simultaneously, Fig. 2.5 .The geometry of the structure is supposed to be fixed. The cross sectional areas A_1, A_2, and A_3 are the design variables. The admissible stresses have to be within the range

$$-15 \le \sigma_i \le 20 \tag{2.5}$$

and the objective function is the volume where a = 100 is taken for numerical evaluations in the sequel:

$$f = \sum_{i=1}^{3} A_i L_i = a\left(\sqrt{2}A_1 + A_2 + \sqrt{2}A_3\right) \tag{2.6}$$

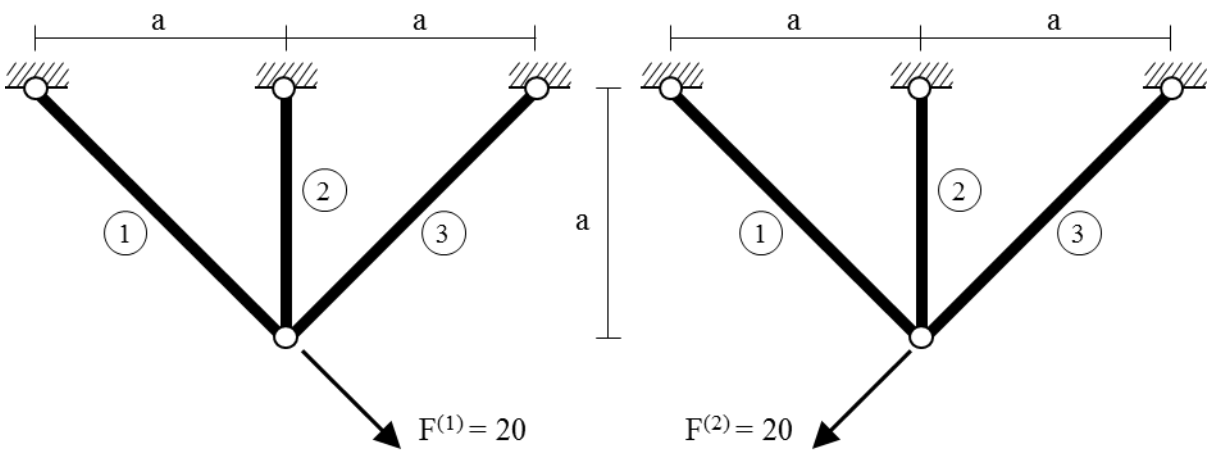


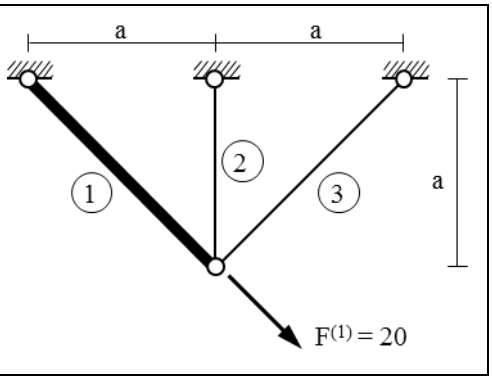
Fig. 2.5: Three bar truss subjected to two load cases.

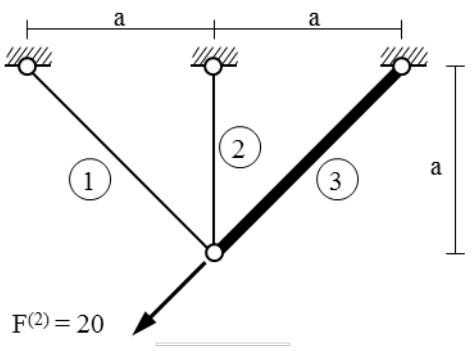
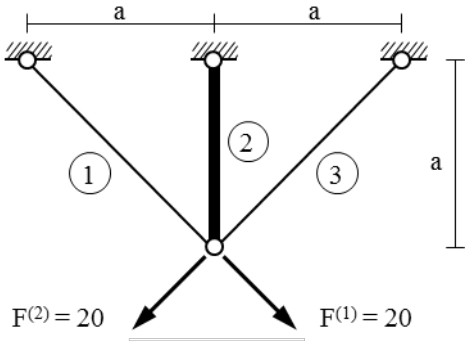
The problem would be simple if only one load case would act or both simultaneously. In all three cases two of the bars would be unnecessary and disappear during the optimization process. The cross section of the remaining bar can be determined by the stress criterion:

$$A_i = \frac{N_i}{\beta_U} \tag{2.7}$$

The bar is, consequently, fully stressed and the optimal design is called a *fully stressed design*. At least for sizing problems it seems to be intuitively correct that a minimum weight or volume design must be fully stressed. This postulate was, therefore, used as an *optimality criterion* representing least weight designs. Optimization methods which are derived from criterions like these are called *optimality criterion methods* or, shorter, *OC methods*. If an optimality criterion can be found, these methods are usually very robust. However, they cannot be generalized and usually apply only for those circumstances they have been designed for.

Fully stressed designs of the three bar truss for single load cases:

load case 1 only $A_1 = \frac{F^{(1)}}{\beta_U} = \frac{20}{20} = 1$ $A_2 = 0$ $A_3 = 0$	

load case 2 only $A_1 = 0$ $A_2 = 0$ $A_3 = \frac{F^{(2)}}{\beta_U} = \frac{20}{20} = 1$	
load case 1 and 2 simultaneously $A_1 = 0$ $A_2 = \frac{1}{2}\sqrt{2}\,\frac{F^{(1)} + F^{(2)}}{\beta_U} = \sqrt{2}\,\frac{20}{20} = \sqrt{2}$ $A_3 = 0$	

It was Schmit in the early 1960's who created the above three bar truss to demonstrate that a minimum weight or volume design due to several non-simultaneous load cases is not a fully stressed design. The problem formulation is straight forward:

$$\begin{aligned} &\text{minimize} \quad f = \sum_{i=1}^{3} A_i L_i = a\left(\sqrt{2}A_1 + A_2 + \sqrt{2}A_3\right) \\ &\text{such that} \quad -15 \le \sigma_i^{(k)} \\ &\qquad\qquad \sigma_i^{(k)} \le 20 \;; \qquad i = 1,2,3; \quad k = 1,2 \end{aligned} \tag{2.8}$$

The problem can be significantly reduced. Since the load cases as well as the geometry of the structure are symmetric it follows that the cross sections have to be symmetrically distributed. I.e. considering just one load case and setting $A_1 = A_3$ is enough. Doing that and choosing load case $F^{(1)}$ it is obvious that bars 1 and 2 are in tension and bar 3 in compression. This means that the number of constraints can be reduced to three. The problem is now reduced to:

$$\text{minimize} \quad f = \sum_{i=1}^{3} A_i L_i = 100\left(2\sqrt{2}A_1 + A_2\right)$$

$$\text{such that} \quad g_1 = \sigma_1 - 20 = \frac{20\left(\sqrt{2}A_1 + A_2\right)}{2A_1A_2 + \sqrt{2}A_1^2} - 20 \le 0$$

$$g_2 = \sigma_2 - 20 = \frac{20\sqrt{2}A_1}{2A_1A_2 + \sqrt{2}A_1^2} - 20 \le 0 \qquad (2.9)$$

$$g_3 = -\sigma_3 - 15 = \frac{20A_2}{2A_1A_2 + \sqrt{2}A_1^2} - 15 \le 0$$

$$\text{and} \quad A_i \ge 0\,; \quad i = 1, 2 \quad \text{(lower bounds)}$$

The graphical representation is given in Fig. 2.6.

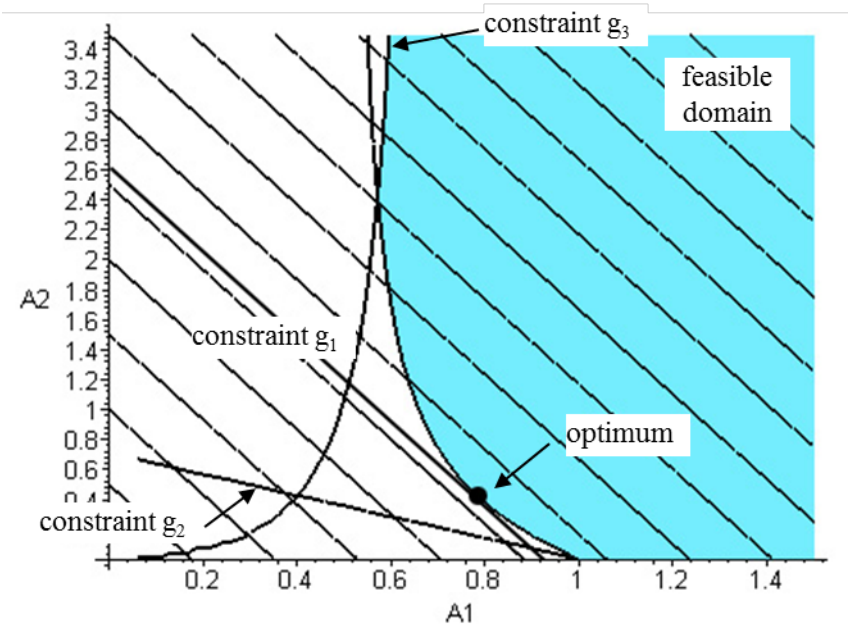


Fig. 2.6: Design space of three bar truss problem

The feasible domain is restricted from below by the lower bound on A_2, and the constraints on the stresses in bars 1 and 3. Constraint g_2 only touches the feasible domain at the point $A_1 = 1$, $A_2 = 0$ and does not play any role for the problem. Obviously, the optimal solution is restricted by constraint g_1 only. The optimal values are:

$$A_1 = 0.788 \quad A_2 = 0.410 \quad f^* = 263.9$$

Constraint g_1 is said to be *active* at the solution whilst the others and the variable bounds are *passive* or *inactive*. In other words: the minimum weight design is not a fully stressed design in the case of multiple load cases. Only member 1 for load case 1 and member 3 for load case 2 reach their stress limit. For both load cases member no. 2 is not fully activated, nevertheless, the design is lighter than for the case that member no. 2 would have been eliminated. Structural optimization led to a solution which contradicts the intuitive lightest fully stressed design. This little example may serve as a further motivation that structural optimization indeed can improve structural design beyond engineering experience.

Another observation of this example is that the objective function with respect to sizing variables is mildly nonlinear. For weight or volume, the objective is even a linear function of the cross section area. The nonlinear character of sizing problems is mainly due to the constraints. Sizing problems of industrial significance are often very large, i.e. they have many unknowns up to several thousand or ten thousands (e.g. bar cross sections, sheet thickness, etc.) and many constraints. From the point of problem formulation sizing problems can be classified as comparatively simple.

2.3 Shape optimization: A two bar truss

Consider the following problem:

Minimize the weight of the truss given in Fig. 2.7 under the load P subject to constraints on the displacement u and the allowable stresses due to material strength and buckling. Modify geometry and member cross sections.

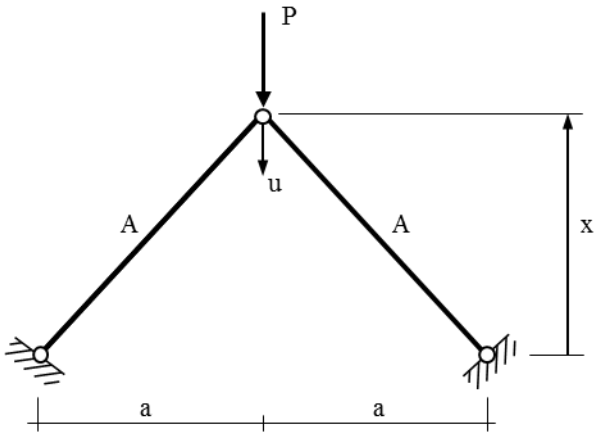


Fig. 2.7: Two bar truss.

The problem is defined by two *optimization variables*: the sizing variable A (cross sectional area) and the vertical coordinate x of the truss vertex (geometrical variable) and by the *state variable* u (vertical displacement under the load). The problem is a shape optimization problem. In total there are four constraints: one *equality constraint* describing the state of equilibrium of internal and external forces, one *inequality constraint* controlling the admissible displacement, and two *inequality constraints* which control the stresses with respect to the material yield strength and Euler member buckling, respectively. Additionally, *lower bounds* on the variables are given to prevent the kinematic case x = 0 and the physically absurd case A <= 0. *Upper bounds* are not necessary for this example.

The problem can be stated explicitly:

variables: structural height x
cross sectional area A

objective: weight

$$f(x, A) = 2\gamma A\sqrt{a^2 + x^2} \rightarrow \min \qquad (2.1)$$

γ ... specific weight

bounds:

$$x \geq x_L \geq 0 \qquad (2.2)$$

$$A \geq A_L \geq 0 \qquad (2.3)$$

equality constraint: equilibrium $K\,u = P$

$$2\,EA\,\frac{x^2}{\left(a^2+x^2\right)^{\frac{3}{2}}}\,u = P \tag{2.4}$$

E ... Young's modulus

inequality constraints:

1.) displacement: $$u = \frac{P}{2\,EA}\,\frac{\left(a^2+x^2\right)^{\frac{3}{2}}}{x^2} \le u_{max} \tag{2.5}$$

2.) yield stress: $$\sigma = -\frac{P}{A}\,\frac{\sqrt{a^2+x^2}}{x} \ge -\beta_s \tag{2.6}$$

β_s ... admissible compressive yield strength

3.) Euler buckling stress: $$\sigma = -\frac{P}{A}\,\frac{\sqrt{a^2+x^2}}{x} \ge -\frac{\pi^2 EcA}{a^2+x^2} \tag{2.7}$$

c ... sectional shape factor
$c = 1/(4\pi)$ for circular cross section

Obviously, the constraints depend on the inverse of the cross section A. This relation is the motivation for several *approximation concepts* which will be discussed in a subsequent chapter. It reflects the well-known fact that the member stresses in statically determinate trusses are inverse functions of the considered member section only. Objective and all constraints are non-linear functions of the geometrical variable. In general, shape optimization problems are non-linear with respect to geometrical variables.

Four cases will be considered:

Case a) Constraints on displacement and yield stress, Fig. 2.8:

additional parameters: P = 1 and u_{max} = 0.8

optimal solution: x* = 2.0 A* = 0.707 f* = 4.0

The stress constraint is limiting the solution, the constraint is said to be *active*. Fig. 2.8 illustrates the situation. It shows contour lines of the objective functions as well as the constraints 1 and 2 as functions of the structural height x (horizontally) the cross sectional area A (vertically). The *feasible domain* is restricted to be above constraint 2 and right of constraint one. Any solution (x, A) which is inside the feasible domain satisfies all constraints, it is *feasible*. The optimal solution is the best solution within the feasible domain. In the present case the optimal solution is characterized by identical tangential directions of constraint 2 and the contour line of the objective.

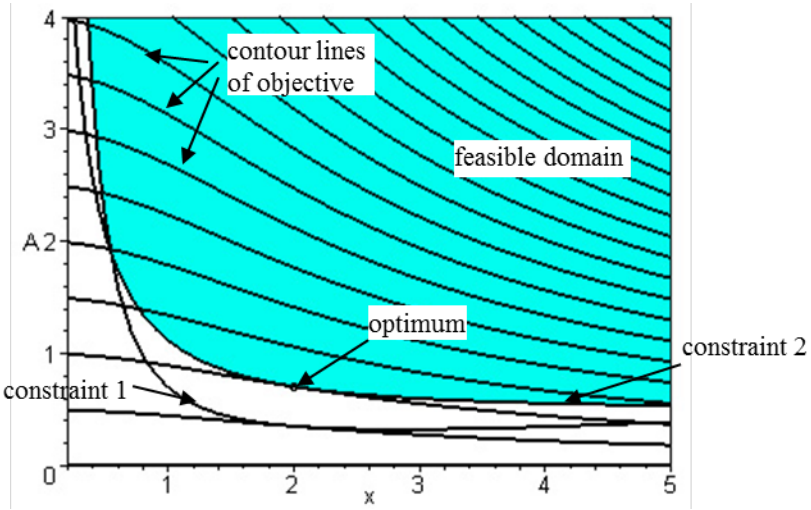


Fig. 2.8: Two bar truss, case a), displacement and strength constraints

Case b) Constraints on yield and buckling stress, Fig. 2.9:

additional Parameters: P = 1

optimal solution: x* = 1.551 A* = 0.816 f* = 4.130

Both constraints are active. The solution is located at a corner of the feasible domain. Since in a problem of n variables the corner is defined by the intersection of n constraints the solution is identically defined by the set of theses constraints (if their gradients are linearly independent at this point). Here in this example, two active constraints define the optimal solution of two variables.

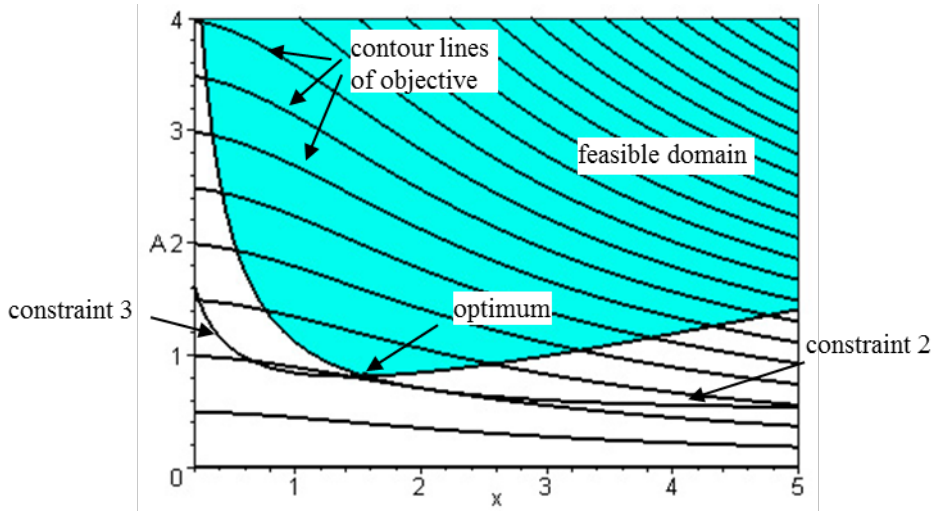


Fig. 2.9: Two bar truss, case b), yield and buckling stress constraints

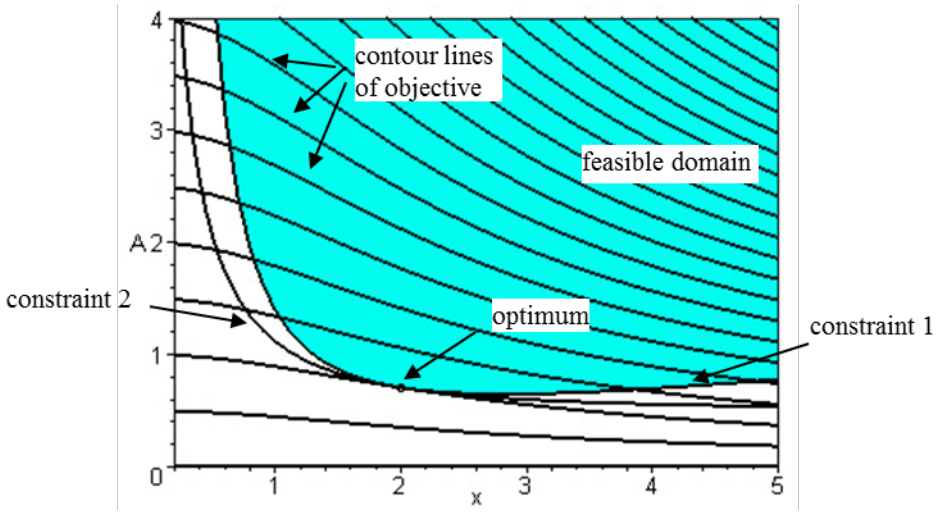


Fig. 2.10: Two bar truss, case c) redundant constraints

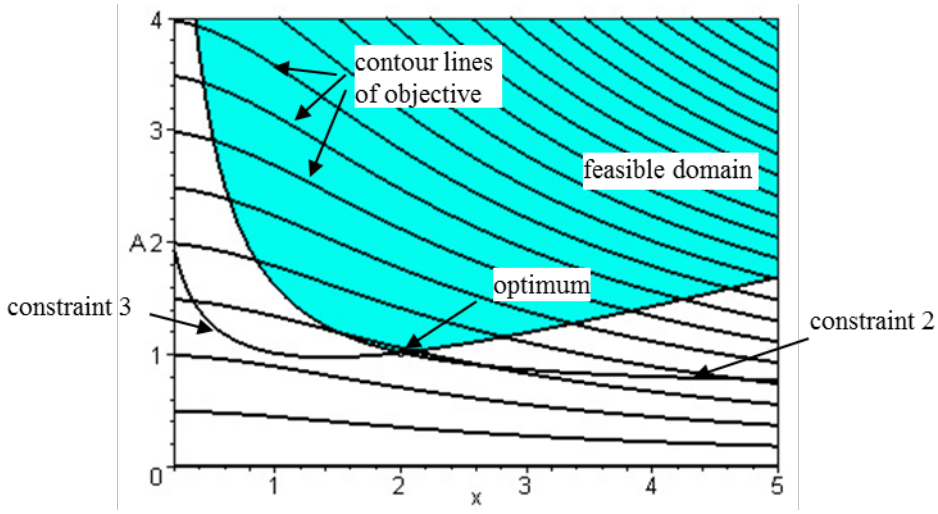


Fig. 2.11: Two bar truss, case d) redundant constraint

The objective functions of shape optimal design problems are in general nonlinear functions of the shape variables. That makes this class of problems mathematically more complex than sizing problems. The source of complexity stems from the necessary effort of problem formulation. One has to select shape variables and consider their interaction very carefully with the formulation of the problem such as to prevent the failure of the optimization process because the variables become physically meaningless or inconsistent. For the present problem that means if x = 0 the truss is kinematic. This is prevented by defining a lower bound of x which is greater than zero. More difficult is to prevent e.g. self-penetration of complex 3D shapes during the optimization process. Shape optimization of three dimensional structures is a very challenging task and demands great skill to define the characteristic variables which allow a large but at the same time non-degenerated variety of shapes.

2.4 Multimodal problems

The optimal orientation of fibers in laminates with respect to structural behavior is a favorite optimization task. Typically, the orientation is defined by an angle α, Fig. 2.12.

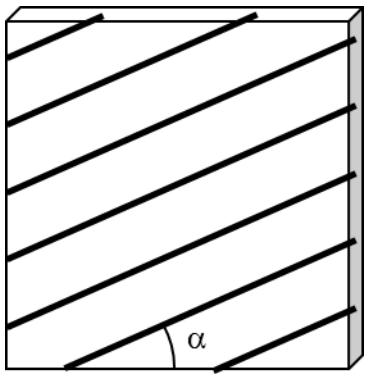


Fig. 2.12: Fiber reinforced laminate, fiber orientation angle α

As a consequence, the structural response is a periodic function with respect to α. This leads to several problems in the context of optimization which will be demonstrated by the following simple example.

Again, consider a truss structure composed by a quadratic frame and a diagonal D which roughly represents one fiber of a laminate, Fig. 2.13.

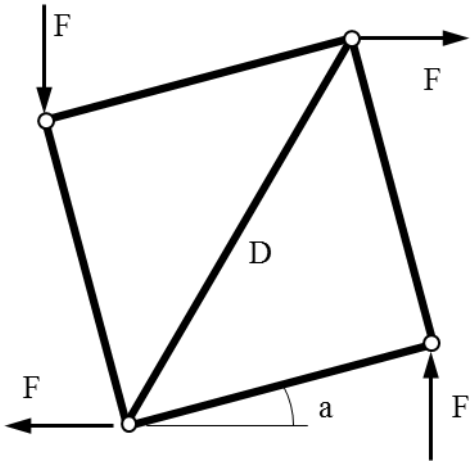


Fig. 2.13: Quadratic truss with diagonal D.

The optimal angle α is to be determined for which the cross sectional area A of diagonal D can be chosen to be the smallest where different upper σ_U (tension) and lower σ_L (compression) admissible stresses have to be considered. That represents different fiber failure modes due to tension and compression. The optimization problem can be written as:

$$\begin{aligned} &\text{minimize} && f(\alpha, A) = A \\ &\text{such that} && g_1(\alpha, A) = D - A\,\sigma_U = \sqrt{2}\,F(\cos\alpha - \sin\alpha) - A\,\sigma_U \le 0 \\ & && g_2(\alpha, A) = -D + A\,\sigma_L = -\sqrt{2}\,F(\cos\alpha - \sin\alpha) + A\,\sigma_L \le 0 \end{aligned} \tag{2.8}$$

where D is the diagonal force and the constraints are formulated in terms of forces rather than stresses. For the numerical evaluation let us assume F = 10.0, σ_U = 4.5, and σ_L = -3.0 . The problem is graphically represented in Fig. 2.14. Because of the periodic characteristics of the constraints the problem has many *local minima* instead of one *global minimum*. Here, all local minima are of equal quality and represent a zero diagonal for $\alpha = \pi/4 + n\pi$.

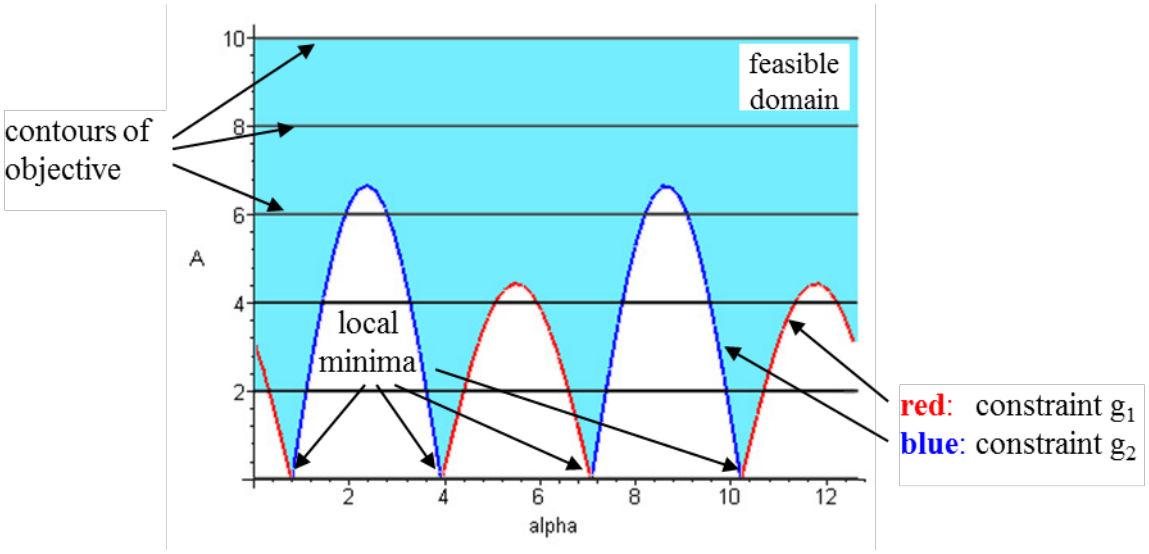


Fig. 2.14: “Fiber orientation problem” non-convex feasible domain.

The boundary of the feasible domain has a typical non-convex shape allowing for more than one solution at the constraint intersections. Unless no additional bounds for the angle are prescribed numerical algorithms will be trapped arbitrarily into one minimum. The solution is dependent on the initial values. The situation gets worse as the objective function values differ from minimum to minimum and one tries to find the overall global minimum out of all the local ones. There exists no numerical algorithm which guarantees to find a global optimum. However, several clever search algorithms which most of them are based on random techniques as e.g. evolutionary strategies and genetic algorithms have a relatively high chance to find the global optimum for the price of a high numerical effort.

As fiber orientation problems are surely non-convex many other structural optimization problems are likely to be non-convex. However, in most cases it is impossible to get information about the problem properties, typically because the structural responses (stresses, displacements, eigen-frequencies, etc.) are implicit functions of the optimization variables and are determined by finite element or other numerical simulation methods. As a consequence, one has to accept that it remains open whether the global optimum is found or not within the chosen design window. From an engineering point of view that is by far good enough, in particular if we take into account the uncertainties of the design window.

3 Optimization using Spread Sheets

Simple optimization problems can efficiently be solved using spread sheet programs, as e.g EXCEL™. It is required that objective and constraints can be programmed in the spread sheet cells or are available as Visual Basic modules. Spread sheets can also be very useful for multi-objective optimization problems to visualize the feasible domain in the space of the objective functions.

3.1 Using the EXCEL™ solver

Definition of variables, objectives, and constraints. The basic principle of spread sheet calculation is that all data is stored in cells and is referenced by the cell address or name. In the context of optimization, the content of a cell might either be interpreted as optimization variable or as value of constraints and objective which in turn have to be defined as functions of the optimization variables. Other more advanced software, e.g. ANSYS (which we will learn to know in the course) principally uses the same methodology. Relevant optimization data (e.g. structural response data as displacements, stresses, eigen-values etc.) are referenced to user defined parameters (instead of cells) which in turn are used by the solver for optimization. Here we again recognize the basic step of modeling the optimization problem: identification of relevant data and their organization in data structures. Spread sheet calculation is a very efficient means to get used with that.

Optimization algorithm. The optimization module of EXCEL consists of a couple of very robust optimization algorithms (Generalized Reduced Gradient method for non-linear problems, the Simplex method with branch and bound for linear problems) which allow for the solution of a broad range of problems, including discrete values of optimization variables. See the EXCEL help facilities for further information about to activate and use the “Solver”.

Among others the following options are available:

- constrained and unconstrained optimization
- equality and inequality constraints, variable bounds
- several levels of sub-problem modeling: linear, quadratic
- linear programming, general reduced gradients
- discrete optimization
- variable scaling
- several report options (e.g. solution sensitivity, iteration history, etc.)

Application. In the following we assume basic knowledge of EXCEL and refer to Excel 2010 and later, assuming that the Solver is activated, Fig. 3.8. Again, we will use the 3 bar truss problem to demonstrate the usage. Moreover, an example file is available from our webpage: 3barTrussExample.xls.

First, the problem is organized as shown in the following figures 3.1 and 3.2. Cells had been given names, e.g. “ex1” is the name of the first variable and refers to cell address “D1” etc. Names can be given by using the “Naming field” (Namenfeld). The optimization method is called by selecting the tab `Data (Daten)` Tab and choosing the `Solver` Button. The cell addresses of objective, variables, and constraints have to be supplied, Fig. 3.3, where inequality constraints may be defined by the “less than” or “greater than” manner, Fig. 3.4. Equality constraints are defined by choosing the “=” operator. Moreover, constraints can be defined for using discrete numbers. Further, more sophisticated control parameters are e.g. concerned with iteration control, Fig. 3.5. The standard numbers have been proofed to be a good choice. Occasionally, the solution accuracy has to be adapted. For the given problem and the indicated initial

values the solver needs about 7 steps to converge, the result is shown in Fig. 3.6. The method terminates offering the option for more detailed reports, Fig. 3.7.

	description	initial value	solution
3	A_1 cross section, bar 1	1,0000	1,0000
4	A_2 cross section, bar 2	1,0000	1,0000
5	A_3 cross section, bar 3		1,0000
6	load F	20,0000	
7	length a	100,0000	
8	admissible compression stress	-15,0000	
9	admissible tension stress	20,0000	
10	σ_1 stress in bar 1		14,1421
11	σ_2 stress in bar 2		8,2843
12	σ_3 stress in bar 3		-5,8579
13	constraint g_1, bar 1 compression		-29,1421
14	constraint g_2, bar 1 tension		-5,8579
15	constraint g_3, bar 2 compression		-23,2843
16	constraint g_4, bar 2 tension		-11,7157
17	constraint g_5, bar 3 compression		-9,1421
18	constraint g_6, bar 3 tension		-25,8579
19	objective, volume		382,8427

Fig. 3.1: Problem set up, initial data

	description	initial value	solution
3	A_1 cross section, bar 1	1	1
4	A_2 cross section, bar 2	1	1
5	A_3 cross section, bar 3		=exs1
6	load F	20	
7	length a	100	
8	admissible compression stress	-15	
9	admissible tension stress	20	
10	σ_1 stress in bar 1		=F*(exs2+WURZEL(2)*exs1)/(2*exs2+WURZEL(2)*exs1)/exs1
11	σ_2 stress in bar 2		=F*(WURZEL(2)*exs1)/(2*exs2+WURZEL(2)*exs1)/exs1
12	σ_3 stress in bar 3		=-F*exs2/(2*exs2+WURZEL(2)*exs1)/exs1
13	constraint g_1, bar 1 compression		=-sig1+sigL
14	constraint g_2, bar 1 tension		=sig1-sigU
15	constraint g_3, bar 2 compression		=-sig2+sigL
16	constraint g_4, bar 2 tension		=sig2-sigU
17	constraint g_5, bar 3 compression		=-sig3+sigL
18	constraint g_6, bar 3 tension		=sig3-sigU
19	objective, volume		=2*WURZEL(2)*a*exs1+a*exs2

Fig. 3.2: Problem set up, equations and references shown

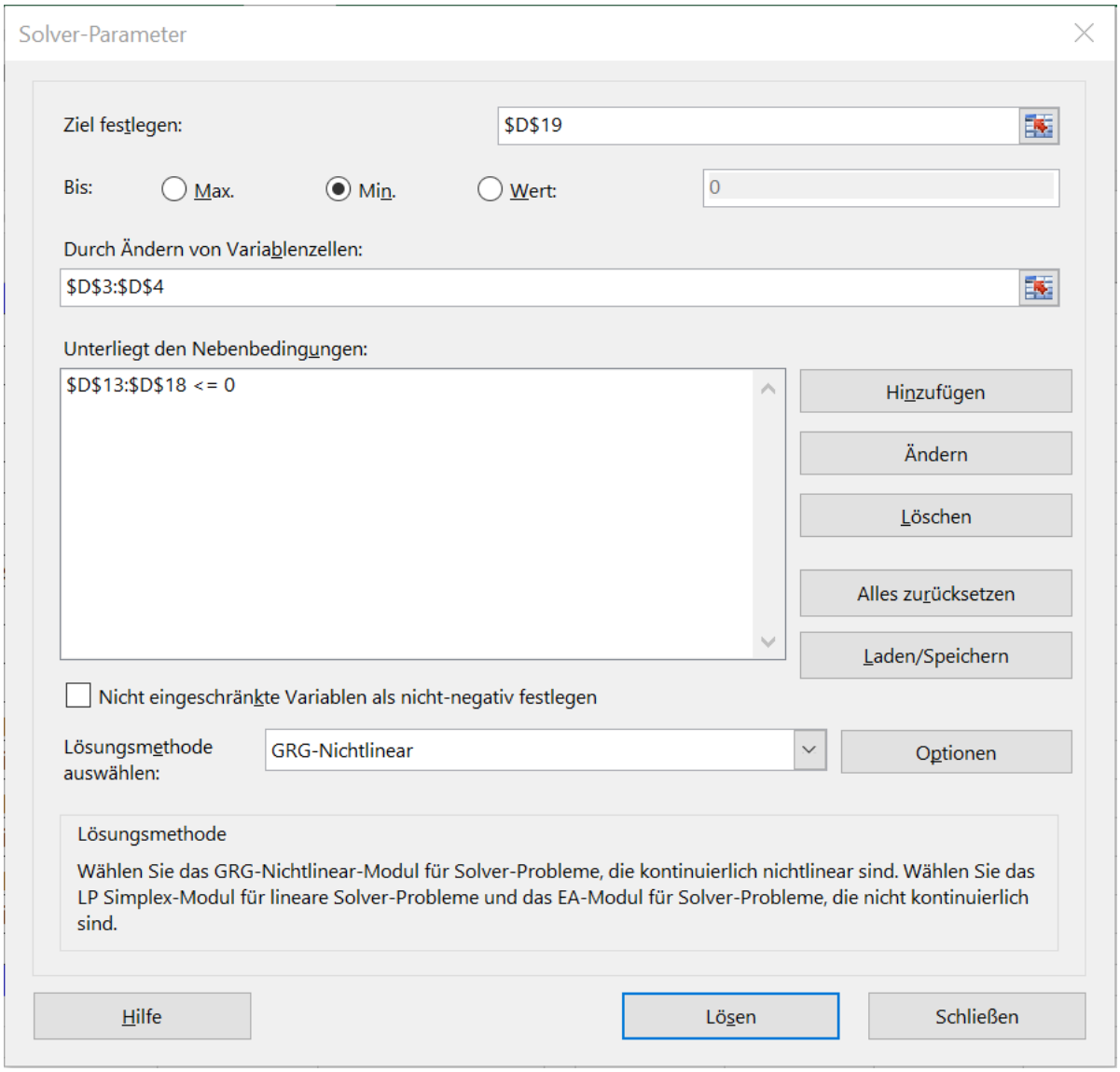


Fig. 3.3: Defining solver parameters: objective, variables, constraints.

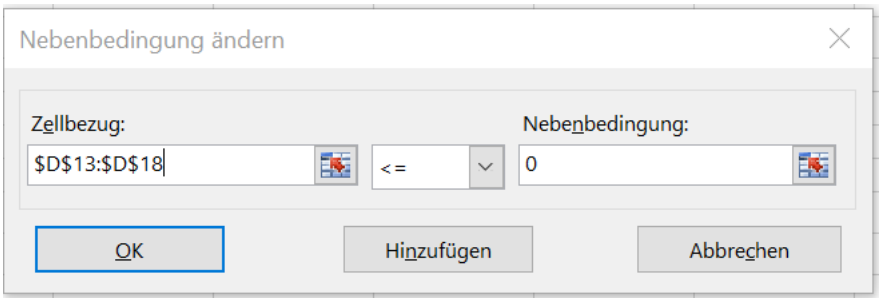


Fig. 3.4: Constraint specification

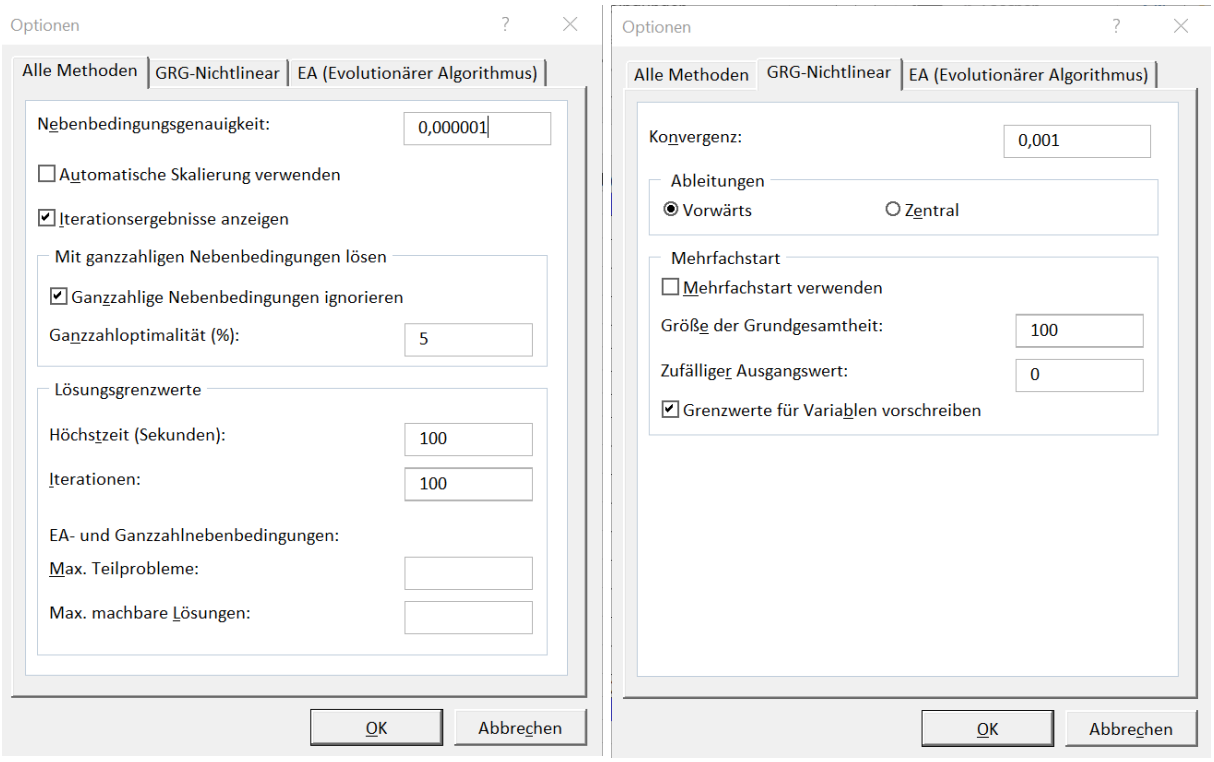


Fig. 3.5: Iteration control parameters for GRG (Generalized Reduced Gradient).

	A	B	C	D	E
1					
2		**description**	**initial value**	**solution**	
3		A_1 cross section, bar 1	1,0000	0,7886	
4		A_2 cross section, bar 2	1,0000	0,4083	
5		A_3 cross section, bar 3		0,7886	
6		load F	20,0000		
7		length a	100,0000		
8		admissible compression stress	-15,0000		
9		admissible tension stress	20,0000		
10		σ_1 stress in bar 1		20,0000	
11		σ_2 stress in bar 2		14,6400	
12		σ_3 stress in bar 3		-5,3600	
13		constraint g_1, bar 1 compression		-35,0000	
14		constraint g_2, bar 1 tension		0,0000	
15		constraint g_3, bar 2 compression		-29,6400	
16		constraint g_4, bar 2 tension		-5,3600	
17		constraint g_5, bar 3 compression		-9,6400	
18		constraint g_6, bar 3 tension		-25,3600	
19		objective, volume		263,8958	
20					

Fig. 3.6: Optimization results

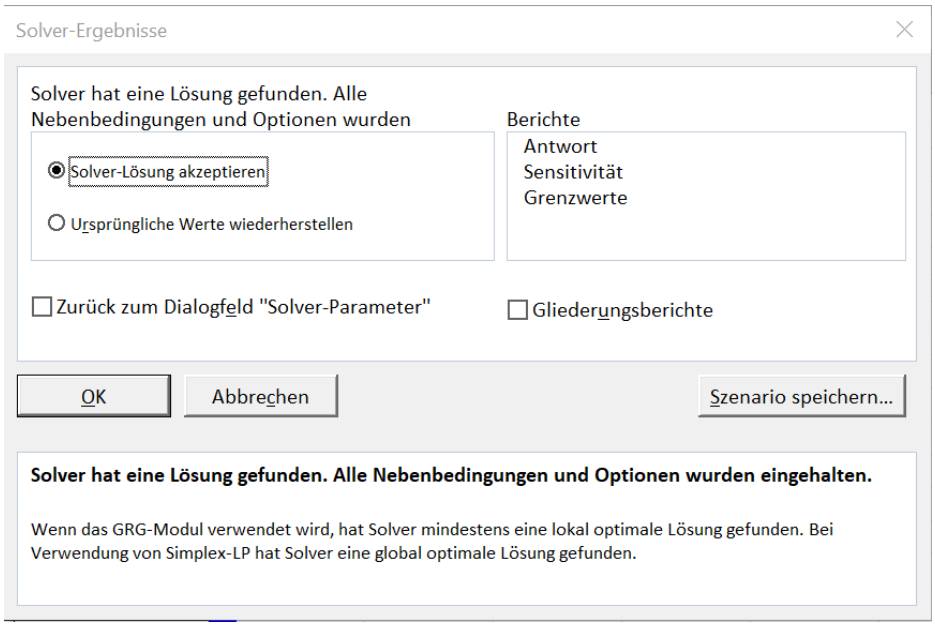


Fig. 3.7: Final report, with the option to get more information.

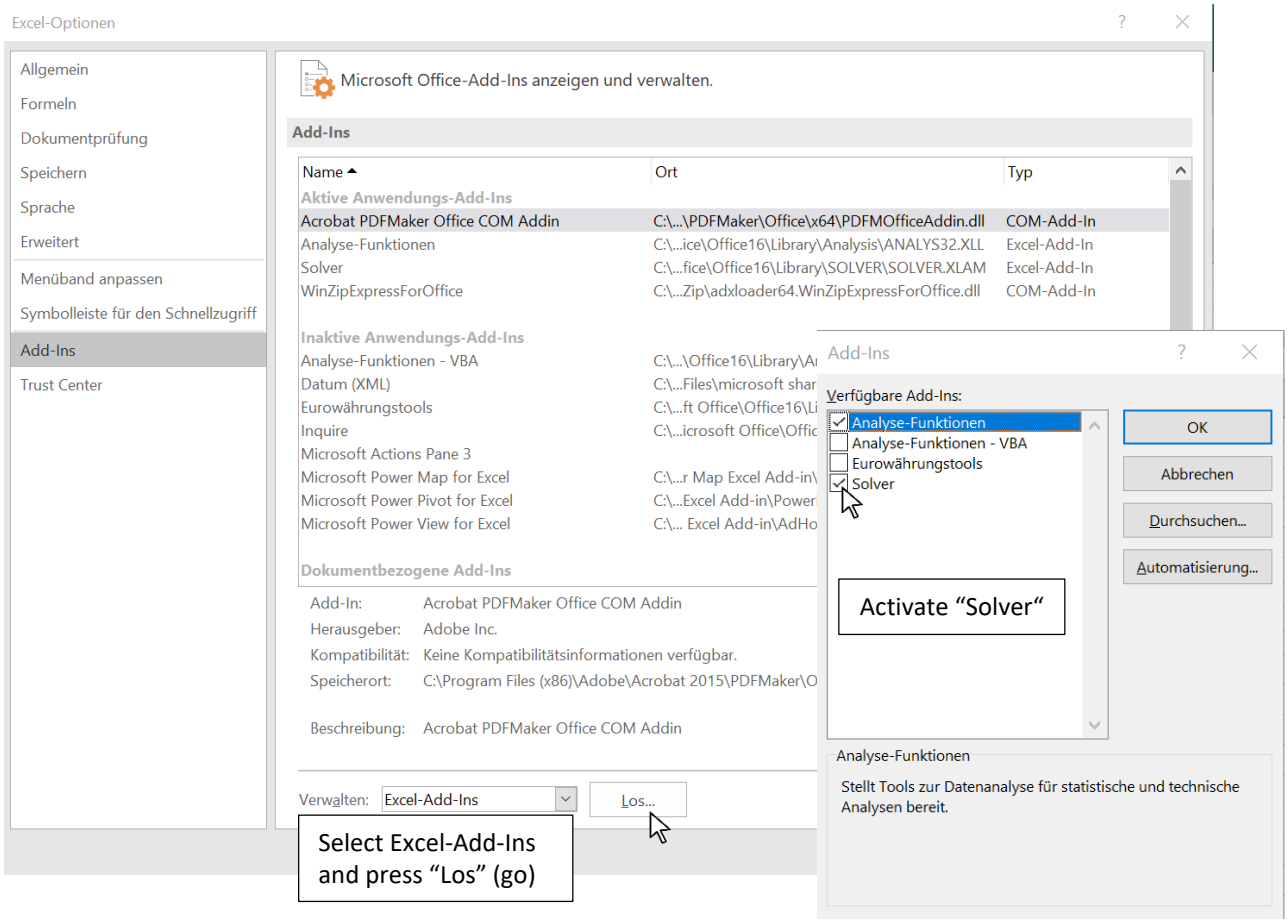


Fig. 3.8: Activate the solver following `Datei(File)->Options->Add-Ins`.

3.2 Multi-objective optimization: data evaluation

	A	B	C	D	E	F	G	H
1		Volume	Energy					
2		263,895843	282,842712					
3		297,925257	250,536047					
4		332,676228	224,365343					
5		368,842233	202,365699					
6		407,137508	183,33122					
7		448,378619	166,468723					
8		493,576073	151,224948					
9		544,062128	137,192082					
10		601,694493	124,051353					
11		669,213043	111,535507					
12		669,213043	111,535507					
13		691,043308	108,194854					
14		712,873572	105,359572					
15		734,703837	102,923046					
16		756,534102	100,806681					
17		778,364366	98,9512644					
18		800,194631	97,3113352					
19		822,024896	95,8514142					
20		843,85516	94,5434052					
21		865,685425	93,36477					
22		865,685425	93,36477					
23		821,24098	99,978984					
24		776,796536	107,748238					
25		732,352091	117,021706					

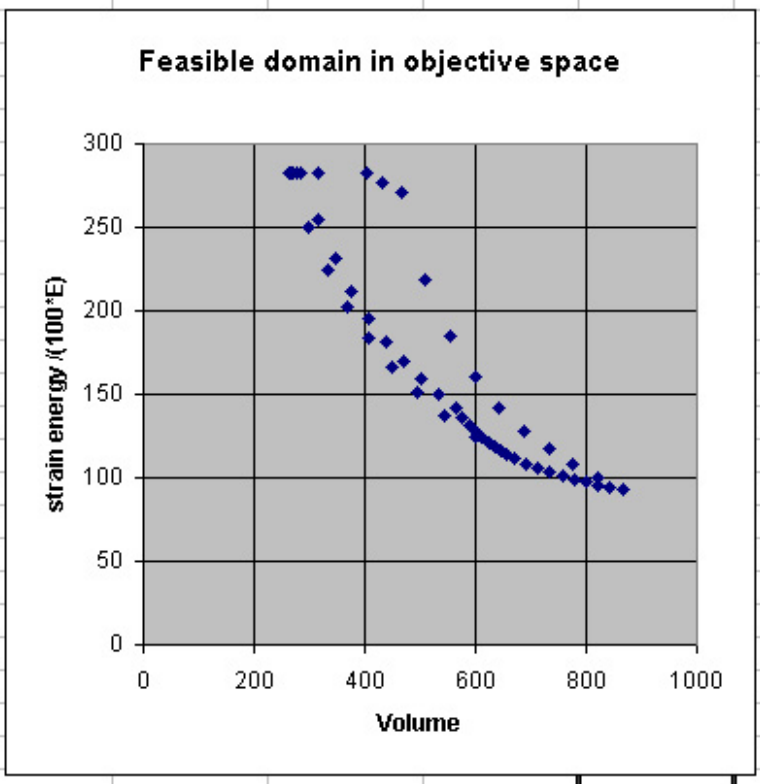


Fig. 3.9: EXCEL diagram manager is used to evaluate the feasible domain in the objective space. Data collected from 64 individual optimization runs with varied combinations of the two objectives “volume” and “strain energy”. The functional efficient edge can be identified as lower left band of the point cloud. Each point represents the result of an individual optimization run. (Refer to chapter 0, Multi-objective optimization.)

4 Basic Mathematics

This chapter discusses the main mathematical topics of optimization. The intention is to give a rather simple introduction and not a rigorous mathematical presentation. After working through this chapter the reader should be able to know as much as necessary to use algorithms and computer methods confidently and to handle problematic cases.

4.1 Problem formulation, problem classes

Optimization problems can be stated in the most general form as:

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}); \quad \mathbf{x} \in \mathbf{R}^n \\ &\text{such that} && g_j(\mathbf{x}) \le 0; \quad j = 1,\ldots, p \\ & && h_j(\mathbf{x}) = 0; \quad j = 1,\ldots; q \end{aligned} \tag{4.1}$$

where f, g_j and h_j are the objective function, the inequality and the equality constraints, respectively. They are functions of the n optimization variables x_i, i = 1,...,n which are arranged in the vector of variables **x**. The vector **x** is of dimension n, i.e. $\mathbf{x} \in \mathbf{R}^n$**.** Alternative to the above formulation inequality constraints can be written in "greater than" fashion. Arranging the constraints in vectors **g** and **h** (4.1) can be written even shorter:

$$\text{minimize} \quad \left\{ f(\mathbf{x}) \,\middle|\, \mathbf{g}(\mathbf{x}) \le 0, \mathbf{h}(\mathbf{x}) = 0; \quad \mathbf{g}: \mathbf{R}^n \to \mathbf{R}^p, \quad \mathbf{h}: \mathbf{R}^n \to \mathbf{R}^q \right\} \tag{4.2}$$

See Fig. 4.1 for graphical representation for several special optimization problem classes.

Nonlinear problems. An optimization problem is called "nonlinear" if at least one of the functions f, g_j or h_j is a nonlinear function of the optimization variables x_i. Problems of structural optimization are usually nonlinear as we have already seen for the simple truss problems in the introduction. To give an example, stress $\sigma = N/A$ in a truss member is a nonlinear function of the cross sectional area A. If σ is taken e.g. as constraint the optimization problem is nonlinear with respect to A although the static calculation might be due to a linear theory of first order. Special cases of nonlinear problems are those with linear constraints which motivates the development of special efficient solution algorithms. The case of linear objective and nonlinear constraints does not differ much from the general case. Special algorithms are, therefore, not available.

Quadratic programming. Quadratic programming problems are characterized by a quadratic objective and linear constraints. They can be written as:

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{e}^T\mathbf{x} \\ &\text{subject to} && \mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \le 0 \\ & && \mathbf{h}(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} \le 0 \end{aligned} \tag{4.3}$$

Quadratic programming as well as linear programming problems can be solved in a finite number of iteration steps. Quadratic problems are important as sub-problems of iterative solution procedures, so called "sequential quadratic programming" or SQP methods. Pure quadratic problems almost do not exist in structural optimization.

Linear programming. Linear problems are of linear objective and constraints:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) = \mathbf{e}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \le 0 \\ & \mathbf{h}(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} \le 0 \end{aligned} \tag{4.4}$$

Linear programming can be used as sub-problem of an iterative procedure. Consequently it is called “sequential linear programming” or SLP analogously to SQP. In pure form linear programming problems appear for applications in structural design with the problem of elastic-plastic design of frames. Linear programs are more important in operational research.

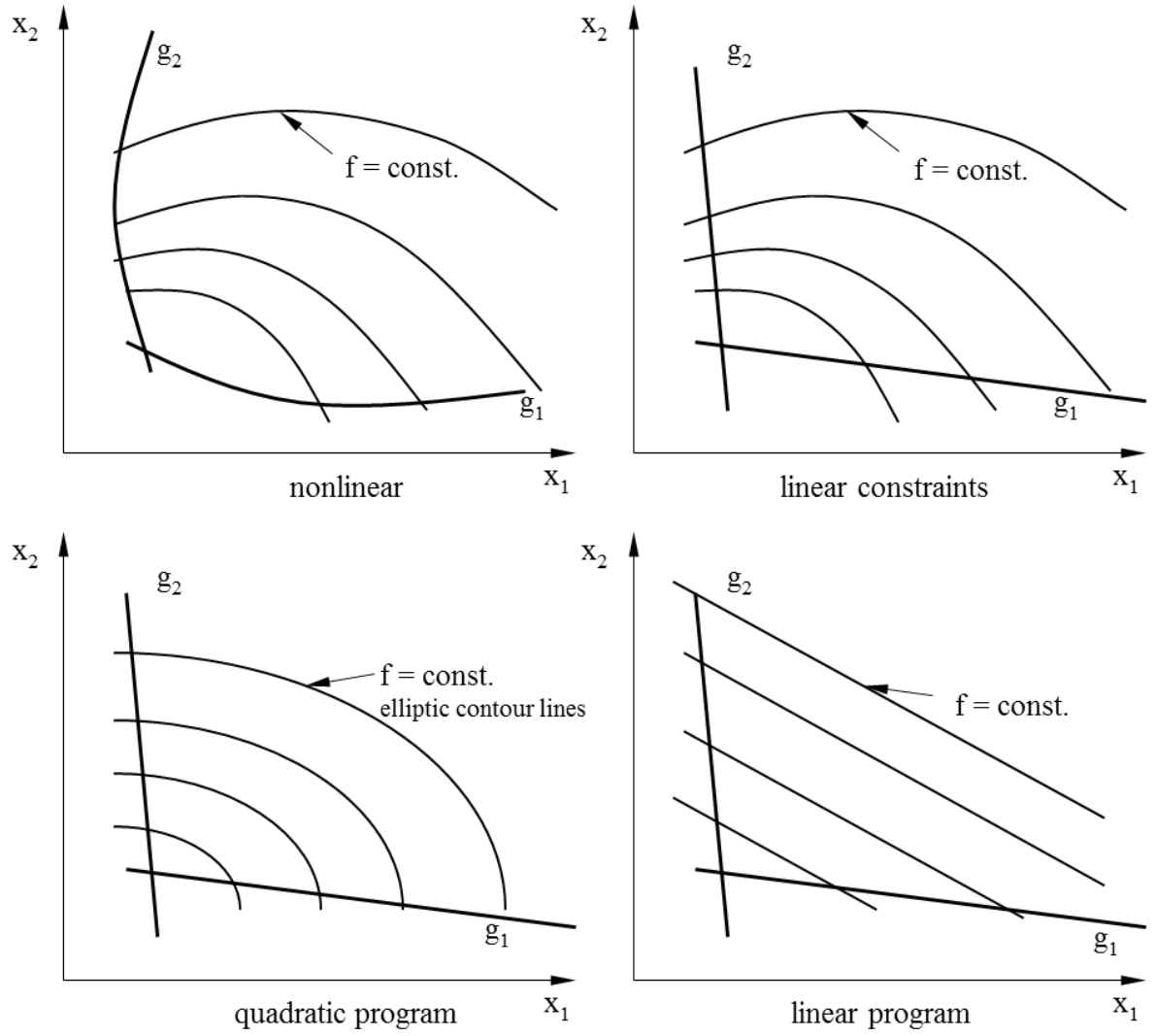


Fig. 4.1: Several optimization classes.

Constrained vs. unconstrained optimization. The basic formulation 4.1 defines a constrained problem. Unconstrained problems are very rare in structural optimization, whereas form finding might be an exception. E.g. minimal weight design is characterized by thin and light members which are critical for material strength or buckling. Sizing problems are in general constrained problems. A typical unconstrained problem is e.g. the determination of minimal surfaces, i.e.

surfaces of minimal area within a given boundary. The knowledge of minimal surfaces is important for the design of lightweight membrane structures and tents.

Variable bounds. A special case of constraints are bounds on variables. These are important as they define limits with respect to physical and technical requirements. Variables are bounded from above as well as from below. Therefore, we speak of upper and lower bounds. Bounds are linear with respect to a single variable. That is the reason why they are treated specially. The basic formulation may be modified with respect to bounds as:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}); \quad \mathbf{x} \in \mathbf{R}^n \\ \text{such that} \quad & g_j(\mathbf{x}) \le 0; \quad j = 1,\ldots, p \\ & h_j(\mathbf{x}) = 0; \quad j = 1,\ldots, q \\ & \underline{x}_i \le x_i \le \overline{x}_i; \quad i,\ldots, n \end{aligned} \tag{4.5}$$

Continuity requirements. For the proof of local optimality conditions continuous second order derivatives are assumed. Many successful so called *gradient* and *Newton type* algorithms are based on this assumption. However, many engineering problems are discontinuous within the second and first derivatives, or even in the function itself. Oscillation or failure of algorithms might be due to discontinuities. Nevertheless, many algorithms can successfully be applied for weakly discontinuous problems also.

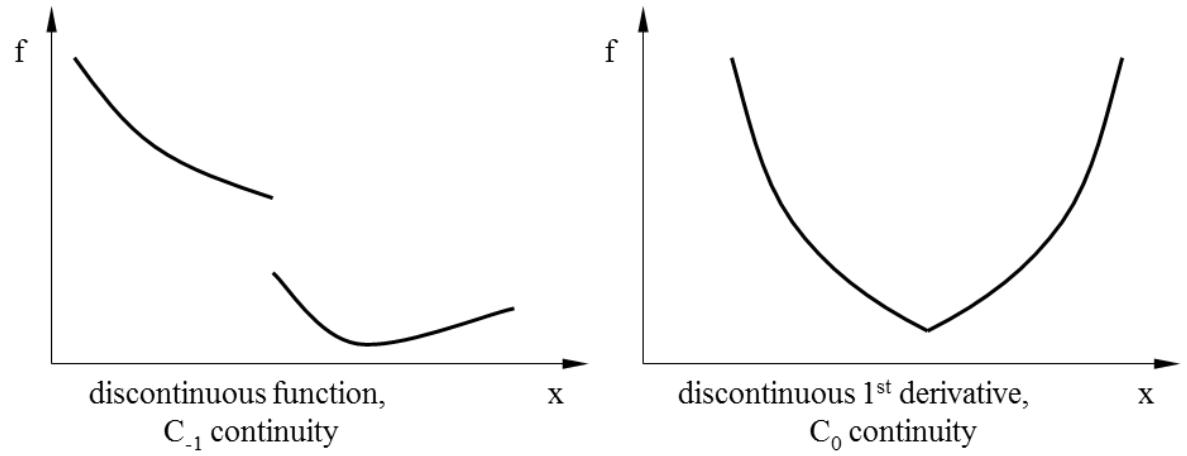


Fig. 4.2: Continuity properties of a one dimensional function.

Second order derivatives. If second order derivatives exist, they can be arranged in a matrix called *Hessian matrix* or short *Hessian*. The Hessian is symmetrical and of dimension n. The Hessian of the objective f(**x**) is:

$$\nabla^2 f(\mathbf{x}) = \mathbf{H} = H_{ij} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_j} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & & & \\ \cdots & & & & \\ \frac{\partial^2 f}{\partial x_i \partial x_1} & & \frac{\partial^2 f}{\partial x_i \partial x_j} & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & & \cdots & & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \tag{4.6}$$

Discrete optimization. The most severe case of discontinuous problems are discrete optimizations problems. Here, the optimization variables can take discrete values only. A typical engineering application example is the design for distinct cross sections taken from a set of available profiles (steal constructions).

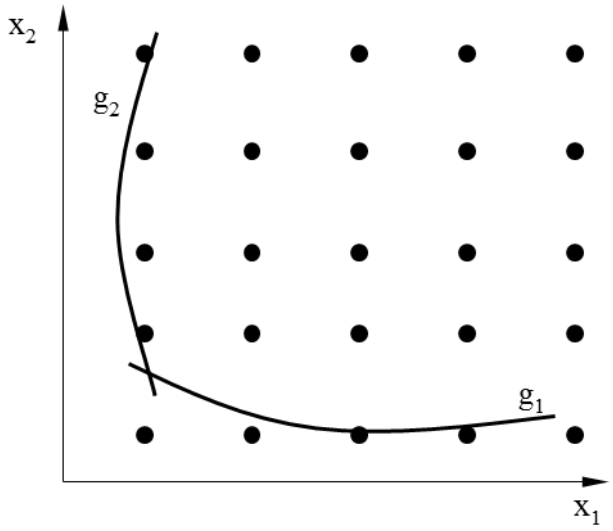


Fig. 4.3: Discrete optimization, objective evaluated for discrete values of x1 and x2

Feasible domain. A certain choice of variables **x** is *feasible* if it satisfies all constraints. The feasible domain defines a subset $\mathbf{F}^n$ of the space $\mathbf{R}^n$:

$$\mathbf{F}^n = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \le 0, \mathbf{h}(\mathbf{x}) = 0\} \tag{4.7}$$

Constraints are called *active* if they are zero at the optimal solution $\mathbf{x}^*$. Equality constraints are always active. Inequality constrains which are satisfied but not zero at $\mathbf{x}^*$ are *passive* or *non-active*. A solution **x** inside of the feasible domain is called *unrestricted* while a solution which is on the boundary (i.e. at least one constraint is active) is called *restricted*. Consequently, optimal solutions are called restricted or unrestricted if they are determined by constraints or not. Constraints which are not active at the optimum are irrelevant or redundant and can be omitted from the problem without any harm, which is called *constraint deletion*. Again, the definition of active constraints is:

$$
\begin{aligned}
\text{optimal solution } \mathbf{x}^*: \quad & f(\mathbf{x}^*) \rightarrow \min \\
\text{subject to} \quad & g_j(\mathbf{x}^*) \le 0; \quad j = 1, \ldots, p \\
& h_j(\mathbf{x}^*) = 0, \quad j = 1, \ldots, q
\end{aligned}
\tag{4.8}
$$

$$
\begin{aligned}
\text{active constraints}: \quad & g_j(\mathbf{x}^*) = 0 \text{ and } h_j(\mathbf{x}^*) = 0 \\
\text{non-active constraints}: \quad & g_j(\mathbf{x}^*) < 0
\end{aligned}
$$

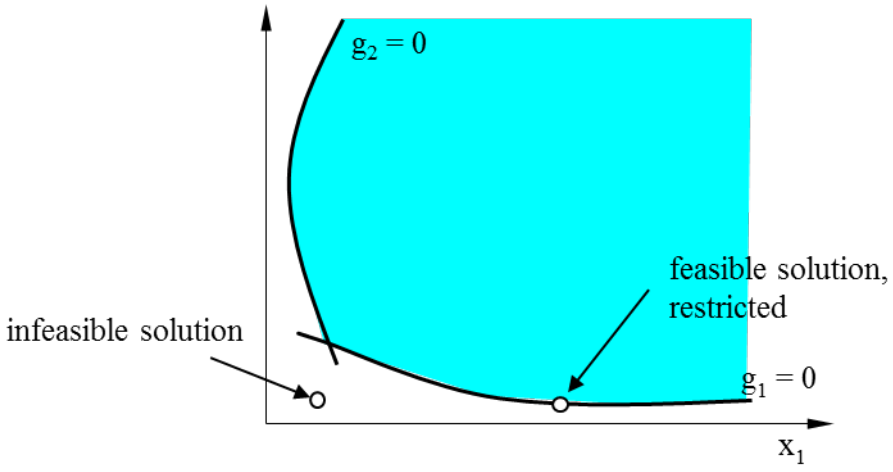


Fig. 4.4: Feasible domain, feasible and infeasible solutions.

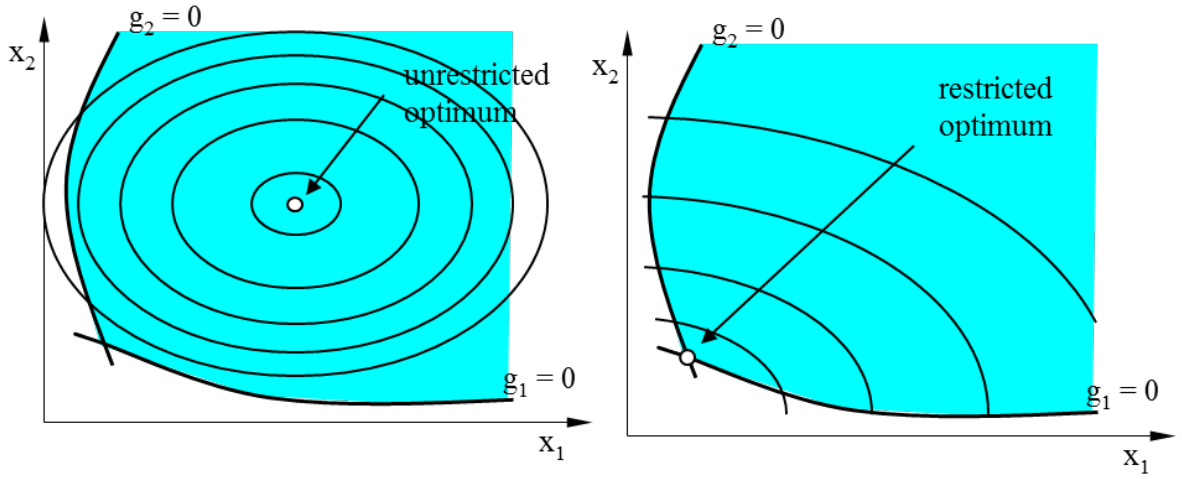


Fig. 4.5: Restricted and unrestricted optima.

4.2 Convexity, local and global optima

The property of convexity of objective and constraints is important for the information whether one or even more solutions are possible. If objective and all relevant constraints are convex the problem has exactly one solution. A function is convex if any two points on the function surface can be joined by a straight line which is above the function and does not intersect the function.

$$f(\varepsilon a + (1-\varepsilon)b) \le \varepsilon f(a) + (1-\varepsilon) f(b) \; ; \quad 0 \le \varepsilon \le 1 \tag{4.9}$$

The function is called *concave* if the line is below the function, and it is called *non-convex* if the line intersects, Fig. 4.6. This definition holds as well for function contour lines. If a contour line is convex the included domain is also called convex. The feasible domain can be characterized to be convex as well. With this definition in mind we recognize that the feasible domain of the 3-bar truss problem is indeed non-convex because of constraint g_3 (compression in bar no. 3).

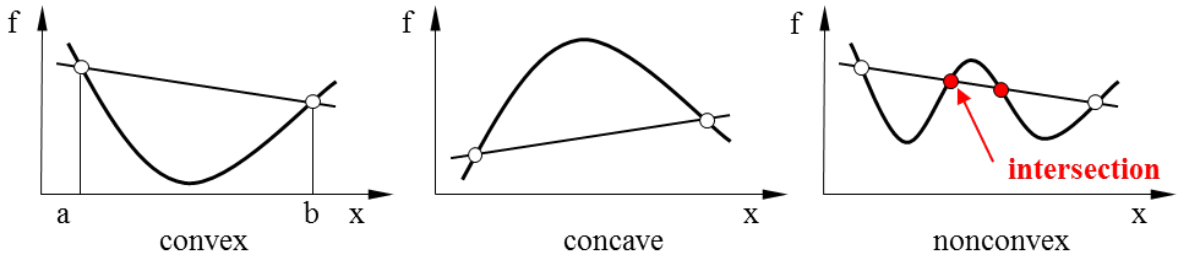


Fig. 4.6: Convex and non-convex functions of one variable

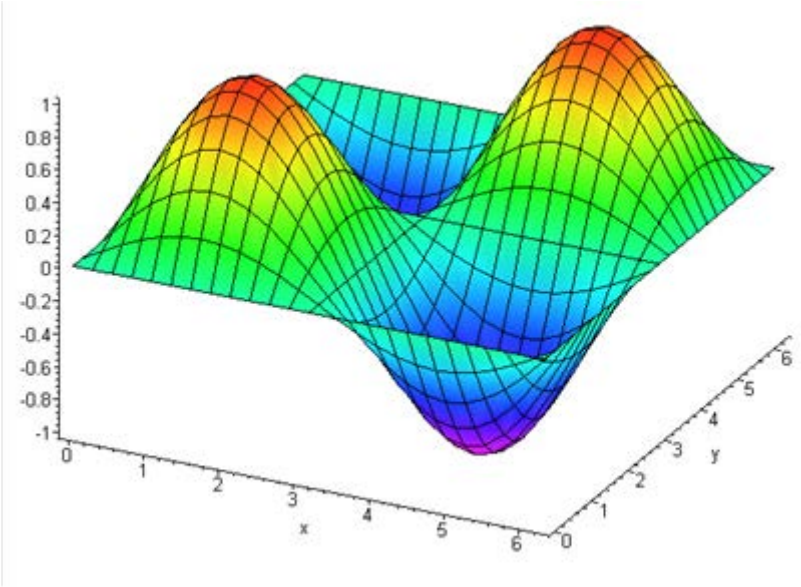


Fig. 4.7: Non-convex function of two variables.

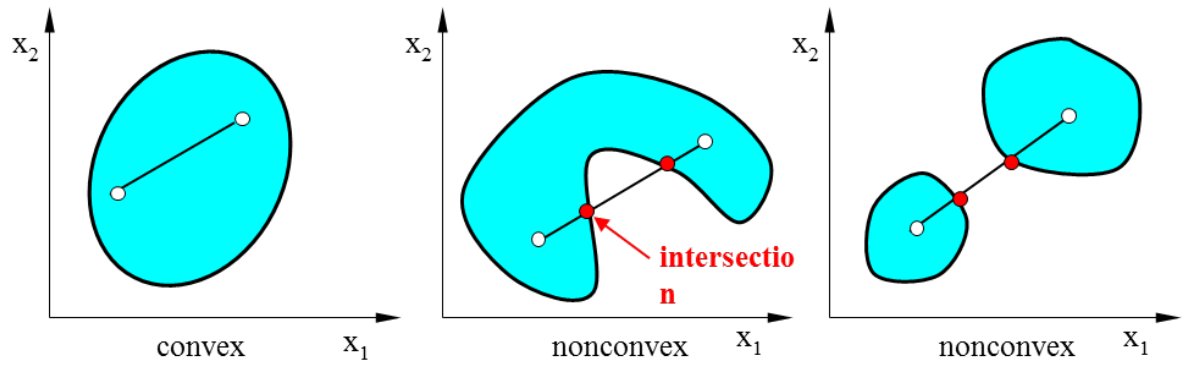


Fig. 4.8: Convex and non-convex domains.

Local and global optima. A solution $\mathbf{x}^{**}$ is called *global optimum* if no better solution can be found in the whole feasible domain:

$$f(\mathbf{x}^{**}) < f(\mathbf{x}) \quad \forall \quad \mathbf{x} \in \mathbf{F}^n \tag{4.10}$$

whereas a local optimum $\mathbf{x}^{*}$ is the best solution within a sufficiently small surrounding $U(\mathbf{x}^{*})$ around $\mathbf{x}^{*}$:

$$f(\mathbf{x}^{*}) < f(\mathbf{x}) \quad \forall \quad \mathbf{x} \in U(\mathbf{x}^{*}) \tag{4.11}$$

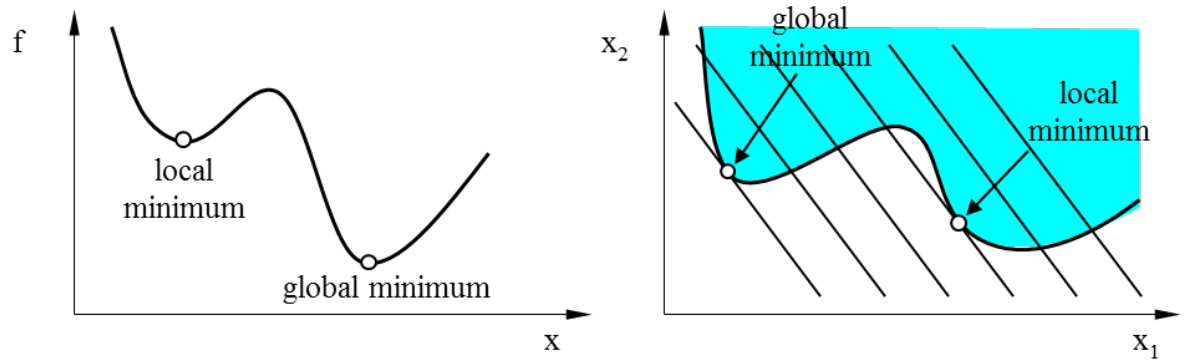


Fig. 4.9: Local and global optima of non-convex function (left) and non-convex domain(right)

4.3 The Lagrange function

The Lagrange function is the key of optimization theory. We try to develop the idea by an example.

Consider the problem:

$$\begin{aligned} &\text{minimize} \quad f(x) = x^2 + 1 \\ &\text{such that} \quad g(x) = 4(1 - x) \le 0 \end{aligned} \tag{4.12}$$

The problem is constrained, the solution is $x^* = 1$, as can be seen from Fig. 4.10.

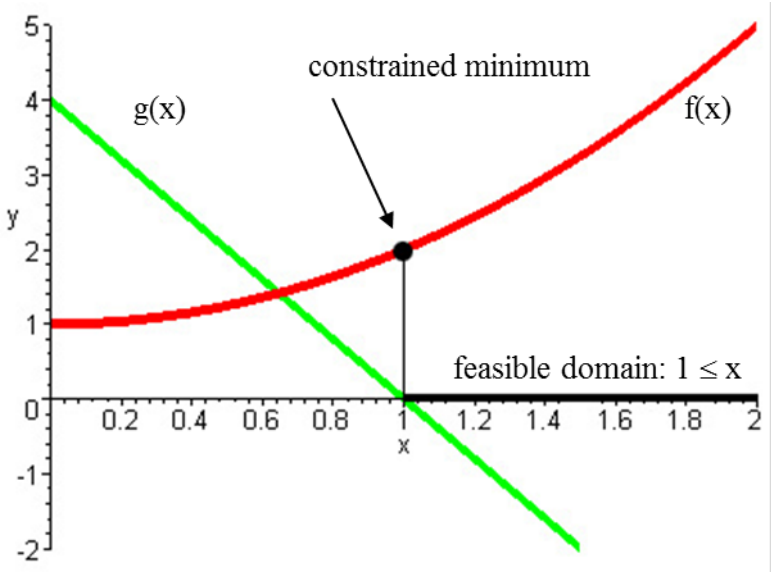


Fig. 4.10: Visualization of problem (4.12)

Assuming continuous first derivatives the solution of unconstrained optimization problems is defined by the stationary condition of vanishing first derivatives:

$$\nabla_x f = \frac{\partial f}{\partial x_i} = 0; \quad i = 1,\ldots, n \tag{4.13}$$

We would like to have an analogous condition for constrained optimization. The idea is as follows, developed as an example for problem (4.12). Consider an auxiliary function L which is composed of f and g as

$$L(x,\lambda) = f(x) + \lambda\, g(x) \tag{4.14}$$

Note, that at the optimum $L(x^*) = f(x^*)$ for arbitrary λ because $g(x^*) = 0$. The question is now: how has λ to be chosen such that the unconstrained minimization of L gives the same result as the original problem 4.12.

The first derivative of L with respect to x is:

$$\frac{dL}{dx} = \frac{d}{dx}\left(x^2 + 1 + \lambda\, 4(1-x)\right) = 2x - 4\lambda \tag{4.15}$$

Imposing the stationary condition at $x^* = 1$:

$$\left.\frac{dL}{dx}\right|_{x=x^*,\,\lambda=\lambda^*} = 2x^* - 4\lambda^* = 0 \tag{4.16}$$

gives $\lambda^* = 1/2$. A final check verifies that for $\lambda^* = 1/2$ the unconstrained minimization of $L(x,\lambda^*)$ w.r.t. x gives the correct result.

But what is to do if the solution x^* is not known in advance? First, for a given λ we determine the minimum of L w.r.t. x by applying the stationary condition:

$$\left.\frac{dL}{dx}\right|_{\lambda} = 2x - 4\lambda = 0 \quad \rightarrow \quad x = 2\lambda \tag{4.17}$$

In a second step the active constraint has to be satisfied:

$$\begin{aligned} &g(x = 2\lambda) = 0 \\ &4\,(1 - 2\lambda) = 0 \quad \rightarrow \quad \lambda = 1/2 \end{aligned} \tag{4.18}$$

From the definition of the Lagrangian as $L = f + \lambda g$ we realize that:

$$g = \frac{dL}{d\lambda} \tag{4.19}$$

In total we have derived two equations for the determination of the two variables x and λ:

$$\begin{aligned} &\frac{dL(x,\lambda)}{dx} = 0 \\ \text{and} \qquad &\frac{dL(x,\lambda)}{d\lambda} = \frac{f(x) + \lambda\, g(x)}{d\lambda} = g(x) = 0 \end{aligned} \tag{4.20}$$

Both equations are stationary conditions with respect to the *primal* variable x and the *dual* variable the so called *Lagrange multiplier* or *Lagrange parameter* λ.

For the above example we get:

$$\left.\begin{aligned} &L(x,\lambda) = x^2 + 1 + \lambda\, 4(1 - x) \\ &\frac{dL(x,\lambda)}{dx} = 2x - 4\lambda = 0 \\ &\frac{dL(x,\lambda)}{d\lambda} = 4 - 4x = 0 \end{aligned}\right\} \quad \rightarrow \quad (x^*, \lambda^*) = (1,\ 1/2) \tag{4.21}$$

Furthermore it is of interest to know whether L is maximal or minimal at the solution (x^*, λ^*). This information is derived from the matrix of second order derivatives of L:

$$\nabla^2 L = \begin{bmatrix} \frac{\partial^2 L}{\partial x^2} & \frac{\partial^2 L}{\partial x\, \partial \lambda} \\ \frac{\partial^2 L}{\partial \lambda\, dx} & \frac{\partial^2 L}{\partial \lambda^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} + \lambda \frac{\partial^2 g}{\partial x^2} & \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial x} & 0 \end{bmatrix} \tag{4.22}$$

Obviously, the matrix is not positive definite, i.e. the Lagrangian is a saddle surface in the enlarged space (x, λ), Fig. 4.11. Usually, only the upper left part of $\nabla^2 L$ is called the Hessian of the Lagrangian.

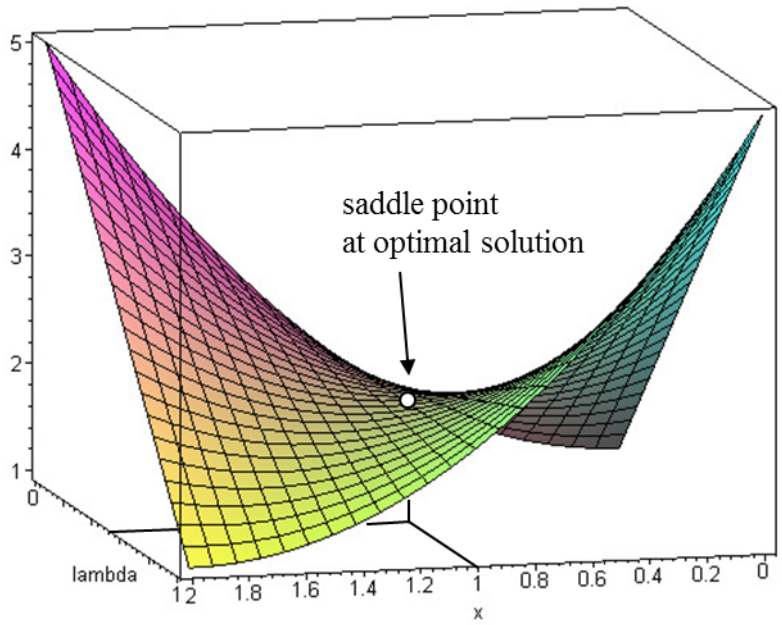


Fig. 4.11: The Lagrange function of problem (4.12)

4.4 The dual function

Again, we assume objective function f and constraints **g** and **h** to be at least C_1-continuous and refer to the previous example. The optimal solution as a saddle point of the Lagrangian is either defined as the minimum of $L(x,\lambda^*)$ w.r.t. x where λ is set to λ^* (primal view) or as the maximum of the *dual function* $D(\lambda)$ with respect to λ (dual view). The dual $D(\lambda)$ is determined as the minimum of $L(x,\lambda)$ with respect to x for given λ:

$$D(\lambda) = \min_x L(x,\lambda) \tag{4.23}$$

The dual function approaches the saddle point from below following the base line of the flume, Fig. 4.12. At the optimum it holds:

$$\max_\lambda D(\lambda) = \min_x L(x,\lambda^*) \tag{4.24}$$

For the problem (4.12) the dual is determined as $D(\lambda) = -4\lambda^2 + 4\lambda + 1$

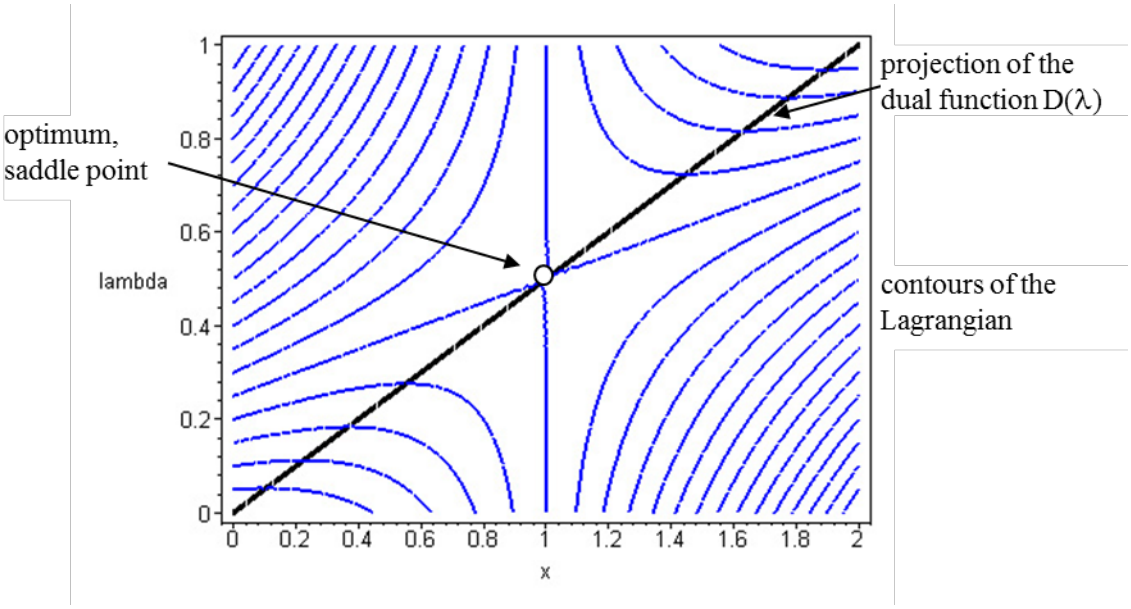


Fig. 4.12: Projection of the dual function D(λ).

Naturally, descent methods fail for finding saddle points. So called *dual algorithms* make use of the dual function to split the numerical procedure to determine saddle points in a repeated sequence of minimization and maximization, so descent methods can be applied for the sub-problems:

1.) minimize L(x,λ) with respect to x, determine x(λ) as function of λ
2.) substitute x(λ) into L to get the dual function $D(\lambda) = L(x(\lambda), \lambda)$
3.) maximize D(λ) with respect to λ

4.5 The Karush-Kuhn-Tucker conditions

The solution of the general constrained optimization problem (4.1) satisfies the necessary Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned}
\frac{\partial L}{\partial x_i} &= \frac{\partial f}{\partial x_i} + \sum_{j=1}^{p} \lambda_j \frac{\partial g_j}{\partial x_i} + \sum_{j=1}^{q} \mu_j \frac{\partial h_j}{\partial x_i} = 0 \; ; \quad i = 1, \ldots, n \\
\frac{\partial L}{\partial \mu_j} &= h_j(\mathbf{x}) = 0 \qquad\qquad ; \quad j = 1, \ldots, q \\
\lambda_j \frac{\partial L}{\partial \lambda_j} &= \lambda_j g_j(\mathbf{x}) = 0 \qquad\qquad ; \quad j = 1, \ldots, p \\
\lambda_j &\geq 0 \qquad\qquad ; \quad j = 1, \ldots, p
\end{aligned} \tag{4.25}$$

where $L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{j=1}^{p} \lambda_j g_j(\mathbf{x}) + \sum_{j=1}^{q} \mu_j h_j(\mathbf{x})$ is the Lagrangian, with $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ the vectors of Lagrange multipliers of inequality and equality constraints, respectively. In addition to the stationary conditions the KKT-conditions supply a criterion to select active inequality constraints.

This is given in lines 3 and 4 of (4.25). It states that at the optimum either the constraints are active, i.e. $g_j = 0$ and $\lambda_j > 0$, or inactive which means $g_j < 0$ and $\lambda_j = 0$.

There is a nice geometrical interpretation of the KKT-conditions. It says that at a constrained optimum the linear combination of the objective and constraint gradients vanish, Fig. 4.13. Since the multipliers of active inequality constraints have to be positive the negative gradient $-\nabla f$ must be inside the cone which is spanned by the gradients of active constraints. The gradient ∇f is defined as:

$$\nabla f = \frac{\partial f}{\partial \mathbf{x}} \tag{4.26}$$

and can be interpreted as a vector facing upward in increasing direction of the objective. The gradients ∇g_j of constraints are defined analogously.

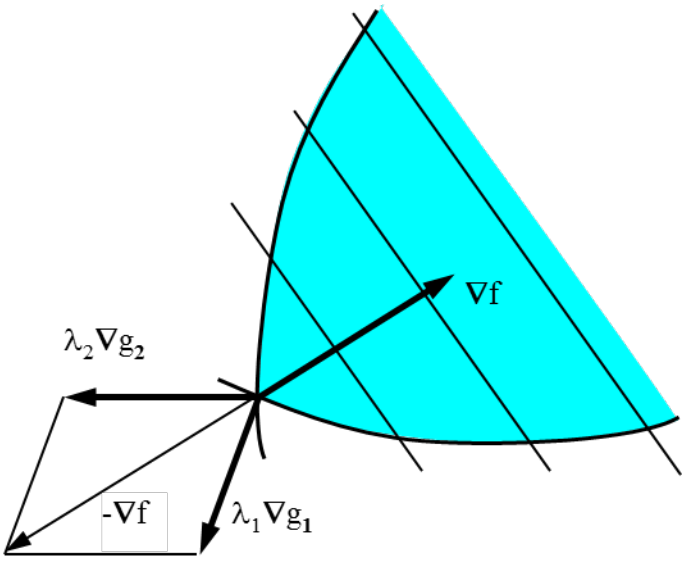


Fig. 4.13: Graphical representation of the Karush-Kuhn-Tucker conditions.

As prerequisite of the KKT-conditions the constraints must be qualified, i.e. their gradients must exist at the optimum. If for some reason during the modeling procedure a constraint is multiplied by itself then the constraint is not qualified anymore:

$$\tilde{g}_j = (g_j)^2 \quad \rightarrow \quad \nabla\tilde{g}_j = 2g_j\nabla g_j \quad \text{and} \quad \nabla\tilde{g}_j(\mathbf{x}^*) = 2g_j(\mathbf{x}^*)\nabla g_j(\mathbf{x}^*) = 0\,! \quad \text{since} \quad g_j(\mathbf{x}^*) = 0$$

As a consequence, the optimization algorithm may behave strange.

Another problem which can arise are linear dependent gradients of constraints. Then the Lagrange multipliers are not well-defined anymore. From a graphical interpretation it becomes clear that in this case a constraint became redundant and must be deleted from the active set, Fig. 4.14. Most algorithms are insensitive to this effect, because the active constraints are usually considered one after the other. Therefore, not more constraints than necessary are considered. (Compare with the 2 bar truss example 2.3 where these two effects arise for certain choices of problem parameters).

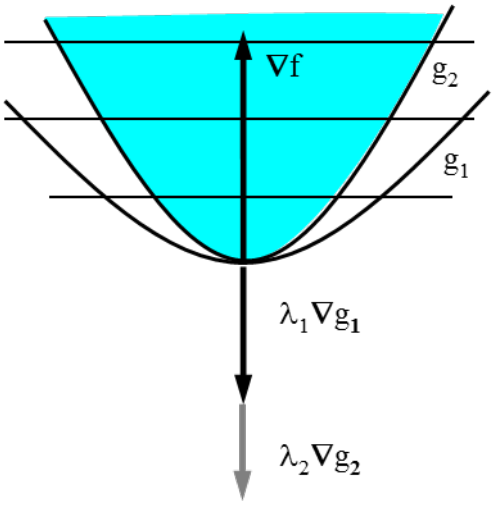


Fig. 4.14: Non-qualified constraints, Lagrange multipliers cannot be uniquely determined. Constraints are redundant.

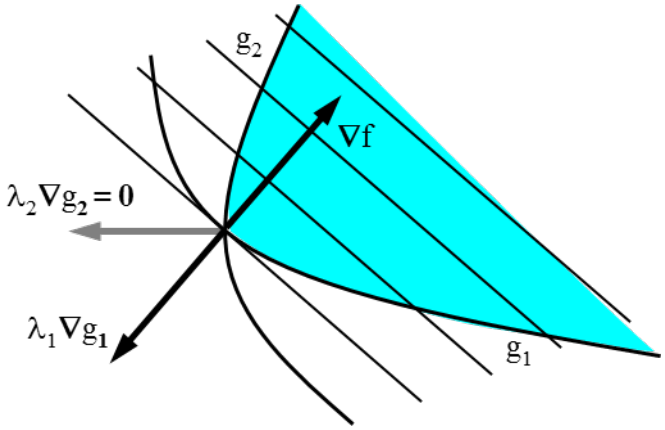


Fig. 4.15: Redundant constraint g_2: unnecessary to define the optimum.

4.6 Interpretation of the Lagrange multipliers, design sensitivity

The Lagrange multipliers tell also about the sensitivity of the objective at the optimum with respect to a modification of the constraint reference value, e.g. the admissible stress in a stress constraint. From the KKT stationary condition for **x** we can determine total derivatives of objective df and constraints dg_j:

$$\frac{\partial f}{\partial \mathbf{x}}\,d\mathbf{x} + \sum_{j=1}^{p} \lambda_j \frac{\partial g_j}{\partial \mathbf{x}}\,d\mathbf{x} = 0$$

$$df + \sum_{j=1}^{p} \lambda_j\, dg_j = 0 \;\rightarrow\; df_{(j)} = -\lambda_j\, dg_j \tag{4.27}$$

That means, that relaxing constraint g_j by an infinitesimal small amount dg_j results in a decrease (i.e. improvement) of the objective of $df_{(j)} = -\lambda_j\, dg_j$, Fig. 4.16. In other words, the Lagrange multiplier is a measure of how sensitive the optimal solution is with respect to a modification of the constraint. A large multiplier means a strong influence of the constraint and vice versa. However, one has to consider the dimension of the constraint. That means, constraints should

be normalized before multipliers are compared. Scaling of the constraint g_j by a factor scg_j results in an inverse scaling of the multiplier since the necessary condition is, of course, still valid:

$$\frac{\partial f}{\partial \mathbf{x}}\,d\mathbf{x}+\sum_{j=1}^{p}\lambda_j\frac{\partial g_j}{\partial \mathbf{x}}\,d\mathbf{x}=\frac{\partial f}{\partial \mathbf{x}}\,d\mathbf{x}+\sum_{j=1}^{p}\frac{\lambda_j}{scg_j}\frac{scg_j\,\partial g_j}{\partial \mathbf{x}}\,d\mathbf{x}=0 \tag{4.28}$$

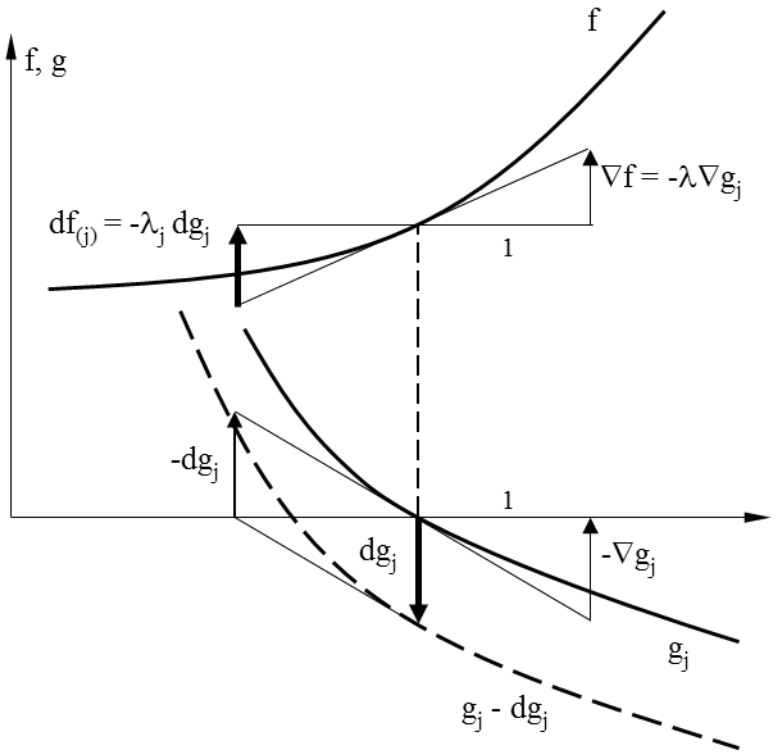


Fig. 4.16: Improvement $df_{(j)}$ of objective by infinitesimal relaxing dg_j of constraint g_j

Example. Consider weight minimization with respect to constraints on the admissible tensile stress of truss members. The stress σ determined as function of the optimization variable $\mathbf{x}$ is not allowed to be larger than the yield strength $\beta_{(1)y}$:

$$\sigma(\mathbf{x}) \le \beta_{(1)y} \quad \rightarrow \quad g(\mathbf{x}) = \sigma(\mathbf{x}) - \beta_{(1)y} \le 0 \tag{4.29}$$

For some reason the material shall be changed to a yield strength $\beta_{(2)y}$. The question is how this change will affect the optimal value of the objective. By comparison we determine the relaxation dg of the constraint:

$$\begin{aligned} g_{(1)}(\mathbf{x}) &= \sigma(\mathbf{x}) - \beta_{(1)y} \le 0 \\ g_{(2)}(\mathbf{x}) &= \sigma(\mathbf{x}) - \beta_{(2)y} \le 0 \\ dg &= g_{(1)} - g_{(2)} = \beta_{(2)y} - \beta_{(1)y} \end{aligned} \tag{4.30}$$

And the improvement of the objective function is determined as:

$$df = -\lambda\, dg = -\lambda\left(\beta_{(2)y} - \beta_{(1)y}\right) \tag{4.31}$$

A better material, i.e. $\beta_{(2)y} > \beta_{(1)y}$, obviously results in a lighter design since $\lambda > 0$ and, therefore, $df < 0$ which indicates a decrease. Note, that the sensitivity analysis is based on linearization. Sensitivity analysis indicates nothing more than a trend-information to detect the importance of constraints. Sensitivity analysis does not save an additional fully nonlinear optimization run.

4.7 Evaluation of Lagrange multipliers

If multipliers are not available after optimization they can be evaluated from the necessary KKT-conditions (4.25) which have to be set up for the active constraints:

$$\nabla f + \nabla \mathbf{g}^T \boldsymbol{\lambda} = 0 \tag{4.32}$$

In general, however, (4.32) gives n equations for only m active constraints. A solution can be achieved by selecting m equations or by a mean square approach:

$$\nabla \mathbf{g}\, \nabla f + \left(\nabla \mathbf{g}\, \nabla \mathbf{g}^T\right)\boldsymbol{\lambda} = 0 \tag{4.33}$$

Example. Again, consider the three bar truss problem. At the optimum we get the following information:

$$\begin{aligned} f &= 263.89 & \frac{\partial f}{\partial A_1} &= 282.2 & \frac{\partial f}{\partial A_2} &= 100.0 \\ g_1 &= -.1e-7 & \frac{\partial g_1}{\partial A_1} &= -21.436 & \frac{\partial g_1}{\partial A_2} &= -7.579 \\ g_2 &= -5.359 & g_3 &= -9.641 \end{aligned}$$

Obviously, constraints g_2 and g_3 are inactive, i.e. we can set up two equations for the remaining unknown Lagrange multiplier λ_1:

$$\begin{Bmatrix} 282.843 \\ 100.0 \end{Bmatrix} + \begin{Bmatrix} -21.436 \\ -7.579 \end{Bmatrix} \lambda_1 = 0$$

which gives by the mean square approach $\lambda_1 = 13.195$.

If the material in member no.1 would be exchanged such that the yield strength would be $\beta_y = 22$ instead of 20 the effect on the objective is:

$$\mathrm{df} = -\lambda_1 \left(\beta_{(2)} - \beta_{(1)}\right) = -13.195 \left(22.0 - 20.0\right) = -26.390$$

indicating a decrease of weight.

5 Algorithms

Typical structural optimization problems have three main problems why they are difficult to be solved: they are nonlinear and constrained, and the relevant constraints are unknown in advance. A one-step solution is in general impossible. The simplest idea to overcome the problem is to take the best solution from a couple of trial solutions. Another pragmatic strategy is to generate a series of intermediate solutions which converge to the final optimum. Naturally, while travelling from an initial to the optimal solution at each iteration-step two important questions arise: which direction shall be taken for the next step, and how far to go in this direction. The many available algorithms differ in how they get answers to these questions. This chapter will supply some knowledge about the background which is necessary to understand the behavior of these algorithms and to use them effectively.

5.1 Classification of algorithms

Algorithms may be distinguished with respect to two main categories:

- what kind of problem do they solve:
  unconstrained or constrained

- which kind of data do they use:
  function values only (*zero order methods* or *direct search methods*),
  or additionally
  gradient (*gradient or first order methods*), or even
  second order information (*second order* or *Newton type methods*)

As many strategies for constrained problems generate unconstrained sub-problems algorithms for the latter kind are important even for structural optimization. In the class of unconstrained problems we distinguish again between minimization methods made for one unknown only, *one-dimensional* methods, and those made for many variables, *n-dimensional methods*.

Methods for constrained problems are again subdivided with respect to the kind of variables (primal and/or dual variables) they prefer. We define four sub-classes:

- *Primal methods*, working directly in the n-dimensional space of the optimization variables **x**. They make no use of Lagrange multipliers and of the KKT-necessary conditions. The most simple algorithms are primal, direct search methods. Evolutionary strategies and genetic algorithms belong to these species. They are preferably useful to handle discrete variables. Successful primal methods which make use of gradient information are known as *method of feasible directions* or *general reduced gradient methods*.
- *Penalty*- and *Barrier-function methods*, working also in the n-dimensional space of the optimization variables. They, however, transform the constrained problem into an unconstrained. The approach is principally simple and quite robust. An old methodology is known as *SUMT* (*s*equential *u*nconstrained *m*inimization *t*echnique) which generates a series of unconstrained sub-problems to finally get a solution near to but not exactly at the optimum. That was one of the reasons why it became unpopular in the meantime. The basic idea, however, came back just recently as *interior point method*. The mathematical basis has been improved and SUMT has now been put into the frame of so called *continuation methods*.

- *Dual methods*, working primarily in the dual m-dimensional space of Lagrange multipliers $\boldsymbol{\lambda}$. The primal optimization variables $\mathbf{x}$ are determined by back substitution. Dual methods split the original optimization problem into two partial problems which have to be solved sequentially. One is unconstrained and formulated in terms of $\mathbf{x}$, the other is formulated in terms of $\boldsymbol{\lambda}$ and is only constrained by simple bounds. In the case of equality constraints it is unbounded too. Because of the simply structured sub-problems methods for unconstrained problems can be applied directly or with minor modifications to handle bounds on variables.
- *Lagrange methods*, working in the full (n+m)-dimensional space of primal and dual variables $\mathbf{x}$, $\boldsymbol{\lambda}$. They directly tackle the Karush-Kuhn-Tucker necessary conditions by solving a sequence of linearized sub-problems. Consistently derived these sub-problems are characterized by a quadratic objective and linear constraints. That's why these kinds of methods are called *SQP*- or *Sequential Quadratic Programming methods*. A simplified variant uses a linear approximation for the objective also and is called *SLP*- or *Sequential Linear Programming*. SQP-methods are considered to be one of the most or even the most sophisticated methods from the mathematical point of view. They have been successfully applied for many structural optimization tasks and are available in almost every structural optimization package. However, they appear to be not robust enough for very large problems. Research on the field is still in progress.

In addition to the mentioned classification there exist many special algorithms which can be associated to any of the classes.

Approximation and *response surface methods* which have been become very popular in the last years do not fit into this frame. Instead, they have to be understood as part of the modeling procedure prior to the problem solution process. Nevertheless, they will be discussed within this chapter because for efficiency they are designed with strong relations to special solution algorithms.

Last not least, *optimality criteria methods* or shorter *OC-methods* should be mentioned. OC-methods state some criteria which the solution has to satisfy to be regarded as optimal. If these criteria coincide with the Karush-Kuhn-Tucker conditions OC-methods have strong similarities to dual methods. However, they differ from several algorithmic aspects. Historically, they have their origin in a heuristic approach to optimization. A well-known example is the *fully stressed design, FSD*. This criterion claims that a structure is optimal if all members are fully stressed. From this requirement very effective recursive procedures can be derived to determine the optimal state of variables. The well-known 3 bar truss example, however, is a famous counterexample to the FSD approach. Applied to this example FSD converges to a non-optimal solution. In general, OC-methods are made for special applications. They are very efficient for these problems also for many variables.

Table 5.1. Classification of algorithms.

Classification of non-linear programming algorithms						
	unconstrained problems		methods for constrained problems			
	1-D minimization	n-D minimization	primal methods	Penalty methods	dual methods	Lagrange methods
zero order	interval search	grid search	grid search	These methods transform constrained problems into unconstrained.	Dual methods split the original problem into a unconstrained w.r.t. the primal variables **x** and a constrained problem w.r.t. λ with bounds on the variables only.	
	polynomial interpolation	Monte Carlo random search	Monte Carlo random search			
		Gauss/Seidel	Complex method			
		Powell's method, Hooke-Jeeves' method	evolutionary strategy			
		simplex method of Nelder and Mead	genetic algorithms			
first order	polynomial interpolation	steepest descent	feasible directions			SQP, quasi Newton
	bisection	conjugate gradient, (Fletcher, Reeves)	generalized reduced gradients			SLP
		quasi Newton, BFGS, DFP				
second order	Newton method	Newton method				SQP, full Newton
		conjugate gradient, (Hestenes, Stiefel)				

5.2 1-D Minimization, line search

One dimensional minimization is an important part of almost all multi-dimensional optimization strategies. Line search is used to determine or correct the step size of descent algorithms. Often it is enough to ensure that the objective is decreased by a line search rather than to exactly evaluate the line minimum. Because of this reason we will roughly sketch the ideas of line search and do not go too much into details. It is referred to the literature for further detailed information.

Interval search. The idea is to reduce successively the interval of uncertainty to a small acceptable value. The simplest strategy is start with equal sized intervals. That interval that contains the minimum will be further reduced in subsequent steps until the interval size reaches an acceptable small value.

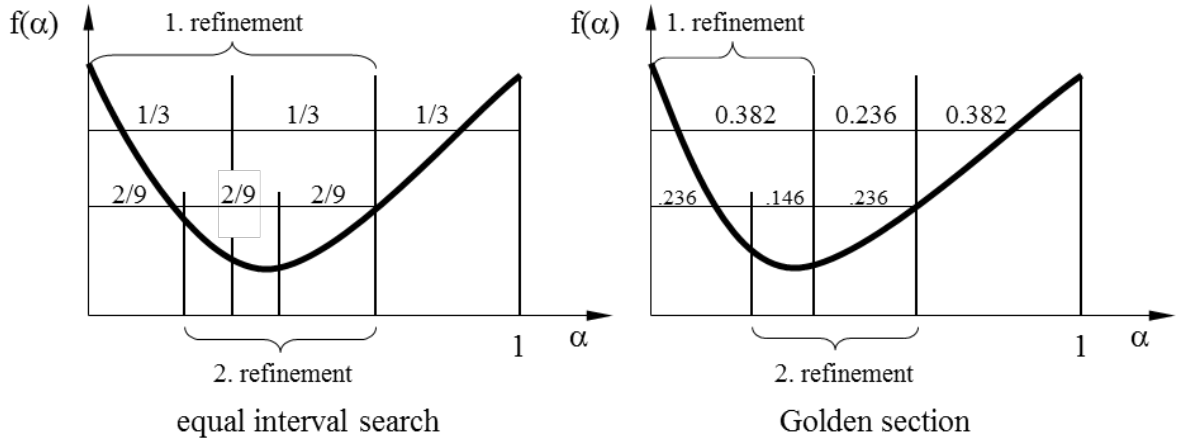


Fig. 5.1: Interval search methods.

Starting with initially three intervals the function is evaluated at the interval boundaries. Those two intervals will further be refined in the next step which contains the line minimum. The procedure is repeated until convergence is accepted. For the equal interval search the function has to be evaluated at two additional points, for the example in the above figures at $\alpha = 2/9$ and $\alpha = 4/9$. The Golden Section methods needs only one additional evaluation because of the special relations of the interval sizes. Note that:

$$\frac{0.236}{0.382} = \frac{0.146}{0.236} = \frac{0.236 - 0.146}{0.146} = \frac{1}{2}\left(\sqrt{5} - 1\right) = 0.618 \tag{5.1}$$

Polynomial interpolation. Polynomial interpolation is preferred if gradient information is available and the line search is used to ensure that the function is sufficiently decreased. The often used quadratic interpolation takes function f_0 and gradient data f_0' at $\alpha = 0$ and the function value f_1 at $\alpha = 1$. The location α_m of the line minimum is approximated as, Fig. 5.2:

$$\alpha_m = \frac{f_0'}{2\left(f_0' + f_0 - f_1\right)} \tag{5.2}$$

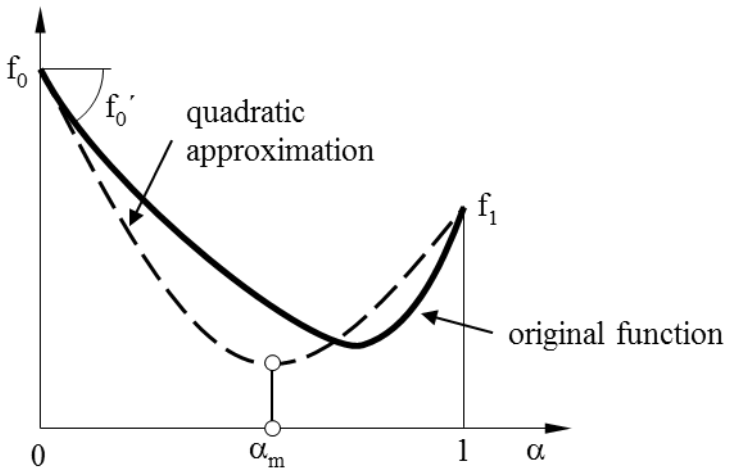


Fig. 5.2: Quadratic curve fitting.

Armijo test. In particular together with quasi Newton methods an exact line search is not necessary. To check that the function value at $\alpha = 1$ is sufficiently smaller than at $\alpha = 0$ will suffice. The Armijo test is satisfied if $f(\alpha=1)$ is below the secant shown in Fig. 5.3. The slope of the secant is defined as the ε multiple of the slope f' at $\alpha = 0$. Often, $\varepsilon = 0.2$ is recommended. If the test fails a quadratic fit is made to approximately determine the line minimum. The test is repeated until it is satisfied.

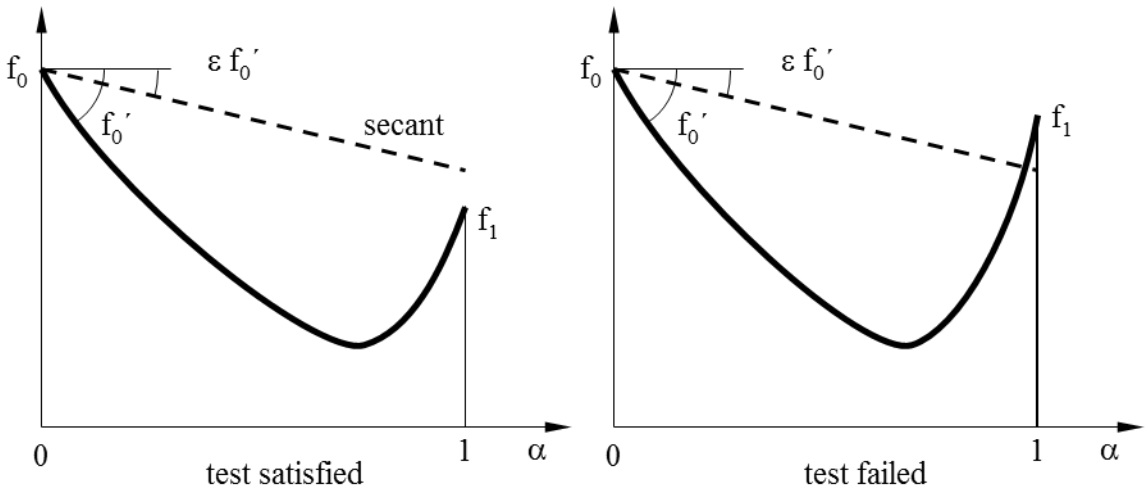


Fig. 5.3: Armijo test.

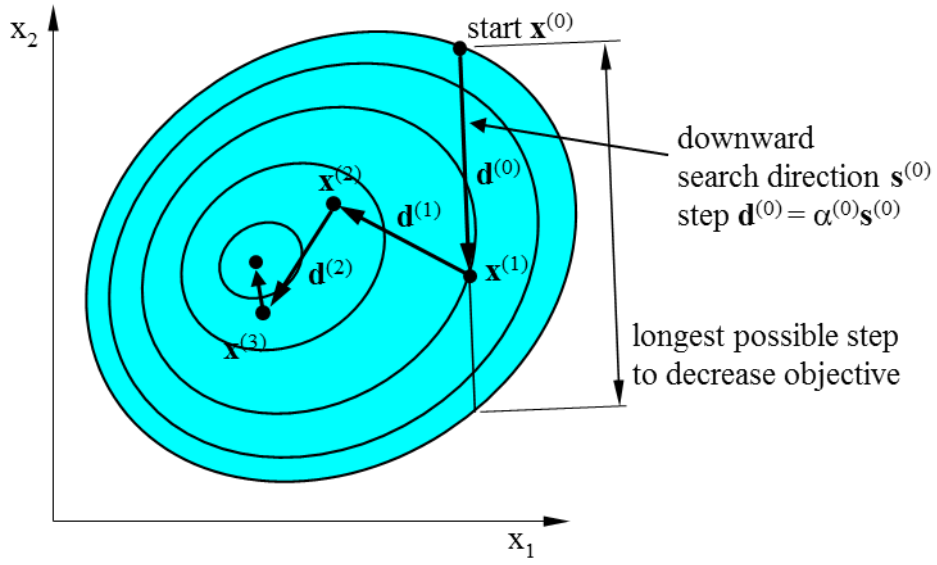
5.3 Descent methods

Most optimization methods are representatives of descent methods no matter what kind of data they use. The basic idea is simple:

- start at a point inside the feasible domain
- choose a descent search direction **s**
- follow this direction such that the function value is sufficiently decreased, however, keep inside the feasible domain
- choose a new search direction and repeat the procedure until the optimum is found

How search directions are determined and how large the step **d** in this direction is made is the secret of the distinct methods. The condition for the step size α is:

$$f\left(\mathbf{x}^{(k)}\right) > f\left(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}\right); \quad \mathbf{d}^{(k)} = \alpha^{(k)}\mathbf{s}^{(k)} \tag{5.3}$$



5.4 Direct search methods

Direct search methods use function values only. They do not evaluate higher order data as gradients or curvature. Therefore, they are preferred for highly discontinuous problems. If the gradients can be determined direct search methods are usually inefficient. Note, however, that the computing time for gradient evaluation dominates the overall process time for most structural optimization problems which for some cases might bring clever direct search methods back into business.

Grid search. That is the simplest approach to optimization one can think of. It works for unconstrained as well as inequality constrained problems. A sufficiently large subset of the n-dimensional space of variables is evaluated at regularly spaced grid points. The best solution, which satisfies all constraints and gives the smallest objective value, is taken as solution. The method is very inefficient. The computing effort increases exponential with the number of variables. The effort can be reduced by randomly generated points (*Monte Carlo method*). The advantage of these methods is that for non-convex problems there exists a chance that they can find the global optimum, however, for a very high price, Fig. 5.4.

Successive coordinate search. This basic descent method successively searches in each coordinate direction. The step size is determined by a line search close to the line minimum, Fig. 5.5.

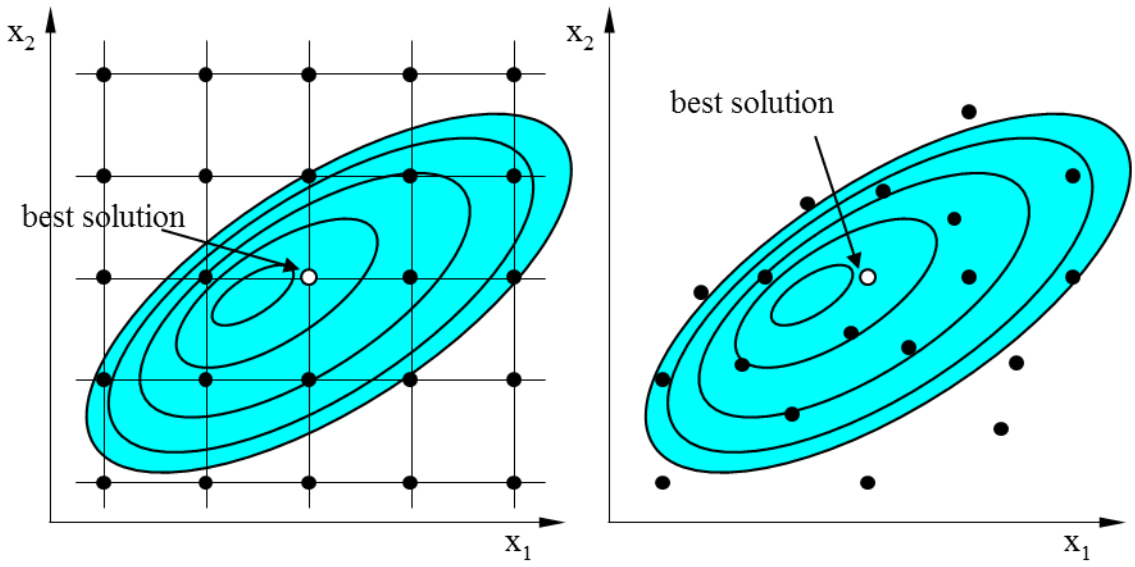


Fig. 5.4: Grid-search (left) and direct search at randomly generated points (right).

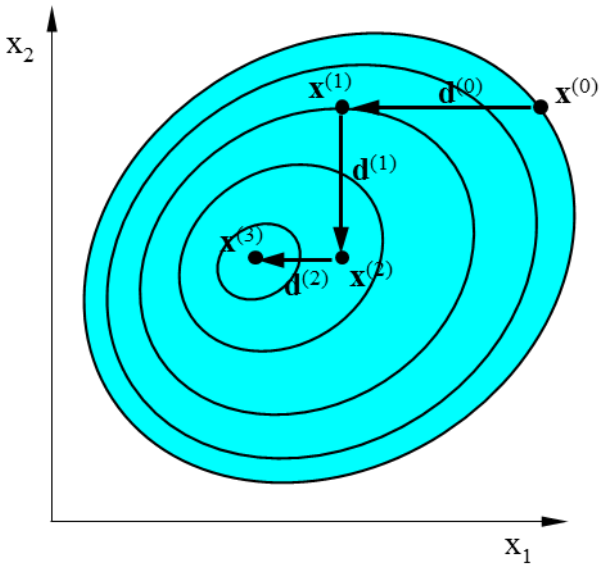


Fig. 5.5: Successive coordinate search.

5.5 Gradient methods

5.5.1 Steepest descent

The steepest descent method is the most basic gradient method for unconstrained problems. It takes as search direction the negative function gradient which is the steepest descent direction. The step size is determined as the line minimum by a line search, Fig. 5.6. The favorite property of the steepest descent method is that the function value is guaranteed to be decreased by each step. Because of this property this method is used by many other methods for a so called *spacer step* to ensure convergence. However, the steepest descent exhibits “zigzagging” for badly conditioned problems.

Subsequent search directions are orthogonal to each other, it holds:

$$\mathbf{s}^{(k)} = -\nabla f\left(\mathbf{x}^{(k)}\right)$$
$$\mathbf{s}^{(k)} \cdot \mathbf{s}^{(k+1)} = 0 \tag{5.4}$$

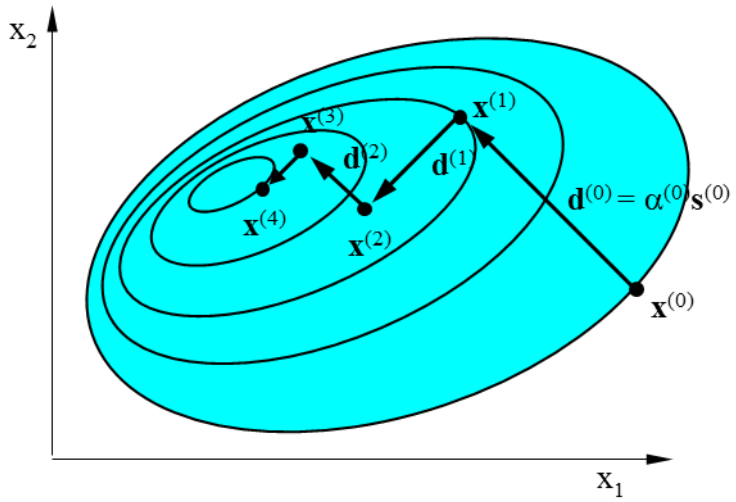


Fig. 5.6: Steepest descent.

Example. Let us discuss the zigzagging behavior of steepest descent. Consider the following quadratic functions and initial values of x and y:

function	initial values, $x^{(0)}$, $y^{(0)}$		number of iterations
$f(x,y) = x^2 + 10y^2$	100.0	10.0	62
$f(x,y) = x^2 + 4y^2$	100.0	10.0	15
$f(x,y) = 4x^2 + 4y^2$	100.0	10.0	1

The curvature of the objective functions influences the number of iterations dramatically. The "narrower" the function valley the more the method tends to oscillate although the type of function does not change, Fig. 5.7. From that observation we may create the idea of artificially "widening" the function by a variable transformation. Let us define a pair of alternative variables $\tilde{x}$ and $\tilde{y}$ by introducing the scaling matrix **T**:

$$\mathbf{x} = \begin{Bmatrix} x \\ y \end{Bmatrix} = \mathbf{T}\tilde{\mathbf{x}} = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} \begin{Bmatrix} \tilde{x} \\ \tilde{y} \end{Bmatrix} \tag{5.5}$$

and write the quadratic function in terms of these:

$$f(\tilde{x},\tilde{y}) = a\,T_1^2\,\tilde{x}^2 + b\,T_2^2\,\tilde{y}^2 = \tilde{a}\,\tilde{x}^2 + \tilde{b}\,\tilde{y}^2 \tag{5.6}$$

In agreement to our observation the best choice of the scaling parameters T_1 and T_2 is such that $\tilde{a} = \tilde{b}$:

$$\tilde{a} = c \quad \rightarrow \quad T_1 = \sqrt{\frac{c}{a}} \qquad \text{and} \qquad \tilde{b} = c \quad \rightarrow \quad T_2 = \sqrt{\frac{c}{b}} \tag{5.7}$$

For the choice of the remaining constant c we compare **T** with the Hessian of f:

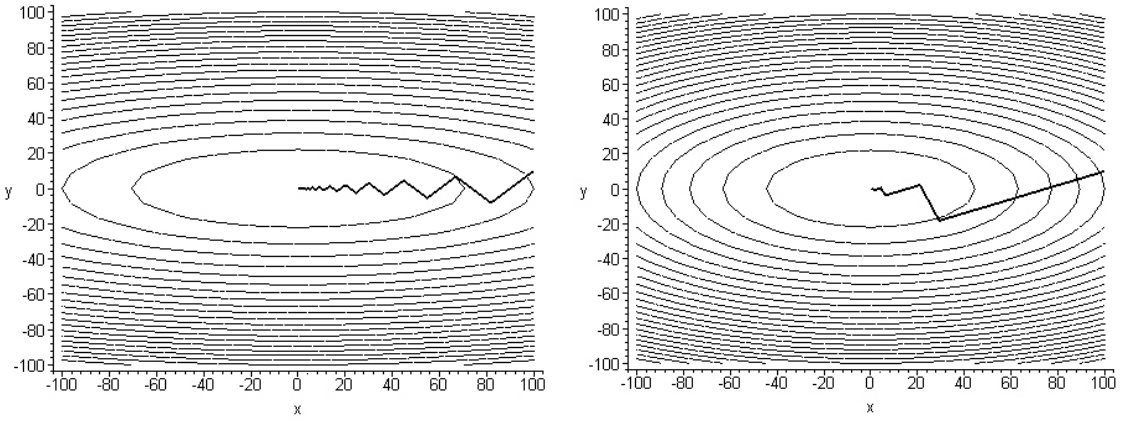
$$\nabla^2 f = \mathbf{H} = \begin{bmatrix} 2a & 0 \\ 0 & 2b \end{bmatrix} \tag{5.8}$$

and realize that for c = 1/2 the scaling matrix **T** is determined by the coefficients of the Hessian **H**:

$$\mathbf{T} = \begin{bmatrix} \frac{1}{\sqrt{H_{11}}} & 0 \\ 0 & \frac{1}{\sqrt{H_{22}}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2a}} & 0 \\ 0 & \frac{1}{\sqrt{2b}} \end{bmatrix} \tag{5.9}$$

In terms of $\tilde{x}$ and $\tilde{y}$ the optimization is done in one iteration. The solution is then transformed back to the original variables x and y by the inverse operation of (5.5).

We can learn from that example that the speed of algorithms can tremendously be influenced by proper variable scaling. That applies in principle to all gradient and quasi Newton methods. It is practically not useful to use the Hessian as basis for the scaling matrix **T** because the calculation of the Hessian takes far too much effort. However, we consider this effect during the modeling stage when we set up the problem. As guidance it is advisable to scale the variables with respect to their typical order of dimension to get normalized variables.



$f(x,y) = x^2 + 10\,y^2$ $\qquad$ $f(x,y) = x^2 + 4\,y^2$

Fig. 5.7: Zigzagging of the steepest descent method.

Line search. The line search is performed to determine the line minimum. For a quadratic function this can be done analytically. For given

quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\,\mathbf{x} + \mathbf{a}^T\mathbf{x}$, the solution $\mathbf{x}_k$ in iteration step k, and a search direction $\mathbf{s}_k$ the function in direction of $\mathbf{s}_k$ is determined as:

$$\begin{aligned} f(\mathbf{x}_k + \alpha\,\mathbf{s}_k) &= f(\mathbf{x}_k) + \alpha\,\mathbf{g}_k^T\mathbf{s}_k + \frac{1}{2}\alpha^2\,\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k \\ \text{with} \qquad \mathbf{g}_k &= \nabla f(\mathbf{x}_k) = \mathbf{Q}\,\mathbf{x}_k + \mathbf{a} \end{aligned} \tag{5.10}$$

The step size α_m is determined as:

$$\begin{aligned} \frac{df(\mathbf{x}_k + \alpha_m\,\mathbf{s}_k)}{d\alpha} &= \mathbf{g}_k^T\mathbf{s}_k + \alpha_m\,\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k = 0 \\ \rightarrow \quad \alpha_m &= -\frac{\mathbf{g}_k^T\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k} \\ \text{and} \qquad \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_m\,\mathbf{s}_k \\ \text{or} \qquad \mathbf{d_k} &= \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_m\,\mathbf{s}_k \end{aligned} \tag{5.11}$$

For steepest descent the search direction $\mathbf{s}_k$ is taken as the negative gradient $-\mathbf{g}_k$.

5.5.2 Conjugate gradient method

The conjugate gradient method takes into account the curvature of the problem. It generates improved search directions and converges faster than the steepest descent method. For the case of quadratic problems the method converges in maximum n steps (n is number of variables). A quadratic problem with two variables (x, y) converges in maximal two iteration steps, Fig. 5.8. With respect to steepest descent the search direction is modified by:

$$\begin{aligned} \mathbf{s}_{k+1} &= -\mathbf{g}_{k+1} + \beta\,\mathbf{s}_k \\ \beta &= \frac{\mathbf{g}_{k+1}^T\mathbf{Q}\,\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k} \end{aligned} \tag{5.12}$$

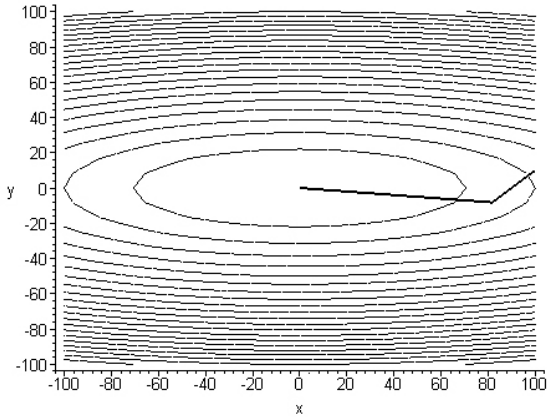
considering the previous search direction $\mathbf{s}_k$ scaled by the factor β. The criterion for the determination of β is that subsequent search directions are conjugate with respect to $\mathbf{Q}$:

$$\mathbf{s}_{k+1}^T\mathbf{Q}\,\mathbf{s}_k = 0 \tag{5.13}$$

An “exact” line search is performed to determine the line minimum. For quadratic problems (5.11) is used. If gradient methods are applied to non-quadratic problems the line search must be done numerically by one of the demonstrated procedures.

The expression for β has been rephrased by Fletcher and Reeves, which is identical to (5.12_2) for quadratic problems, exact line searches and if $\mathbf{s}_0$ is taken as the negative gradient at $\mathbf{x}_0$:

$$\beta = \frac{\mathbf{g}_{k+1}^T\mathbf{g}_{k+1}}{\mathbf{g}_k^T\mathbf{g}_k} \tag{5.14}$$



$f(x,y) = x^2 + 10\,y^2$
initial solution (x,y) = (100.0, 10.0)

convergence in two steps

Fig. 5.8: Conjugate gradient method.

5.5.3 Demonstration of the Fletcher Reeves formula for β

We refer to the quadratic problem (5.10) and (5.12):

$$f(\mathbf{x}_k) = \frac{1}{2}\mathbf{x}_k^T\mathbf{Q}\,\mathbf{x}_k + \mathbf{a}^T\mathbf{x}_k$$
$$\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \mathbf{Q}\,\mathbf{x}_k + \mathbf{a}$$
$$\mathbf{s}_k = -\mathbf{g}_k + \beta\,\mathbf{s}_{k-1}$$

Performing an exact line search the next iterate $\mathbf{x}_{k+1}$ is generated as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{s}_k$$
$$\mathbf{g}_{k+1} = \mathbf{Q}\,\mathbf{x}_{k+1} + \mathbf{a} = \mathbf{Q}(\mathbf{x}_k + \alpha_k\mathbf{s}_k) + \mathbf{a} = \mathbf{g}_k + \alpha_k\mathbf{Q}\,\mathbf{s}_k$$

with $$\alpha_k = -\frac{\mathbf{g}_k^T\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k} = \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k} \qquad (5.15)$$

$$\mathbf{g}_{k+1} = \mathbf{g}_k + \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k}\mathbf{Q}\,\mathbf{s}_k$$

and $$\mathbf{g}_k^T\mathbf{s}_k = -\mathbf{g}_k^T\mathbf{g}_k + \beta\underbrace{\mathbf{g}_k^T\mathbf{s}_{k-1}}_{=0} = -\mathbf{g}_k^T\mathbf{g}_k$$

Because of the exact line search the actual gradient $\mathbf{g}_k$ is orthogonal to the previous search direction $\mathbf{s}_{k-1}$.

Next, we treat the expression for the next gradient $\mathbf{g}_{k+1}$ (5.15$_4$) and can show the equivalence of (5.12) and (5.14):

$$\frac{\mathbf{g}_{k+1}^T}{\mathbf{g}_k^T\mathbf{g}_k}\cdot\Bigg| \quad \mathbf{g}_{k+1} = \mathbf{g}_k + \frac{\mathbf{g}_k^T\mathbf{g}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k}\mathbf{Q}\,\mathbf{s}_k$$

$$\frac{\mathbf{g}_{k+1}^T\mathbf{g}_{k+1}}{\mathbf{g}_k^T\mathbf{g}_k} = \frac{\overbrace{\mathbf{g}_{k+1}^T\mathbf{g}_k}^{=0}}{\mathbf{g}_k^T\mathbf{g}_k} + \frac{\mathbf{g}_{k+1}^T\mathbf{Q}\,\mathbf{s}_k}{\mathbf{s}_k^T\mathbf{Q}\,\mathbf{s}_k} = \beta$$

It remains to prove that subsequent gradients $\mathbf{g}_k$ and $\mathbf{g}_{k+1}$ are orthogonal. Again, we treat the expression for the next gradient $\mathbf{g}_{k+1}$ (5.15$_4$):

$$\mathbf{g}_k^T \cdot \Big| \qquad \mathbf{g}_{k+1} = \mathbf{g}_k + \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{s}_k^T \mathbf{Q}\, \mathbf{s}_k} \mathbf{Q}\, \mathbf{s}_k$$

$$\mathbf{g}_k^T \mathbf{g}_{k+1} = \mathbf{g}_k^T \mathbf{g}_k + \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{s}_k^T \mathbf{Q}\, \mathbf{s}_k} \mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_k = 0$$

$$\text{if} \qquad \mathbf{s}_k^T \mathbf{Q}\, \mathbf{s}_k = -\mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_k$$

The latter can be shown as follows:

$$\text{from} \qquad \mathbf{s}_k = -\mathbf{g}_k + \beta_k\, \mathbf{s}_{k-1}$$

$$\text{and} \qquad \beta_k = \frac{\mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_{k-1}}{\mathbf{s}_{k-1}^T \mathbf{Q}\, \mathbf{s}_{k-1}}$$

$$-\mathbf{s}_k^T \mathbf{Q}\, \mathbf{s}_k = -\mathbf{g}_k^T \mathbf{Q}\, \mathbf{g}_k + 2\beta_k\, \mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_{k-1} - \beta_k^2\, \mathbf{s}_{k-1}^T \mathbf{Q}\, \mathbf{s}_{k-1}$$

$$= -\mathbf{g}_k^T \mathbf{Q}\, \mathbf{g}_k + \beta_k\, \mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_{k-1} + \beta_k \underbrace{\left[\mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_{k-1} - \beta_k\, \mathbf{s}_{k-1}^T \mathbf{Q}\, \mathbf{s}_{k-1}\right]}_{=0}$$

$$= -\mathbf{g}_k^T \mathbf{Q}\, \mathbf{g}_k + \beta_k\, \mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_{k-1} = \mathbf{g}_k^T \mathbf{Q}\, \mathbf{s}_k$$

which completes the proposition.

5.5.4 The method of feasible directions

This method is the extension of steepest descent or conjugate gradients to constrained problems. It is widely used in structural optimization and known to be quite robust. The method is intended to start inside the feasible domain. As long as the solution keeps inside there is no difference to one of the latter methods. If during the iteration the boundary is hit the next search direction must go back into the feasible domain. It has to satisfy two conditions:

(i) it must be feasible (i.e. keeping inside the feasible domain):

$$\nabla \mathbf{g}(\mathbf{x}_0)^T \cdot \mathbf{s} \le 0 \tag{5.16}$$

(ii) usable (i.e. it must reduce the objective):

$$\nabla \mathrm{f}(\mathbf{x}_0)^T \cdot \mathbf{s} \le 0 \tag{5.17}$$

Consequently, the direction must be inside the cone which is limited by the tangents at the domain's boundary and the objective contour line at the boundary intersection, Fig. 5.9. Directions close to the constraint tangent are not desirable since a small move in this direction would violate the constraint. Assuming that the constraint is nonlinear and convex the search direction must additionally be pushed away from the constraint. That is done by adding an additional positive push-off factor θ to modify 5.16:

$$\nabla \mathbf{g}(\mathbf{x}_0) \cdot \mathbf{s} + \theta \le 0 \tag{5.18}$$

This ensures that the angle between the two directions exceeds 90°. For the convergence behavior it is important how large θ is chosen, e.g.:

$$\theta = \left[1 - \frac{g(\mathbf{x})}{\varepsilon}\right]^2; \qquad -0.1 \le \varepsilon \le -0.001 \tag{5.19}$$

For linear constraints take ε close to zero or set θ directly to zero. A known problem of the method is zigzagging along the constraint which alternately becomes active and non-active depending on whether the search is pushed away or comes back again.

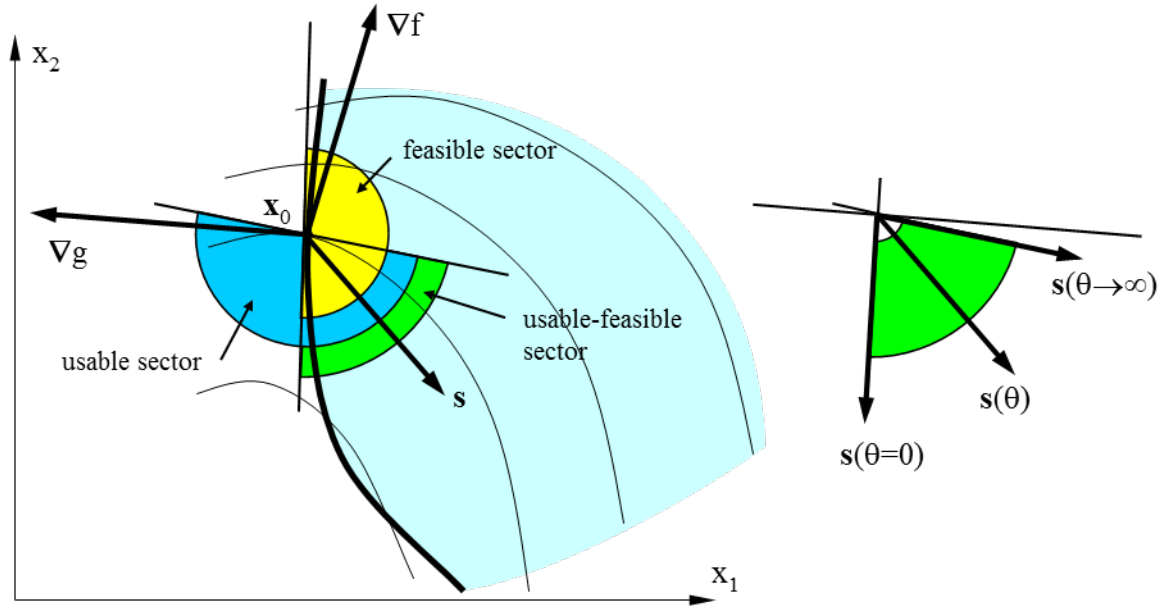


Fig. 5.9: Usable-feasible search direction.

5.5.5 Other gradient methods

Other successful gradient methods are the *generalized reduced gradient method, (GRG)* and the *modified feasible direction method*. Both are able to travel along the active constraint without zigzagging. GRG transforms inequality constraints into equality constraints by introducing so called *slag variables*. The number of variables is increased by the number of inequality constraints. Because of that reason GRG is said to be inefficient for structural optimization.

5.6 Newton and quasi Newton methods for unconstrained problems

Newton methods are derived from linearization of the stationary condition, which is in the unconstrained case:

$$\nabla f(\mathbf{x}) = 0 \tag{5.20}$$

Linearization at $\mathbf{x}_k$ yields:

$$\begin{aligned} \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{d}_k &= 0 \\ \text{with} \qquad \mathbf{d}_k &= \mathbf{x}_{k+1} - \mathbf{x}_k \end{aligned} \tag{5.21}$$

From which the step $\mathbf{d}_k$ to the next solution point $\mathbf{x}_{k+1}$ is determined as:

$$\mathbf{d}_k = -\left[\nabla^2 f(\mathbf{x}_k)\right]^{-1} \nabla f(\mathbf{x}_k) = -\alpha\, \mathbf{S}_k\, \mathbf{g}_k \tag{5.22}$$

where $\mathbf{g}_k$ is the gradient of f at the point $\mathbf{x}_k$ of development and α is a line search parameter.

The generalization of (5.22) allows for the generation of a broad family of algorithms by the specific choice of **S** and α, e.g.:

- Newton-Raphson method: $\mathbf{S}_k = \left[\nabla^2 f(\mathbf{x}_k)\right]^{-1}$ and $\alpha = 1$
- steepest descent: $\mathbf{S}_k = \mathbf{I}$ and α from an exact line search

The classical Newton-Raphson method determines step direction and step length at one calculation. Close to the optimal solution it exhibits a quadratic rate of convergence. A line search can be added to guarantee convergence in extreme situations also. The disadvantage is that the Hessian must be evaluated and inverted in each step which is far to expensive for most of the cases in structural optimization.

Quasi Newton methods substitute for $\mathbf{S}_k$ a matrix which approximates the inverse of the Hessian but is successively generated by first order information.

Second order derivatives are approximated by a difference ratio of gradients. First, consider the one-dimensional case:

$$\frac{d^2 f(x_{k+1})}{dx^2} \cong B_{k+1} = \frac{\dfrac{df(x_{k+1})}{dx} - \dfrac{df(x_k)}{dx}}{\Delta x_k} \quad \rightarrow \quad \Delta x_k\, B_{k+1}\, \Delta x_k = \left(\frac{df(x_{k+1})}{dx} - \frac{df(x_k)}{dx}\right) \Delta x_k \tag{5.23}$$

For n dimensions that yields:

$$\mathbf{d}_k^T \mathbf{B}_{k+1} \mathbf{d}_k = (\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{d}_k \; ; \quad \mathbf{g}_k = \nabla f(\mathbf{x}_k) \tag{5.24}$$

where the difference ratio is taken at beginning and end of the step $\mathbf{d}_k$. The matrix $\mathbf{B}_k$ approximates the Hessian from which an update formula for $\mathbf{S}_k$ can be determined. The best known quasi Newton schemes are *BFGS* and *DFP* which are named after their inventors Broyden, Fletcher, Goldfarb, Shanno and Davidon, Fletcher and Powell, respectively. They are also called *rank 2 update schemes* because the matrix **B** is updated by the sum of two dyadic vector products:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \mathbf{u}\,\mathbf{u}^T + \mathbf{v}\,\mathbf{v}^T \tag{5.25}$$

Among other considerations the proper choice of **u** and **v** is motivated by the quasi Newton secant condition (5.24) and the requirement that subsequent search directions must be conjugated.

As initial guess the matrix $\mathbf{B}_0$ is set to the identity **I**. The first step is, therefore, comparable to steepest descent with the difference of an inexact line search. Subsequent search directions are conjugated, i.e. the method has a strong relation to gradient methods. One of the advantages is

that the line search must not be exact. An Armijo test is enough. Once in a while a steepest descent step might be necessary to guarantee convergence. Most of the common SQP-methods are based on the BFGS update scheme.

5.7 SQP, Sequential Quadratic Programming

SQP is a second order method for constrained problems. Starting point is the linearization of the stationary conditions which now are the Karush-Kuhn-Tucker conditions. For simplicity assume in the sequel equality or active inequality constraints only. The KKT-conditions are:

$$\begin{aligned} \nabla_{\mathbf{x}} L(\mathbf{x},\boldsymbol{\lambda}) &= f(\mathbf{x}) + \nabla \mathbf{g}(\mathbf{x})^T \boldsymbol{\lambda} = 0 \\ \nabla_{\boldsymbol{\lambda}} L(\mathbf{x},\boldsymbol{\lambda}) &= \mathbf{g}(\mathbf{x}) \qquad\qquad = 0 \end{aligned} \tag{5.26}$$

Linearization at ($\mathbf{x}_k$, $\boldsymbol{\lambda}_k$) yields:

$$\begin{aligned} &\nabla f(\mathbf{x}_k) + \nabla \mathbf{g}(\mathbf{x}_k)^T \boldsymbol{\lambda}_k + \nabla^2 f(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + \\ &\qquad \boldsymbol{\lambda}_k^T \nabla^2 \mathbf{g}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) + \nabla \mathbf{g}(\mathbf{x}_k)^T (\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) = 0 \\ &\mathbf{g}(\mathbf{x}_k) + \nabla \mathbf{g}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) \qquad\qquad = 0 \end{aligned} \tag{5.27}$$

Which can be written in matrix notation:

$$\begin{bmatrix} \nabla^2 L_k & \nabla \mathbf{g}(\mathbf{x}_k)^T \\ \nabla \mathbf{g}(\mathbf{x}_k) & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{d}_k \\ \boldsymbol{\lambda}_{k+1} \end{Bmatrix} = -\begin{Bmatrix} \nabla f(\mathbf{x}_k) \\ \mathbf{g}(\mathbf{x}_k) \end{Bmatrix} \tag{5.28}$$

where we use the abbreviations:

$$\mathbf{d}_k = \mathbf{x}_{k+1} - \mathbf{x}_k \quad \text{and} \quad \nabla^2 L_k = \nabla^2 f(\mathbf{x}_k) + \boldsymbol{\lambda}_k^T \nabla^2 \mathbf{g}(\mathbf{x}_k) \tag{5.29}$$

Eq. 5.28 is the extension of 5.22 for constraints. Using a quadratic programming method for the solution of (5.28) the approach can easily be extended for inequality constraints also. Because the quadratic sub-problem has to be solved repeatedly for each iteration step the overall procedure got its name: *Sequential quadratic programming*. The quadratic sub-problem is:

$$\begin{aligned} \text{minimize} \quad & \nabla^T f(\mathbf{x}_k)\mathbf{d}_k + \frac{1}{2}\mathbf{d}_k^T \nabla^2 L_k \mathbf{d}_k \\ \text{such that} \quad & \nabla \mathbf{g}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{g}_k \le 0 \\ & \nabla \mathbf{h}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{h}_k = 0 \end{aligned} \tag{5.30}$$

Instead of the Hessian $\nabla^2 L_k$ of the Lagrangian, usually, a quasi-Newton approximation is used, preferable BFGS. As a consequence a line search becomes necessary. For unconstrained problems the objective is used as descent function in the line search. SQP methods, however, tackle constrained problems. They are working in the (n+m)-dimensional space of primal and dual variables ($\mathbf{x}$, $\boldsymbol{\lambda}$), looking for solutions which satisfy the stationary conditions of the Lagrangian. The Lagrangian, however, cannot be used as descent function for the line search. Remember that the optimum is a saddle point of the Lagrangian, not a minimum which would be necessary for a successful line search procedure. Because of that an auxiliary descent function is used which is called *merit function* which should have a minimum at the optimum or at least close

to it. Most important is that the merit function must be convex near to the optimum. All kinds of Penalty functions are usually used. All of them represent a compromise which implies the possibility of failure of the SQP-method. Fortunately, that happens rarely. However, it does occur and the codes report these strange error reports that a merit function has caused problems. What helps is to restart the problem at the current solution or to reformulate the problem with respect to scaling of variables and objective function.

Scaling of variables. We had already discussed the importance of variable scaling with respect to the convergence behavior of steepest descent. The reason was the approximation of the Hessian as identity matrix, as we just learned from the quasi Newton theory. Since BFGS is as well starting with an identity matrix as approximation of the Hessian the same reasons for variable scaling apply. Experience tells that normalization with respect to the characteristic order of the variables dimension is a good choice, e.g. the inverse of initial values of variables can be taken as scaling factors.

Scaling of the objective. When discussing the steepest descent method we had also realized that the relative scaling of variables is important. We still had the choice of additional uniform scaling by that factor c. Uniform scaling of all variables by the factor c transforms a quasi-Newton approximation as:

$$\begin{aligned} f_{BFGS} &= \frac{1}{2}\mathbf{d}^T\mathbf{B}\,\mathbf{d} + \nabla f^T\mathbf{d} \\ \tilde{f} &= \frac{1}{2}c^2\,\mathbf{d}^T\mathbf{B}\,\mathbf{d} + c\nabla f^T\mathbf{d} \end{aligned} \tag{5.31}$$

As c is still not determined the following is in principle as good as the previous:

$$\hat{f} = \frac{1}{2}\mathbf{d}^T\mathbf{B}\,\mathbf{d} + \frac{1}{c}\nabla f^T\mathbf{d} \tag{5.32}$$

This states, that dividing the function by c has the same effect as multiplying all variables by the same factor c. In other words, scaling of the function and in turn of the function gradient adjusts the function gradient to the approximation **B** of the Hessian. This directly affects the step size at the beginning of the iteration when **B** has not yet been adjusted to the problem. As

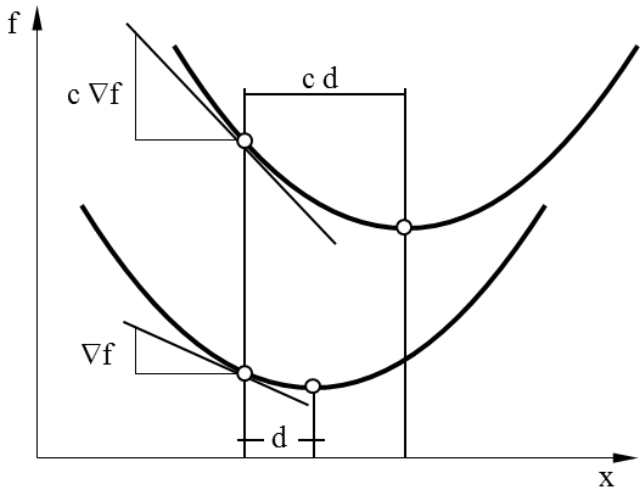


Fig. 5.10:
Effect of objective function scaling on quadratic approximation with constant curvature.

Scaling by factor c enlarges the step to the line minimum by c.

an important consequence the number of additional function calls is affected by objective scaling. It is worthwhile to think about proper objective scaling during the modeling stage. It decides about the efficiency of the solution procedure. Despite of all theoretical complexity SQP methods have been proved as reliable and effcient tools in structural optimization for small to medium size problems.

5.8 SLP, Sequential Linear Programming

Sequential linear programming might also been seen as a Lagrange method because it deals simultaneously with both kinds of variables, the primal and the dual ones. It shares with SQP the idea of directly solving the Karush-Kuhn-Tucker necessary conditions by subsequent iteration steps. Unlike to SQP it generates a linear sub-problem which can be derived from (5.30) by deleting the quadratic term in the objective and adding an additional trust region or move limits for step length control:

$$\begin{aligned} &\text{minimize} && \nabla f(\mathbf{x}_k)\mathbf{d}_k \\ &\text{such that} && \nabla \mathbf{g}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{g}(\mathbf{x}_k) \le 0 \\ & && \nabla \mathbf{h}(\mathbf{x}_k)\mathbf{d}_k + \mathbf{h}(\mathbf{x}_k) = 0 \\ & && \underline{\mathbf{d}}_k \le \mathbf{d}_k \le \overline{\mathbf{d}}_k \qquad \text{additional move limits} \end{aligned} \tag{5.33}$$

The sub-problem (5.33) can efficiently be solved by a linear programming optimizer of which many are readily available. Insofar, SLP is a robust and reliable method which also has successfully been applied for structural optimization. It is insensitive to scaling, however, can suffer from problems with the step length control together with weakly constrained problems. The move limit adaptation rules are principally simple:

- enlarge the trust region if the progress is steady
- shrink the trust region if the method oscillates which indicates that the optimum had been overshot

The following illustrates the move limit adaptation strategy with respect to steady progress for variable x_1 until iteration step 5 and oscillation of variable x_2. Note, that the example is for illustration only. The SLP method is inappropriate for unconstrained problems.

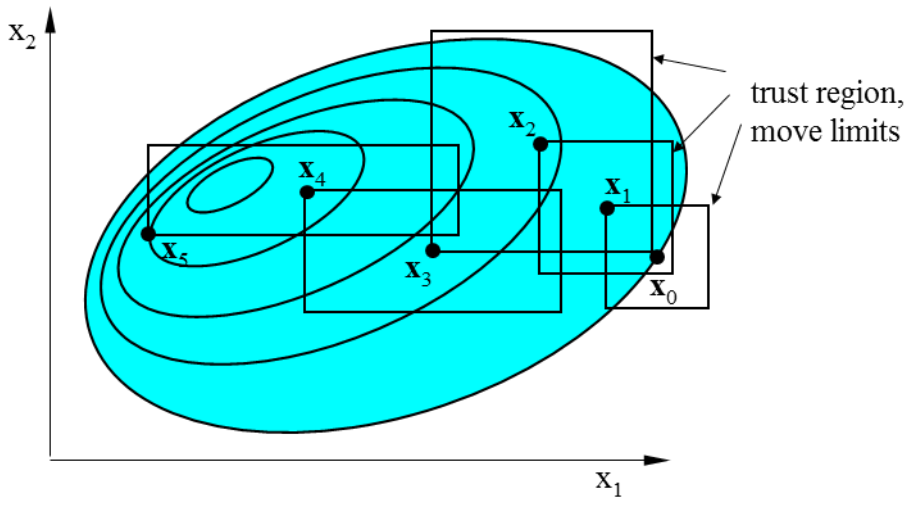


Fig. 5.11: Illustration of move limit adaptation.

5.9 Penalty methods

Penalty methods transform constrained problems into unconstrained. Appropriate methods can then be used. We distinguish so called *interior* and *exterior penalty methods*. With exterior penalty methods some function is added to the objective to penalize infeasible solutions. On the other hand such a function prevents that interior penalty methods leave the feasible domain. That is reason why the interior penalty method is also called *barrier function method*. In both cases the penalty factor adjusts the effect. As the factor approaches infinity the solution is increasingly better approximated while the problem condition worsens. As a consequence numerical problems occur if the factor is too large. The advantage of the interior method is that intermediate solutions always are feasible; the disadvantage is that equality constraints cannot be handled. The latter is no problem for exterior methods, these, however, converge from the exterior, i.e. the infeasible, domain towards the solution. Intermediate solutions can only be used if they are scaled back to the boundary of the feasible domain. The main advantage of both methods is that they can be handled very easily and make only little use of complicated theory. They are very successful in practical application for all kind of problems. A well-known procedure is *SUMT, sequential unconstrained minimization technique*, which generates a series of solutions with increasing penalty factor which allows for controlled approach to the optimum until the solution is accepted or the problem condition collapses. In this context the penalty factor can be understood as a continuation parameter, a feature which this rather old method has in common with the inner point methods. This approach from the interior domain to the solution as interior penalty function methods do but are on a strong mathematical basis in contrast to the more intuitive motivation of SUMT.

Exterior penalty methods, Fig. 5.12. The idea is simple: if a constraint is violated add a penalty to the objective such that the solution is pushed back towards the feasible domain. The objective is modified by the product of penalty factor c and penalty function P(**x**):

$$f_p(\mathbf{x}, c) = f(\mathbf{x}) + c\,P(\mathbf{x}) \tag{5.34}$$

The following variants of P are popular where we use the abbreviation:

$$g_j^+(\mathbf{x}) = h_j(\mathbf{x}); \qquad j = 1,\ldots, q \quad (\text{all equality constraints})$$
$$g_j^+(\mathbf{x}) = \max\left[0, g_j(\mathbf{x})\right]; \qquad j = 1,\ldots, p \quad (\text{all active inequality constraints})$$

name	formula	comments
L_1-penalty function also called "exact" penalty method	$P(\mathbf{x}) = \sum_j \left\|g_j^+(\mathbf{x})\right\|$	advantage: optimum can be found exactly with finite penalty factor disadvantage: discontinuous gradient at the boundary of feasible domain
quadratic penalty function	$P(\mathbf{x}) = \sum_j \left(g_j^+(\mathbf{x})\right)^2$	advantage: continuous gradients reduce numerical problems disadvantage: optimum of modified problem always infeasible

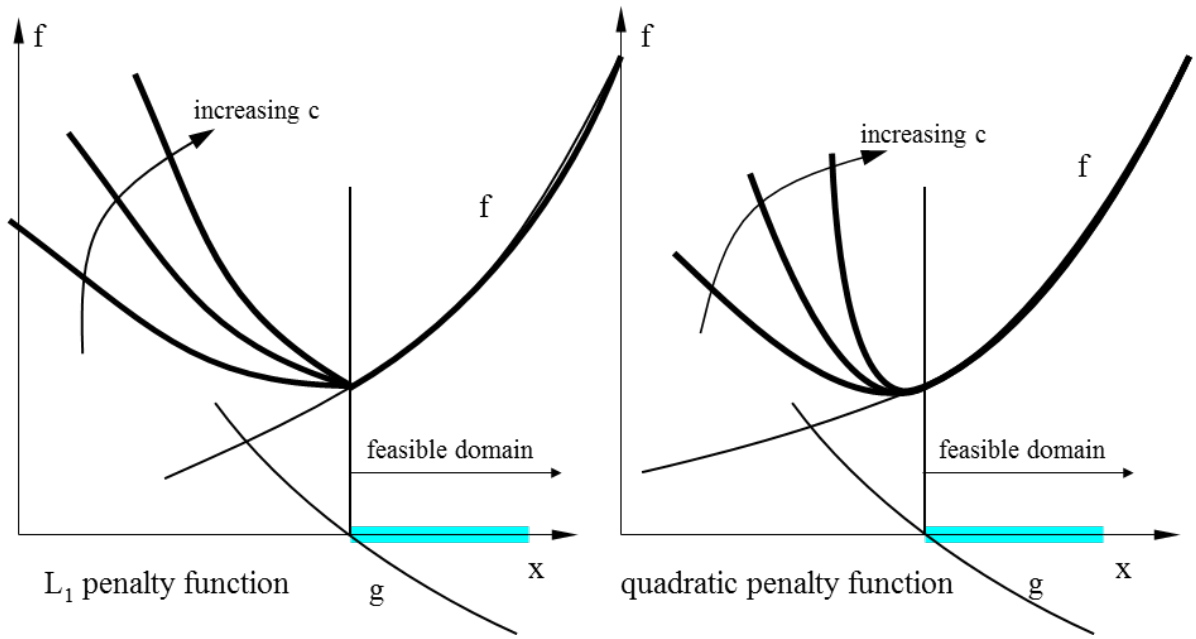


Fig. 5.12: Exterior penalty functions.

Interior penalty functions, Fig. 5.13. The problem is modified as:

$$f_p(\mathbf{x}, c) = f(\mathbf{x}) + \frac{1}{c} B(\mathbf{x}) \tag{5.35}$$

where B(**x**) is the barrier function and c is the penalty factor. Note, that only inequality constraints can be handled. Again, several variants are common:

$$\text{a)} \quad B(\mathbf{x}) = \sum_{j=1}^{p} \frac{1}{-g_j(\mathbf{x})}$$

$$\text{b)} \quad B(\mathbf{x}) = \sum_{j=1}^{p} \frac{1}{\left(-g_j(x)\right)^{\varepsilon}} \quad ; \ \varepsilon > 0, \quad \text{preferably } \varepsilon = 2 \tag{5.36}$$

$$\text{c)} \quad B(\mathbf{x}) = -\sum_{j=1}^{p} \ln\left(-\mathbf{g}_j(\mathbf{x})\right)$$

The method never leaves the feasible domain. A problem might be to find a feasible start solution.

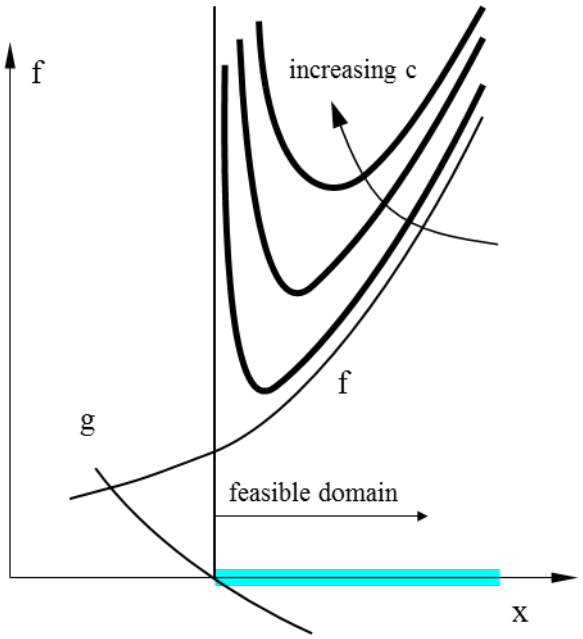


Fig. 5.13: Interior penalty or barrier function method.

5.10 Dual methods

The principal idea has already been discussed together with the dual function. The methods take advantage of the duality of primal and dual variables (Lagrange multipliers). Naturally, descent methods fail for finding saddle points. Dual methods make use of the dual function to split the numerical procedure to determine saddle points in a alternating sequence of minimization and maximization, so descent methods can be applied for the sub-problems:

4.) minimize L($\mathbf{x}$,λ) with respect to $\mathbf{x}$, determine $\mathbf{x}(\lambda)$ as a function of λ
5.) substitute $\mathbf{x}(\lambda)$ into L to get the dual function $D(\lambda) = L(\mathbf{x}(\lambda), \lambda)$
6.) maximize D(λ) with respect to λ

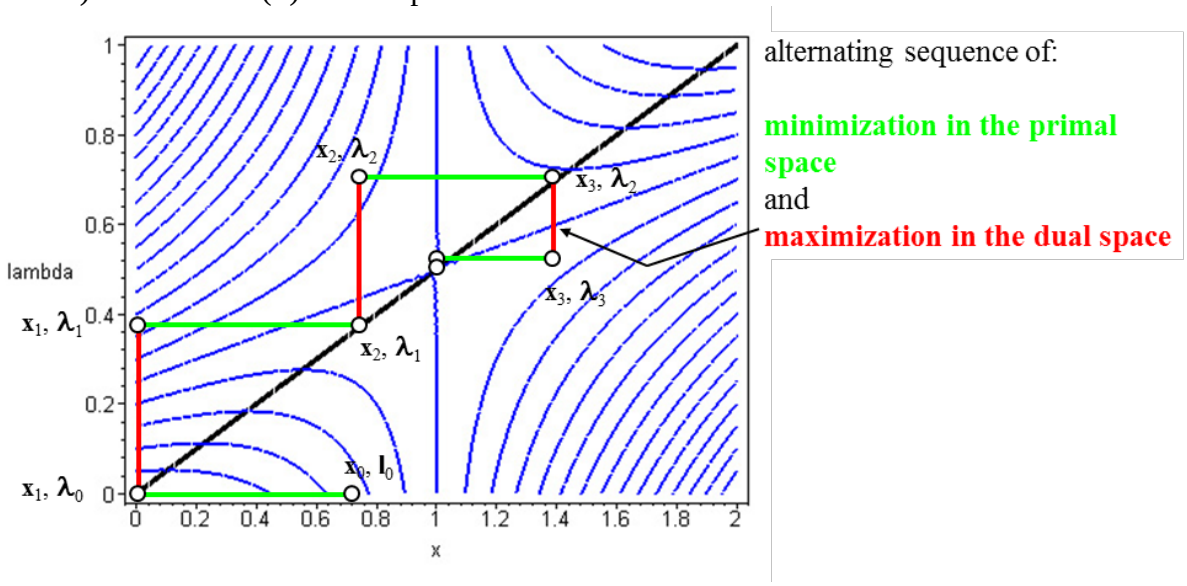


Fig. 5.14: Dual methods: principal iteration sequence.

The two kinds of sub-problems are comparatively simple. Therefore, they can be solved with almost any method which is available for unconstrained or simple constrained problems. Also, basic tools as e.g. quadratic programming methods which play a center role in the context of SQP are often based on dual approaches. Together with *approximation methods* dual methods have been very successful for structural optimization because they are able to exploit the characteristic of the primal sub-problem very efficiently, as e.g. separable problems. Refer to approximation methods.

5.10.1 The augmented Lagrange method

This method is based on augmenting the Lagrange function by an extra penalty term representing the constraints. This explains the name of the method. As a consequence, that results in a special dual solution technique with separated steps of minimization and maximization to determine the primal optimization variables and the dual Lagrange multipliers, respectively. As can be proven mathematically, the method reliably converges to the constrained solution while the penalty parameters can be chosen of finite size which doesn't cause numerical problems. The augmented Lagrange method is a simple but very robust technique which, in particular, is very useful for very large problems. A very prominent example is the simulation of contact mechanics where there are very large numbers of inequality constraints representing the contact conditions.

Let us introduce the method for a problem with one optimization variable x and one equality constraint h(x) = 0:

$$\begin{aligned} f(x) &\rightarrow \min \\ h(x) &= 0 \end{aligned} \tag{5.37}$$

The departure of the method is the Lagrange function, augmented by a quadratic penalty term:

$$L_a(x,\lambda_a,r) = L(x,\lambda_a) + \frac{1}{2} r [h(x)]^2 = f(x) + \lambda_a h(x) + \frac{1}{2} r [h(x)]^2 \tag{5.38}$$

Clearly, the solution is not affected by this modification. Also, because of the quadratic penalty term, the first derivative of the Lagrangian is not affected by this modification at the solution, i.e. the Karush-Kuhn-Tucker conditions remain the same:

$$\begin{aligned} \frac{\partial L}{\partial x} &= \frac{\partial f}{\partial x} + \lambda \frac{\partial h}{\partial x} \\ \frac{\partial L_a}{\partial x} &= \frac{\partial f}{\partial x} + \lambda_a \frac{\partial h}{\partial x} + r\, h(x) \frac{\partial h}{\partial x} = \frac{\partial f}{\partial x} + (\lambda_a + r\, h(x)) \frac{\partial h}{\partial x} \end{aligned} \tag{5.39}$$

Now, comparing the first order derivatives of the Lagrangian and its augmented variant gives the interesting relation between the Lagrange multiplier λ as of the standard formulation and λ_a of the augmented formulation:

$$\lambda = \lambda_a + r\, h(x) \tag{5.40}$$

Obviously, λ_a converges to λ as x converges to the optimal solution x^* where $h(x^*) = 0$. The augmented Lagrange method makes use of this relation and defines the recurrence formula to

determine the Lagrange multiplier $\lambda^{(k+1)}$ of iteration step (k+1) to be determined from the values of the previous step (k) by:

$$\lambda^{(k+1)} = \lambda^{(k)} + r\, h\left(x^{(k)}\right) \tag{5.41}$$

The penalty parameter r must be chosen “large enough” but remains of finite value which defines the strength of the procedure. This can be proven by further mathematical analysis. Practically, one starts with a first suggestion of r. If it appears that the algorithm oscillates the penalty parameter is increased until the algorithm starts to converge.

In the case of inequality constraints the technique is modified to insure that the Lagrange multipliers must not be negative. Considering individual penalty parameters for each constraint the basic formulation is modified as:

$$\begin{aligned} & f(x) \rightarrow \min \\ & g_j(x) \le 0 \\ & L_a(x,\lambda,r) = f(x) + \sum_{j=1}^{m} \lambda_j\, g_j(x) + \frac{1}{2}\sum_{j=1}^{m} r_j \left[\max\left(0, g_j(x)\right)\right]^2 \\ & \lambda_j^{(k+1)} = \max\left(0, \lambda_j^{(k)} + r_j\, g_j\left(x^{(k)}\right)\right) \end{aligned} \tag{5.42}$$

Because of numerical reasons, the algorithm might to be refined to deal with the non-differentiable penalty terms of inequality constraints. Nevertheless, for many applications the algorithm is working well even in the basic formulation as given here.

Finally, the augmented Lagrange method is defined as follows:

1. Select initial values, k = 0, $\lambda = \lambda^{(0)}$, r > 0
2. For fixed $\lambda^{(k)}$ minimize $L_a(x,\lambda^{(k)})$ w.r.t. x to determine $x^{(k)}$
3. Update λ to determine $\lambda^{(k+1)} = \lambda^{(k)} + r\, h(x^{(k)})$ or $\lambda^{(k+1)} = \max(0, \lambda^{(k)} + r\, g(x^{(k)}))$
4. Increment the iteration counter k = k+1
5. Repeat steps 1 to 4 until convergence

The methods is generalized straight forward for cases of many variables and many equality as well as inequality constraints.

Example. Consider the following problem:

$$\begin{aligned} f &= (x-2)^2 + (y-2)^2 \;\rightarrow\; \min \\ \text{s.t.}\quad g_1 &= 7 - x - y \le 0 \\ g_2 &= 1 + x - y \le 0 \end{aligned} \tag{5.43}$$

With the optimal solution:

$$\mathbf{x}^* = [3.0 \quad 4.0]^T; \quad \boldsymbol{\lambda} = [3.0 \quad 1.0]^T; \quad f^* = 5.0; \quad \mathbf{g}^* = [0.0 \quad 0.0^T]$$

The iteration history of the augmented Lagrange method is as follows. Observe how the Lagrange multipliers are updated from step to step.

For $r_1 = 1$ and $r_2 = 1$:

	x	y	λ_1	λ_2	f	g_1	g_2	aug. Lagrange
1	2,5	3	0	0	1,25	1,5	0,5	2,5
2	2,75	3,5	1,5	0,5	2,8125	0,75	0,25	4,375
3	2,875	3,75	2,25	0,75	3,828125	0,375	0,125	4,84375
4	2,9375	3,87499999	2,625	0,875	4,39453119	0,18750002	0,06250001	4,9609375
5	2,96875	3,9375	2,81250002	0,93750001	4,69238281	0,09375	0,03125	4,990234377
6	2,984375	3,96875001	2,90625002	0,96875001	4,84497073	0,04687499	0,015625	4,997558595
7	2,99218749	3,984375	2,95312501	0,984375	4,92218017	0,02343751	0,00781249	4,999389649
8	2,99609375	3,9921875	2,97656252	0,99218749	4,96101381	0,01171874	0,00390625	4,999847412
9	2,99804688	3,99609375	2,98828126	0,99609374	4,98048784	0,00585937	0,00195313	4,999961853
10	2,99902344	3,99804688	2,99414063	0,99804687	4,99023915	0,00292968	0,00097657	4,999990463
11	2,99951172	3,99902344	2,99707031	0,99902344	4,9951184	0,00146484	0,00048828	4,999997616
12	2,99975586	3,99951172	2,99853515	0,99951172	4,99755889	0,00073242	0,00024414	4,999999404
13	2,99987793	3,99975586	2,99926758	0,99975586	4,99877937	0,00036621	0,00012207	4,999999851
14	2,99993896	3,99987793	2,99963379	0,99987793	4,99938967	0,00018311	6,1035E-05	4,999999963
15	2,99996948	3,99993896	2,99981689	0,99993897	4,99969483	9,1553E-05	3,0517E-05	4,999999991

And for $r_1 = 10$ and $r_2 = 10$:

	x	y	λ_1	λ_2	f	g_1	g_2	aug. Lagrange
1	2,90909091	3,81818182	0	0	4,1322314	0,27272728	0,09090909	4,545454545
2	2,99173554	3,98347108	2,72727276	0,90909091	4,91769689	0,02479339	0,00826446	4,996243427
3	2,99924869	3,99849737	2,97520662	0,99173554	4,99248968	0,00225394	0,00075131	4,999968954
4	2,9999317	3,9998634	2,99774606	0,99924869	4,99931701	0,0002049	6,8301E-05	4,999999743
5	2,99999379	3,99998758	2,9997951	0,9999317	4,99993791	1,8628E-05	6,2092E-06	4,999999998
6	2,99999944	3,99999887	2,99998137	0,99999379	4,99999436	1,6903E-06	5,7033E-07	5
7	2,99999994	3,9999999	2,99999828	0,99999949	4,99999948	1,615E-07	3,6987E-08	5
8	3	3,99999999	2,99999989	0,99999986	4,99999995	1,2687E-08	1,1811E-08	5
9	3	3,99999999	3,00000002	0,99999998	4,99999995	1,2687E-08	1,1811E-08	5
10	3	3,99999999	3,00000014	1,0000001	4,99999995	1,2687E-08	1,1811E-08	5

Next, consider the modified problem with $g_1 = 1 - x - y$ which is inactive at the optimal solution $\mathbf{x}^* = [1.5, 2.5]^T$:

For $r_1 = 10$ and $r_2 = 10$ we get the following iteration history:

	x	y	λ_1	λ_2	f	g_1	g_2	aug. Lagrange
1	1,54545455	2,45454545	0	0	0,41322314	-3	0,09090909	0,454545455
2	1,50413223	2,49586777	0	0,90909091	0,49176969	-3	0,00826446	0,499624343
3	1,50037566	2,49962434	0	0,99173554	0,49924897	-3	0,00075131	0,499996895
4	1,50003415	2,49996585	0	0,99924869	0,4999317	-3	6,8301E-05	0,499999974
5	1,50000311	2,4999969	0	0,9999317	0,49999379	-3,00000001	6,2115E-06	0,5
6	1,50000172	2,50000094	0	0,99999381	0,49999921	-3,00000266	7,8645E-07	0,5
7	1,49999904	2,500001	0	1,00000168	0,50000196	-3,00000004	-1,9639E-06	0,5
8	1,5000008	2,49999941	0	0,99998204	0,49999861	-3,00000022	1,3899E-06	0,5
9	1,50000041	2,50000081	0	0,99999594	0,50000039	-3,00000122	-3,9178E-07	0,5
10	1,50000041	2,50000081	0	0,99999202	0,50000039	-3,00000122	-3,9178E-07	0,5

5.11 Optimality criteria methods

In their origins optimality criteria methods (shorter: OC-methods) had been heuristic. The basic idea is to state some criterion which identifies the optimum if it is fulfilled. This criterion may be something else than the necessary Karush-Kuhn-Tucker conditions. Therefore, OC-methods can find the mathematically true solution only if the OC-criterion happens to coincide with the KKT-conditions. A prominent example where that is not the case is the *fully stressed design* of statically determinate structures. It states that structures are of minimum weight if all of their members are fully stressed. This criterion already fails for statically indeterminate structures, multiple load cases, shape optimization etc. (compare with the three bar truss example). For each optimization task another OC-criterion has to be stated. Despite the theoretical weakness OC-methods are very robust and effective for those cases they are prepared for. The lack of generality was tried to overcome by deriving OC-criteria from the KKT-conditions. At this end, OC-methods appear to be simple variants of dual methods. Insofar OC-methods have lost significance for general structural optimization problems.

FSD, fully stressed design. Suppose, one likes to find the cross section dimensions of a structure which elements are fully stressed at least under one load case. From that demand a simple recursion formula is defined which scales the actual cross sections by the ratio of current stress and admissible stress. In the case of trusses and cross sectional area A this formula is:

$$A_i^{(k+1)} = \max_{\text{all load cases}} \left[A_i^{(k)} \frac{\sigma_i^{(k)}}{\sigma_i^{\text{admissible}}} \right]; \quad \text{for all members } i = 1, \ldots, n \tag{5.44}$$

In principle that's all. For statically determinate structures the method terminates after one step. For statically indeterminate structures, however, it will not converge to the correct solution, in general. Typically, a statically determinate structure will be found. The main problem is that the procedure is very tedious until convergence and may even worsen the objective towards the end. Consequently, one likes to terminate the method as soon as it becomes ineffective although the constraints may still be violated. Even so one gets a feasible solution by scaling back the current solution of iteration step k to the boundary of the feasible domain. All members are scaled by the largest stress ratio:

$$A_i = \text{fac}\, A_i; \quad \text{fac} = \max_i \left[\frac{\sigma_i^{(k)}}{\sigma_i^{\text{admissible}}} \right] \text{for all members } i = 1, \ldots, n \tag{5.45}$$

Example. Again, consider the three bar truss problem. The following table gives the iteration history for the first 12 iteration steps and the final solution.

iter.	A_1	A_2	σ_1	σ_2	σ_3	weight	w. (scal)
0	1.000	1.000	14.142	8.284	-5.858	382.8	270.7
1	0.707	0.414	21.877	15.469	-6.408	241.4	264.1
2	0.773	0.320	21.082	16.306	-4.776	250.8	264.4
3	0.815	0.261	20.706	16.882	-3.824	256.7	265.8
4	0.844	0.220	20.498	17.302	-3.196	260.8	267.3
5	0.865	0.191	20.371	17.623	-2.748	263.8	268.7
6	0.881	0.168	20.287	17.876	-2.411	266.0	269.8
7	0.894	0.150	20.228	18.080	-2.149	267.8	270.9
8	0.904	0.136	20.186	18.248	-1.938	269.3	271.8
9	0.912	0.124	20.155	18.389	-1.766	270.5	272.5
10	0.919	0.114	20.131	18.509	-1.622	271.5	273.2
11	0.925	0.105	20.112	18.612	-1.499	272.3	273.8
12	0.931	0.098	20.097	18.702	-1.394	273.0	274.4
n	1.000	0.000	20.000	20.00	0.000	282.8	282.8

We realize: convergence is rather slow and constrains are violated for members 1 and 2 throughout the whole process. The solutions must be scaled back to the boundary of the feasible domain. The scaled variable values are not given in the tables. The following figure illustrates the convergence to the (wrong) statically determinate structure. The best result would have been achieved by FSD at the second step.

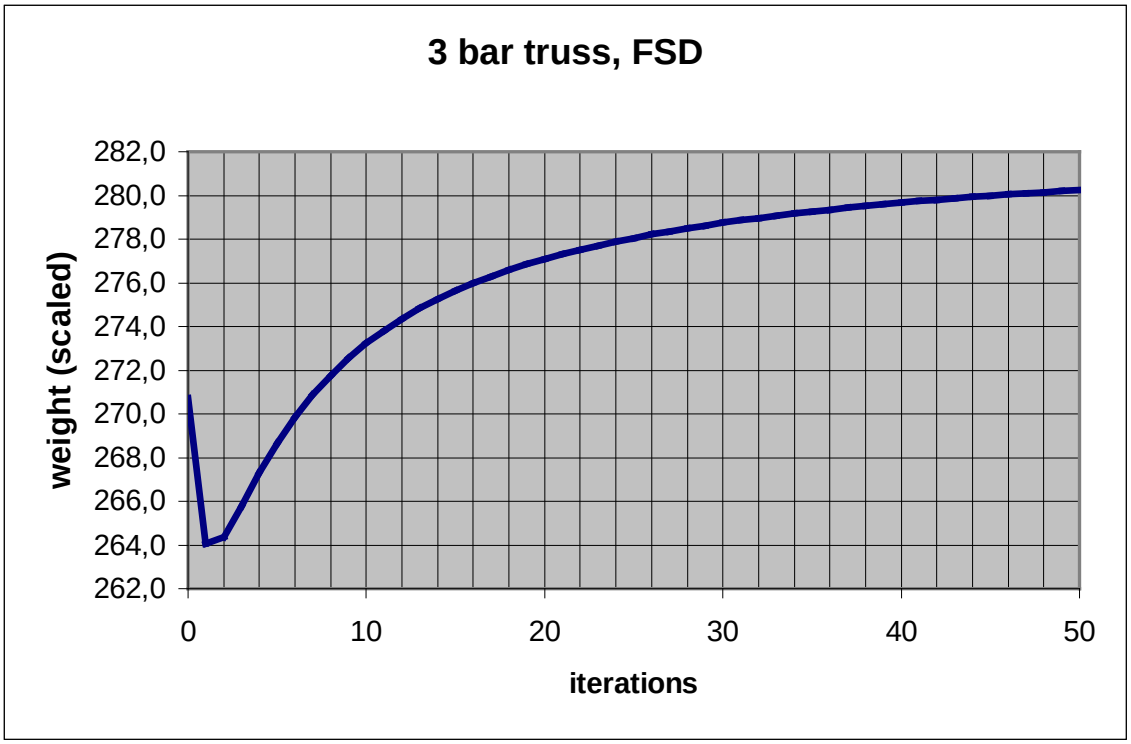


Fig. 5.15: FSD, iteration history, 3 bar truss problem.

Karush-Kuhn-Tucker related OC-criterion. The Karush-Kuhn-Tucker criteria

$$\begin{aligned} \nabla_x L(\mathbf{x},\boldsymbol{\lambda}) &= \nabla_x f(\mathbf{x}) + \nabla_x \mathbf{g}(\mathbf{x})^T \boldsymbol{\lambda} = 0 \\ \nabla_\lambda L(\mathbf{x},\boldsymbol{\lambda}) &= \mathbf{g}(\mathbf{x}) = 0 \end{aligned} \tag{5.46}$$

may be used in the context of OC-methods. For the set of active constraints we can rewrite the KKT-conditions as:

$$\begin{aligned} T_i &= -\frac{\boldsymbol{\lambda}^T \dfrac{\partial \mathbf{g}(\mathbf{x})}{\partial x_i}}{\dfrac{\partial f(\mathbf{x})}{\partial x_i}}; \quad \text{for all variables } i = 1,...,n \\ G_j &= \frac{\tilde{g}_j(\mathbf{x})}{\bar{g}_j} \quad ; \quad \text{for all active constraints} \end{aligned} \tag{5.47}$$

where $\tilde{g}_j(\mathbf{x})$ and $\bar{g}_j$ denote the variant and constant part of $g_j(\mathbf{x}) = \tilde{g}_j(\mathbf{x}) - \bar{g}_j \le 0$, respectively. Using the ratios T_i and G_j the recurrence relations are defined as:

$$\begin{aligned} x_i^{(k+1)} &= \left[T_i^{(k)}\right]^{1/\beta} x_i^{(k)} \\ \text{and} \quad \lambda_j^{(k+1)} &= \left[G_i^{(k)}\right]^{1/\delta} \lambda_j^{(k)} \end{aligned} \tag{5.48}$$

where at the optimum T_i and G_j approach one. Good experience is reported by setting $\beta = 2$ and $\delta = 1$.

The sequence of procedural steps is:

1.) detect active constraints
2.) initialize $\boldsymbol{\lambda}$ by a mean square solution of $\nabla \mathbf{g} \nabla f + \nabla \mathbf{g} \nabla \mathbf{g}^T \boldsymbol{\lambda} = 0$, or be other criteria
3.) keep $\boldsymbol{\lambda}$ fixed, determine $\mathbf{x}$ from 5.48
4.) keep $\mathbf{x}$ fixed, determine $\boldsymbol{\lambda}$ from 5.48
5.) repeat steps 3 and 4 as long as required

Note the formal identity with the dual methods. The only differences are the solutions procedures in the steps 3 and 4 where the OC-method uses the simple recurrence relations. Application is very robust for the cost of slow convergence.

Example. Consider the problem:

$$\begin{aligned} \text{minimize} \quad & f(x) = x^2 + 1 \\ \text{such that} \quad & g(x) = \frac{1}{x} - 1 \le 0 \\ \text{initial values:} \quad & x^{(0)} = \frac{1}{4}; \qquad \lambda^{(0)} = \frac{1}{32} \\ \text{procedural constants:} \quad & \beta = 4 \qquad \delta = 1 \end{aligned}$$

The following table demonstrates the first 10 iteration steps:

i	x	λ	df/dx	dg/dx	T	$\tilde{g}$	G	g
0	0.2500	0.0313	0.5000	-16.0000		4.0000	4.0000	3.0000
1		0.1250			1.4142			
	0.3536		0.7071	-8.0000		2.8284	2.8284	1.8284
2		0.3536			1.4142			
	0.5000		1.0000	-4.0000		2.0000	2.0000	1.0000
3		0.7071			1.2968			
	0.6484		1.2968	-2.3784		1.5422	1.5422	0.5422
4		1.0905			1.1892			
	0.7711		1.5422	-1.6818		1.2968	1.2968	0.2968
5		1.4142			1.1144			
	0.8593		1.7186	-1.3543		1.1637	1.1637	0.1637
6		1.6458			1.0671			
	0.9170		1.8340	-1.1892		1.0905	1.0905	0.0905
7		1.7947			1.0386			
	0.9524		1.9049	-1.1024		1.0499	1.0499	0.0499
8		1.8843			1.0219			
	0.9733		1.9466	-1.0556		1.0274	1.0274	0.0274
9		1.9361			1.0123			
	0.9852		1.9704	-1.0302		1.0150	1.0150	0.0150
10		1.9651			1.0068			
	0.9919		1.9838	-1.0164		1.0082	1.0082	0.0082

The following figure illustrates the relations of OC- and dual methods. One can see the alternating solution steps in primal and dual direction. One also can see the convergence problems near the optimum.

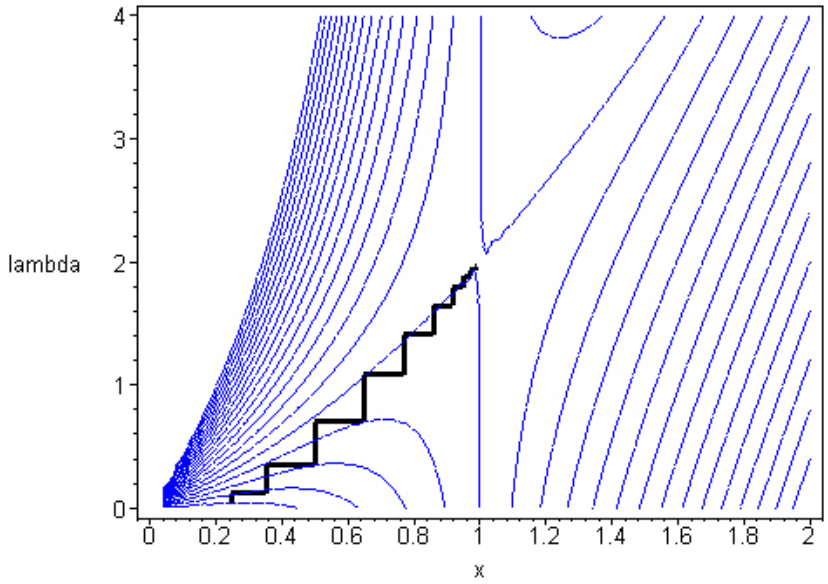
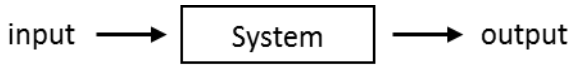


Fig. 5.16: Iteration history of KKT based OC-method towards the saddle of the Lagrangian.

6 Sensitivity Analysis

6.1 Types of Sensitivity Analysis

In general, sensitivity analysis of any numerical or experimental systems deals with the question of how output parameters vary if input parameters are modified.



In the context of structural optimization sensitivity analysis is used to determine the gradient of response functions which are used as objective or constraints. In the context of this manuscript we call this type of sensitivity analysis *response sensitivity analysis*. It is explained in more details in the sequel of this chapter.

Also of interest is to know how the result of an optimization, i.e. the design, is changed if the reference value of a constraint is modified. That's why we talk here of *design sensitivity analysis*. The sensitivity of a constraint is expressed by the related value of the corresponding Lagrange multiplier. The method is explained in chapter 4.6. It should be noted that often *response sensitivity analysis* is also called *design sensitivity*. It is always recommended to carefully check what the authors mean.

6.2 Response Sensitivity Analysis

Optimal designs are characterized by the necessary Karush-Kuhn-Tucker conditions

$$\frac{dL}{d\mathbf{x}} = \frac{df}{d\mathbf{x}} + \boldsymbol{\lambda}^T \frac{d\mathbf{g}}{d\mathbf{x}} = 0 \tag{6.1}$$

i.e. first order derivatives of objective f and constraints **g** w.r.t. the optimization variables **x** have to be determined. Also, to find optimal solutions using higher order methods (e.g. steepest descent, conjugated gradients, SQP, etc.) the evaluation of first order derivatives is important.

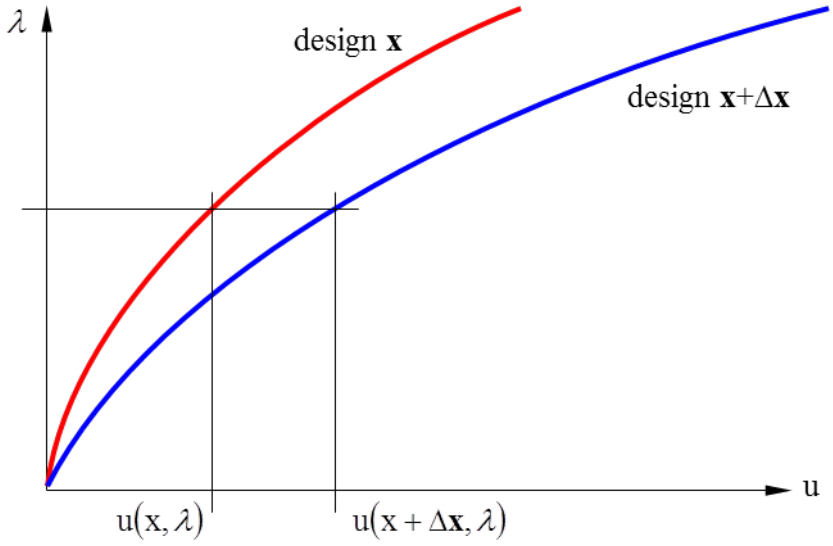


Fig. 6.1: Effect of design change on structural response

As we ask for the derivatives of structural response functions f and **g** (e.g. weight, strain energy, stresses, displacements, eigenvalues, etc.) we realize that, in general, these functions are dependent of the design variables **x** as well as the "response variables" **u** which usually are the

structural displacements. In turn, however, the response variables **u** themselves are dependent of the design variables **x**. That becomes intuitively obvious if we realize that by a design change $\Delta\mathbf{x}$ the stiffness of the structure is affected and, consequently, the structure now exhibits smaller or larger displacements, Fig. 6.1.

In mathematical terms the nested dependencies state as:

$$f = f\left(\mathbf{x}, \mathbf{u}(\mathbf{x})\right) \tag{6.2}$$

When differentiating f w.r.t. **x** we have to consider the chain rule of differentiation:

$$\frac{df}{d\mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \frac{\partial f}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{x}} \tag{6.3}$$

Example. Consider stress σ as response function f. Using the method of finite elements the stress vector is calculated as:

$$\boldsymbol{\sigma} = \mathbf{C}\,\mathbf{B}\,\mathbf{u} \tag{6.4}$$

where **C** is the material matrix and the operator matrix **B** describes the strain-displacement relation:

$$\boldsymbol{\varepsilon} = \mathbf{B}\,\mathbf{u} \tag{6.5}$$

which is a function of the structural geometry.

E.g., consider a bar element and linear analysis:

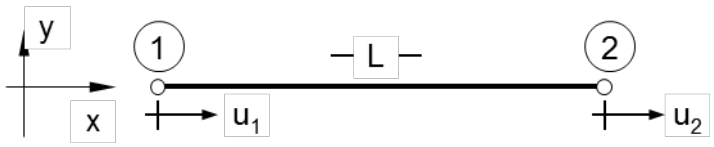


Fig. 6.2: Bar element.

The longitudinal strain ε is defined as:

$$\varepsilon = \frac{u_2 - u_1}{L} = \mathbf{B}\,\mathbf{u} = \frac{1}{L}\left[-1,\ 1\right]\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad \text{where} \quad L = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2 + \left(z_2 - z_1\right)^2} \tag{6.6}$$

which is a function of the geometry measured in terms of the nodal coordinates $\mathbf{x}_i$ of the nodes i=1,2.

Assuming constant material parameters, the derivative of $\boldsymbol{\sigma}$ w.r.t. **x** (shape optimization) now writes as:

$$\frac{d\boldsymbol{\sigma}}{d\mathbf{x}} = \underbrace{\mathbf{C}\frac{d\mathbf{B}}{d\mathbf{x}}\mathbf{u}}_{\frac{\partial\boldsymbol{\sigma}}{\partial\mathbf{x}}} + \underbrace{\mathbf{C}\mathbf{B}}_{\frac{\partial\boldsymbol{\sigma}}{\partial\mathbf{u}}}\frac{\mathbf{du}}{\mathbf{dx}} \tag{6.7}$$

In case of the simple bar element we have:

$$\frac{d\mathbf{B}}{d\mathbf{x}} = -\frac{1}{L^2}\frac{dL}{d\mathbf{x}}\left[-1,\ 1\right] \tag{6.8}$$

$$\frac{dL}{dx_1} = \frac{1}{2L}\,2\underbrace{\left(x_2 - x_1\right)}_{\Delta x}(-1) = -\frac{\Delta x}{L}$$

$$\frac{dL}{dx_2} = \frac{\Delta x}{L} \tag{6.9}$$

$$\frac{d\mathbf{B}}{d\mathbf{x}} = -\frac{1}{L^3}\begin{bmatrix} \Delta x & -\Delta x \\ -\Delta x & \Delta x \\ \hline \Delta y & -\Delta y \\ -\Delta y & \Delta y \\ \hline \Delta z & -\Delta z \\ -\Delta z & \Delta z \end{bmatrix}$$

In the general case of several stress components arranged in the stress vector $\boldsymbol{\sigma}$ $d\mathbf{B}/d\mathbf{x}$ appears to be a matrix with three indices:

$$\sigma_i = C_{ij}\,B_{jk}\,u_k$$

$$\frac{d\sigma_i}{dx_p} = C_{ij}\underbrace{\frac{dB_{jk}}{dx_p}}_{3\ \text{indices}}u_k + C_{ij}\,B_{jk}\frac{du_k}{dx_p} \tag{6.10}$$

which generates problems for display.

6.3 Sensitivity of the response variables

We saw that a major part of response sensitivity analysis is to determine the derivatives of the response variables **u** (typically nodal displacements) w.r.t. the design variables **x**. These we get from the equilibrium condition which writes for the case of the displacement finite element method as:

$$\mathbf{K}\,\mathbf{u} = \mathbf{P} \tag{6.11}$$

with **K** ... system stiffness matrix

P ... load vector

u ... vector of nodal displacment components

Derivation w.r.t. **x** yields:

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\left(\mathbf{K}\,\mathbf{u}\right) = \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}\mathbf{x}}$$
$$\frac{\mathrm{d}\mathbf{K}}{\mathrm{d}\mathbf{x}}\,\mathbf{u} + \mathbf{K}\,\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} = \frac{\mathrm{d}\mathbf{P}}{\mathrm{d}\mathbf{x}} \tag{6.12}$$

$$\therefore \quad \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}\mathbf{x}} = \mathbf{K}^{-1}\left(\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}\mathbf{x}} - \frac{\mathrm{d}\mathbf{K}}{\mathrm{d}\mathbf{x}}\,\mathbf{u}\right) \tag{6.13}$$

In analogy to $\mathbf{u} = \mathbf{K}^{-1}\,\mathbf{P}$ the right hand side of (6.13) is called *pseudo load vector* **P***. Evaluation of d**u**/d**x** can be understood as a static multi load case analysis with as many load cases as design variables. For that reason direct equation solvers are often preferred to iterative solvers in the context of structural optimization because **K** is factorized once. The factorization is available for all pseudo load cases.

Direct approach.

On the way to determine the derivative of a response function $\mathrm{g}_\mathrm{j}\left(\mathbf{x},\mathbf{u}\left(\mathbf{x}\right)\right)$ with respect to $\mathrm{x_i}$

$$\frac{\mathrm{dg_j}}{\mathrm{dx_i}} = \frac{\partial \mathrm{g_j}}{\partial \mathrm{x_i}} + \frac{\partial \mathrm{g_j}}{\partial \mathbf{u}}\,\frac{\mathrm{d}\mathbf{u}}{\mathrm{dx_i}}$$
$$= \frac{\partial \mathrm{g_j}}{\partial \mathrm{x_i}} + \frac{\partial \mathrm{g_j}}{\partial \mathbf{u}}\,\mathbf{K}^{-1}\left(\frac{\mathrm{d}\mathbf{P}}{\mathrm{dx_i}} - \frac{\mathrm{d}\mathbf{K}}{\mathrm{dx_i}}\mathbf{u}\right) \tag{6.14}$$

the direct sensitivity analysis means:

first, determine d**u**/d$\mathbf{x}_\mathrm{i}$ by solving $\mathbf{K}^{-1}\mathbf{P}_\mathrm{i}^{*}$ and

then inserting d**u**/d$\mathbf{x}_\mathrm{i}$ into (6.14). I.e., we have to solve $\mathbf{K}^{-1}\mathbf{P}_\mathrm{i}^{*}$ n times (n … number of design variables.

Adjoint method.

On the other hand we can invert the order of operations. Consider again

$$\frac{\mathrm{dg_j}}{\mathrm{dx_i}} = \frac{\partial \mathrm{g_j}}{\partial \mathrm{x_i}} + \left[\frac{\partial \mathrm{g_j}}{\partial \mathbf{u}}\,\mathbf{K}^{-1}\right]\left(\frac{\mathrm{d}\mathbf{P}}{\mathrm{dx_i}} - \frac{\mathrm{d}\mathbf{K}}{\mathrm{dx_i}}\mathbf{u}\right) \tag{6.15}$$

Now, we first solve for the so called *adjoint variables* $\tilde{\mathbf{u}}_\mathrm{j}$:

$$\tilde{\mathbf{u}}_\mathrm{j} = \mathbf{K}^{-1}\,\frac{\partial \mathrm{g_j}}{\partial \mathbf{u}} \tag{6.16}$$

and then insert the result into the remaining formula. We have to solve (6.16) m times (m … number of response functions g_j).

Depending on which number is larger, n or m, either the direct or the adjoint method is preferable. Together with active set strategies only the gradients of active constraints have to be determined.

Pseudo Force Vector.

In particular for shape optimal design the evaluation of the pseudo force vector is dominating the numerical effort. The derivation of the stiffness matrix is quite time consuming. In the case of standard isoparametric elements it appears to be 3 times more as the generation of the stiffness matrix itself. Consider the definition of an element stiffness matrix:

$$\mathbf{k}_e = \int_V \mathbf{B}^T \mathbf{C} \mathbf{B} |\mathbf{J}| \, dV \tag{6.17}$$

and follow the product rule:

$$\frac{d\mathbf{k}_e}{dx_i} = \int_V \left[\frac{d\mathbf{B}^T}{dx_i} \mathbf{C} \mathbf{B} |\mathbf{J}| + \mathbf{B}^T \mathbf{C} \frac{d\mathbf{B}}{dx_i} |\mathbf{J}| + \mathbf{B}^T \mathbf{C} \mathbf{B} \frac{d|\mathbf{J}|}{dx_i} \right] dV \tag{6.18}$$

For some cases (e.g. specific shell formulations) the problem becomes even worse because the constitutive matrix **C** might also be a function of **x** even for linear elastic material. Then we get an additional term in (6.18).

In the discussion of whether the direct or the adjoint method is faster we should not forget that everything may be dominated by the evaluation of the pseudo force vector which needs the same effort for both approaches.

Sizing.

Things change dramatically in the case of sizing if the stiffness matrix depends much simpler on **x**, e.g. linear in the case of truss bars:

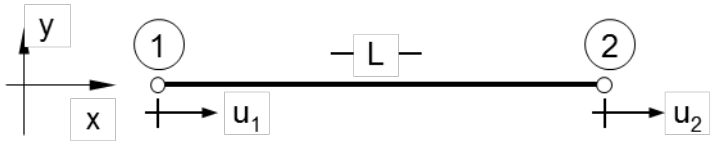
$$\mathbf{k}_e = \frac{EA}{L} [\cdot \cdot] \tag{6.19}$$

where the cross sectional area A is the sizing variable x_i. It follows that:

$$\frac{d\mathbf{k}_e}{dx_i} = \frac{1}{x_i} \mathbf{k}_e \tag{6.20}$$

The numerical effort to generate $d\mathbf{K}/dx_i$ is negligible.

Example. Again a bar



$$\sigma = \mathrm{E}\,\varepsilon = \mathrm{E}\frac{\mathrm{u}_2 - \mathrm{u}_1}{\mathrm{L}}$$

$$\mathrm{g} = \sigma - \beta_\mathrm{y} \le 0$$

$$\frac{\partial \mathrm{g}}{\partial \mathbf{u}} = \frac{\mathrm{E}}{\mathrm{L}}\begin{pmatrix}-1 & 1\end{pmatrix} = \mathbf{C}\,\mathbf{B}$$

$$\frac{\mathrm{dg}}{\mathrm{dA}} = \frac{\partial \mathrm{g}}{\partial \mathbf{u}}\frac{\mathrm{d}\mathbf{u}}{\mathrm{dA}}$$

$$\mathbf{k}_\mathrm{e} = \frac{\mathrm{EA}}{\mathrm{L}}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}; \quad \frac{\mathrm{d}\mathbf{k}_\mathrm{e}}{\mathrm{dA}} = \frac{\mathrm{E}}{\mathrm{L}}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}$$

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{dA}} = \mathbf{K}^{-1}\left(\frac{\mathrm{d}\mathbf{P}}{\mathrm{dA}} - \frac{\mathrm{E}}{\mathrm{L}}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\mathbf{u}\right)$$

6.4 Finite Difference Approximations

The coding of the sensitivity analysis might be difficult or even impossible because of the complexity of the underlying element formulation. Or the source code is not available. In these cases a numerical approximation by finite difference analysis might be useful. Often, the accuracy of a finite difference approximation might be satisfactory.

The first order derivative with respect to variable x_i of a function f can be approximated by:

$$\frac{\partial \mathrm{f}}{\partial \mathrm{x}_\mathrm{i}} \cong \frac{\mathrm{f}\left(\mathrm{x}_\mathrm{i} + \Delta \mathrm{x}_\mathrm{i}\right) - \mathrm{f}\left(\mathrm{x}_\mathrm{i}\right)}{\Delta \mathrm{x}_\mathrm{i}} \tag{6.21}$$

In terms of structural optimization that means a full finite element analysis including the set up and factorization of the system stiffness matrix for the variation of each variable x_i. Very soon this procedure becomes very time consuming.

As an alternative the so-called *semi-analytical sensitivity analysis* was developed. It is defined as mixture of the *analytical* method described in chapter 7.2 and the *numerical* method of (6.21). The compromise is to develop all equations analytically but to replace the derivatives of the element stiffness matrices by the numerical approximation. The advantage of the approach is that no additional element code has to be prepared.

The semi-analytical sensitivity analysis is defined by (refer to (6.13)):

$$\frac{d\mathbf{u}}{d\mathbf{x}} = \mathbf{K}^{-1}\left(\frac{d\mathbf{R}}{d\mathbf{x}} - \frac{\Delta\mathbf{K}}{\Delta\mathbf{x}}\mathbf{u}\right) \tag{6.22}$$

The advantage of simpler coding of the stiffness matrix derivative must be paid by numerical errors which may increase if the mesh is refined (refer to the next paragraph). Because of that the semi-analytical sensitivity analysis can be used only in a refined form. An approach is shown in the sequel.

6.5 What is wrong with the Semi-analytical Sensitivity Analysis?

To repeat, the semi-analytical sensitivity analysis makes use of the finite difference approximation of the stiffness matrix derivative:

$$\frac{d\mathbf{u}}{d\mathbf{x}} = \mathbf{K}^{-1}\left(\frac{d\mathbf{R}}{d\mathbf{x}} - \frac{\Delta\mathbf{K}}{\Delta\mathbf{x}}\mathbf{u}\right) \tag{6.23}$$

What seems so convincing to do if the analytical derivative of **K** is not available, however, hides a severe problem. In contradiction to the usual convergence behavior of finite element analysis which means that the error fades out with mesh refinement, the error of semi-analytical sensitivity analysis can increase! For example, consider the problem of a Bernoulli cantilever beam with a moment applied at the tip, Fig. 6.3.

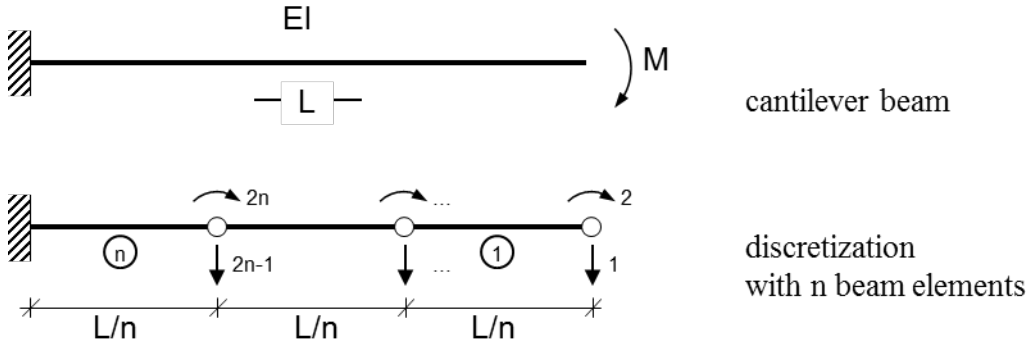


Fig. 6.3: Cantilever beam, finite element discretization.

The displacement derivative with respect to a change of the cantilever length L is determined by formula 6.23 using the semi-analytical sensitivity analysis. The forward difference scheme with $\Delta L = 0.0001\ L$ is applied. With increasing number of elements the following results for du_1/dL are determined and compared with the exact analytical result which is scaled to one, see also Fig. 6.4. Surprisingly, the error of the semi-analytical analysis increases with increasing mesh refinement.

Table 6.1. Derivative du_1/dL for several sensitivity analysis schemes.

element	exact	semi-analytical	mod. semi-analytical
1	1.0	.999650	.999810
2	1.0	.998900	.999802
3	1.0	.997650	.999801
4	1.0	.995901	.999800
5	1.0	.993651	.999800
6	1.0	.990902	.999800
7	1.0	.987653	.999800
8	1.0	.983904	.999800
9	1.0	.979656	.999800
10	1.0	.974907	.999800
11	1.0	.969659	.999800
12	1.0	.963910	.999800
13	1.0	.957662	.999800
14	1.0	.950914	.999800
15	1.0	.943666	.999800
16	1.0	.935919	.999800
17	1.0	.927671	.999800
18	1.0	.918924	.999800
19	1.0	.909677	.999800
20	1.0	.899929	.999800

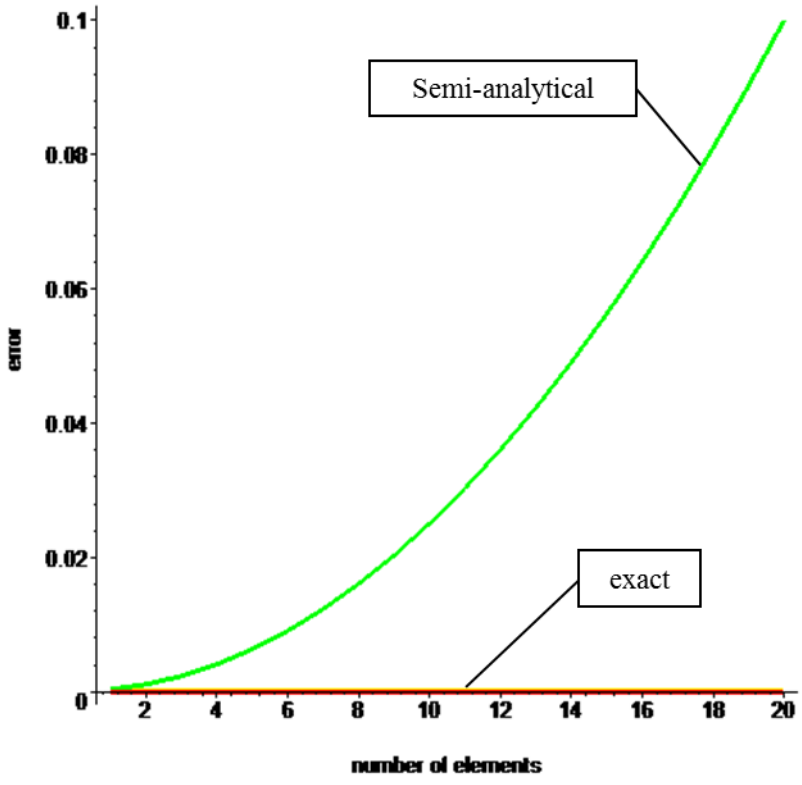


Fig. 6.4: Cantilever beam: Evolution of error with increasing number of elements.

The reason of the dramatically bad behaviour of the semi-analytical sensitivity analysis stems from the errors which arise with respect to the rigid body rotation behaviour of the approximated stiffness matrix derivative. This error is more amplified for smaller than for larger elements. Let us look at the formulas.

Let us assume that a free, unsupported structure can undergo rigid body translations and rotations - the so called “zero eigen-vectors“ $\boldsymbol{\phi}$ - which do not introduce any forces, i.e.:

$$\mathbf{K}\,\boldsymbol{\phi} = 0 \tag{6.24}$$

Deriving 6.24 with respect to design variables **x**

$$\frac{d\mathbf{K}}{d\mathbf{x}}\,\boldsymbol{\phi} + \mathbf{K}\,\frac{d\boldsymbol{\phi}}{d\mathbf{x}} = 0 \tag{6.25}$$

and pre-multiplying by $\boldsymbol{\phi}^T$ yields:

$$\boldsymbol{\phi}^T\,\frac{d\mathbf{K}}{d\mathbf{x}}\,\boldsymbol{\phi} + \boldsymbol{\phi}^T\,\mathbf{K}\,\frac{d\boldsymbol{\phi}}{d\mathbf{x}} = 0 \tag{6.26}$$

Applying 6.24 gives as result the rigid body condition which the stiffness matrix derivative satisfies:

$$\boldsymbol{\phi}^T\,\frac{d\mathbf{K}}{d\mathbf{x}}\,\boldsymbol{\phi} = 0 \tag{6.27}$$

Let us check this result for the special case of a Bernoulli beam element. The degrees of freedom are defined as:

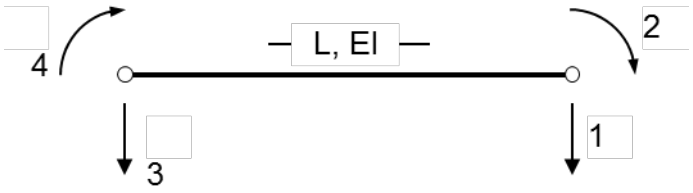


Fig. 6.5: Bernoulli beam element.

The stiffness matrix takes the well-known format:

$$K = \begin{bmatrix} \frac{12EI}{L^3} & -\frac{6EI}{L^2} & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{4EI}{L} & \frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{12EI}{L^3} & \frac{6EI}{L^2} & \frac{12EI}{L^3} & \frac{6EI}{L^2} \\ -\frac{6EI}{L^2} & \frac{2EI}{L} & \frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix}$$

and the derivative with respect to L:

$$\frac{dK}{dL}=\begin{bmatrix} -\frac{36EI}{L^4} & \frac{12EI}{L^3} & \frac{36EI}{L^4} & \frac{12EI}{L^3} \\ \frac{12EI}{L^3} & -\frac{4EI}{L^2} & -\frac{12EI}{L^3} & -\frac{2EI}{L^2} \\ \frac{36EI}{L^4} & -\frac{12EI}{L^3} & -\frac{36EI}{L^4} & -\frac{12EI}{L^3} \\ \frac{12EI}{L^3} & -\frac{2EI}{L^2} & -\frac{12EI}{L^3} & -\frac{4EI}{L^2} \end{bmatrix}$$

It is readily shown that the rigid body conditions are satisfied for the eigen vectors:

$$\text{rigid body translation}: \quad \boldsymbol{\phi}_t^{\mathbf{T}} = \begin{bmatrix} b & 0 & b & 0 \end{bmatrix}^T$$

$$\text{rigid body rotation}: \quad \boldsymbol{\phi}_r^{\mathbf{T}} = \begin{bmatrix} \frac{L}{2}\vartheta & \vartheta & -\frac{L}{2}\vartheta & \vartheta \end{bmatrix}^T$$

Now, let us take a look at the forward finite difference approximation $\Delta\mathbf{K}/\Delta L$ where the length increment is taken as Δ:

$$\frac{\Delta K}{\Delta L}=\begin{bmatrix} -\frac{12EI\left(3L^2+3L\Delta+\Delta^2\right)}{\left(L+\Delta\right)^3L^3} & \frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & \frac{12EI\left(3L^2+3L\Delta+\Delta^2\right)}{\left(L+\Delta\right)^3L^3} & \frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} \\ \frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & -\frac{4EI}{\left(L+\Delta\right)L} & -\frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & -\frac{2EI}{\left(L+\Delta\right)L} \\ \frac{12EI\left(3L^2+3L\Delta+\Delta^2\right)}{\left(L+\Delta\right)^3L^3} & -\frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & -\frac{12EI\left(3L^2+3L\Delta+\Delta^2\right)}{\left(L+\Delta\right)^3L^3} & -\frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} \\ \frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & -\frac{2EI}{\left(L+\Delta\right)L} & -\frac{6EI\left(2L+\Delta\right)}{\left(L+\Delta\right)^2L^2} & -\frac{4EI}{\left(L+\Delta\right)L} \end{bmatrix}$$

Applying the rigid body test by pre- and post-multiplying with the eigen vectors gives:

$$\begin{aligned} \boldsymbol{\phi}_t^T \frac{\Delta\mathbf{K}}{\Delta L}\boldsymbol{\phi}_t^T &= 0 \\ \boldsymbol{\phi}_r^T \frac{\Delta\mathbf{K}}{\Delta L}\boldsymbol{\phi}_r^T &= \frac{12\,EI\,\Delta\,\vartheta^2}{\left(L+\Delta\right)^3} \neq 0 \quad !!! \end{aligned} \tag{6.28}$$

and shows the error w.r.t. the rigid body rotations. As L becomes smaller the error, obviously, becomes larger which is the reason for the bad properties of the semi-analytical sensitivity analysis. All finite element formulations have this defect where the stiffness components are dependent of different powers of the design variable, i.e. for the beam 1/L, $1/L^2$, and $1/L^3$.

There are several remedies given in the literature which all try to fix the rigid body test. A most straight forward idea is the following: Add to $\Delta\mathbf{K}/\Delta L$ an additional matrix which takes away the error and which just is generated by the dyadic product of $\boldsymbol{\phi}_r$:

$$\boldsymbol{\phi}_r^T\left(\frac{\Delta\mathbf{K}}{\Delta L}+a\boldsymbol{\phi}_r\boldsymbol{\phi}_r^T\right)\boldsymbol{\phi}_r=0 \tag{6.29}$$

which yields the factor a:

$$a=-\frac{48\,EI\,\Delta}{\vartheta^2\,(L+\Delta)^3\left(L^2+4\right)^2} \tag{6.30}$$

$\Delta\mathbf{K}/\Delta L$ modified as in 6.29 gives the robust and mesh independent results which are reported in the last column of Table 6.1.

7 Multi criteria optimization

7.1 Definition

Many optimization problems are characterized by more than one objective function. We are speaking of *multi-objective* or *multi criteria optimization*. If we define p objective functions, they can be arranged in a vector format:

$$\mathbf{f}(\mathbf{x}) = \begin{Bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \dots \\ f_p(\mathbf{x}) \end{Bmatrix}; \quad \mathbf{x} = x_1, x_2, \dots, x_n \tag{7.1}$$

Therefore, we speak also of *vector optimization*. In general, objectives are conflicting. E.g., a structure is either the cheapest or the lightest, i.e. the objectives "costs" and "weight" are conflicting. In such cases all we can optimize is a compromise of all the different objectives.

7.2 Dealing with conflicting objective functions

7.2.1 The compromise function

The simplest way to deal with conflicting objectives is to define one composite objective which replaces all the objectives. Weight factors w_i determine the importance of objectives f_i:

$$F(\mathbf{x}) = \sum_{i=1}^{p} w_i f_i(\mathbf{x}) + C \tag{7.2}$$

The optimal solution is dominated by the choice of weighting factors which presumes the kind of compromise. The constant C might be introduced to shift the compromise function. It does not affect the optimal solution.

Example, (n = 1, p = 2). objectives: $f_1 = (3-x)^2 + 10$ $f_2 = (8-x)^2 + 5$

compromises[1]: $F_1 = f_1 + f_2 - 25$ $x^* = 5.5$

$F_2 = 5f_1 + f_2 - 60$ $x^* = 3.833$

$F_3 = f_1 + 3f_2 - 32$ $x^* = 6.75$

The optimal solution depends on the choice of weighting factors:

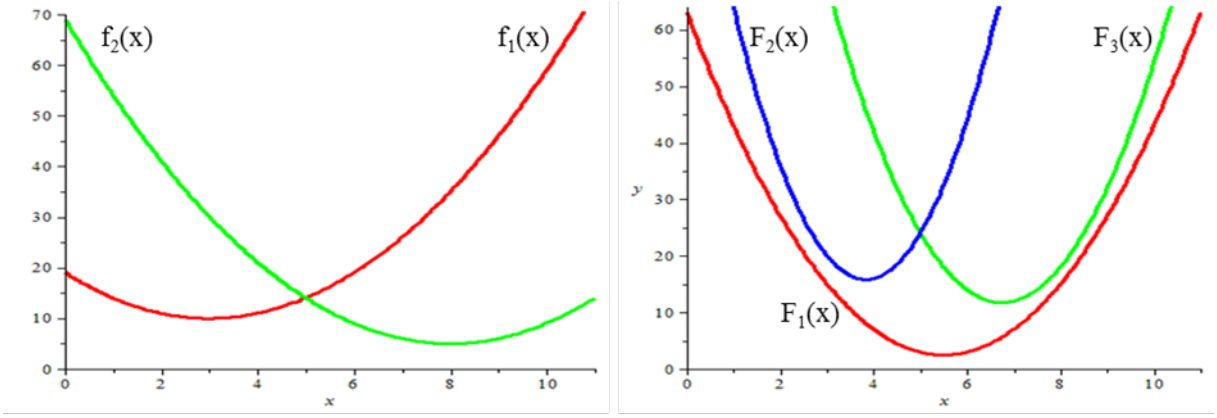


Fig. 7.1: Two objectives f_1 and f_2 (left) and three combinations F_1, F_2 and F_3 (right)

[1] the constants are added to shift the functions for visualization; they don't affect the solution

7.2.2 Hierarchical approach

The second intuitive way is to select the most important objective as the only one and to impose limits on the others. Thus, the problem is reduced to the standard format.

Example, (n = 1, p = 2). Choose f_1 as the only objective function. Take f_2 to define the constraint $g = f_2 - 15 = (8 - x)^2 - 10 \le 0$. The solution is defined as $x^* = 4.84$ where $g = 0$, Fig. 7.2.

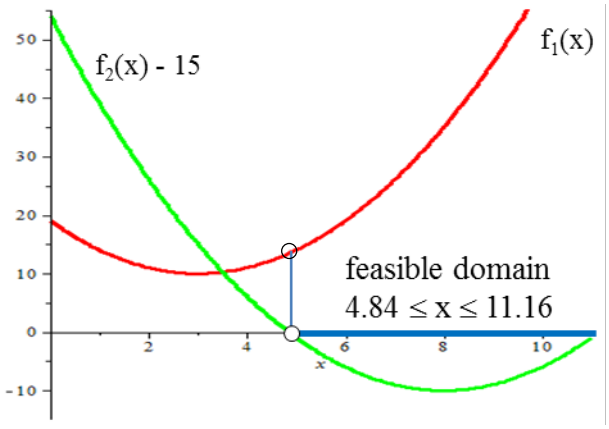


Fig. 7.2: Optimal solution using f_2 as constraint

7.2.3 Pareto optimization

A systematic approach to find an optimal compromise is a strategy called *Pareto optimization* with respect to the Italian engineer and social scientist Vilfredo Pareto (1896). A solution **x*** is defined to be Pareto optimal if for any other choice of variables **x** at least one of all objective functions becomes worse. This definition describes in general a set of possible solutions from which a final choice has to be done. This reflects that all acceptable compromises are of equal quality.

Example. Consider the above functions f_1 and f_2. Choosing x from the interval $\{3 \le x \le 8\}$ for increasing x, obviously, $f_2(x)$ decreases while $f_1(x)$ increases. Thus, $\{x \mid 3 \le x \le 8\}$ defines the set of Pareto optimal solutions, Fig. 7.3.

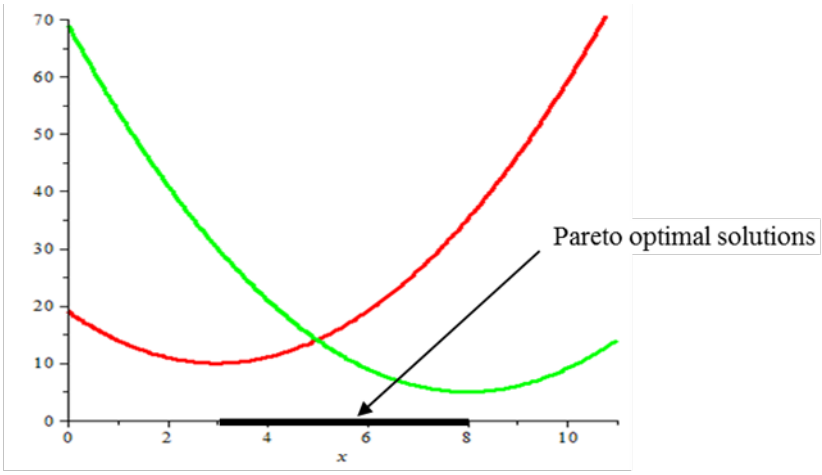


Fig. 7.3: Set of Pareto optimal solutions.

7.3 The space of objective functions

A certain design is expressed in terms of the design variables x_i. The design variables span what is called the *design space* or *space of variables*. In terms of the multi-objective optimization each objective f_p is a function of the design variables $f_p(x_i)$, i.e. a chosen design x_i results in a combination of individual objectives f_p which span the *space of objective functions*.

Example. One design variable (n = 1), two objective functions (p = 2).

objectives: $f_1 = (3-x)^2 + 10$ $f_2 = (8-x)^2 + 5$

point wise transformation from design space x into objective space (f_1, f_2)

Obviously, the one dimensional design space transforms into a curve in the objective space where x can be interpreted as the curve parameter.

design space	objective space	
x	f_1	f_2
0	19	69
1	14	54
2	11	41
3	10	30
4	11	21
5	14	14
6	19	9
7	26	6
8	35	5
9	46	6
10	59	9

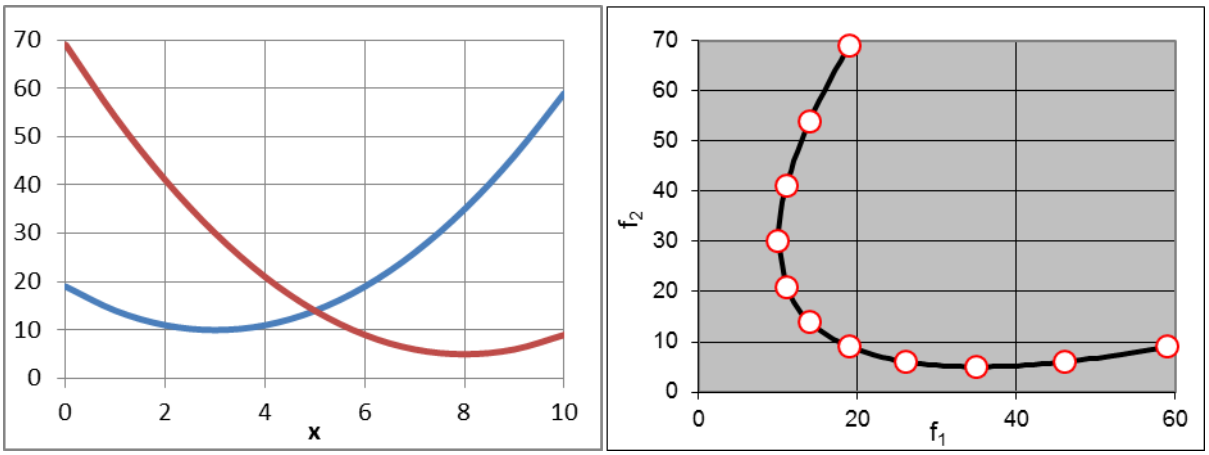


Fig. 7.4: Design space (left), Space of objective functions (right)
dots represent specific designs from x = 0 to x = 10

Example. Two design variables (n = 2), two objective functions (p = 2).

objectives: $f_1 = (x+1)^2 + (y+1)^2 + 5$ $\quad \mathbf{x}^* = [-1 \quad -1]^T$

$f_2 = 2(x-2)^2 + (y-3)^2 + 2xy$ $\quad \mathbf{x}^* = [\ 1 \quad\ 2]^T$

point wise transformation from design space (x, y) into objective space (f_1, f_2)

The two dimensional design space transforms into a two dimensional objective space which is bounded by the black curve from left and from below. This curve is called *functional efficient edge*. Along the functional efficient edge the feasible domain is "folded", i.e. there exist several designs (x,y) which are represented by the same values of f_1 and f_2. As an example refer to the two bottom lines of the table to the right.

design space		objective space	
x	y	f_1	f_2
1	-1	9	16
-6	0	31	137
-4	6	63	33
2	-2	15	17
-2	-1	6	52
2	-3	18	24
-5	6	70	47
1	-4	18	43
6	-5	70	36
-3	3	25	32
-6	-5	46	252
0	-5	22	72
-1	-3	9	60
-2.871	-0.2941	9	60

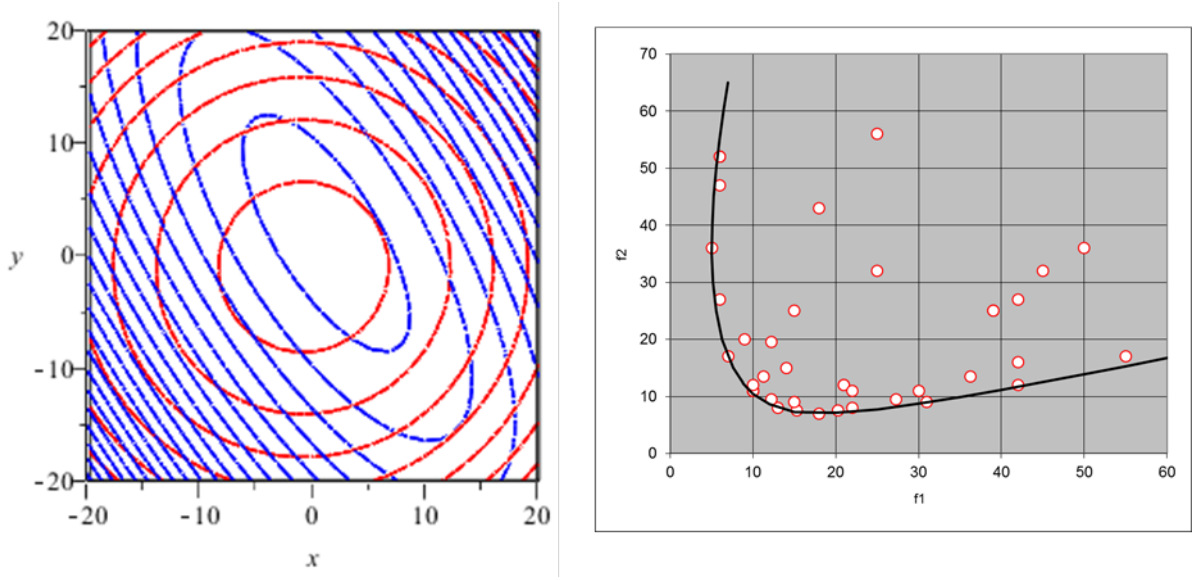


Fig. 7.5: Design space (left), Space of objective functions (right)
dots represent objective values of specific designs (x,y)

7.4 Pareto optimal solutions in the space of objective functions

If we transform the problem into the space of objective functions the Pareto optimal set can be clearly identified. The curve in Fig. 7.6 represents all pairs $\{f_1(x), f_2(x)\}$ where x is acting as a curve parameter, refer to Fig. 7.4. Obviously, above $f_2 = 30$ and right of $f_1 = 35$ both functions are increasing simultaneously. The Pareto optimal solutions are defined to be right of the vertical and above of the horizontal tangent. In that range of the curve one function is increasing whilst the other one is decreasing which characterizes the conflicting situation.

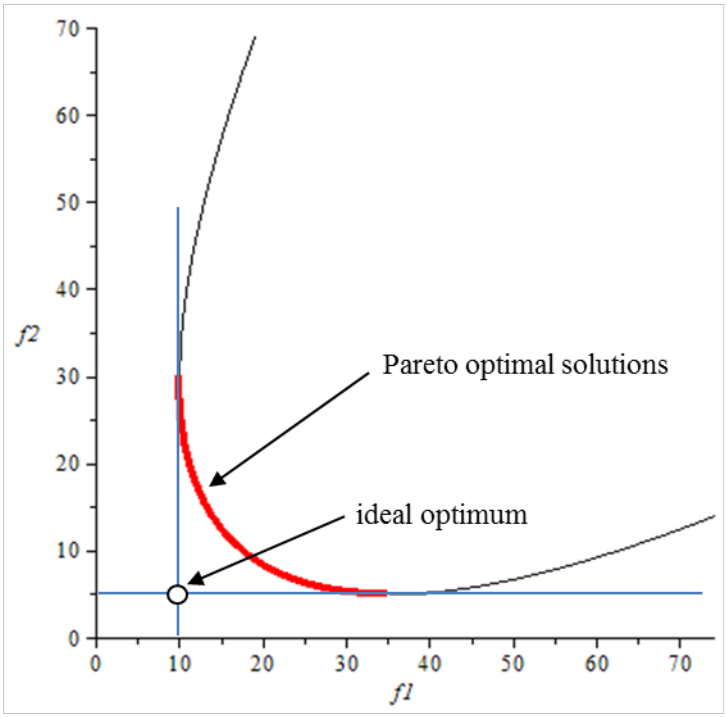


Fig. 7.6: Pareto optimal set in the space of objective functions.

7.5 Design quality

For the standard case of optimization with one objective function a design $\mathbf{x}_A$ is better than another design $\mathbf{x}_B$ if it holds $f(\mathbf{x}_A) < f(\mathbf{x}_B)$. For the multi criteria case the situation is more complicated. The following definitions hold:

Design A is said to dominate design B, i.e. A < B, if:

$$f_j(\mathbf{x}_A) < f_j(\mathbf{x}_B), \quad j \in P = \{1, 2, ..., p\}$$

Vice versa, design B is dominated by design A, i.e. B > A, if:

$$f_k(\mathbf{x}_B) > f_k(\mathbf{x}_A), \quad k \in P = \{1, 2, ..., p\}$$

Two designs A and B are said to perform equally if the two conditions do not hold, i.e.:

$$f_j(\mathbf{x}_A) < f_j(\mathbf{x}_B) \;\; j \in J \quad \text{and} \quad f_k(\mathbf{x}_A) > f_k(\mathbf{x}_B) \;\; k \in K, \quad J \cap K = \emptyset \quad \text{and} \; J \cup K = P$$

That is displayed for the case of a two dimensional objective space as follows. Compared to design A represented by related objective values another designs located in any of the four

displayed quadrants have the specified properties, as there are “dominating”, “dominated” or “equally performing, Fig. 7.7.

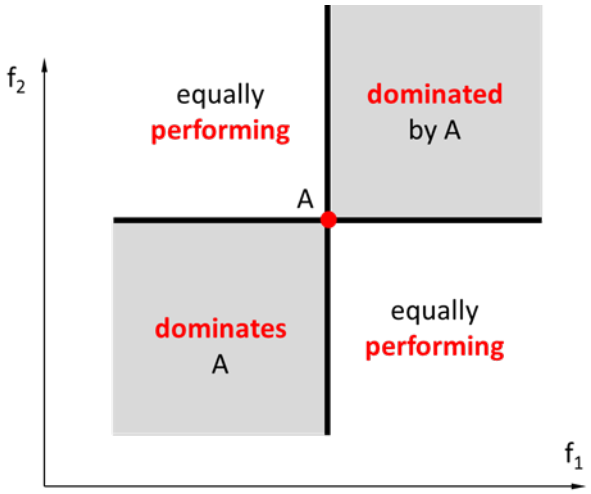


Fig. 7.7: Related properties of different design regarding design quality.

7.6 Methods to identify Pareto optimal solutions

The Pareto optimal solutions are those solutions of the functional efficient edge which are performing equally, i.e. any Pareto optimal solution is neither dominating nor is it dominated by its neighbors on the functional efficient edge. That is visualized by applying the “cross lines” of Fig. 7.7. If the functional efficient edge crosses the “equally performing” quadrants than that part of the edge indicates the *Pareto optimal set of solutions* which represent the set of best compromises. Still, however, one gets more than one solution, i.e. further decisions have to be taken to arrive at a final design which might be realized.

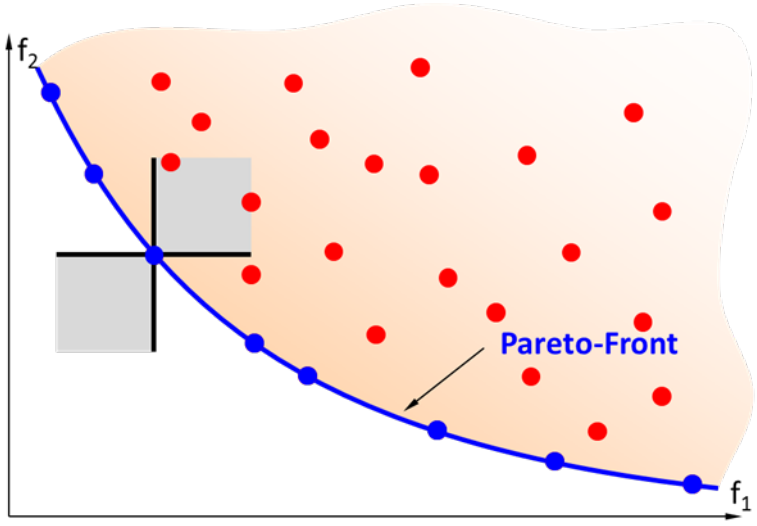


Fig. 7.8: Identifying Pareto optimal solutions at the functional efficient edge.

7.7 Selecting from the Pareto set – The min-max solution

From the Pareto set which represents the set of all reasonable compromises a best one can be defined. For that reason the quality of a solution $f_i(\mathbf{x})$ is defined by its normalized distance d_i from the individual optimum f_i^*:

$$d_i(\mathbf{x}) = \frac{f_i(\mathbf{x}) - f_i^*}{f_i^*}; \quad i = 1, \ldots, p \tag{7.3}$$

In the case that the individual optimum is zero the denominator of (7.3) has to be replaced by a number which represents the characteristic dimensions of the objective.

Now, the best compromises can be defined e.g. by the minimum of the largest deviation (L_∞ norm, *min-max optimum*):

$$\text{minimize} \quad \max_{i=1,\ldots,p}(d_i(\mathbf{x})) \tag{7.4}$$

Another possibility is to define the point defined by the minimum distance from the reference point $\mathbf{f}^* = (f_1^*, f_2^*, \ldots, f_p^*)$ (L_2 or Euclidean norm):

$$\text{minimize} \quad \sum_{i=1}^{p} d_i^2 \tag{7.5}$$

Example. Taking the individual minima of the above functions f_1 and f_2 as reference point:

$$f_1^* = 10 \text{ and } f_2^* = 5 \tag{7.6}$$

Using the Euclidean norm the optimal compromise is evaluated as:

$$\text{minimize} \quad \sum_{i=1}^{p} d_i^* = (f_1 - f_1^*)^2 + (f_2 - f_2^*)^2 \tag{7.7}$$

from which we get the optimal result $x^* = 5.5$.

For this example the min-max optimum is identical, i.e. $x^* = 5.5$.

For two objective functions the two mentioned optimality criterions can be graphically demonstrated.

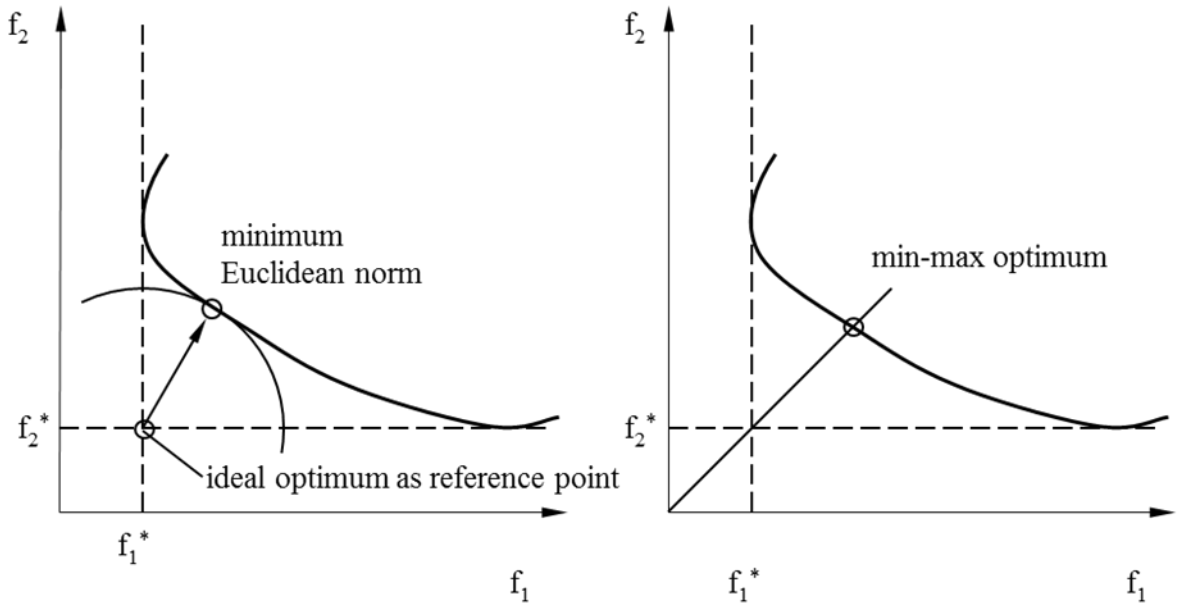


Fig. 7.9: Choice of optima from functional efficient edge.

Further methods can be derived by choosing other reference points as indicated above. Each choice, however, will prefer another solution from the Pareto set. The practical experience with respect to the considered application may be incorporated.

Non-convex objective functions give non-convex edges of the feasible domain in the space of objective functions. The functional efficient edge is split into several pieces.

Example. Consider the two objective functions:

$$f_1(x) = 60 - 40x^2 + \frac{110}{9}x^2 - \frac{40}{27}x^3 + \frac{5}{81}x^4 \quad \text{and} \quad f_2(x) = (4-x)^2 + 5$$

The Pareto optimal solutions are graphically represented:

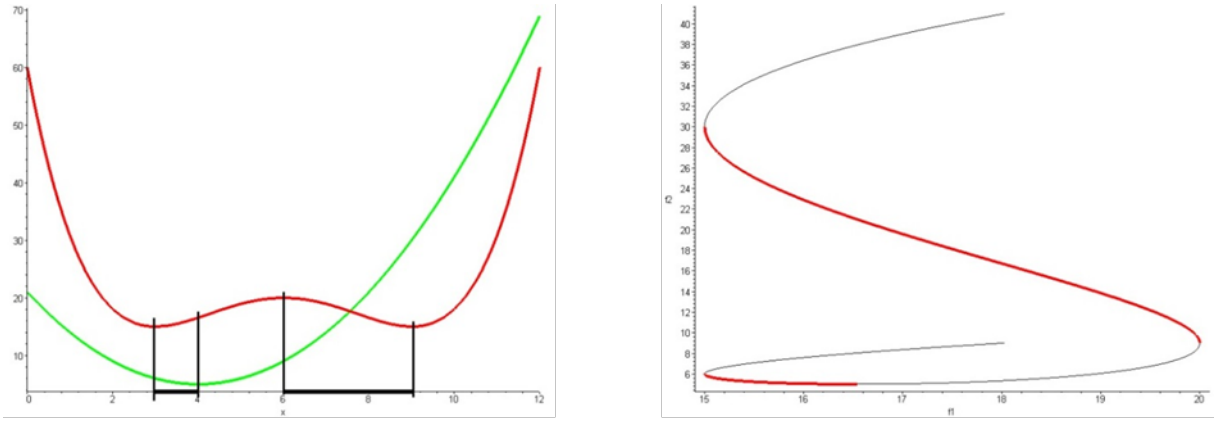
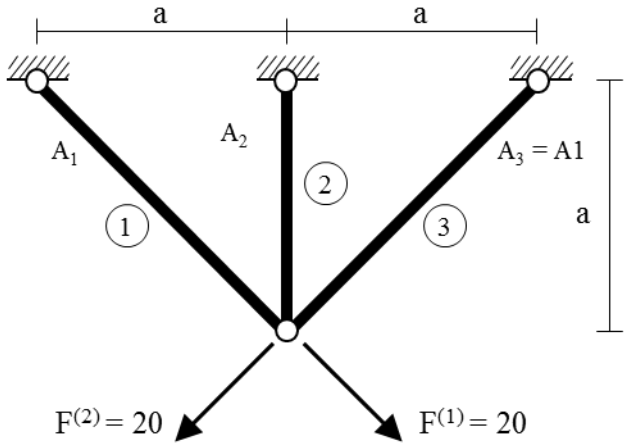


Fig. 7.10: Non-convex objective functions: Pareto optimal solutions in design space (left) and objective function space (right)

7.7.1 Multi-objective optimization of the three bar truss

For the further explanation of multi-objective optimization let us again consider the three bar truss problem.



Additional to the setting as given before we now consider an additional objective function, the strain energy π:

$$\pi = \frac{1}{2}\sum_{i=1}^{3}\frac{N_i^2\,\ell_i}{E\,A_i} \tag{7.8}$$

Again, the cross sections A_1 and A_2 are the optimization variables. Because of symmetry A_3 is equal to A_1. Minimization of the strain energy means to maximize the stiffness. This goal is achieved by increasing the structural mass, i.e. by setting the cross sectional areas to their upper bound. Obviously, the objective "strain energy" is in conflict to weight or volume. The complete problem states as:

$$\text{minimize} \quad \left\{ \begin{array}{c} V = \sqrt{2}\,200\,A_1 + 100\,A_2 \\ \pi = \dfrac{1}{2E}\left(\dfrac{\sqrt{2}\,100\,N_1^2}{A_1} + \dfrac{100\,N_2^2}{A_2} + \dfrac{\sqrt{2}\,100\,N_3^2}{A_1} \right) \end{array} \right\}$$

$$\text{subject to:} \quad g_1 = 20\frac{A_2 + \sqrt{2}\,A_1}{2\,A_1A_2 + \sqrt{2}\,A_1^2} - 20 \le 0 \tag{7.9}$$

$$g_3 = 20\frac{A_2}{2\,A_1A_2 + \sqrt{2}\,A_1^2} - 15 \le 0$$

$$0 \le A_1 \le 2\,; \quad 0 \le A_2 \le 3$$

Additionally, we assume a = 100. As we have seen the stress in member 2 is not critical, thus we can reduce the set of constraints to g_1 and g_3. Looking at the graphical representation we identify the different locations of the objective functions: the minimum with respect to the volume V on g_1 as before and the minimum with respect to π at the upper right corner of the feasible domain where both Areas (A_1 and A_2) reach their upper bound.

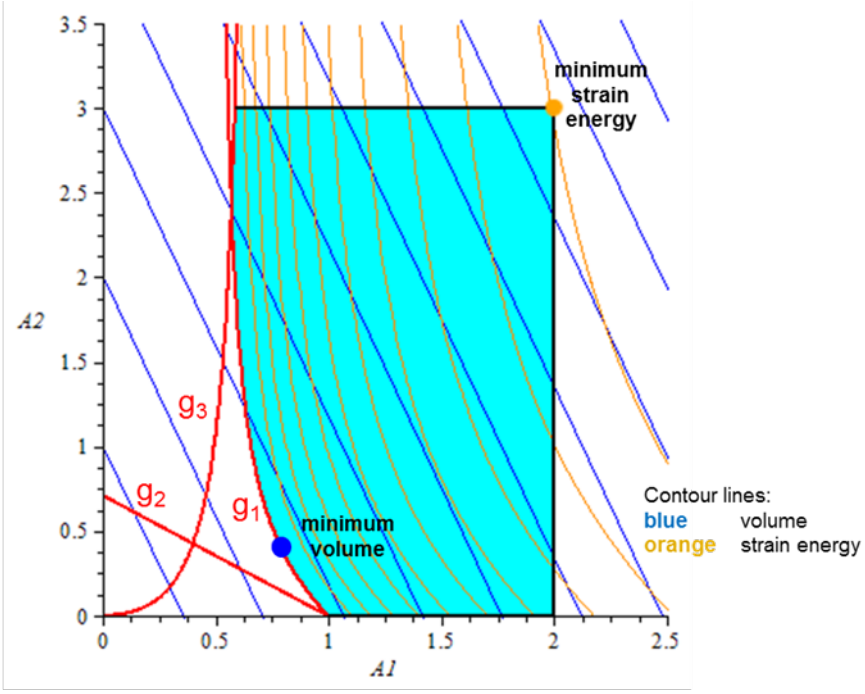


Fig. 7.11: 3 bar truss: optimal solutions for volume and strain energy

To find a good compromise of the both conflicting objectives, first, the functional efficient edge of the feasible domain must be identified.

Usually, in structural optimization the problem functions are not explicitly known as in this simple example. The objective functions are determined point wise by complicated and time consuming numerical analyses for feasible values of the optimization variables. For the case of two objectives the results can be plotted as points in a x-y graph, Fig. Fig. 7.12.

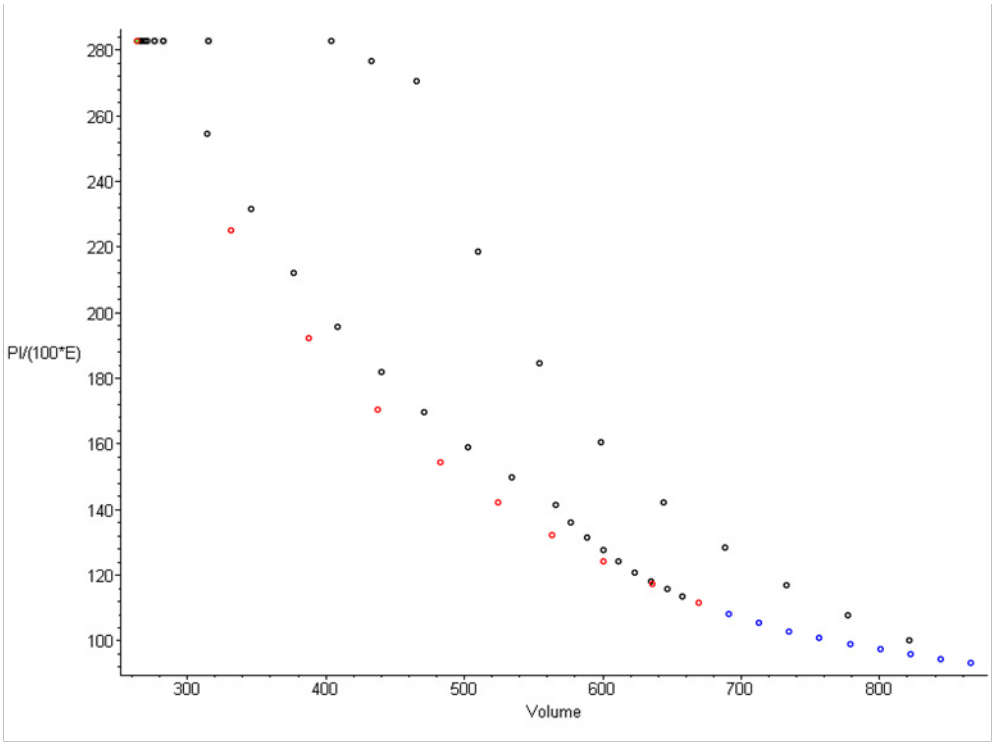


Fig. 7.12: Point wise evaluation of the feasible domain in the space of objectives.

For more than one variable the feasible domain in the objective space is a plane with curved boundaries. The band of lowest and left most points gives a representation of the functional efficient edge.

Note, that the functional efficient edge is not necessarily identical with a boundary of the feasible domain in the design space. The functional efficient edge links the individual minima more or less directly by some curve. Mapped back into the design space this curve still connects both minima more or less directly. For the considered example the map of the functional efficient edge runs at least partially through the feasible domain in the design space. In the design space the functional efficient edge consists of two straight lines which connect both minima, Fig. 7.13. One line runs through the feasible domain, the other along the upper bound of variable A_1.

The set of best compromises must not necessarily consist of constrained solutions. As a matter of fact the min-max optimum is inside the feasible domain as indicated in the figure. The feasible domain is split by the functional efficient edge into two parts. Since in the objective space the functional efficient edge restricts the feasible domain, the two parts below and above the edge are mapped onto each other, compare the hatched domains in Figs. 7.13 and 7.15. The mapping is irreversible. Or, with other words, designs which are located left and right of the functional efficient edge (in the design space) are both not optimal compromises.

Designs on the functional efficient edge can be numerically determined by applying the weighted sum approach. It can be shown that the solutions which are found by replacing the objective functions by the composition

$$F(\mathbf{x}) = \sum_{p} w_i \, f_i(\mathbf{x}) \tag{7.10}$$

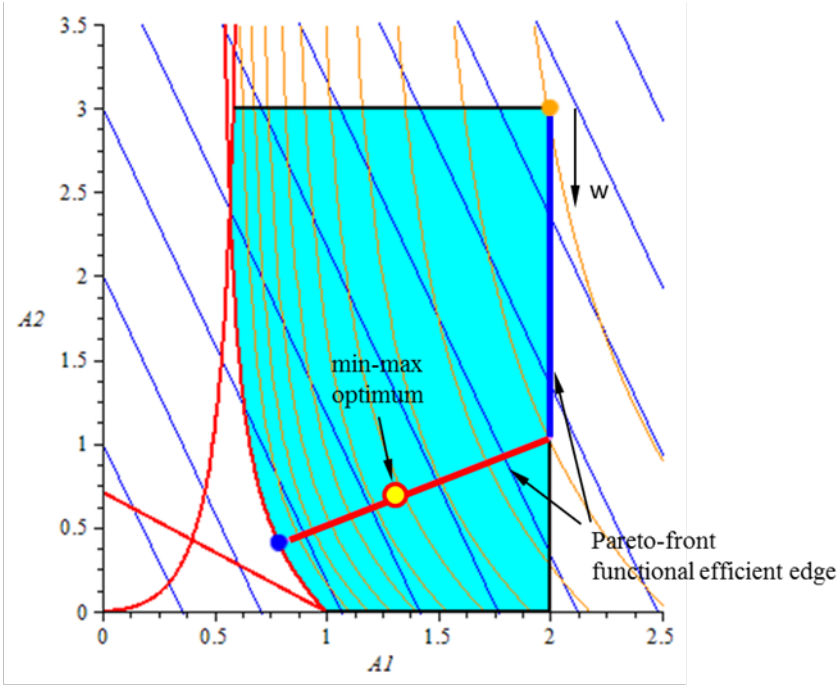


Fig. 7.13: Functional efficient edge in the design space.

are on the functional efficient edge. In the case of non-convex problems it is possible, however, that the efficient edge cannot be detected completely by this approach. For two objectives one can modify the weighted sum as

$$F(\mathbf{x}) = w\,\frac{f_1(\mathbf{x})}{f_1^*} + (1-w)\,\frac{f_2(\mathbf{x})}{f_2^*};\quad 0 \le w \le 1 \tag{7.11}$$

where w now plays the role of a curve parameter. Again, f_i^* are the individual minimum objective values which are used to normalize the effect on the composed function F. If w is set to the bounds (0, 1) the individual objective functions and their solutions are approached.

The functional efficient edge is found pointwise for specified w by solving the modified problem:

$$\begin{aligned} \text{minimize}\quad & F(\mathbf{x}, w) = w\,\frac{V(A_1, A_2)}{V^*} + (1-w)\frac{\pi(A_1, A_2)}{\pi^*} \\ \text{subject to}\quad & g_1(A_1, A_2) \le 0 \\ & g_1(A_1, A_2) \le 0 \\ & 0 \le A_1 \le 2;\quad 0 \le A_2 \le 3 \end{aligned} \tag{7.12}$$

The solution $A_1^*(w)$, $A_2^*(w)$ defines one point on the functional efficient edge in the design space. In the space of objectives the efficient edge is defined by

$$V^*(w) = V\left(A_1^*(w), A_2^*(w)\right) \quad \text{and} \quad \pi^*(w) = \pi\left(A_1^*(w), A_2^*(w)\right) \tag{7.13}$$

The plot of V^* and π^* over the curve parameter w is shown in Fig. 7.14. The slope discontinuities are due to a change of active constraints:

range	active constraints	individual minimum of
$.000 \le w \le .126$	bounds: $A_1 \le 2$ and $A_2 \le 3$	strain energy
$.126 \le w \le .320$	bound: $A_1 \le 2$	
$.320 \le w \le .752$	no constraint active	
$.752 \le w \le 1.0$	constraint g_1, stress in member 1	volume

Since the objectives remain constant in the first and last sector, these sectors collapse into a point when mapped into the space of objectives. Comparing with the graph of the efficient edge in the space of objectives, Fig. 7.15, the visual range varies there between $.126 \le w \le .752$. The bounds define the effective start and ending points.

In practice, one optimization analysis has to be performed for each choice of w, the solutions are plotted into a graph to detect the functional efficient edge, Fig. Fig. 7.12. Note, that the start and ending points are not necessarily related to w = 0 and w = 1. In fact the efficient edge can vary between a small range of w, which, usually, has to be found by trial and error. The choice of compromises can be done conveniently from the graph. In the case of more than two objectives the evaluation must be done numerically, e.g. applying the Euclidean norm to evaluate a compromise which is as close as possible to a desired reference design.

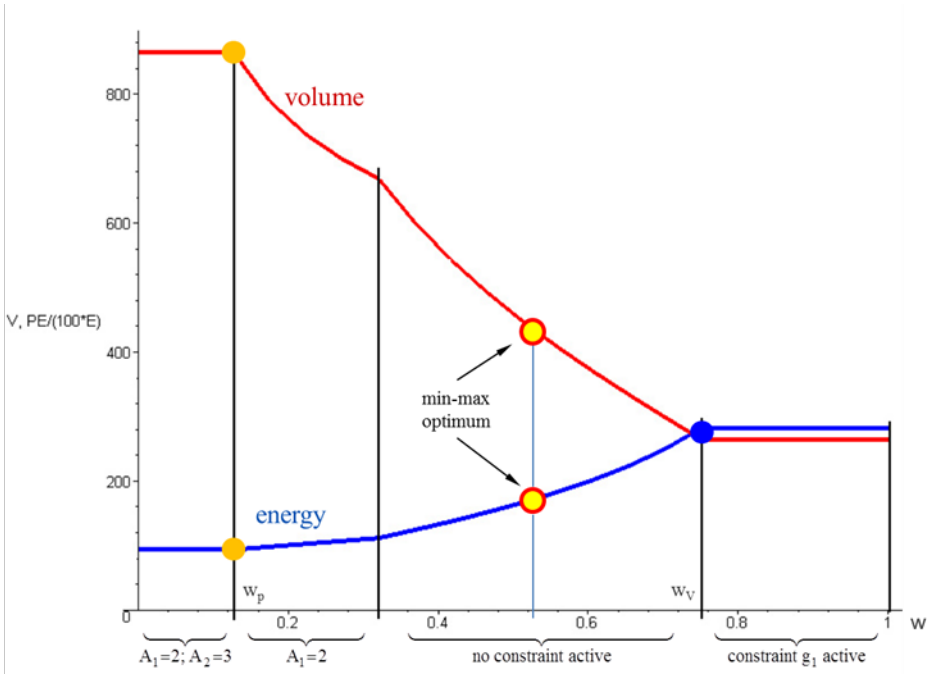


Fig. 7.14: Trace of the functional efficient edge for $0 \leq w \leq 1$

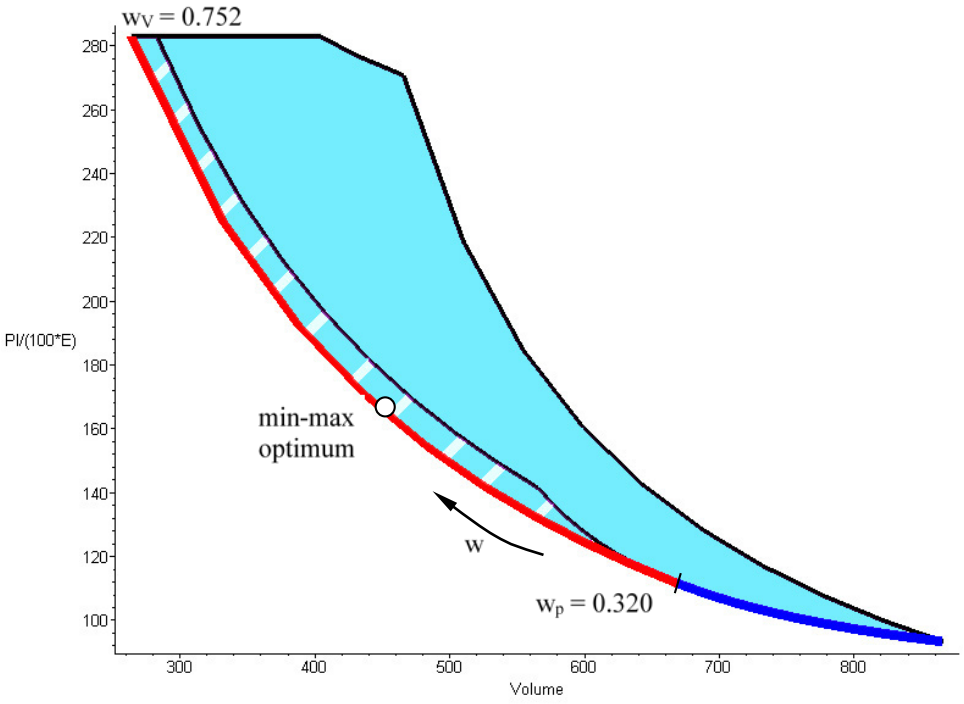


Fig. 7.15: Feasible domain in the objective space; the thin solid lines reflect the boundaries of the feasible domain in the design space.

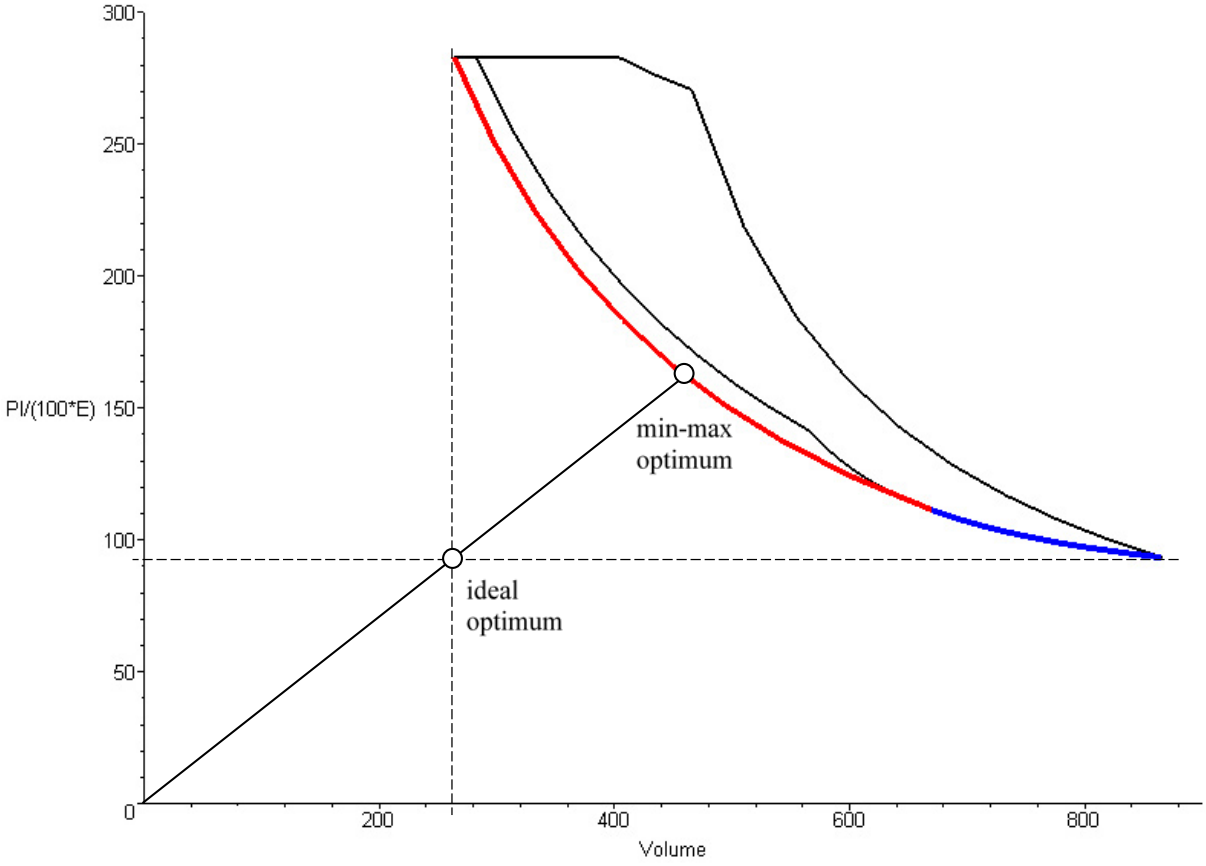


Fig. 7.16: Graphical construction of the min-max optimum

8 Shape Optimization

This chapter is coincident with the contribution about
"Shape optimization" by Kai-Uwe Bletzinger
for the Encyclopedia of Computational Mechanics, 2nd edition.

Encyclopedia of Computational Mechanics Second Edition,
Edited by Erwin Stein, René de Borst and Thomas J.R. Hughes.
Volume 2: *Solids and Structures*.
ISBN: 978-1-119-00379-3.

8.1 Introduction

8.1.1 The challenge of shape optimization

Numerical shape optimization of structures and technical objects is a challenge. It combines engineering and physical insight into the behavior of the object with the mathematics of optimization methods and the esthetics of shape. Even those who have been educated to be as rational as can be are not free from emotions about shape. The experience tells that optimized shapes are not accepted if they do not look “good”. It is very difficult to promote new, optimal shapes which differ from the traditional ones although they prove to be better in all technical aspects. Just remind the inverted hanging model as a generator of traditional optimal shapes of shells and domes and test yourself.

Another very important reason supporting the emotions about shape optimization is that very often there exist more than one optimal solution among them the global and many local ones. In engineering even local optima are very attractive as the finalized solution. Remind that the formulation of numerical optimization problems typically needs to simplify the model, e.g. regarding loading, boundary conditions etc. Experience tells that exploring the design space by optimizing several idealized but less constrained problems gains more insight into the potential of the object under inspection than solving one optimization problem which is overloaded by too many constraints.

Also, many shape optimal design problems are difficult to be mathematically formulated. They appear to be vague and need additional regularization. From an application point of view that reflects the engineering and esthetical freedom of shape optimization. As an example, refer to the optimal design of a stiffened plate by creating deep drawn beads. No optimal solution might be found without some regularization regarding the surface curvature.

There is an enormous potential for shape optimization in all fields of engineering and other applications. Still, however, mainly because of the three mentioned and closely related reasons shape optimization is in its infancy compared to topology optimization and other fields of numerical simulation and computational mechanics. But it is obvious that time is changing and shape optimization slowly but steadily is intruding everyday design procedures.

This text will give some basic information about those aspects which make shape optimization different from other problem types of optimization. It will concentrate on those topics which deal with the control of shape (continuous and parametric), the regularization of problem formulation, the treatment of the infinity of design space and some further aspects which are specific to numerical and parametric shape optimization. As a consequence, there will nothing be presented about general optimization theory or algorithms, specific aspects of implementation and applications. In particular, a unified approach will be displayed which allows to see the floating transition between free shape optimization and CAD based parametric shape optimization on a sound theoretical basis.

8.1.2 Types of shape optimization

Surprisingly enough, the term “shape optimization” is differently interpreted among the various engineering and mathematical disciplines. In engineering, it is a standard task to design details of the structural layout and shape against the material limits, the impact of load and many other constraints. Typical design objectives are chosen to save material, cost or to improve structural

efficiency. Shape optimization of that type, hereupon, shall be called *technical design optimization*. It is characterized by the vast amount of constraints and small degrees of freedom to modify the shape. That is in dramatic contrast to another type of shape optimization which shall be called *form finding*. In that context, the methods of shape optimization are used to explore the design space for new, ideally so far unknown shapes. Consequently, those problems are characterized by a very large, ideally infinite number of degrees of freedom and very few, often none constraints. There is no doubt that the true potential of shape optimization is form finding. Most of the available mathematical theory about shape optimization deals with form finding as well. Many of the technical design optimization problems can be treated by standard parametric optimization and will not be treated here. Instead, we will start with the treatment of continuously defined shape and derive methods to control and formulate shape in this context. Further on we will show how those methods can be downsized to parameterized shape optimization as applied for technical design optimization. There is a smooth transition between the extremes.

8.1.3 Numerical methods for shape optimization

The choice of an adequate numerical method for the solution of a shape optimization problem depends on the number of design parameters versus the number of constraints and the mathematical properties of the underlying physical problem. Gradient methods are the technique of choice for problems with many design parameters, in particular form finding problems. In the context of fluid optimization gradient methods are standard.

The gradients of objective and constraints are determined by sensitivity analysis. Adjoint sensitivity methods are preferable if the number of design parameters exceeds the number of response functions which are the objective and constraint functions. Adjoint methods are able to determine the gradients of a response function for the effort of one additional solution of the state equation, independent of the number of design parameters. The sensitivity analysis is a general technique of numerical optimization and system identification. As well it is a basic technology for stochastic and reliability analysis. Insofar it is not a specific technology for shape optimization and will not be discussed in details, here. In the case of interest, please, refer to the vast available literature. Instead, some comments will be given regarding the semi-analytical sensitivity analysis which is an efficient alternative to the complete analytical sensitivity analysis of structures in the specific context of shape optimization.

Since all approaches need discretization to be solved numerically there are two options: Starting from a variational formulation, (i) first linearize, then discretize or, (ii) first discretize, then linearize. The first line allows for basic theoretical analysis and is preferred in mathematics and for specific and/or computational extensive problems as many can be found in fluid optimization. In contrast, the second line easily allows to create a most modular software environment which nests different tools for the individual tasks of shape optimization, as there are geometrical design, physical simulation and mathematical programming. That approach is the technical standard of commercially available optimization tools for structural optimization.

Newton and sequential quadratic programming methods are rarely used for large scale shape optimization such as form finding. Steepest descent and versions of modified gradient methods define the state of the art. Together with gradient smoothing or gradient filtering they are converging fast and robust, in particular for very large problems.

Regarding the theoretical basis of methods for optimization and sensitivity analysis there is very intensive methodological research done in mathematics in all directions.

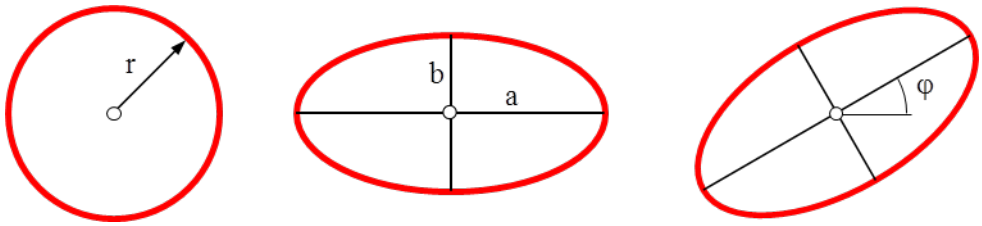


Fig. 8.1: Alternative models of a circle and its deformation into an ellipse.

All other kinds of direct and gradient free methods including evolutionary and genetic algorithms as well as methods for multi criteria problems might also be used for shape optimization which fit to the specific problem at hand. Response surface methods are used for discontinuous problems such as shape optimization in the context of crash simulation, as one prominent example.

8.1.4 Modeling of shape optimization problems

The geometric modeling for shape optimization differs from standard geometrical modeling which is a preprocessing step for numerical analysis. As an example, consider a circle, Fig. 8.1. It is uniquely defined by one parameter which is the radius r. However, if the circle is the initial geometry of a subsequent shape optimization additional geometric parameters have to be considered. The finally available shape degrees of freedom and the potential optimal shape completely depend on the vision of the modeler about what should be optimized or on arbitrary decisions depending on the software tool used. The circle might be alternatively modelled as an ellipse with two or three parameters (the lengths of two axes a,b and the rotation φ). Or, if it is modelled as a NURBS there exist further more alternatives regarding the number and positions of the control nodes and the related weights, Fig. 8.2. The example may illustrate the challenge for complex realistic problems. The extra parameters for the subsequent optimization complicate the modelling of the initial model. Eventually, discipline and labor time are necessary to set up a geometric model which is ready for shape optimization. Experience tells that this is one of the major reasons which prevent shape optimization to be performed. In contrast, setting up models for topology optimization is much simpler which helps to explain its success. Node based optimization also helps to simplify the modeling effort which opens great potentials.

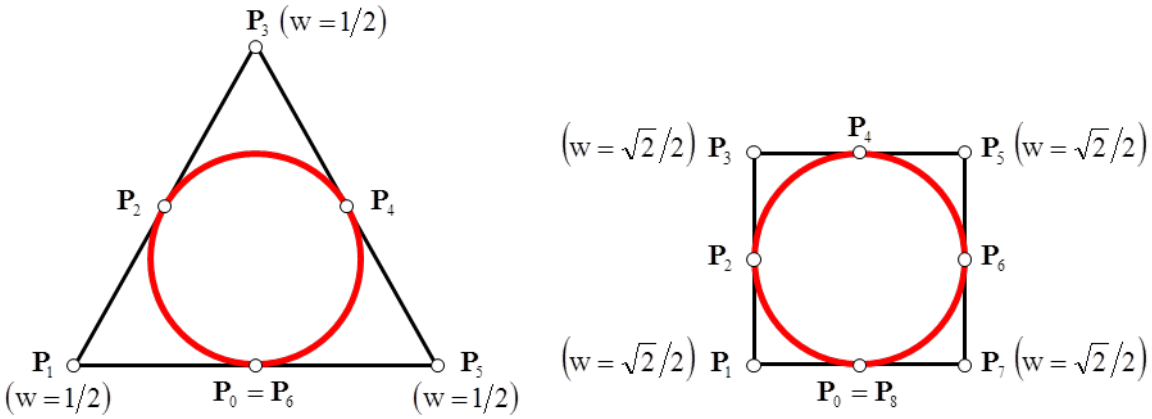


Fig. 8.2: Alternative NURBS models of a circle.

8.1.5 Shape control methods

Shape control methods are of basic importance for shape optimization. In principle, one divides into so-called *node based* and *CAD based* shape optimization. The options are also referred to as *parameter-free* (which means free from CAD parameters) and *parametric* shape optimization.

The node based methods directly use the geometric parameters of the discretization grid for the physical simulation. The advantage is the largest possible design space for the given grid. Shape is controlled by directly dealing with the grid nodes together with specific regularization techniques such as gradient smoothing or filtering. The formulation of node based optimization problems is most efficient. It typically needs no considerably extra effort on top on formulating the analysis model.

CAD-based methods use the CAD parameters as the design parameters. They define the actual industrial standard for shape optimization methods combining CAD preprocessors with simulation packages. As an advantage the optimal shape is directly formulated in CAD. Compared to node based methods no additional effort is necessary to retrieve the optimal shape back into CAD. Also, CAD-based shape optimization models use small numbers of design parameters. That is an advantage regarding the numerical effort and eventually allows to use other methods than gradient methods for optimization. On the other hand, however, reducing the design space increases the demand for carefully selecting the remaining design parameters, i.e. the CAD parameters. Experience tells that it is almost impossible to completely consider the consequence for shape optimization when CAD parameters are selected during the geometrical modelling procedure, refer to the previous chapter. That is a major issue if different departments work on geometrical and optimization modelling. Also, CAD models are not unique. Identical geometries can be created by different CAD parameterizations, Fig. 8.2. Taken for subsequent shape optimization, typically, those alternative models will generate different optimal solutions as a consequence of the arbitrary model decisions. Often it appears that CAD models have to be re-parameterized to be suitable for shape optimization. CAD based methods are preferable for technical design optimization rather than form finding.

The IGA based optimization technologies belong to the CAD-based methods and share their pros and cons. Beyond the technological aspects they have brought considerable additional power into the basic research of shape optimization.

Another in practice well accepted shape control technique is morphing. There exist several commercial implementations. Originally developed in computer graphics it is most simple to be applied technique. The object to be morphed is embedded into a morphing box and shares the box' strain field when the morphing box is deformed. Morphing boxes may also be applied to parts of the object only. Think of Pinocchio's nose as a typical application. The design parameters are the handles of the morphing box. Only few parameters are necessary to morph even geometrically complex objects. This is the advantage of the technique. One of the challenges is the unique definition of the morphing box for repeated application. Other problematic issues are the definition of geometrical boundary conditions of completely embedded objects and the combination of several boxes. Morphing box techniques are good choices for quick shape optimization in restricted design spaces with (very) few design parameters.

8.1.6 Shape optimization vs. sizing and topology optimization

Shape optimization is classified regarding complexity to be between sizing and topology optimization. Sizing treats the optimization of cross section or thickness parameters with fixed geometry. In contrast and following the original understanding, topology optimization deals with searching for the best alternative among all possibilities including topological discontinuities. For methodological reasons, topology optimization problems are typically transferred into sizing or shape optimization problems, e.g. density variations or level set techniques, respectively, to create void regions. Topology optimization shares with form finding the large number of design parameters. Shape optimization deals with topologically continuous objects. The difference between sizing and shape optimization is typically due to the underlying model. As an example, the thickness optimization, i.e. sizing, of a plate turns into a shape optimization problem if the plate is modelled as a 3D solid.

8.1.7 Shape optimization in the course of the design chain

The distinct stages of product development are supported by different types and methods of optimization, Fig. 8.3. On the time line there are: Conceptual design, preliminary design, detailed design and production. Topological optimization and sizing are used in conceptual and the detailed design stages, respectively. Shape optimization is the only method which is found in all stages, although with different focus. During conceptual design form finding is used as a creative mean to find new solutions. In the later stages parametric shape optimization is used for technical design optimization when the basic layout has evolved. As the CAD model evolves through the design stages CAD-based shape optimization is the choice in later stages, in particular for the detailed optimization or the design modification during production or revisions. Still, there are great unexploited potentials for form finding.

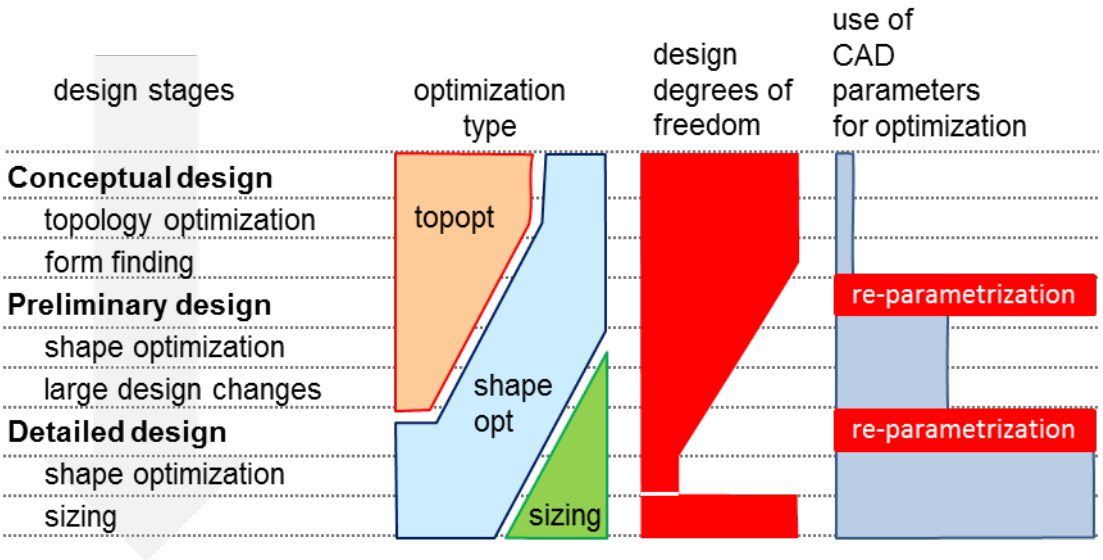


Fig. 8.3: Optimization during design stages; decreasing shape design degrees of freedom; increasing number of CAD parameters during design evolution.

8.1.8 A short review of shape optimization

Indeed, shape optimization is as old as rational engineering. Galileo Galilei is mentioned as one of the first who applied mechanical insight to the design of structures and laid first tracks in the direction where we are today. There is much to be told about the history and the beauty of shape optimal design but is beyond the scope of this text about the computational aspects of shape optimization.

Physical experiments for optimal structural design such as the hanging model have successfully been applied since centuries and still are today, in particular in architecture and civil engineering where form finding is of great importance [1, 2]. Design for maximum structural efficiency is the typical driving goal. A very special branch of shape optimization in architecture deals with the form finding of pre-stressed membrane and cable net structures which perfectly combine structural efficiency with esthetic elegance. Frei Otto was the pioneer creating those structures by physical experiments with minimal surfaces [3]. The Munich Olympic roof 1972 was the event to develop basic numerical methods in that regard. Linkwitz [4], Haug [5] and the group of Argyris [6] independently developed form finding, non-linear simulation and optimization methods for membranes.

One of the first papers in the specific field of general computational shape optimization was published by Zienkiewicz and Campbell in 1973 [7] about the shape optimization of 2D structures. They proposed a node-based approach and used the coordinates of the finite element nodes as design parameters. A preliminary version of the CAD-based approach had been published by Imam in 1982 about the shape optimization of 3D structures using so-called design elements [8]. The use of B-splines to control shape was reported by Braibant and Fleury in 1984 [9]. One of their major motivations was to carefully control the smoothness of shape. The state of the art in parametric shape optimization which had been reached at the end of the 1980s is reported in [10]. Recently, the isogeometric analysis (IGA) [11] motivated contributions about shape optimization by groups from mathematics and engineering [12-16]. As the same CAD-geometry is used for analysis and design, IGA is a variant of the CAD-based optimization approach.

Alternatively, Azegami developed the traction method for the node-based shape optimization of structures [17]. The basic idea is the smoothing of the shape gradient together with adjoint sensitivity analysis [18]. Also recently, other groups started to intensively work on node-based methods for structural optimization together with gradient filtering [19, 20]. Essentially, those methods are similar to the filtering methods in topology optimization [21].

In computational fluid dynamics and mathematics the node-based approach was preferred. The works of Pironneau [22], Mohammadi [23, 24] and Jameson [25-27] were groundbreaking contributions, both with respect to the adjoint sensitivity analysis as well as the gradient treatment for a robust descent method based on the Sobolev smoothing. Stück and Rung [28] introduced a global Gaussian type filter as an alternative. Regarding adjoint sensitivity analysis algorithmic [29, 30] and continuous approaches have been developed [31, 32]. Alternatively to the node-based optimization, also morphing boxes find application in practice, e.g. in aerospace shape optimization [33].

Actually, it appears that all the mentioned separated tracks of shape optimization merge. Besides the selected citations there is a vast and most valuable amount of further literature.

8.2 Continuous shape optimization

8.2.1 Shape evolution

Following the ideas of Hadamard [34], the understanding of shape optimization is based on the idea of design evolution through the evolutionary *pseudo time* t. Pseudo time represents not more than the sequence of steps of evolution in contrast to a dynamic process including inertia effects, Fig. 8.4. Consequently, through design evolution, the volume Ω of a body as well as its surface Γ are functions of t. The initial design is related to time t_0. At the surface we introduce material surface coordinates $\xi = \{\xi_1,\ldots, \xi_{d-1}\}$ in $\mathbb{R}^d$ which uniquely define *material points* at the surface. Typically, we consider d = {2,3} of 2- and 3-dimensional bodies. Material coordinates are invariant of pseudo time t. The position in Euclidean space of a material point at every instance t is given by its spatial position vector $\mathbf{x}(\xi,t) \in \mathbb{R}^d$. As we trace the path of the movement of a material point through time we identify what is called the *design trajectory*. It is important to realize that every material point may move normal and tangential to the actual surface $\Gamma(t)$. Consequently, at the body surface the tangent of the design trajectory may be inclined relative to the surface normal vector $\mathbf{n}(\xi,t) \in \mathbb{R}^d$.

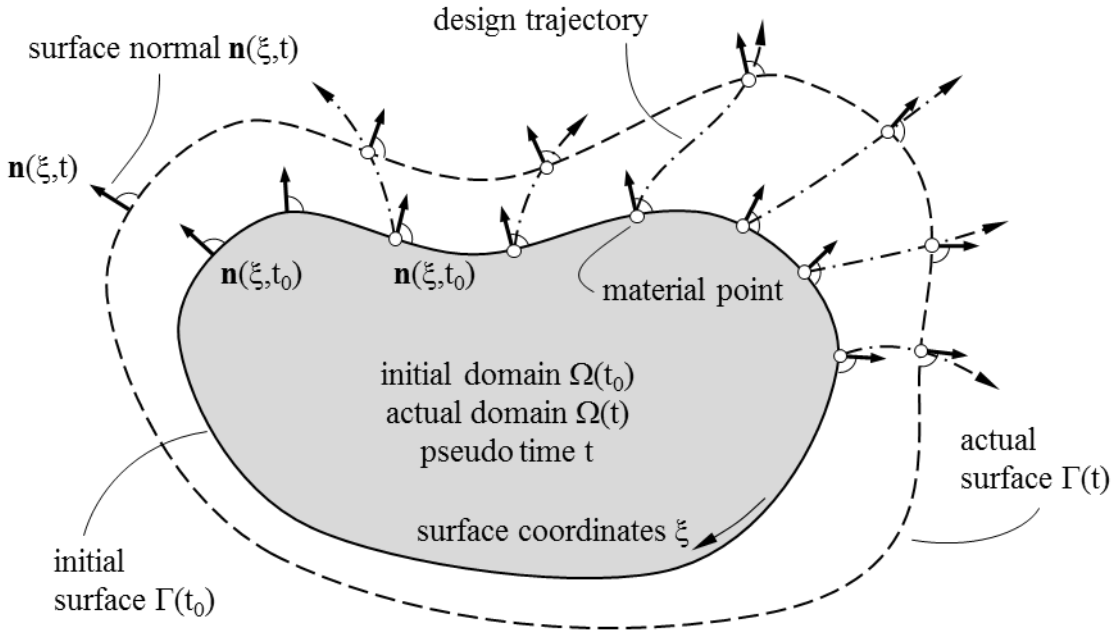


Fig. 8.4: Design evolution of a body.

8.2.2 Design control field

The shape modification is driven by a design control field $s(\xi,t) \in \mathbb{R}$. The field may be understood as curve parameters of the design trajectories. To a trajectory at surface coordinates ξ a "design control" s(ξ,t) is related, Fig. 8.5. Regarding shape optimization, the coordinates at surface material point ξ are defined to be functions of the design control field:

$$\mathbf{x}(\xi,t) = \mathbf{x}(s(\xi,t)) = \begin{Bmatrix} x_1(s(\xi,t)) \\ x_2(s(\xi,t)) \\ x_3(s(\xi,t)) \end{Bmatrix} \tag{8.1}$$

An example of a reasonable choice for s is to take the spatial coordinate $z \in \mathbb{R}$ in normal direction aligned to surface normal vector **n**. That choice considers the fact that relevant modifications of shape are in direction of the surface normal only. As well, the controls may be aligned to other distinct "move directions" which might be chosen due to the specific design problem. Most generally, the design control field is an abstract scalar field which controls the design evolution.

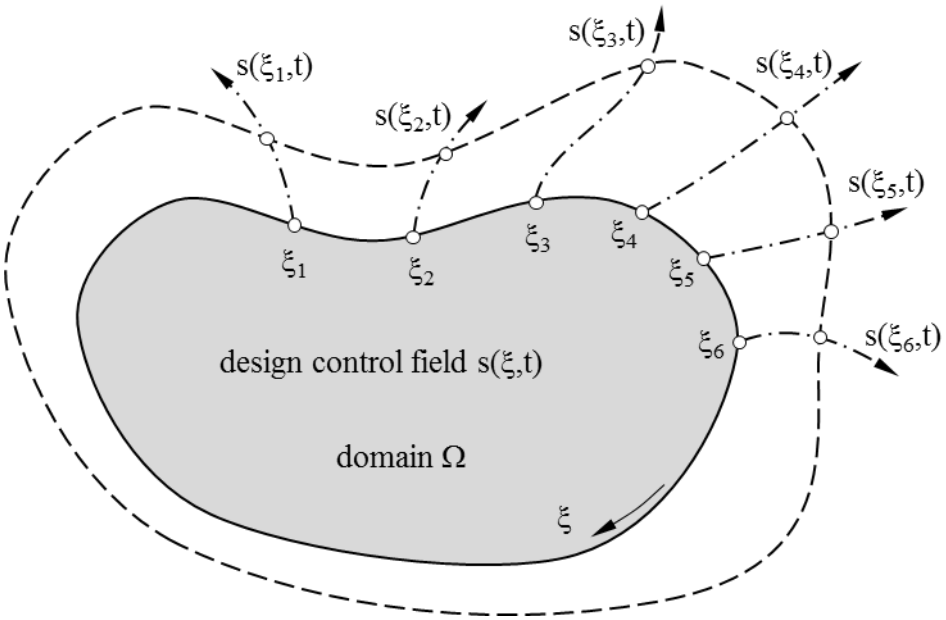


Fig. 8.5: Scalar design control field s.

8.2.3 Design velocity

The change in time of the spatial position **x** of a material point ξ defines what is called the *design velocity* **v**:

$$\mathbf{v}(\xi, t) = \frac{d\mathbf{x}}{dt} = \frac{d\mathbf{x}}{ds}\frac{ds}{dt} \tag{8.2}$$

The term velocity is adapted from the natural understanding of the definition of velocity and is applied to the design change of the body's shape. It is not to be mixed up with a true velocity. The design velocity vector is tangential to the design trajectory. In general, the

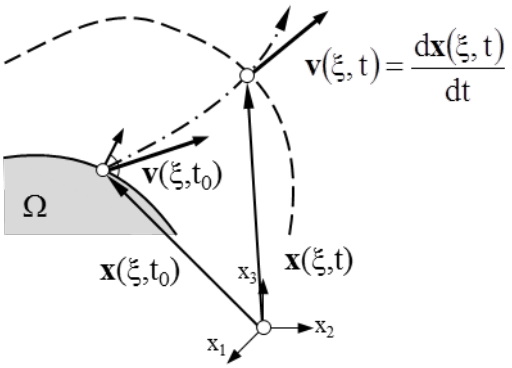


Fig. 8.6: Definition of design velocity **v**.

design velocity is not normal to the actual surface of the body. Material points may move normal as well as tangential to the surface. It is most important to realize that in the case that all material points at the surface are moving tangentially without any normal component they will remain on the actual surface. Consequently, the shape of the body will not be modified and all properties remain unchanged. Regarding shape optimization only the normal components of the design velocity will result in a modification of shape and are relevant for shape optimization. Tangential movements of material points, however, are important to control the quality of the surface description which, finally, will affect the quality of surface discretization as the basis of numerical solution techniques.

8.2.4 Change of volume, volumetric problems

The volume of the body is given by the volume integral:

$$V(t) = \int_{\Omega(t)} d\Omega \tag{8.3}$$

Note, that the domain is varying with time. Applying the theorem of Gauss the time derivative of the volume is determined as the integral over the surface Γ considering the surface design velocity **v** and the surface normal vector **n**:

$$\frac{dV(t)}{dt} = \int_{\Omega(t)} \operatorname{div} \mathbf{v}\, d\Omega = \int_{\Gamma(t)} \mathbf{v} \cdot \mathbf{n}\, d\Gamma \tag{8.4}$$

Again, it becomes clear that only the normal components of surface modifications are relevant for the change of design.

8.2.5 Design objective

Consider an integral design objective Φ with volume density f_Ω, e.g. mass as the volume integral of material density ρ. Considering the design control field s the design objective is determined as:

$$\Phi(t) = \Phi(s(t), t) = \int_{\Omega(t)} f_\Omega(s(t), t)\, d\Omega \tag{8.5}$$

Alternatively, consider situations where the surface Γ is the body itself, e.g. the shape optimization of shells. The surface Γ is bounded by the edge Ψ where the vectors **m** are normal to the edge Ψ but tangential to the surface Γ, Fig. 8.7. Now, the design density is given as a surface density f_Γ and the design Φ objective is:

$$\Phi(t) = \Phi(s(t), t) = \int_{\Gamma(t)} f_\Gamma(s(t), t)\, d\Gamma \tag{8.6}$$

The surface density f_Γ is derived from the volume density f_Ω by pre-integration through the body thickness h in direction of the normal coordinate z by:

$$f_\Gamma = \int_h f_\Omega\, dz \tag{8.7}$$

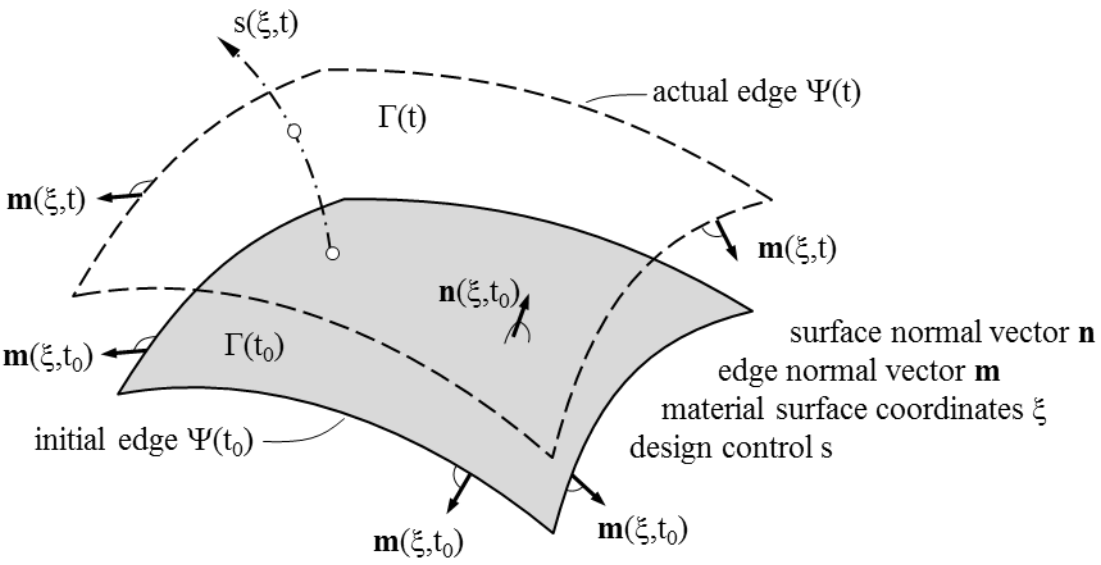


Fig. 8.7: Surface problem.

8.2.6 Normal and tangential coordinates

Surface curvilinear spatial coordinates are identified which are related to the surface normal and tangential vectors **n** and **t** = {$\mathbf{t}_1,\ldots,\mathbf{t}_{d-1}$} in $\mathbb{R}^d$, respectively. In normal direction a "normal coordinate" z is introduced. The normal coordinate is a scalar field z(s(ξ,t)) as a function of the actual state of the design control s(ξ,t) at the material point ξ and pseudo time t, Fig. 8.8. As a special choice of design control s, the surface coordinate field z might be taken.

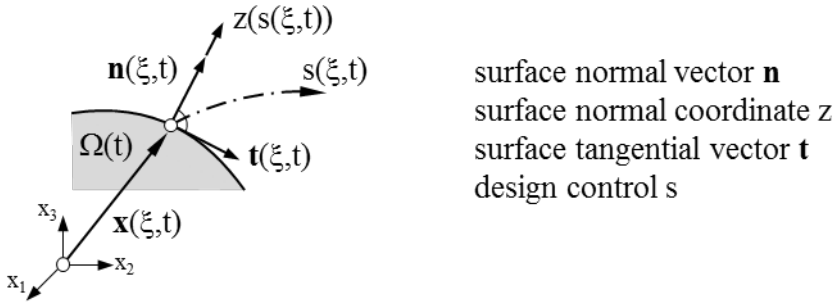


Fig. 8.8: Surface normal coordinates z.

8.2.7 Shape derivative

When taking the time derivative of objective Φ one has to consider the change of the volume as well. The change of volume is related to the surface normal component of the design velocity **v** at the surface and the divergence theorem can be applied:

$$\frac{d\Phi(t)}{dt} = \frac{d}{dt}\left[\int_{\Omega(t)} f_\Omega(t)\,d\Omega\right] = \int_{\Omega(t)}\left(\frac{\partial f_\Omega}{\partial t} + \frac{df_\Omega}{d\mathbf{x}}\mathbf{v} + f_\Omega\,\mathrm{div}\,\mathbf{v}\right)d\Omega = \int_{\Omega(t)}\frac{\partial f_\Omega}{\partial t}\,d\Omega + \int_{\Gamma(t)} f_\Omega\,(\mathbf{v}\cdot\mathbf{n})\,d\Gamma \quad (8.8)$$

In the context of shape optimization the time derivative is called *shape derivative*.

8.2.7.1 Surface driven volumetric problems

Consider the shape optimization of a volumetric problem which is driven only by the modification of the surface. Then the design density f_Ω is not a direct function of time and it holds:

$$\frac{\partial f_\Omega}{\partial t} = 0 \quad \text{and}$$
$$\frac{d\Phi}{dt} = \int_{\Gamma(t)} f_\Omega \left(\mathbf{v}\cdot\mathbf{n}\right) d\Gamma = \int_{\Gamma(t)} f_\Omega \frac{dz}{dt}\, d\Gamma \tag{8.9}$$

with (8.7) in mind we can define $f_\Omega = df_\Gamma/dz$ and arrive at:

$$\frac{d\Phi}{dt} = \int_{\Gamma(t)} \frac{df_\Gamma}{dz}\frac{dz}{dt}\, d\Gamma \tag{8.10}$$

8.2.7.2 Surface problems

In contrast to the volumetric problem before, now, through the design evolution the surface Γ is moving with the body but is growing tangentially to the surface across the edge Ψ. Taking the shape derivative of Φ, now, the flux across the edges is reflected by a tangential divergence term ($f_\Gamma \operatorname{div}_\Gamma \mathbf{v}$) considering derivatives with respect to tangential spatial coordinates:

$$\frac{d\Phi}{dt} = \frac{d}{dt}\left[\int_{\Gamma(t)} f_\Gamma(t)\, d\Gamma\right] = \int_{\Gamma(t)} \left(\frac{\partial f_\Gamma}{\partial t} + \frac{df_\Gamma}{d\mathbf{x}}\mathbf{v} + f_\Gamma \operatorname{div}_\Gamma \mathbf{v}\right) d\Gamma \tag{8.11}$$

Splitting the design evolution into surface normal and tangential directions the latter may be reformulated as:

$$\frac{d\Phi}{dt} = \int_{\Gamma(t)} \left(\frac{\partial f_\Gamma}{\partial t} + \left(\frac{\partial f_\Gamma}{\partial z} - \kappa f_\Gamma\right)\frac{dz}{dt}\right) d\Gamma + \int_{\Psi(t)} f_\Gamma \left(\mathbf{v}\cdot\mathbf{m}\right) d\Psi \tag{8.12}$$

where $\kappa = -\operatorname{div}_\Gamma \mathbf{n}$ (twice the surface mean curvature) reflects the change of area content of a curved surface when it is modified in normal direction. The second term reflects the flux across the edge considering the design velocity in edge normal direction **m**.

Again restricting to surface driven problems, the objective density is time invariant and it is $\partial f_\Gamma/\partial t = 0$. Also, if focusing on evolution in surface normal direction the boundary term vanishes and it remains which is formally equivalent to (8.10):

$$\frac{d\Phi}{dt} = \int_{\Gamma(t)} \left(\frac{\partial f_\Gamma}{\partial z} - \kappa f_\Gamma\right)\frac{dz}{dt}\, d\Gamma = \int_{\Gamma(t)} \frac{df_\Gamma}{dz}\frac{dz}{dt}\, d\Gamma \tag{8.13}$$

8.2.8 Shape and design gradient of surface driven problems

The shape derivative is related to the change of design controls through the chain rule of differentiation:

$$\frac{d\Phi}{dt} = \int_{\Gamma(t)} \frac{df_\Gamma}{ds}\frac{ds}{dt}\,d\Gamma = \int_{\Gamma(t)} \frac{df_\Gamma}{dz}\frac{dz}{ds}\frac{ds}{dt}\,d\Gamma\,; \quad \frac{df_\Gamma}{dz} = \nabla f_\Gamma \cdot \mathbf{n} \tag{8.14}$$

where ∇f_Γ and df_Γ/ds are the *shape* and *design gradient* of objective function Φ, respectively. Basically, the derivatives of the objective surface density f_Γ with respect to the surface normal coordinate z are to be considered. That information is to be calculated locally at every surface material point as basic data of sensitivity analysis. Recall the alternative definitions of the shape derivative for surface and volumetric problems. The negative of the normal component of the shape gradient defines the steepest descent direction of optimal shape update.

8.2.9 Shape derivative considering the state equations

Additional state equations S[u] = 0 as functions of the state variables u represent equality constraints. Typically, they are partial differential equations. Regarding dynamics and/or fluid mechanics, at this point we consider quasi static states. Introducing adjoint variables u^* which represent Lagrange multipliers the objective function is modified again considering the two cases of shape variant volumetric and surface problems and an appropriate inner product $\langle\cdot,\cdot\rangle$:

$$\begin{aligned} \tilde{\Phi} &= \Phi + \left\langle u^*, S[u] \right\rangle = \int_\Omega f_\Omega(s,u)\,d\Omega + \int_\Omega u^* S[u,s]\,d\Omega \\ \text{or } \tilde{\Phi} &= \Phi + \left\langle u^*, S[u] \right\rangle = \int_\Gamma f_\Gamma(s,u)\,d\Gamma + \int_\Gamma u^* S[u,s]\,d\Gamma \end{aligned} \tag{8.15}$$

Straight forward, referring to (8.10) and (8.13), applying the chain rule of differentiation the shape derivative yields:

$$\begin{aligned} \frac{d\tilde{\Phi}}{dt} &= \int_\Gamma \left(f_\Omega + u^* S\right)\frac{dz}{dt}\,d\Gamma + \int_\Omega \left(\frac{\partial f_\Omega}{\partial u} + u^*\frac{\partial S}{\partial u}\right)\frac{du}{dt}\,d\Omega \\ \text{or } \frac{d\tilde{\Phi}}{dt} &= \int_\Gamma \left(\frac{\partial f_\Gamma}{\partial z} - \kappa f_\Gamma + u^*\left(\frac{\partial S}{\partial z} - \kappa S\right)\right)\frac{dz}{dt}\,d\Gamma + \int_\Gamma \left(\frac{\partial f_\Gamma}{\partial u} + u^*\frac{\partial S}{\partial u}\right)\frac{du}{dt}\,d\Gamma \end{aligned} \tag{8.16}$$

The time derivative du/dt of the state variables can be eliminated by solving the adjoint problem for u^*. Considering the test function v it is given in weak form as:

$$\int_\Psi \left(\frac{\partial f_\Psi}{\partial u} + u^*\frac{\partial S}{\partial u}\right) v\,d\Psi = 0 \tag{8.17}$$

Referring to the type of problem, the integration domain Ψ might be either the volume Ω or the surface Γ.

The adjoint problem is linear as it is defined at the actual state of design. For structural problems the operator $\partial S/\partial u$ is identified as the stiffness operator, however, applied to u^* in reverse order. Alternatively, using the adjoint operator $\partial S^x/\partial u$ the weak form might be written as considering the definition of an appropriate inner product $\langle\cdot,\cdot\rangle$:

$$\left\langle \frac{\partial S^x}{\partial u} u^*, v \right\rangle = -\left\langle \frac{\partial f_\Psi}{\partial u}, v \right\rangle \tag{8.18}$$

where the derivatives $\partial f_\Psi/\partial u$ of the objective density with respect to the state variables represent the adjoint source term or, for structural optimization, the adjoint load term.

For further details of the adjoint analysis it is referred to the vast amount of literature in that regard. In particular, it should be noted that it must be carefully distinguished between the continuous, the discretized continuous and the discrete variants of adjoint analysis. As a matter of fact, if not carefully developed they may not converge to identical results. A major question of adjoint analysis deals with the existence of adjoint differential equations which do not exist for every objective function. From a practical point of view, however, since a weak form of the adjoint problem, as sketched above, might always be defined this problem is not a limitation for shape optimal design.

8.2.10 Relation of design controls s and surface normal coordinates z

Most generally, the surface normal coordinate z at material point ξ_0 may be determined as the weighted mean of design controls s in the vicinity $\|\xi\text{-}\xi_0\| \le r$ applying the filter function A:

$$z(\xi_0) = \int_\Gamma A(\xi - \xi_0)\, s(\xi)\, d\Gamma \tag{8.19}$$

Alternatively, the filter might be defined as a function of the spatial surface coordinates. The filter function A is defined to be positive within the filter radius r and zero outside, as well it is normalized, Fig. 8.9:

$$\begin{aligned} &A(\xi - \xi_0):\quad \{A \ge 0\,;\, \|\xi - \xi_0\| \le r\,;\, A = 0\,;\, \text{else}\} \\ &\int_{\|\xi - \xi_0\| \le r} A(\xi - \xi_0)\, d\Gamma = 1 \end{aligned} \tag{8.20}$$

Consequently the derivative of z at position ξ_0 with respect to design control s at position ξ_1 yields:

$$\frac{dz(\xi_0)}{ds(\xi_1)} = A(\xi_1 - \xi_0) \tag{8.21}$$

and the design gradient at position ξ_1 is:

$$\frac{df_\Gamma}{ds}(\xi_1) = \int_\Gamma \frac{df_\Gamma}{dz}\frac{dz}{ds}\bigg|_{\xi_1} d\Gamma = \int_\Gamma \frac{df_\Gamma}{dz} A(\xi_1 - \xi)\, d\Gamma = \int_\Gamma A^\times(\xi - \xi_1)\frac{df_\Gamma}{dz}\, d\Gamma \tag{8.22}$$

Note, that the derivative is filtered by the adjoined filter $A^\times(\xi - \xi_1) = A(\xi_1 - \xi)$. For symmetric filters it holds $A^\times(\xi - \xi_1) = A(\xi - \xi_1)$.

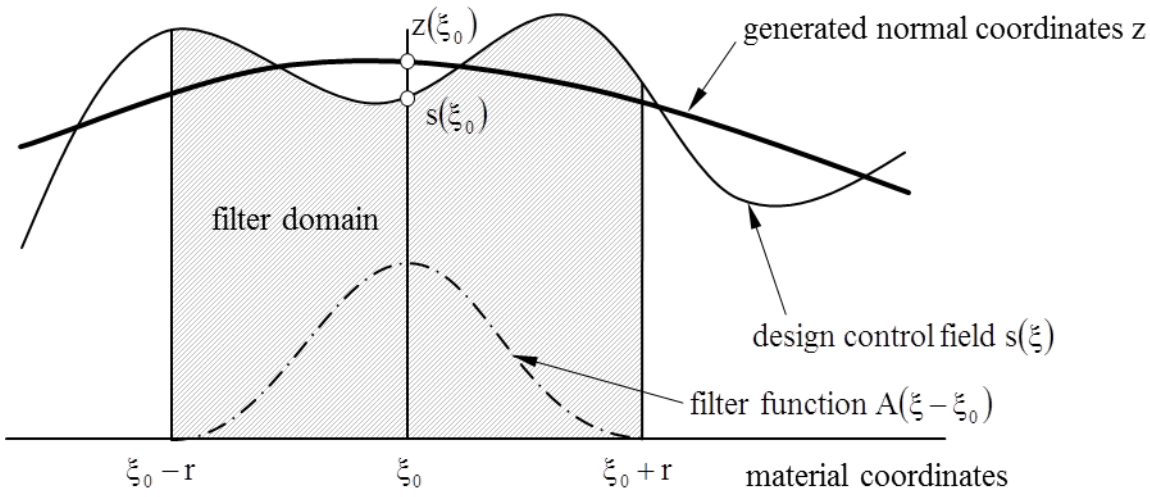


Fig. 8.9: Generating normal coordinates z by filtering of design controls s.

8.2.11 Tangential gradients

Consider at every surface point a set of normal and tangent vectors $\{\mathbf{n}, \mathbf{t}_1, \dots, \mathbf{t}_{d-1}\}$ and related coordinates $\{z, \tau_1, \dots, \tau_{d-1}\}$ which define an orthonormal basis in $\mathbb{R}^d$, Fig. 8.8. Then the shape gradient of the surface objective density f_Γ may be decomposed into normal and tangential partitions of which the tangential one defines the tangential gradient $\nabla_t f_\Gamma$:

$$\nabla f_\Gamma = (\nabla f_\Gamma \cdot \mathbf{n})\mathbf{n} + \sum_{i=1}^{d-1} (\nabla f_\Gamma \cdot \mathbf{t}_i)\mathbf{t}_i = \frac{df_\Gamma}{dz}\mathbf{n} + \nabla_t f_\Gamma \tag{8.23}$$

As stated before the tangential gradient is irrelevant for shape optimization and, typically, it even vanishes. In the view of mesh quality control the tangential gradient might be taken of an auxiliary function h_Γ,. i.e. $\nabla f_\Gamma = \nabla h_\Gamma$, which is designed to control the mesh quality, e.g. as to guarantee equal element size after discretization or to minimize the element distortion. Eventually, refer to chapter 8.3.8 for an example about to choose h.

8.2.12 Update of the control field

The control field is updated in a descent direction d_s, e.g. the negative design gradient df_Γ/ds. Note the adjoint or backward use of the filter A for the shape gradient:

$$d_s = -\frac{df_\Gamma}{dz}\frac{dz}{ds} = -\int_\Gamma A^\times \frac{df_\Gamma}{dz} d\Gamma \tag{8.24}$$

Straight forward, applying the forward filter operation (8.19) the update d_z of the normal coordinate z is determined as

$$d_z = \int_\Gamma A d_s \, d\Gamma = -\int_\Gamma \int_\Gamma A\, A^\times \frac{df_\Gamma}{dz} d\Gamma \, d\Gamma \tag{8.25}$$

which defines a two pass filter operation. Note, that the control field must not be resolved to arrive at this end.

8.2.13 Shape update

The final shape update velocity $\mathbf{v}_d$ is composed from a combination of the normal update d_z and the negative tangential gradient:

$$\mathbf{v}_d = d_z \mathbf{n} \frac{dz}{dt} - \sum_{i=1}^{d-1} \left(\left(\nabla h_\Gamma \cdot \mathbf{t}_i \right) \mathbf{t}_i \frac{d\tau_i}{dt} \right) \tag{8.26}$$

Still, the dependence of the tangential coordinates τ_i on the design control field s has to be defined following pragmatic arguments concerning the mesh quality and implementation issues.

Finally, the shape update in direction of the steepest descent is:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \mathbf{v}_d \, \Delta t \tag{8.27}$$

where Δt might be adapted by a line search. Alternatively, shape optimization is performed in a staggered procedure: (i) shape update in normal direction, (ii) mesh adaptation applying standard mesh smoothers. Of course other optimization techniques including Newton methods might be applied. Nevertheless it appears that for shape optimization steepest descent is converging fast and robust even for the largest problems.

8.2.14 Shape gradient smoothing

Regarding the design update rule (8.27) the smoothness of the resulting shape depends on the smoothness of the shape update $\mathbf{v}_d$ and the shape gradient as its kernel. If the smoothness of the latter is reduced then the smoothness of the resulting shape will also be reduced during the shape evolution, eventually leading to instable procedures and highly oscillating, distorted meshes.

As a remedy, the smoothness of the shape gradient as the essential ingredient of the shape update is to be controlled and adapted to the smoothness of the intended shape. Filtering operations take this role where (8.22) and (8.25) are known as explicit local filtering. The traction method by Azegami [35] is an alternative which makes use of the partial differential equations of elasticity. The method might be understood as a generalization of classical splines which take the bending stiffness of elastic lines to control the shape smoothness. The traction method is a global method affecting the entire surface. Stück and Rung have worked on explicit global Gaussian filters for the shape gradient smoothing [28].

Another, very well-known alternative is the implicit Sobolev smoothing as introduced by Jameson, Pironneau, Mohammadi and many others [23-25]. That procedure is motivated by problems with objectives which depend on the shape **x** and derivatives of **x**.

As an example following Jameson's arguments, consider a one dimensional problem with x the spatial surface coordinate field, z the field of normal coordinates representing the geometry and $z' = dz/dx$ the first order spatial derivatives:

$$\Phi = \int_\Omega f_\Omega(z, z') d\Omega \quad \rightarrow \quad \min \tag{8.28}$$

Following the arguments of the previous chapters the variation of Φ is determined as the surface integral

$$\delta\Phi = \int_{\Gamma}\left(\frac{\partial f_{\Gamma}}{\partial z}\delta z + \frac{\partial f_{\Gamma}}{\partial z'}\delta z'\right)d\Gamma \tag{8.29}$$

After applying partial integration

$$\delta\Phi = \int_{\Gamma}\left(\frac{\partial f_{\Gamma}}{\partial z} - \frac{d}{dx}\frac{\partial f_{\Gamma}}{\partial z'}\right)\delta z\, d\Gamma = \int_{\Gamma}\nabla f_{\Gamma}\,\delta z\, d\Gamma \tag{8.30}$$

and comparing with (8.10) we observe that the shape gradient ∇f_{Γ} is a function of z, z′ and z″ and, therefore, its smoothness might be reduced depending on z″.

As a remedy, a weighted Sobolev inner product is introduced replacing (8.30) by

$$\delta\Phi = \int_{\Gamma}\left(\nabla_m f_{\Gamma}\,\delta z + \varepsilon\left(\nabla_m f_{\Gamma}\right)'\delta z'\right)d\Gamma \tag{8.31}$$

which is applied to the modified shape gradient $\nabla_m f_{\Gamma}$. Again, after partial integration the modified shape gradient is identified as:

$$\nabla_m f_{\Gamma} - \frac{\partial}{\partial x}\varepsilon\frac{\partial}{\partial x}\left(\nabla_m f_{\Gamma}\right) = \nabla f_{\Gamma} \tag{8.32}$$

This defines an implicit filtering scheme to determine the modified shape gradient $\nabla_m f_{\Gamma}$ from the original ∇f_{Γ} by solving another system of equations. Eventually, finite differencing is applied to approximate the second order derivatives with respect to x. Practically, some few steps of an iterative solution scheme are good enough. Referring to Mohammadi, the number ε is chosen such that the filter effect is local and can be determined from the norm of the second order derivatives of the shape update or an appropriate approximation of them. For the shape update the modified, filtered gradient $\nabla_m f_{\Gamma}$ is used following the procedures mentioned above. The technique is easily expanded to multi-dimensional cases and proves to be very successful.

Standard CAD methods act as filters as well by directly controlling the smoothness of the shape and its update. As a matter of fact, their smoothing properties had been major arguments for their introduction [9].

8.2.15 Example for gradient smoothing

The necessity and the effect of gradient smoothing are demonstrated for the iterative solution of the classical Brachistrone example, also refer to [26]. The problem states as:

$$\begin{aligned}\Phi &= \int_{x=1}^{25}\sqrt{\frac{1+z'^2}{z}}\,dx \;\rightarrow\; \min \\ \text{s.t.:}\quad & z_1 = z(x=1) = 0; \quad z(x=25) = 1.2\end{aligned} \tag{8.33}$$

From (8.30) the shape gradient is

$$\nabla f_\Gamma = -\frac{1+z'^2+2z\,z''}{2\left(z\left(1+z'^2\right)\right)^{3/2}} \tag{8.34}$$

Obviously, the shape gradient suffers from degradation of smoothness.

As a possible alternative for a numerical solution, the shape is discretized with quadratic B-splines at 13 grid points with spacing $\Delta x = 2$. The inner 11 points are related to the free vertical coordinates to be determined. Additional points and B-splines on either side are considered and C^1-continously linked to control the boundary tangential conditions, Fig. 8.10.

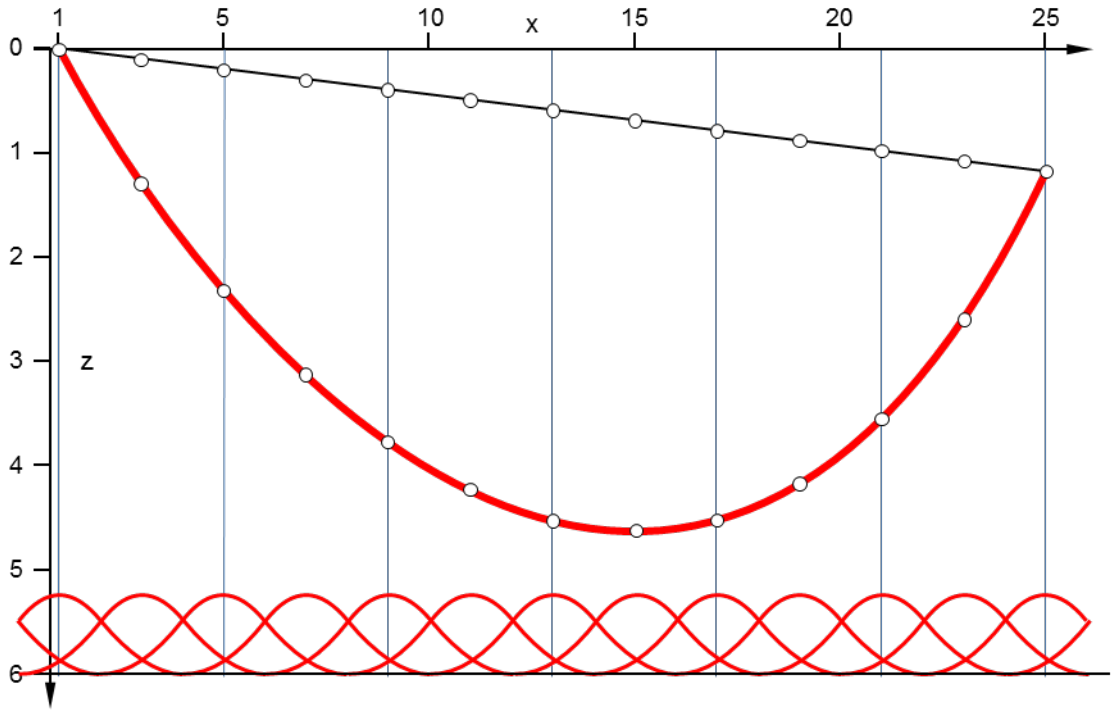


Fig. 8.10: Brachistrone problem, discretization with quadratic B-splines.

The algorithmic constants chosen for Sobolev smoothing and filtering are:

$$\text{Sobolev smoothing}: \quad \lambda = 25; \quad \varepsilon = 1$$

$$\text{explicit linear filter}: \quad r_{\text{filter}} = 2$$

The shape gradient ∇f_Γ and, alternatively, the Sobolev smoothed $\nabla_m f_\Gamma$ from solving (8.32) and the geometry update value d_z (8.25) are repeatedly determined at inner nodes i = {1...11} to create the series of shape update, either by:

$$z_i^{(k+1)} = z_i^{(k)} - \lambda\, \nabla f_\Gamma\left(x_i\right)^{(k)}$$

$$z_i^{(k+1)} = z_i^{(k)} - \lambda\, \nabla_m f_\Gamma\left(x_i\right)^{(k)}$$

$$z_i^{(k+1)} = z_i^{(k)} + \lambda\, d_z\left(x_i\right)^{(k)}$$

For smoothing and filtering the discrete gradient values at the inner nodes are multiplied with the respective filter matrices **F**. For the Sobolev smoothing:

$$\mathbf{F}^{-1} = \begin{bmatrix} 1+2\varepsilon & -\varepsilon & 0 & 0 & \dots & 0 \\ -\varepsilon & 1+2\varepsilon & -\varepsilon & 0 & \dots & 0 \\ 0 & -\varepsilon & 1+2\varepsilon & -\varepsilon & \dots & 0 \\ \vdots & & & & \dots & 0 \\ 0 & \dots & 0 & -\varepsilon & 1+2\varepsilon & -\varepsilon \\ 0 & & \dots & 0 & -\varepsilon & 1+2\varepsilon \end{bmatrix} \tag{8.35}$$

and, for the explicit filtering considering the two pass procedure which represents twice the chain rule, first, to determine the design gradient and, second, to determine the shape update from the design update:

$$\begin{aligned} \hat{F}_{ij} &= \int_x A(x, x_i) B_j(x) dx \; ; \quad i, j = \{1, \dots, 11\} \\ \mathbf{F} &= \hat{\mathbf{F}} \hat{\mathbf{F}}^T \end{aligned} \tag{8.36}$$

Where A is a linear hat filter function with radius $r_{filter} = 2$ and B_j are the shape functions of the quadratic B-splines centered at the inner nodes $x_j = \{3, \dots, 23\}$. It appears, that both filter matrices have basically the same properties. Refer to Fig. 8.11 which shows the distribution of the center row entries of the related filter matrices.

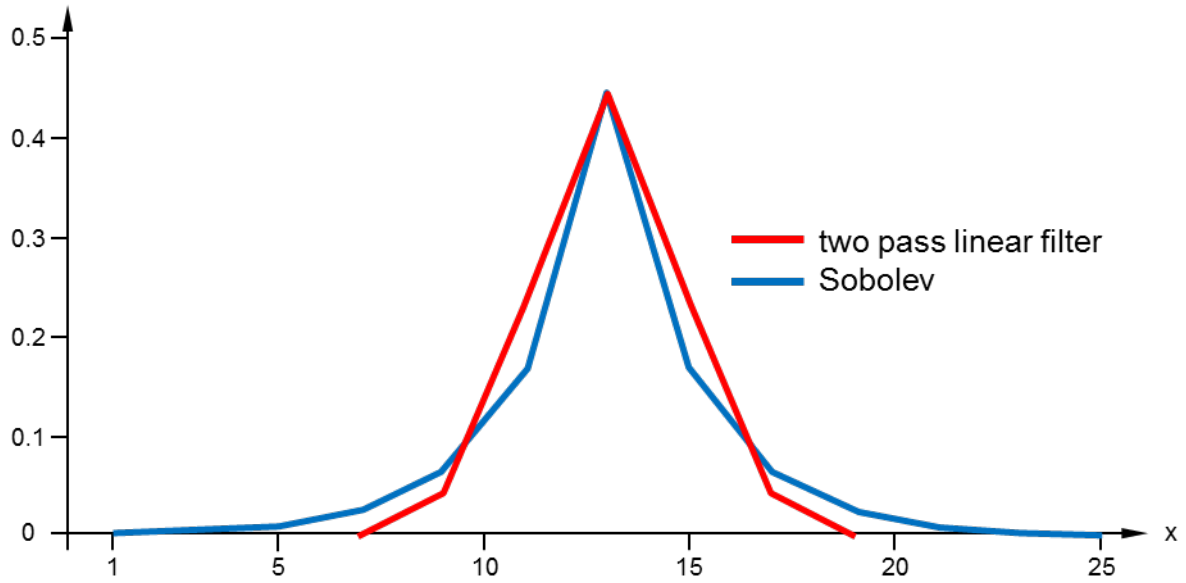


Fig. 8.11: Brachistrone problem, entries of filter matrix center row.

Finally, note the stabilizing effect of the smoothing and filtering on the iteration history. Without that operation the iteration is diverging, Fig. 8.12. The filtering technique serves two tasks: (i) generating the geometry from the control field and (ii) as a smoother to stabilize the iteration.

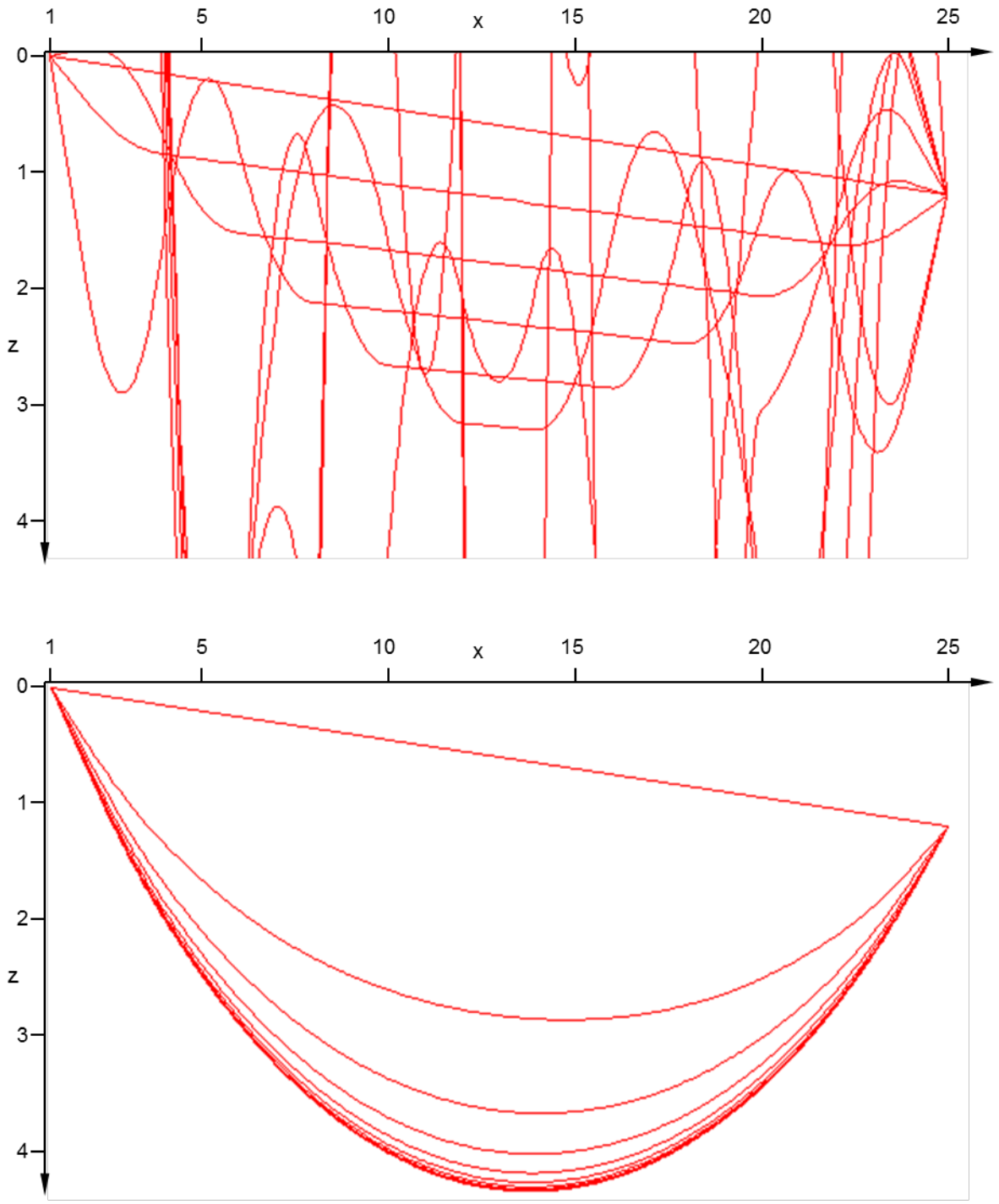


Fig. 8.12: Brachistrone problem; iterative evolution of shape;
divergence without smoothing / filtering (top, the ten first steps);
convergence with Sobolev smoothing or explicit filtering (bottom).

8.3 Discretization of the geometry

8.3.1 Discretization of the filtering operation, design vertices and morphing

Consider shape functions N_j and discrete design parameters s_j to discretize the design control field:

$$s(\xi, t) \approx s_h = N(\xi)_j \, s_j(t) \tag{8.37}$$

Then, substituting into (8.19) yields:

$$z_h(\xi_0) = \int_\Gamma A(\xi - \xi_0) N_j(\xi) s_j \, d\Gamma \tag{8.38}$$

The discrete control parameters s_j are taken out of the integral. The remaining integral of filter function A and shape functions N_j result in so called *morphing functions* B_j which relate the discrete control parameter s_j to the generated normal coordinates, i.e. the shape:

$$z_h(\xi_0) = \int_\Gamma A(\xi - \xi_0) N_j(\xi) \, d\Gamma \, s_j = B_j(\xi_0) s_j \tag{8.39}$$

Chosing A and N_j to be polynomial functions of degree p and q, respectively, then B_j is of degree (p+q+1). That is indicated by super indices:

$$B_j^{(p+q+1)} = \int_\Gamma A^{(p)}(\xi - \xi_0) N_j^{(q)}(\xi) \, d\Gamma \tag{8.40}$$

Obviously, if $A^{(1)}$ and $N^{(1)}$ are chosen as linear functions the morphing functions $B^{(3)}$ are of degree 3 and piecewise cubic. Indeed, (8.40) appears to be an alternative approach to the well-known cubic B-splines, and, more general, to B-splines of degree (p+q+1). The morphing functions B_j are equivalent to the well-known shape functions of standard CAD methods. The design control field $s_h(\xi)$ is identified to be equivalent to the control polygon of standard splines.

8.3.2 Implicit splines

More generally, a hierarchy of splines is defined by recursively applying a constant Filter $A^{(0)}$ to the previous morphing function starting from a constant morphing function $B^{(0)}$ generating morphing functions $B^{(1)}$, $B^{(2)}$, $B^{(3)}$ etc., refer to (8.41) and Fig. 8.13. This definition is equivalent to the original definition of subdivision splines by Catmull and Clark [36, 37]. Note, that morphing functions $B^{(d)}$ are piecewise C_{d-1}-continuous. They share this property with standard spline functions.

Consider the special case of a 1D piecewise linear design control field s of degree q = 1 as well as a linear “hat” function for $A^{(1)}$ of degree p = 1. Then the filter operation generates a piecewise cubic morphing function $B^{(3)}$ for z which can be identified as a standard cubic B-spline, Fig. 8.13 bottom.

$$
\begin{aligned}
B^{(1)} &= \int_{\Gamma} A^{(0)}\, B^{(0)}(\xi)\, d\Gamma \\
B^{(2)} &= \int_{\Gamma} A^{(0)}\, B^{(1)}(\xi)\, d\Gamma \\
B^{(3)} &= \int_{\Gamma} A^{(0)}\, B^{(2)}(\xi)\, d\Gamma \\
B^{(p+q+1)} &= \int_{\Gamma} A^{(p)}\, B^{(q)}(\xi)\, d\Gamma
\end{aligned}
\tag{8.41}
$$

8.3.3 Implicit open splines

Filters may be modified near to the edges to exactly control edge values of the geometry at the edge which is equivalent to natural open splines with zero curvature at the edge. As an example for linear filters refer to Fig. 8.14. At the free edge the reflected filter A_r is introduced. As during the filter process the regular filter intersects the free edge, the filtering is modified at the left or right free edge, respectively:

$$
\begin{aligned}
z(\xi_0) &= \int_{\Gamma} A(\xi - \xi_0)\, s(\xi) + A_r(\xi - \xi_0)\,(s(\xi) - 2s(\xi_L))\, d\Gamma; \quad \xi_L \le \xi \le \xi_L + r \\
z(\xi_0) &= \int_{\Gamma} A(\xi - \xi_0)\, s(\xi) + A_r(\xi - \xi_0)\,(s(\xi) - 2s(\xi_R))\, d\Gamma; \quad \xi_R - r \le \xi \le \xi_R
\end{aligned}
\tag{8.42}
$$

Note, the special treatment of the control value $s(\xi_L)$ or $s(\xi_R)$ at the left or right free edge which is equivalent to the reduction of continuity requirements by multiple knots of the standard explicit splines. An example is given in Fig. 8.15 for a piecewise linear control function and an open cubic B-spline generated by applying the presented procedure. Other modified filters may be created to generalize the technique for other boundary conditions and surface splines.

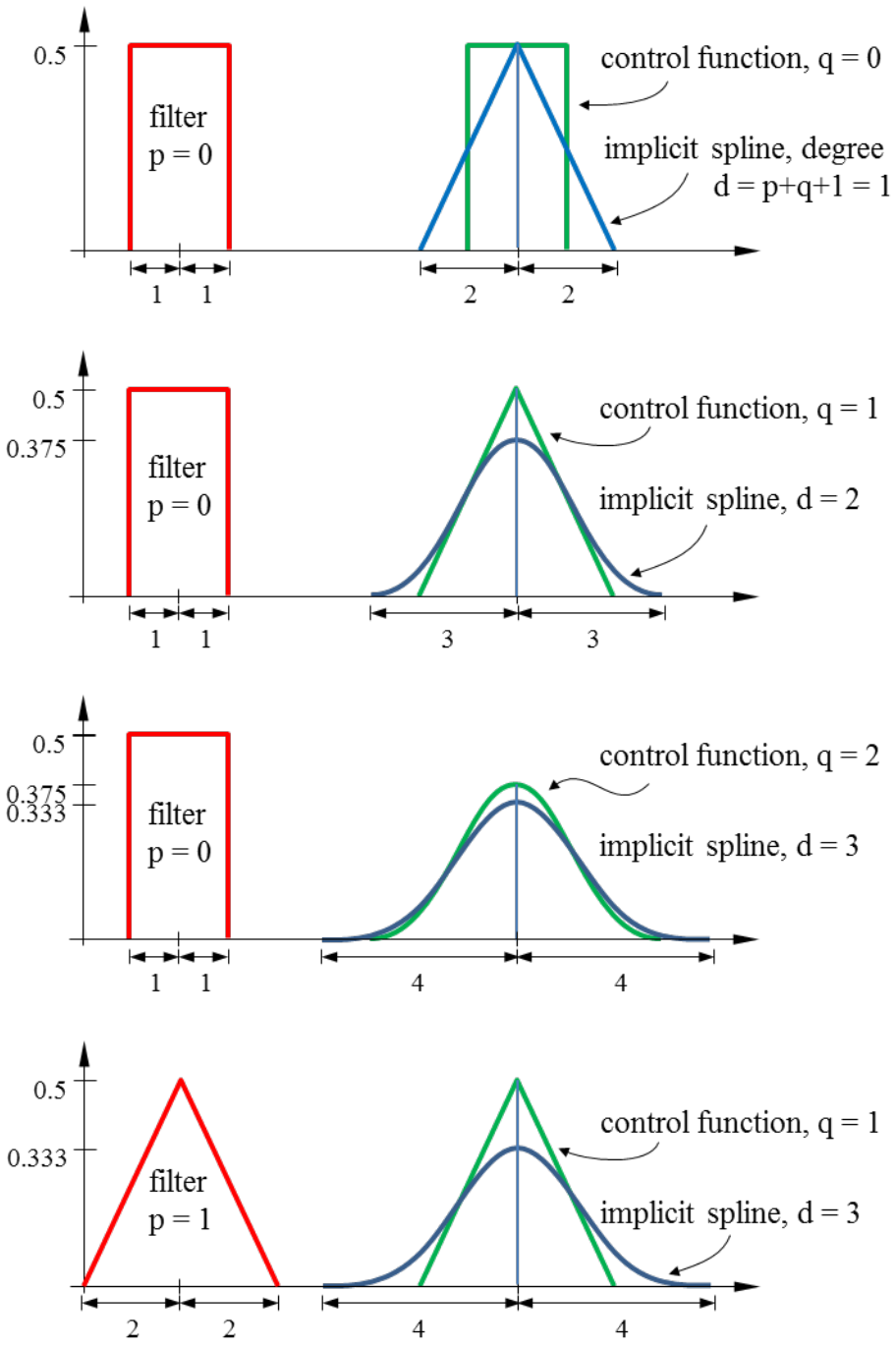


Fig. 8.13: Hierarchy of implicitly generated B-splines.

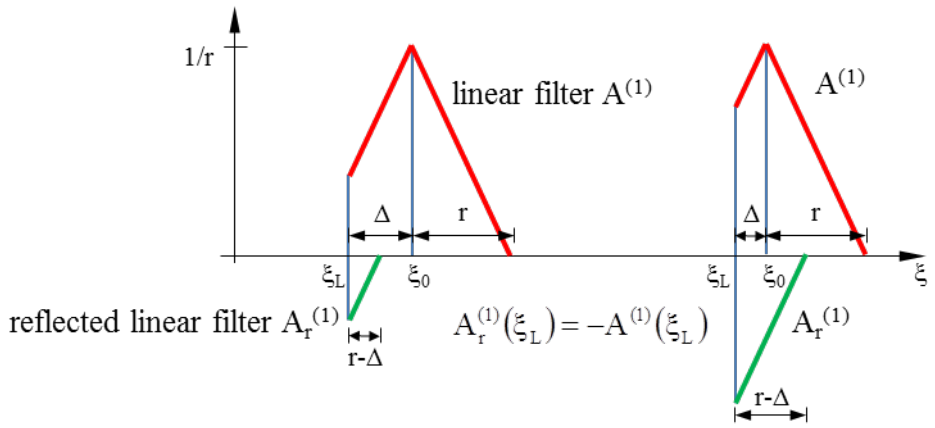


Fig. 8.14: Reflected filter function $A_r^{(1)}$ at left free "edge" located at ξ_L

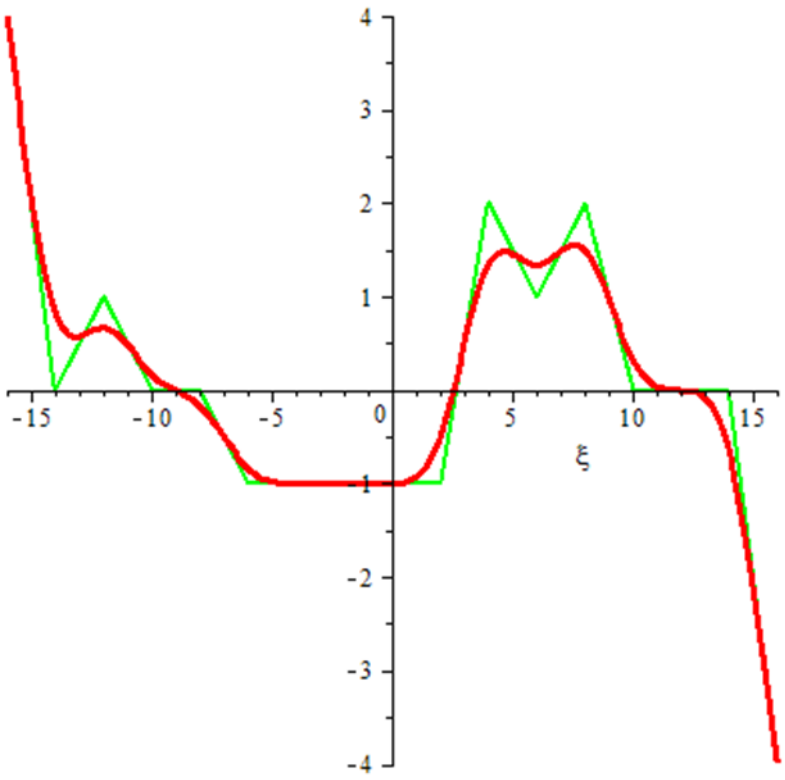


Fig. 8.15: Applying reflected filter functions $A_r^{(1)}$ to exactly interpolate a linear control field at the left and right free "edges". The interpolated geometry is a cubic B-spline of degree 3 and C^2 continuity.

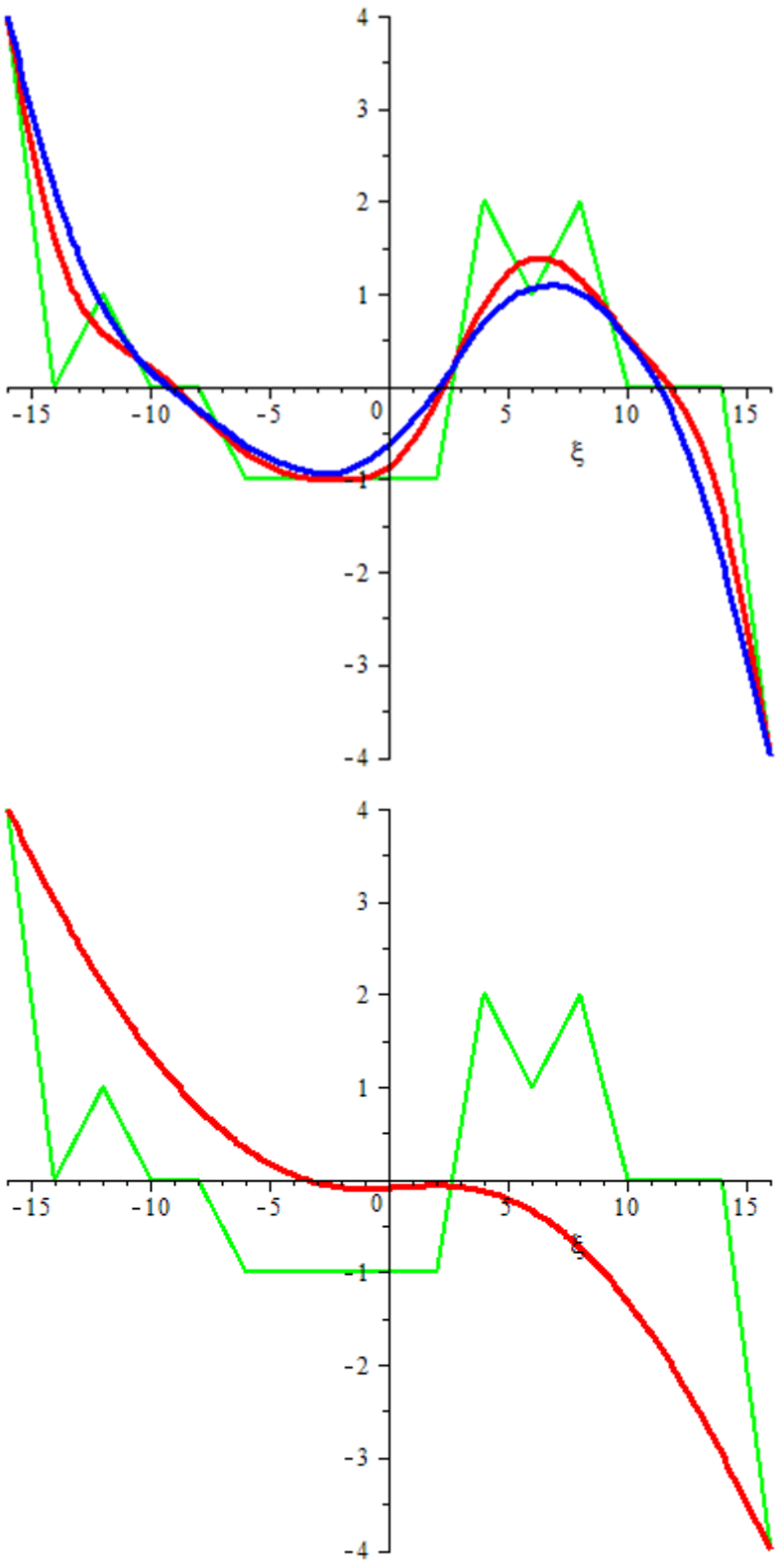


Fig.8.15 continued: Variations of the procedure with increasing filter radii r = 4 (red), 6 (blue), r = 16 (bottom figure)

8.3.4 Alternative discretization of the shape derivative

Let F be the approximation of objective function Φ as a consequence of replacing the design control function s by its approximation s_h. Referring to the shape derivative (8.14) and considering (8.37) the variation $\delta\Phi$ of the objective function is approximated by δF:

$$\delta\Phi(s) \approx \delta F(s_h) = \frac{dF}{ds_i}\delta s_i = \int_\Gamma \frac{df_\Gamma}{ds} N_i \, \delta s_i \, d\Gamma \tag{8.43}$$

Introducing the filter function A as the moderator between geometry z and design control field s and applying (8.21), now, double integrals have to be considered. Several alternative equivalent notations of the discrete design derivative dF/ds_i can be developed which yield the identical result:

$$\frac{dF}{ds_i} = \int_\Gamma \frac{df_\Gamma}{ds}\frac{ds}{ds_i} d\Gamma = \int_\Gamma \left(\int_\Gamma \frac{df_\Gamma}{dz} A \, d\Gamma \right) N_i \, d\Gamma \tag{8.44}$$

$$\frac{dF}{ds_i} = \int_\Gamma \frac{df_\Gamma}{dz}\frac{dz}{ds_i} d\Gamma = \int_\Gamma \frac{df_\Gamma}{dz} \left(\int_\Gamma A \, N_i \, d\Gamma \right) d\Gamma \tag{8.45}$$

$$\frac{dF}{ds_i} = \int_\Gamma \frac{Df_\Gamma}{Ds}\frac{Ds}{Ds_i} d\Gamma = \int_\Gamma \left(\int_\Gamma \frac{df_\Gamma}{dz} N \, d\Gamma \right) A_i \, d\Gamma \tag{8.46}$$

The last variant has been developed from exchanging the *general shape function* N and filter function A. Shape functions N_i are similar to filter functions as they are centered at a defined material position ξ_i on a compact support. The following definitions hold:

$$\begin{aligned} &A = A(\xi - \xi_c); \quad N_i = N(\xi - \xi_i); \quad \xi_c \ldots \text{mat. coor. of varying center} \\ &N = N(\xi - \xi_c); \quad A_i = A(\xi_i - \xi); \quad \xi_i \ldots \text{mat. coor. of control node i} \end{aligned} \tag{8.47}$$

The specific filter A_i is referring the specific position ξ_i with assigned design parameter s_i to the varying center position ξ. The general shape function $N(\xi-\xi_c)$ is centered at any varying position ξ_c and is used as filter to define the *equivalent design gradient* Df_Γ/Ds:

$$\frac{Df_\Gamma}{Ds}(\xi_1) = \int_\Gamma \frac{df_\Gamma}{dz} N(\xi - \xi_1) d\Gamma \tag{8.48}$$

It remains to determine the discretization of the shape gradient df_Γ/dz. Most generally, another discretization of the geometry z might be introduced with shape functions R_j and discrete coordinates z_j, e.g. by applying standard finite element techniques. Together with the nodal values df_Γ/dz_i of the discretized shape gradient it yields:

$$z(\xi) \approx z_h(\xi) = \sum_i R_i(\xi) z_i \quad \text{and} \quad \left(\frac{df_\Gamma}{dz}\right)_h \approx \sum_i R_i(\xi) \frac{df_\Gamma}{dz_i} \tag{8.49}$$

Eventually, the same grids and shape functions N_i, R_j might be used for the discretization of geometry z and design control field s. Also, refer to chapters 8.5.5.3 and 8.5.7.

8.3.5 Vertex morphing and node based shape optimization

Vertex morphing refers to (8.44) as the preferred approach. It is developed from the indirect control of shape (8.19) by implicitly defined splines and the use of (8.39). Then, (8.22) is used to determine the discrete approximation of the design gradient df_Γ/ds which is the basis for the design update. Indeed, filtering the design gradient is equivalent to the implicit treatment of splines as shape controlling functions. Spline geometries can be controlled without applying CAD methods [20, 38].

As the kernel of vertex morphing the components of the discrete design gradient are approximated by:

$$\frac{df_\Gamma}{ds} \approx \sum_i N_i \frac{df_\Gamma}{ds_i}; \quad \text{and} \quad \frac{df_\Gamma}{ds_i} = \int_\Gamma \frac{df_\Gamma}{dz} A_i \, d\Gamma = \int_\Gamma \frac{df_\Gamma}{dz} A(\xi_i - \xi) d\Gamma \tag{8.50}$$

Note the adjoint use of the filter when comparing with the filter procedure (8.19) of the geometry (refer to chap. 8.2.2 for the definition of the adjoint filter A^x). If the filter A is symmetrical and regular grids are used it holds $A = A^x$ and:

$$\frac{df_\Gamma}{ds_i} = \int_\Gamma A(\xi_i - \xi)\frac{df_\Gamma}{dz} d\Gamma = \int_\Gamma A^\times(\xi - \xi_i)\frac{df_\Gamma}{dz} d\Gamma = \int_\Gamma A(\xi - \xi_i)\frac{df_\Gamma}{dz} d\Gamma \tag{8.51}$$

Pragmatically, the latter reflects the practical approach where the same, symmetrical filters are applied for the filtering of geometry as well as the gradient even on non-regular grids. Obviously, variations of the filtering procedure create variations of the discrete shape gradients and derivatives. For standard filters the negative shape gradient proofs to be a descent directions and a valid choice for optimization.

With vertex morphing the discrete design controls s_j are identified to be the design handles or the *vertices* of the design model. The setup of an optimization model is most efficient if design control field and analysis model share the same discretization grid and shape functions N or R which simplifies the implementation as well. Only the choice of a filter has to be added to the analysis model to create the design model from the analysis model. Typically, if a predefined type of filter is used (e.g. a piecewise linear hat filter) only the filter radius has to be defined additionally. Eventually, the filter may vary through the design domain. Even the largest optimization problems are easily created and controlled. For obvious reasons the procedure is also called *node based shape optimization*, Fig. (8.16).

It is important to note that the variability of the resulting shape is characterized by the density of the design grid used to discretize the design control field s. The filter helps to control the continuity properties of the resulting shape. The filter does not affect the global and local solutions and leaves them unchanged. Typically, of course, in the case of non-convex problems different local solutions are found for different filters as they modify the descent directions. The design space and its variety of alternative local optima are easily explored with repeated optimization and varied filters.

8.3.6 Transition to CAD based shape optimization

Referring to (8.39) the transition towards CAD based shape control is obvious:

$$z_h(\xi_0) = \int_\Gamma A(\xi - \xi_0) N_j(\xi)\, d\Gamma\, s_j = B_j(\xi_0) s_j \tag{8.39}$$

Indeed, the morphing functions B_j happen to be identical to standard CAD spline functions if the filter function A and shape functions N_j are properly selected. Moreover, the geometry z may be evaluated at certain grid points ξ_i defining the discrete coordinates z_i at the nodes of a simulation grid together with the elements of the mapping matrix B_{ij} and the design parameters s_j:

$$z_i = \int_\Gamma A(\xi - \xi_i) N_j(\xi)\, d\Gamma\, s_j = B_j(\xi_i) s_j = B_{ij} s_j \tag{8.52}$$

Straight forward, the discrete design derivative is determined from (8.45) wherein the morphing functions are identified:

$$\frac{dF}{ds_i} = \int_\Gamma \frac{df_\Gamma}{dz}\frac{dz}{ds_i} d\Gamma = \int_\Gamma \frac{df_\Gamma}{dz}\left(\int_\Gamma A\, N_i\, d\Gamma\right) d\Gamma = \int_\Gamma \frac{df_\Gamma}{dz} B_i\, d\Gamma \tag{8.53}$$

Note, the difference between CAD-based and node-based optimization is the treatment of the spline approximation, either explicitly or implicitly, respectively. Also, note, that the direct application of CAD morphing functions generate the discrete design derivative which must further be treated to give the discrete design gradient as the basis of the design update (refer to chapter 8.5.7).

The CAD-based procedure allows to freely choosing individual dimensions n_s and n_z for the discretization of s and z, respectively. Standard CAD approaches are characterized by a small number n_s of design parameters and **B** to be a rectangular matrix with n_s columns to prolong from the small design space to the large geometry space, Fig. (8.16).

The property of the filter function A is twofold: It is a smoother from the numerical point of view and equivalent to CAD shape functions from the shape control point of view. Always, the filter is essential to stabilize the numerical procedure. Vice versa, standard CAD functions also act as smoothers to stabilize the numerical optimization procedure which was a major argument for their introduction. Again, with the different choice of numbers n_s of discrete design parameters the problem and its optimal solutions are modified, whereas different choices of the filter affect the iteration history and the convergence to one or the other local optimum but not the problem and its global and local optimal solutions.

As a major advantage, the CAD-based approach makes use of all CAD related features as a standard design approach. This is the reason why they are well accepted for industrial applications [39].

The disadvantages of CAD-based shape optimization are a consequence of the, typically, small dimensions of the design space together with the non-uniqueness of CAD models. Eventually, they allow to generating the same initial geometry with different models and parameters. Consequently, they will diverge to different optima with the optimization procedure. One has to

carefully think about the choice of specific models and parameters as they tremendously decide about the final result of optimization. Typically, if large design modifications shall be allowed, several repeated optimization runs together with the re-parameterization of the CAD-model might be necessary to arrive at an accepted result. That is the reason why CAD-based shape optimization has been accepted so far for little geometrical adjustments of given models only. CAD-based methods are not the right choice for intensive design space exploration and to find completely new and innovative solutions of which the proper parameterization cannot be known in advance.

8.3.7 Transition to morphing boxes

The transformation (8.52) formally applies as well to the shape procedure using morphing boxes with handles s_j. Then, the body characterized by its analysis grid with nodal coordinates z_i is completely embedded into the morphing boxes, Fig. (8.16). From the theoretical point of view and the above discussion there is no methodological difference compared with the CAD based approach. The advantage of using morphing boxes is that the geometry of the morphing boxes is arbitrary simple and easy to be set up. Even the most complex bodies are easily embedded which is the striking argument for everyday application. There are three major challenges with that technique: (i) in general, the exact position of morphing boxes relative to the embedded object is difficult to be defined although it should be traced for a unique problem documentation, (ii) in general the body does not share common boundaries with the morphing boxes which complicates the shape control of edges and faces, and (iii) definition and control of continuity conditions of individual control fields across the common boundaries of adjacent morphing boxes to ensure at least first order geometrical continuity during optimization. Every CAD patch might be used as morphing box. The morphing box technique is well accepted for industrial use and problems with a comparatively small number of design variables [33].

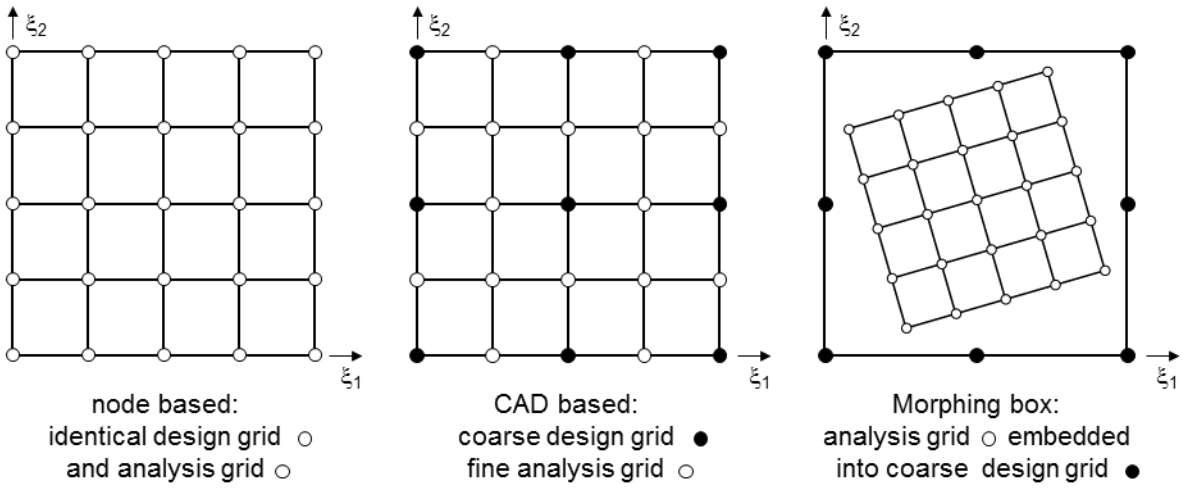


Fig. 8.16: Design and analysis grids.

8.3.8 Form finding of membrane structures as a special field of node-based shape optimization

Membrane structures are made from material which can transfer tensile forces only. That includes 2D textiles (isotrop, anistrop, woven) and foils as well as 1D cables which are composed to cable nets. Those structures have to be pre-stressed. The form finding deals with determining the equilibrium shape of the given pre-stress field and given boundary conditions. Those may be fixed edge positions or edge forces which are introduced by attached pre-stressed edge cables.

In the case of an isotropic and constant pre-stress state it appears that the problem might equivalently be stated as finding the shape for the minimum area content and given boundaries. Referring to (8.12) we deal with the specific case $f_\Gamma = 1$ and the stationary condition:

$$\delta\Phi = \int_\Gamma -\kappa\,\delta z\,\mathrm{d}\Gamma + \int_\Psi \left(\mathbf{m}\cdot\delta\mathbf{x}\right)\mathrm{d}\Psi = 0 \tag{8.54}$$

After partial integration and considering that tangential variations vanish anyway the domain term can be written in terms of the 3D spatial surface geometry $\mathbf{x}$

$$\delta\Phi = \int_\Gamma \mathrm{div}_\Gamma \delta z\,\mathrm{d}\Gamma = \int_\Gamma \mathbf{I} : \frac{\partial\left(\delta\mathbf{x}\right)}{\partial\mathbf{x}}\,\mathrm{d}\Gamma = 0 \tag{8.55}$$

where, $\mathbf{I}$ is the identity tensor.

The expression is generalized by considering a weighted metric and replacing the identity tensor by the Cauchy stress tensor $\boldsymbol{\sigma}$. We arrive at

$$\delta\Phi = h\int_\Gamma \boldsymbol{\sigma} : \frac{\partial\left(\delta\mathbf{x}\right)}{\partial\mathbf{x}}\,\mathrm{d}\Gamma = 0 \tag{8.56}$$

which, obviously, represents the virtual work of the surface pre-stress field $\boldsymbol{\sigma}$ in equilibrium where h is the membrane thickness and Γ the surface in the deformed, i.e. current configuration. That is the so-called soap film analogy which states the equivalence of the equilibrium equation with the minimal surface condition in the case of a isotropic, constant stress field, i.e. $\boldsymbol{\sigma}$ = const., and a constant membrane thickness h. The latter is an accepted approximation because the thickness is very small compared to the structure span width. The edge cable pre-stress forces are considered equivalently and added to the overall virtual work equation.

For the simulation of membrane structures, compared to (8.54), the virtual work expression (8.56) is more appropriate as it deals with stresses and forces which are the important values for construction.

Discretizing and solving (8.56) for the full 3D geometry reveals the independence of tangential geometric variations, leading to singular equations. In practice, however, restriction to the normal coordinate z (where we started from) is not an option as the surface tangential deformation of the edge cables have to be considered which makes a tangential surface deformation necessary. Also, practical shapes may be very complex and decomposition in normal and tangential directions may lead to inefficient procedures.

Alternatively, we add an additional term controlling the tangential mesh deformation, as in (8.26). The tangential deformation is determined by comparing the initial undeformed surface Γ_0 with the actual deformed surface Γ by introducing the deformation gradient $\mathbf{F} = d\mathbf{x}/d\mathbf{X}$ where $\mathbf{X}$ is the undeformed geometry. The virtual equation may now be reformulated as

$$\delta\Phi = h\int_{\Gamma} \boldsymbol{\sigma} : \frac{\partial(\delta\mathbf{x})}{\partial\mathbf{x}}\, d\Gamma = h\int_{\Gamma_0} \det\mathbf{F}\left(\boldsymbol{\sigma}\cdot\mathbf{F}^{-T}\right) : \delta\mathbf{F}\, d\Gamma_0 = h\int_{\Gamma_0}\left(\mathbf{F}\cdot\mathbf{S}\right) : \delta\mathbf{F}\, d\Gamma_0 = 0 \qquad (8.57)$$

with $\mathbf{S}$ the second Piola-Kirchhoff stress tensor.

Prescribing $\mathbf{S}$ instead of $\boldsymbol{\sigma}$ adds tangential stiffness and the tangential mesh deformation can uniquely be determined from solving the discretized equations (8.57). Even more, the equations are linear. However, the resulting shape differs from the true solution as long as $\boldsymbol{\sigma}$ differs from $\mathbf{S}$. Therefore, the procedure is repeated after adapting the deformed geometry to be the undeformed geometry of the next step:

$$\begin{aligned}
&\text{initialize}: \quad k := 0; \quad \mathbf{X}^{(k)} := \mathbf{X}^{(0)}, \quad \mathbf{S} = \text{prescribed surface stress}\\
&\text{do}\\
&\quad \text{solve} \quad \delta\Phi = h\int_{\Gamma_0^{(k)}}\left(\mathbf{F}^{(k)}\cdot\mathbf{S}\right) : \delta\mathbf{F}^{(k)}\, d\Gamma_0^{(k)} = 0 \quad \rightarrow \quad \mathbf{x}^{(k)}\\
&\quad \text{if} \quad \left\|\mathbf{x}^{(k)} - \mathbf{X}^{(k)}\right\| < \text{acc} \quad \text{stop} \qquad (8.58)\\
&\quad \mathbf{X}^{(k+1)} := \mathbf{x}^{(k)}; \quad \Gamma_0^{(k+1)} = \Gamma^{(k)}\\
&\quad k := k+1\\
&\text{end do}
\end{aligned}$$

The Cauchy stress $\boldsymbol{\sigma}$ is converging to the intended stress as the deformation converges to zero. The method is called *updated reference strategy* (URS) because of obvious reasons [40]. URS is basically a most simple procedure to solve the minimal surface problem including the control of the tangential mesh deformation. Therefore, URS is also taken as an alternative to derive tangential gradients as in (8.26) for most general shape optimization problems. Then the surface stress $\mathbf{S}$ is an artificial stress, introduced only to control the mesh deformation. In detail, URS can be interpreted as a weighted Laplacian smoother and is applied as general mesh regularization tool, also [41].

As a matter of fact, the 2nd Piola-Kirchhoff stress tensor $\mathbf{S}$ is equivalent to the force density of cables. URS is the generalization of the *force density method* (FDM) by Linkwitz [6]. The FDM was developed on the occasion of designing the Munich Olympic roof. At those days there didn't exist any numerical tool to do the form finding and node-based shape optimization. FDM and URS are most efficient tools, converging fast and robust for even the largest problems.

Surprisingly, worldwide the activities about the development of numerical methods for the form finding of membrane structures took place mainly independent from the work about computational shape optimization. Whereas those form finding methods of the first days of the 1970

years already could deal with thousands of shape parameters. A number, the parallel developed general approaches not even reached two decades later.

8.4 Node-based shape optimization examples

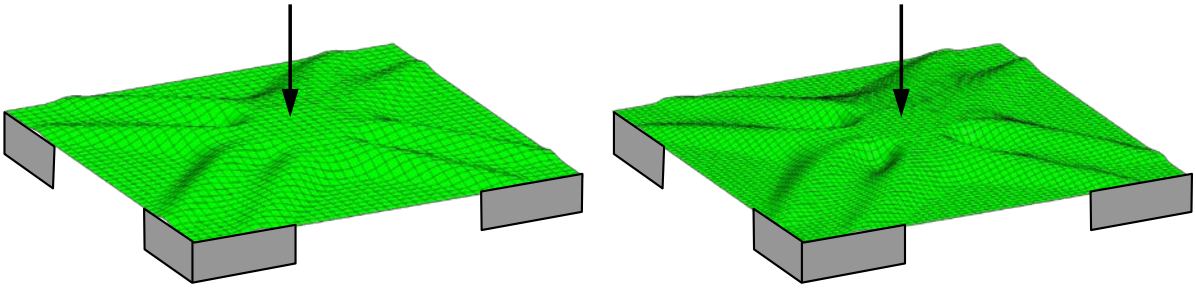
8.4.1 Bead design of plates and shells

This and the following examples demonstrate the success of the vertex morphing method for the application to thin structures and surfaces in 3D [20, 42]. The presented ideas are analogically applied moving nodes in shape normal direction whilst the filters are extended to be rotationally symmetric hat filters. Additionally, the mesh quality has been controlled by applying a weighted, anisotropic Laplace smoother which has been developed by the group applying the mechanical analogy of virtual surface stresses [41, 43]. The optimization as well as the structural analysis has been done applying the own software CARAT++, which is an efficient, object-oriented and parallel implementation, including highly efficient semi-analytical [44] and adjoint sensitivity analysis, robust, gradient based optimization techniques (variants of conjugate gradients), and reliable non-linear finite element models in particular for shell and membrane structures.

All examples are highly non-convex and the optimal results represent local minima. Which local minimum may be found depend extremely on the applied regularization scheme, i.e. the chosen filter methodology, how it is implemented and, most important, how large the filter radius is chosen, not to mention the effects of the type of finite element discretization and the modeling of loads and supports. Therefore, the examples are not reported in all details. Instead, the focus is on potential fields of application and the success of the approach for challenging industrial problems as an alternative to CAD-based shape optimization, in particular for the first stages of product development when a large design space is mandatory.

The first example demonstrates the mesh independence of the method, Fig. 8.17. A quadratic plate is loaded in the center and supported at the corners. The question is to find the optimal topology of stiffening beads. A filter radius is chosen as large as half of the width of support. Additionally, a constraint on the maximum bead depth is given. As shown, the optimal solution is characterized by the filter but it is mesh independent. The choices of filter type and size are additional degrees of design freedom which may be used to explore the design space. Note the smooth final surface although only local radial filters are applied. No post-processing is applied.

Fig. 8.18 shows the result of a joint project together with Adam Opel GmbH. The optimal distribution of beads has been determined to maximize the five lowest eigen-frequencies of a thin metal sheet. The number of iterations appears always to be not more than 40 for every problem size. The number of optimization variables has been appr. 50.000 lateral (shape relevant) and 100.000 tangential (mesh relevant) variables.



40x40 elements 60x60 elements

Fig. 8.17: Optimal bead design for max. stiffness of an initially plane sheet.

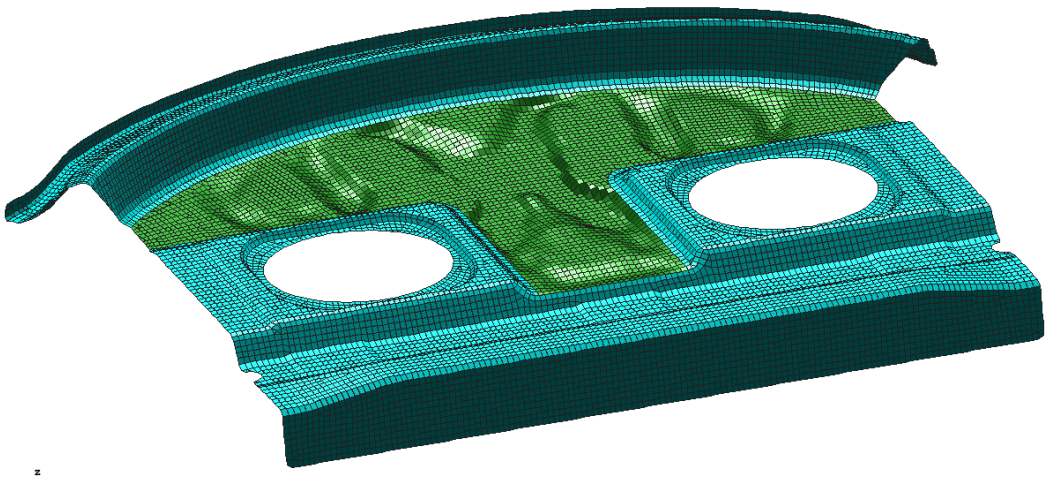


Fig. 8.18: Bead optimization of a thin metal sheet for the automotive industry. Maximization of the five smallest eigen values.

8.4.2 Optimal design of a cylindrical roof

The shape of a cylindrical roof (length = 2.000mm) subjected to self-weight is optimized for stiffness, Figs. 8.19 and 8.20. Mass is indirectly controlled by setting the thickness constant. A very fine mesh is used for analysis and design with approximately 100.000 lateral and 200.000 tangential design degrees of freedom. The well-known shape of the inverted hanging chain is found as global optimum for any choice of filter radius if the Poisson effect is neglected ($\nu = 0$). For $\nu \neq 0$ several local minima are found depending on the choice of filter radius r. Note the speed of convergence as well as the marginal differences of design objective values for the various different solutions, Fig. 8.20.

(a) (b) (c)

Fig. 8.19: Cylindrical roof; initial circular cross section (a), global optimum, $\nu = 0$ (b), local minimum, $\nu \neq 0$, r = 100mm (c)

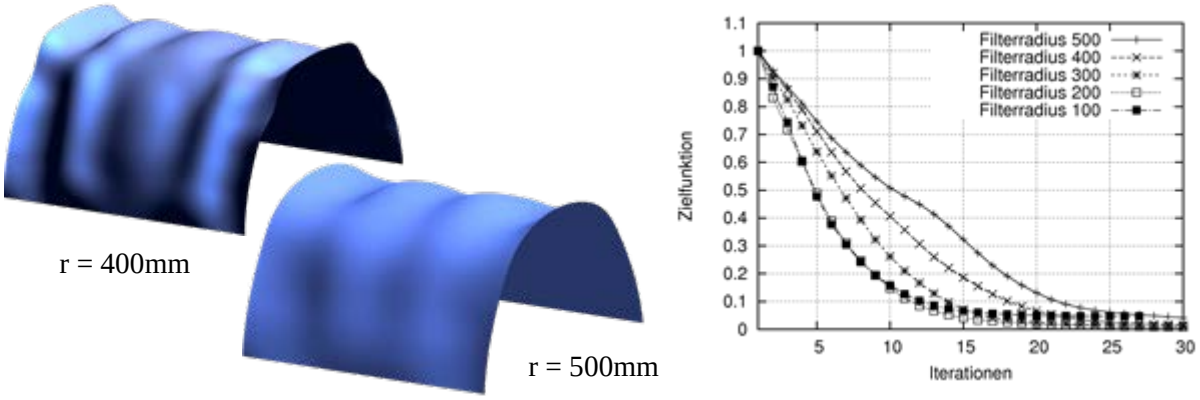


Fig. 8.20: Cylindrical roof; local minima with varied filter radius, $\nu \neq 0$ convergence history

8.4.3 Staggered optimization of a fiber reinforced composite shell

The shape of a bend cantilever is determined, assuming a composite shell with two layers of fiber reinforcement, Figs. 8.21 and 8.22. The filter technique has been applied to regularize the fiber optimization as well. The objective is maximum stiffness; altogether there are about 80,000 shape and fiber angle variables.

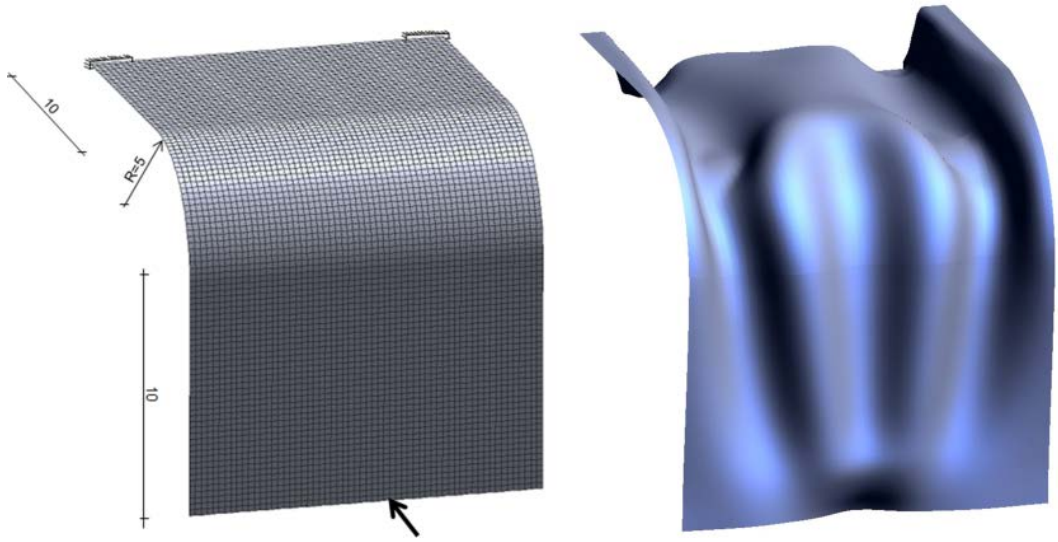


Fig. 8.21: Staggered shape and fiber optimization of a bend cantilever. Initial shape and loading (left), optimal shape (right).

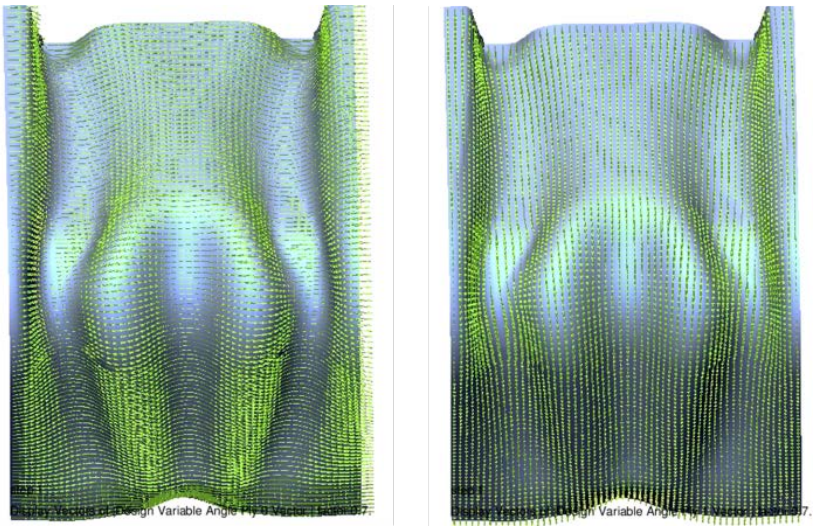


Fig. 8.22: Staggered shape and fiber optimization of a bend cantilever. Optimal fiber orientation, bottom layer (left), top layer (right).

8.4.4 CFD applications

The geometry of the VW-Passat side mirrors has been improved in several different scenarios, where the whole mirror or parts have been allowed to be modified, Fig. 8.23. In all cases the goal was to reduce the drag of the complete car by shape modifications of the mirrors only. Therefore, a complete model of the car had to be simulated in an appropriate numerical wind tunnel using OpenFoam, the adjoint solver of ICON, and CARAT++ for optimization. Regarding optimization, both mirrors had been treated individually, i.e. no symmetry conditions had been applied. That gives 32,000 design parameters for each mirror, i.e. 64,000 in total. Most important was to maintain the mirror feature lines throughout the optimization as to preserve the specific characteristics of a Passat mirror, Fig. 8.24. The total drag of the car could be reduced up to 0.6%. The geometry has been provided by Volkswagen and the adjoint solvers by ICON who had been partners of the EU-project FLOWHEAD. In further applications, which are not displayed here, the complete car body had been optimized which comes together with up to 3.5 Mio shape parameters.

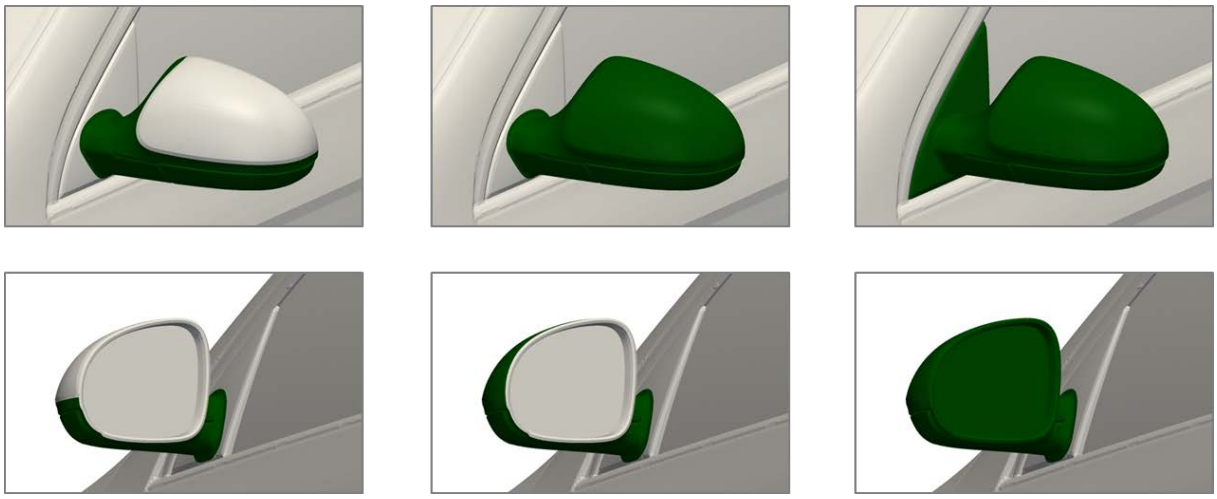


Fig. 8.23: Selected design scenarios for the VW-Passat side mirror. The dark parts are allowed to be modified by shape optimization.

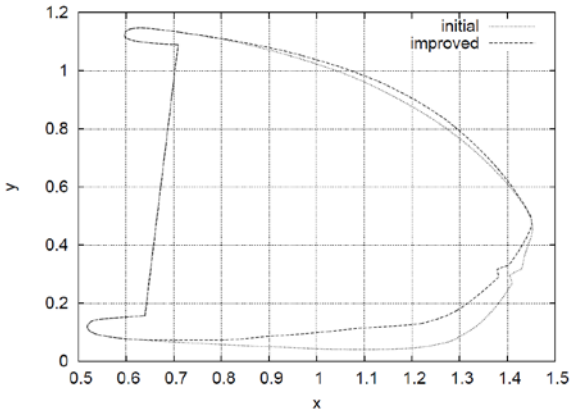


Fig. 8.24: Shape optimization of the side mirrors for drag reduction of the complete car referring to the center column of Fig. 8.23. Longitudinal section of the mirror body. The shape is morphed whilst the displayed feature lines are maintained. The shape of the mirror itself (left straight line) has been constrained to guarantee the usability. Therefore, the optimizer was prevented to simply remove the mirrors to reduce drag.

8.4.5 Form finding of anisotropic pre-stressed membrane structure

This example refers to the optimal shape of a weighted minimal surface which reflects the practical case of textile roofs with local high regions, Fig. 8.25. At the top there is a hole with an inner, fixed edge. At the lower edges there are pre-stressed cables. The ratio of the hole-radius at the top to the structural height is a critical value. If it is tool large than there does not exist an "ideal" minimal surface which is an equilibrium shape of a constant isotropic stress. Therefore, structures like this have to be determined with an anisotropic ratio of pre-stress. From geometrical reasons, however, the ratio cannot be constant everywhere but has carefully to be adapted. Consequently, there is no equilibrium shape if the ratio is forced to be constant but other than one. That is displayed in the figure. Starting from an initial geometry which is used to define the mesh topology and the boundary conditions the URS procedure does not converge with a constant stress ratio but results in a highly distorted mesh as shown in the middle of the figure. The large distorted elements indicate in which direction the pre-stress has to be increased and how the ratio of pre-stress in radial and meridian directions has to be adapted to arrive at a reasonable solution as shown at bottom of the figure with locally adapted stress ratios where 3/1 is the mean value.

There exist solutions only together with additional regularizing decisions about the stress ratios and directions which have to be controlled by the designer, eventually by repeatedly trying different values of the mean stress ratios or the stress directions. As consequence, the optimization method must be efficient and robust dealing with very large numbers of design parameters where there are three coordinates at every node. URS and other versions of the generalized force density method meet those requirements. Parametric CAD-based optimization, including IGA, is no option since the generalized minimal surfaces cannot be approximated sufficiently by coarse grids of spline patches.

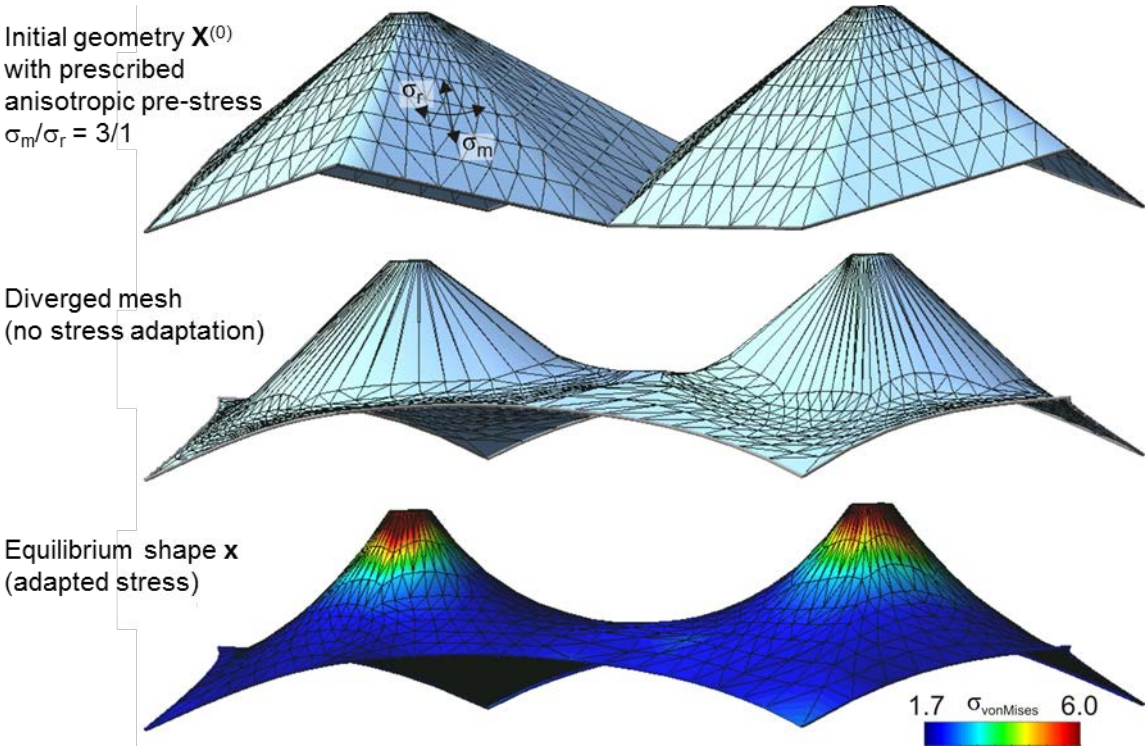


Fig. 8.25: Form finding of an anisotropic pre-stressed textile membrane structure.

8.5 Parametric shape optimization

8.5.1 Interdisciplinary nature of shape optimization

Parametric optimization is understood as a synthesis of basically three models which are (i) the optimization model, (ii) the design model and (iii) the analysis model also known as the three-column concept [45]. Methods from different disciplines are combined as there are (i) design modeling, e.g. by CAD; (ii) CSD and CFD, e.g. by FEM and/or Finite Volumes; (iii) behavior sensitivity analysis, either direct or adjoint; (iv) mathematical optimization; and (v) interactive computer graphics [10].

8.5.2 The design model

The design is defined by a finite number n_s of design control parameters which are collected in the vector **s**.

8.5.2.1 Design patches

Applying CAD concepts the body is composed by design patches and related shape functions. Within each patch the geometry **x** as a function of the material patch coordinates ξ is parametrized by shape functions B_j and the coordinates $\mathbf{r}_j$ of control nodes j. The generation rule is equivalent to (8.39) now extended to all spatial coordinates[2]:

$$\begin{aligned} &\mathbf{x}(\xi) = \sum_j B_j(\xi)\mathbf{r}_j \\ &\mathbf{x}^T = \{x_1,...,x_d\};\ \mathbf{r}_j^T = \{\hat{x}_1,...,\hat{x}_d\}_j;\ \xi = \{\xi_1,...,\xi_d\};\ \text{in } \mathbb{R}^d \end{aligned} \tag{8.59}$$

As shape functions B_j all available patch shape functions may be applied. In particular, that includes B-splines and NURBS [46, 47] as the CAD standard but also interpolating polynomials as they are known from finite elements. The concept of isogeometric analysis (IGA) perfectly fits into this approach. The coordinates $\mathbf{r}_j$ of the control nodes are assigned as design parameters and are collected in the vector **s** of discrete design parameters. With the related shape functions B_j properly arranged in the matrix **B** one can write:

$$\begin{aligned} &\mathbf{x}(\xi) = \mathbf{B}(\xi)\mathbf{s} \\ &\mathbf{B}_{d\times n_s} \text{ in } \mathbb{R}^d;\ \mathbf{s}^T = \{s_1,...s_{n_s}\};\ n_s \text{ ... number od design parameters} \end{aligned} \tag{8.60}$$

8.5.2.2 Geometrical continuity

During shape optimization it might be necessary to maintaining the C_1 geometrical or G_1-continuity between adjacent CAD patches. This must be enforced by constraints between the positions of patch control nodes on either side of the common control node, edge or face. The strong form might be preferable to explicitly resolve the parameter relations. As an example, to support the G^1-continuity of two adjacent cubic splines (e.g. Bézier- or B-spline) it is sufficient to force the common node and the next control nodes on either side to stay on a common line 14, Fig. 3. This condition results in a linear expression in terms of the co- ordinates of three design control nodes where δ is additional free parameter:

[2] In chap. 8.5 parameterized fields are not specially marked, e.g. **x** stands for the parameterized geometry.

$$\mathbf{r}_1^{(2)} = \mathbf{r}_0^{(2)} + \delta\left(\mathbf{r}_n^{(1)} - \mathbf{r}_{n-1}^{(1)}\right) \tag{8.61}$$

This technique may be extended to G^2-continuity nesting the common node and the two next nodes on either side as well as to control the geometrical continuity of surface patches. Formally, the topological relations between the affected control nodes may be formulated by means of superimposed additional patches also called *continuity elements*. An equivalent approach has been developed for IGA where they became known as bending strips. Nevertheless, the technique is limited to the connection of spline curves as, generally, the explicit formulation of geometric continuity for surface patches under all conditions of surface modification is practically impossible. In particularly, that is true for higher continuity requirements and non-conforming meshes of adjacent patches. Alternatively, the weak form of the continuity equations must be used. Then they have to be added as equality constraints to the overall optimization problem. Note, that applying implicit splines by using filters and the vertex morphing the surface smoothness is implicitly controlled without additional efforts.

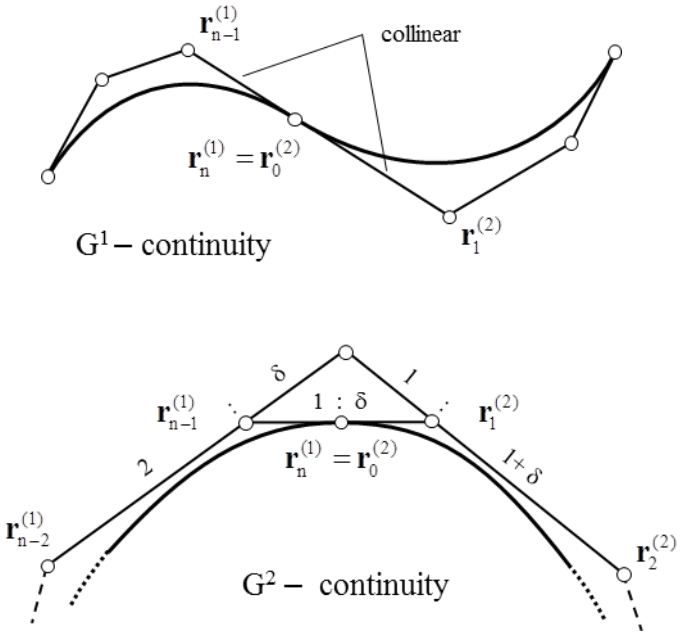


Fig. 8.26: Geometrical G^1- and G^2-continuity of Bézier (top) and B-Spline (bottom) curves.

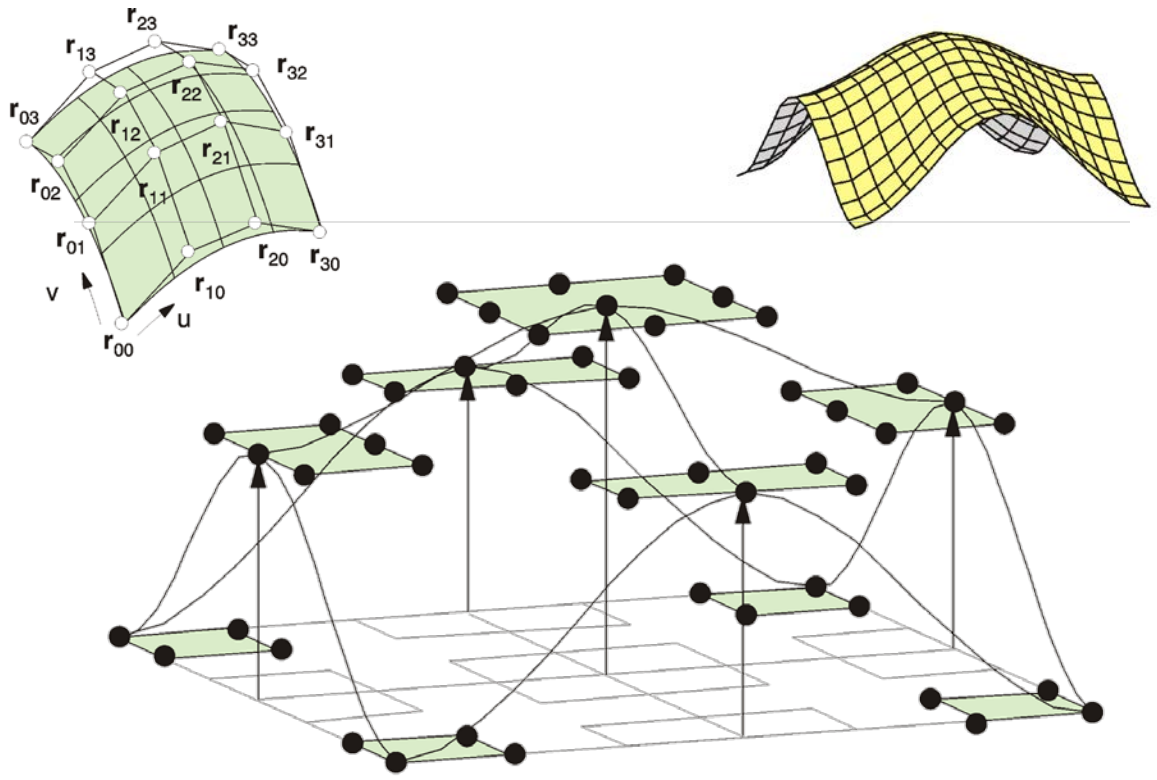


Fig. 8.27: G_1-continuous composition of 4 Bézier-patches; individual patch (top left); generated surface (top right); G^1-continuity elements (green) control the bilinear continuity conditions of adjacent control nodes (bottom)

8.5.3 Parameter linking

Eventually, the design model parameters have to satisfy further additional conditions. Important examples are symmetry which has to be maintained during optimization, prescribed move directions of control nodes or the combination of different coordinate types and systems. Those conditions are driven by the modeling decisions which vary for each specific application and reflect constructive limits, design code rules or manufacturing constraints. Also the esthetic appearance might be treated, e.g. by the control of the surface curvature through the specific treatment of the parameters.

The basic idea is a generalization of (8.60) and states the control of the current coordinate field $\mathbf{x}$ by superimposing to the initial geometry $\mathbf{x}^0$ some add-on $\Delta\mathbf{x}$ which is a function of the parameters collected in vector $\mathbf{s}$:

$$\mathbf{x}_i = \mathbf{x}_i^0 + \Delta\mathbf{x}_i(\mathbf{s}) \tag{8.62}$$

The lower index i refers to node i of the analysis grid with material coordinates ξ_i. Again, it holds that $\mathbf{x}^T = \{x_1,\ldots,x_d\}$ and $\xi^T = \{\xi_1,\ldots,\xi_{d-1}\}$ in $\mathbb{R}^d$ assuming that only coordinates at the body surface Γ are controlled. Interior points might be controlled equivalently, then one more material coordinate has to be considered.

For the analysis node i the add-on might be related to the parameters **s** as:

$$\Delta\mathbf{x}_i(\mathbf{s}) = \mathbf{L}_{xs}\mathbf{s} + \mathbf{H}_{xr}(\mathbf{r}) \tag{8.63}$$

where $\mathbf{L}_{xs}$ is a linear linking matrix and $\mathbf{H}_{xr}$ is a linking function which maps the coordinates of the control node to the coordinates of analysis node i. The coordinates of the control nodes are collected in the one-dimensional vector **r**:

$$\mathbf{r}^T = \left\{\mathbf{r}_1^T, \ldots, \mathbf{r}_{nc}^T\right\}; \quad nc \ldots \text{number of control nodes} \tag{8.64}$$

Furthermore, there is the relation between the control node coordinates **r** and the parameters **s** which is assumed to be linear:

$$\mathbf{r} = \mathbf{r}^0 + \mathbf{L}_{rs}\mathbf{s} \tag{8.65}$$

Equivalently, the vector $\mathbf{r}^0$ represents initial values of the control node coordinates and $\mathbf{L}_{rs}$ is the linking matrix with constant entries which represents a linear mapping of the parameters **s** to **r**.

The linking function $\mathbf{H}_{xr}$ may resolve (8.59) and represent the morphing functions B_j which states a linear mapping:

$$\Delta\mathbf{x}_i = \mathbf{H}_{xr}(\mathbf{r}) = \sum_j B_j(\xi_i)\mathbf{r}_j \tag{8.66}$$

If the cylindrical or spherical coordinates are used to describe the design model, an additional transformation is necessary to generate final Cartesian coordinates. Now, $\mathbf{H}_{xr}$ are non-linear functions of **r**. They may be decomposed as:

$$\mathbf{H}_{xr}(\mathbf{r}) = \mathbf{T}\left(\sum_j B_j\,\mathbf{r}_j\right) \tag{8.67}$$

where **T**(**y**) denote non-linear transformations. In the case of cylindrical coordinates $\mathbf{y}^T = \{\rho,\theta,\zeta\}$, **T**(**y**) is defined as:

$$\mathbf{T}(\mathbf{y}) = \begin{Bmatrix} \rho\cos\theta \\ \rho\sin\theta \\ \zeta \end{Bmatrix} \tag{8.68}$$

The combination of linear and non-linear parameter relations have to be considered with the sensitivity analysis:

$$\frac{d\mathbf{x}_i}{d\mathbf{s}} = \frac{d\Delta\mathbf{x}_i}{d\mathbf{s}} = \mathbf{L}_{xs} + \frac{d\,\mathbf{H}_{xr}}{d\mathbf{r}}\mathbf{L}_{rs} = \mathbf{L}_{xs} + \frac{d\,\mathbf{T}}{d\mathbf{y}}\frac{d\mathbf{y}}{d\mathbf{r}}\mathbf{L}_{rs} \tag{8.69}$$

8.5.3.1 Linear linking of design model parameters

The following linking rules are useful in practical applications:

- Prescribed move directions:

$$\mathbf{r}_j = \mathbf{r}_j^0 + \mathbf{a}\, s_i \tag{8.70}$$

- Linear combinations:

$$\mathbf{r}_j = \mathbf{r}_j^0 + \sum_i \alpha_i \mathbf{a}_i\, s_i \tag{8.71}$$

- Symmetry:

$$\mathbf{r}_j = \mathbf{r}_j^0 + \left(\mathbf{I} - 2\mathbf{n}\mathbf{n}^T\right)\mathbf{a}\, s_i \tag{8.72}$$

- Projection:

$$\mathbf{r}_j = \mathbf{r}_j^0 + \left(\mathbf{d}\,\mathbf{d}^T\right)\mathbf{a}\, s_i \tag{8.73}$$

Refer to Fig. 8.28 for the definition of vectors and scalars. The parameters s_i refer to the relevant entries of the parameter vector **s**. Geometrical continuity conditions as explained before are another examples of linear linking rules.

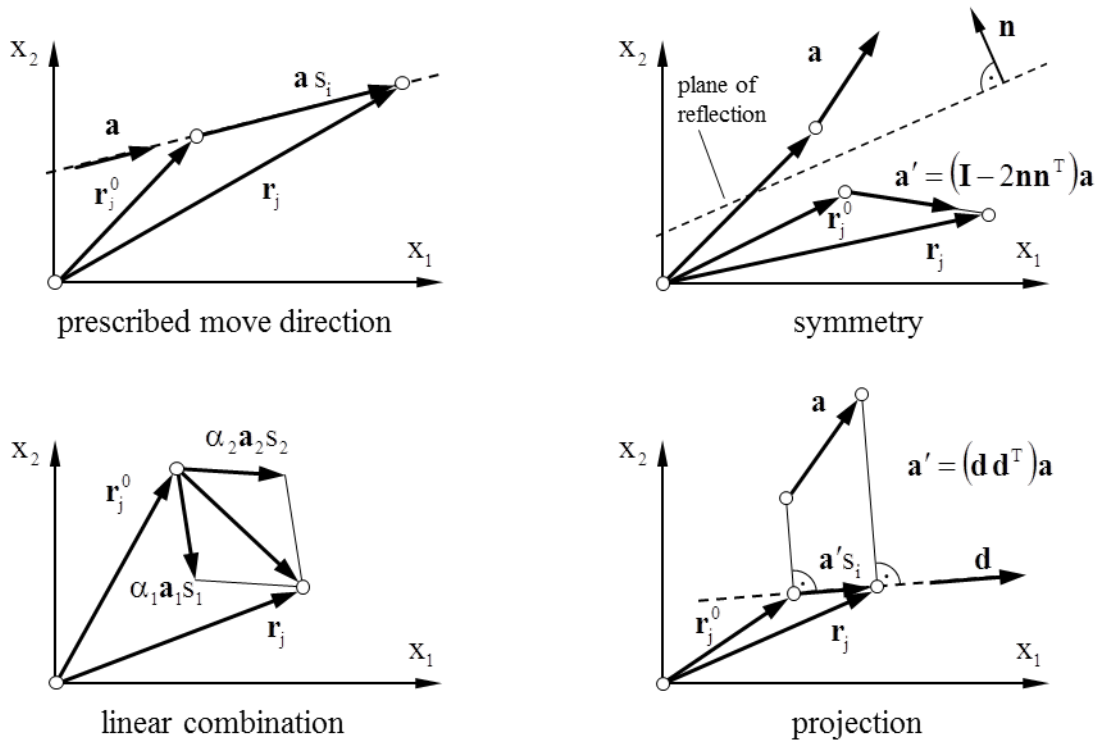


Fig. 8.28: Linear control node linking

8.5.4 The optimization model

The discretized optimization problem states as:

$$\begin{aligned} &\text{minimize} && \mathrm{F}(\mathbf{s},\mathbf{u}) \\ &\text{subject to} && \mathbf{G}(\mathbf{s},\mathbf{u}) \le 0 \\ & && \mathbf{H}(\mathbf{s},\mathbf{u}) = 0 \end{aligned} \tag{8.74}$$

with F the objective function, G and H sets of inequality and equality constraints. They are functions of the vector **s** of design variables and **u** which is the vector of the n_{dof} discrete state variables. F is the approximation of its continuous equivalent objective function Φ as a consequence of replacing all relevant fields by their discretized equivalents, in particular the geometry. The state variables **u** are the solution of the set **S** of n_{dof} state equations which, typically, constitute the discretized weak form of partial differential equations:

$$\mathbf{S}(\mathbf{s},\mathbf{u}) = 0 \tag{8.75}$$

For the example of linear elasticity the state equations represent the weak form of equilibrium and are specified as

$$\mathbf{S} = \mathbf{K}(\mathbf{s})\mathbf{u} - \mathbf{R}(\mathbf{s}) = 0 \tag{8.76}$$

with **K** and **R** as the stiffness matrix and the load vector, respectively. Both are functions of the design parameters **s**.

The optimal solution of (8.74) is evaluated from the stationary conditions of the Lagrange function L:

$$\mathrm{L} = \mathrm{F} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{G} + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{H} + \mathbf{u}^{*\mathrm{T}}\mathbf{S} \tag{8.77}$$

which are known as the Karush-Kuhn-Tucker conditions:

$$\begin{aligned} \mathrm{L}_{\mathrm{s}} &= \mathrm{F}_{\mathrm{s}} + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{G}_{\mathrm{s}} + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{H}_{\mathrm{s}} + \mathbf{u}^{*\mathrm{T}}\mathbf{S}_{\mathrm{s}} = 0 \\ \boldsymbol{\lambda}^{\mathrm{T}}\mathrm{L}_{\lambda} &= \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{G} = 0; \quad \boldsymbol{\lambda} \ge 0 \\ \mathrm{L}_{\mu} &= \mathbf{H} = 0 \\ \mathrm{L}_{\mathrm{u}^*} &= \mathbf{S} = 0 \\ (\cdot)_{\mathrm{a}} &= \frac{\partial(\cdot)}{\partial \mathrm{a}} \end{aligned} \tag{8.78}$$

and **λ**, **μ** and $\mathbf{u}^*$ are the vectors of Lagrange multipliers whereas $\mathbf{u}^*$ is also known the vector of discrete adjoint variables.

In principle, any kind of nonlinear programming method might be applied to solve (8.74). Typically, shape optimization problems are highly nonlinear in **s** which has to be considered for the specific choice of a method. Gradient methods are preferable for large shape optimization problems with many design parameters. Gradient information may also be used to setup the surrogate model of response surface methods.

8.5.5 Discrete sensitivity analysis

This paragraph is partially redundant with chapter 0 about Sensitivity Analysis.

The variants of sensitivity analysis deal with the different treatment of the state equations [48]. As an example, that will be demonstrated for the derivative of the objective function F and the state equations of linear elasticity.

8.5.5.1 Direct sensitivity analysis

Consider the state variables **u** to be functions of the design parameters **s**. Then, starting from

$$\begin{aligned} F &= F(\mathbf{s},\mathbf{u}(\mathbf{s})) \\ \mathbf{S} &= \mathbf{S}(\mathbf{s},\mathbf{u}(\mathbf{s})) \end{aligned} \tag{8.79}$$

we apply the chain rule of differentiation:

$$\begin{aligned} \frac{dF}{d\mathbf{s}} &= F_s + F_u \frac{d\mathbf{u}}{d\mathbf{s}} \\ \frac{d\mathbf{S}}{d\mathbf{s}} &= \mathbf{S}_s + \mathbf{S}_u \frac{d\mathbf{u}}{d\mathbf{s}} = 0 \end{aligned} \tag{8.80}$$

Solving (8.80_2) for d**u**/d**s** and substituting into (8.80_1) yields:

$$\begin{aligned} \frac{d\mathbf{u}}{d\mathbf{s}} &= -\mathbf{S}_u^{-1}\mathbf{S}_s \\ \frac{dF}{d\mathbf{s}} &= F_s - F_u\mathbf{S}_u^{-1}\mathbf{S}_s \end{aligned} \tag{8.81}$$

And, considering the specific example of linear elasticity:

$$\begin{aligned} \mathbf{S} &= \mathbf{K}\mathbf{u} - \mathbf{R} = 0 \\ \frac{d\mathbf{u}}{d\mathbf{s}} &= -\mathbf{K}^{-1}(\mathbf{K}_s\mathbf{u} - \mathbf{R}_s) = \mathbf{K}^{-1}\mathbf{R}^* \;;\quad \mathbf{R}^* = \mathbf{R}_s - \mathbf{K}_s\mathbf{u} \\ \frac{dF}{d\mathbf{s}} &= F_s - F_u\,\mathbf{K}^{-1}(\mathbf{K}_s\mathbf{u} - \mathbf{R}_s) \end{aligned} \tag{8.82}$$

For obvious reasons, $\mathbf{R}^*$ is also called *pseudo load vector*.

8.5.5.2 Discrete adjoint sensitivity analysis

Consider the Lagrangian with $\mathbf{u}^*$ as the discrete adjoint variables which take the roles of Lagrangian multipliers and treat the parameters **s** and the state variables **u** as independent:

$$L = F(\mathbf{s},\mathbf{u}) + \mathbf{u}^{*T}\mathbf{S}(\mathbf{s},\mathbf{u}) \tag{8.83}$$

Taking the total derivative

$$dL = L_s ds + L_u du + L_{u^*} du^* = \left(F_s + \mathbf{u}^{*T}\mathbf{S}_s\right)ds + \left(F_u + \mathbf{u}^{*T}\mathbf{S}_u\right)du + \mathbf{S}\,du^* \tag{8.84}$$

the adjoint variables $\mathbf{u}^*$ are determined from $L_u = 0$ as:

$$\mathbf{u}^* = -\mathbf{S}_u^{-T} F_u \tag{8.85}$$

which states the linear adjoint problem

$$\min \; F^* = \tfrac{1}{2}\mathbf{u}^{*T}\mathbf{S}_u^T\mathbf{u}^* - F_u\mathbf{u}^* \tag{8.86}$$

with the adjoint operator matrix $\mathbf{S}_u{}^T$. Typically, structural problems are self adjoint which reflects the symmetry of the stiffness matrix $\mathbf{K} = \mathbf{S}_u$. Finally, back substituting (8.85) into (8.84) gives:

$$\frac{dF}{ds} = F_s + \mathbf{u}^{*T}\mathbf{S}_s = F_s - \left(\mathbf{S}_u^{-T}F_u\right)^T\mathbf{S}_s \tag{8.87}$$

which is identical with (8.81).

It depends on the number n_r of response functions which are the objective function and all constraints {F, **G**, **H**} compared to the number n_s of design parameters **s** whether the direct or the adjoint sensitivity analysis is numerically more efficient. The adjoint sensitivity analysis is more efficient if $n_s > n_r$ which is typically the case for preliminary design, node based shape optimization and form finding. For CAD based shape optimization, later stages of design when design degrees of freedom are limited but the number of constraints increase and when it holds that $n_s < n_r$ then the direct sensitivity analysis is to be preferred.

8.5.5.3 Discrete design derivative

Recall the definition of the continuous objective function Φ generalized for the two considered cases where D stands for the related integration domain (volume Ω or surface Γ) and f_D is the appropriate objective density:

$$\Phi = \int_D f_D \, dD \tag{8.88}$$

Consequently, the variation $\delta\Phi$ of the objective due to a variation of the geometry $\delta\mathbf{x}$ at the surface Γ is:

$$\delta\Phi = \int_\Gamma \frac{df_\Gamma}{d\mathbf{x}}\delta\mathbf{x}\,d\Gamma\,; \quad \delta\mathbf{x} = \frac{d\mathbf{x}}{dt}\delta t = \mathbf{v}\,\delta t \tag{8.89}$$

After discretizing the geometry variation $\delta\mathbf{x}$ in terms of the varied design parameters $\delta\mathbf{s}$ by applying (8.45)

$$\delta\mathbf{x} = \mathbf{B}\,\delta\mathbf{s} \tag{8.90}$$

we arrive at the discretized variation δF of the objective function:

$$\delta F = \frac{dF}{d\mathbf{s}}\,\delta\mathbf{s} = \int_{\Gamma} \frac{df_{\Gamma}}{d\mathbf{x}}\,\mathbf{B}\,\delta\mathbf{s}\,d\Gamma \tag{8.91}$$

Then, the discrete version of the design derivative dF/d**s** is defined as, compare with (8.53):

$$\frac{dF}{d\mathbf{s}} = \int_{\Gamma} \mathbf{B}^{T}\,\frac{df_{\Gamma}}{d\mathbf{x}}\,d\Gamma \tag{8.92}$$

In discrete shape optimization, typically, the discrete version of the design derivative is evaluated directly by the sensitivity analysis and the gradient $df_{\Gamma}/d\mathbf{x}$ remains to be evaluated by subsequent operations, if necessary.

8.5.5.4 Semi-analytical sensitivity analysis

An important part of sensitivity analysis is the evaluation of the partial derivatives $\mathbf{S}_s$ of the state equations with respect to the design parameters **s**. Basically, the derivatives of the operator matrix and the source terms of **S** have to be evaluated. For structural problems that are the system stiffness matrix **K** and the system load vector **R**, respectively. The derivatives $\mathbf{K}_s$ and $\mathbf{R}_s$ are assembled from element matrices and vectors following the standard procedures. In particular, determining the derivatives $\mathbf{k}_s$ of the element stiffness matrix may be very elaborate. In particular, that is the case for highly developed and sophisticated structural finite elements, e.g. ANS and EAS elements for shell analysis where many additional operations are performed. Indeed, because of the product and chain rule of differentiation, evaluating the exact analytical derivatives of element stiffness matrices $\mathbf{k}_s$ may take considerable more computational effort than evaluating the stiffness matrix itself. Then, the exact derivative $\mathbf{k}_s$ should be replaced by a finite difference approximation $\Delta\mathbf{k}$ to arrive at considerable more efficient code. From practical experience it can be stated that the numerical errors by the finite difference step can be neglected if the corrections considering the rigid body rotation test have been made (refer to the next paragraph). Measures of code efficiency depend on the specific implementation and cannot be generalized. From the experience with own implementations the differences in computational time may easily be of factor two and more for the most sophisticated element formulations. To prefer the semi-analytical sensitivity analysis against the exact one is a pragmatic decision.

(8.82) may be modified by the use of the finite difference approximations $\Delta\mathbf{K}$ and $\Delta\mathbf{R}$ of the system stiffness matrix and load vector:

$$\frac{dF}{d\mathbf{s}} \approx F_s - F_u\,\mathbf{K}^{-1}\left(\Delta\mathbf{K}\,\mathbf{u} - \Delta\mathbf{R}\right) \tag{8.93}$$

8.5.6 The rigid body rotation test

This paragraph is partially redundant with chapter 0 about Sensitivity Analysis.

The phenomenon described hereon is specific for shape optimization of structures. It is a consequence of the change of finite element orientation by a variation of the design parameters.

Let us consider a symmetric matrix **K** of size (*ndof,ndof*), e.g. an element stiffness matrix **k**. With respect to base vectors the matrix might be written as the sum of dyadic products adopting Einstein's summation convention. Choosing eigenvectors ϕ_i as base vectors **K** is written as

$$\mathbf{K} = \Lambda_{ij}\boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j; \quad i, j = 1,..., ndof; \quad \boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j := \boldsymbol{\phi}_i \boldsymbol{\phi}_j^T \tag{8.94}$$

With

$$\begin{aligned} \boldsymbol{\phi}_i \cdot \boldsymbol{\phi}_j = \delta_{ij} &= \begin{cases} 1; & \text{if } i = j \\ 0; & \text{if } i \neq j \end{cases} \\ \text{and} \qquad \Lambda_{ij} &= \begin{cases} \lambda_i; & \text{if } i = j \\ 0; & \text{if } i \neq j \end{cases} \end{aligned} \tag{8.95}$$

where λ_i are the eigenvalues of **K**.

The set of zero eigenvectors or rigid body modes $\boldsymbol{\varphi}_i$ is defined by

$$RBM = \{i;\ ndof - nrbm + 1 \le i \le ndof\}; \quad \text{if } i \in RBM \quad \text{then} \quad \boldsymbol{\varphi}_i := \boldsymbol{\phi}_i \ \text{ and } \ \lambda_i = 0 \tag{8.96}$$

applying a decreasing order of (positive) eigenvalues where the *nrbm* zero eigenvalues are sorted at the end of all *ndof* eigenvalues.

Vice versa the remaining set of deformation modes is defined as:

$$DFM = \{i; 1 \le i \le ndof - nrbm\}; \quad \text{if } i \in DFM \quad \text{then} \quad \lambda_1 \ge \lambda_i \ge \lambda_{ndof-nrbm} > 0 \tag{8.97}$$

Finally, the full set EGM of all eigenmodes is defined as:

$$EGM = DFM \cup RBM = \{i; 1 \le i \le ndof\} \tag{8.98}$$

Applying the product rule of differentiation the derivative of **K** is:

$$\frac{d\mathbf{K}}{ds} = \mathbf{K}_{,s} = \Lambda_{ij,s}\boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j + \Lambda_{ij}\boldsymbol{\phi}_{i,s} \otimes \boldsymbol{\phi}_j + \Lambda_{ij}\boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_{j,s}; \quad i, j \in EGM \tag{8.99}$$

The derivatives of eigenmodes can be generated by linear combination of eigenmodes in turn

$$\boldsymbol{\phi}_{i,s} = a_{ij}\boldsymbol{\phi}_j; \quad a_{ij} = \boldsymbol{\phi}_{i,s} \cdot \boldsymbol{\phi}_j; \quad a_{ij} = -a_{ji} \tag{8.100}$$

and the derivative $\mathbf{K}_{,s}$ related to eigenmodes of **K** as special choice of bases yields to be (indices in brackets are not considered for summation):

$$\begin{aligned} \mathbf{K}_{,s} &= \left(\Lambda_{ij,s} + \Lambda_{ik}a_{kj} + \Lambda_{kj}a_{ki}\right)\boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j = \Lambda_{ij|s}\ \boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j \\ &= \left(\lambda_{(i),s}\delta_{ij} + \lambda_{(i)}a_{ij} + \lambda_{(j)}a_{ji}\right)\boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j = \lambda_{(i)|s}\ \boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j; \quad i, j \in EGM \end{aligned} \tag{8.101}$$

That defines the co-variant derivatives of matrix components $\Lambda_{ij|s}$.

It is important to note that with the change of s there come modifications of eigenvalues λ_i and eigenmodes $\boldsymbol{\phi}_i$ in value and direction, respectively. The latter is the reason for off-diagonal fill-in of matrix coefficients $\Lambda_{ij|s}$. However, the number of rigid body modes *nrbm* always remains

constant. In other words zero eigenvalues remain to be zero as well as their derivatives as the design variable s is varied:

$$\lambda_i = 0 \text{ and } \lambda_{i,s} = 0 \quad \text{for} \quad i \in RBM \tag{8.102}$$

As a consequence, which can be seen from (8.101), the subset of components of $\mathbf{K}_{,s}$ which relate to rigid body modes always vanish to be zero for any choice of s:

$$\Lambda_{ij|s} = 0; \ \Lambda_{ij,s} = 0; \ \Lambda_{ij} = 0 \quad \text{for} \quad i, j \in RBM \tag{8.103}$$

That result is well known and reflects the rigid body test which $\mathbf{K}_{,s}$ has to satisfy [Mljenek, Lund, …] for all rigid body modes $\boldsymbol{\varphi}_i$:

$$\boldsymbol{\varphi}_i \mathbf{K}_{,s} \boldsymbol{\varphi}_j = 0 \quad \text{for} \quad i, j \in RBM \tag{8.104}$$

It is explained by starting with the rigid body eigenvalue problem

$$\mathbf{K} \boldsymbol{\varphi}_m = 0 \quad \text{for} \quad m \in RBM \tag{8.105}$$

Derivation of (8.105) with respect to s gives what here is called the rigid body condition:

$$\mathbf{K}_{,s} \boldsymbol{\varphi}_m + \mathbf{K} \boldsymbol{\varphi}_{m,s} = 0 \quad \text{for} \quad m \in RBM \tag{8.106}$$

and, again using modal decomposition for clarity:

$$\begin{aligned} \Lambda_{ij|s} \, \boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j \cdot \boldsymbol{\varphi}_m + \Lambda_{ij} \, \boldsymbol{\phi}_i \otimes \boldsymbol{\phi}_j \cdot \boldsymbol{\varphi}_{m,s} &= 0; \quad \text{for } i, j \in EGM \text{ and } m \in RBM \\ \Lambda_{im|s} \, \boldsymbol{\phi}_i + \Lambda_{ij} a_{mj} \, \boldsymbol{\phi}_i &= 0 \\ \Lambda_{im|s} \, \boldsymbol{\phi}_i + \lambda_{(i)} a_{mi} \, \boldsymbol{\phi}_i &= 0 \end{aligned} \tag{8.107}$$

Because of the zero eigenvalues of rigid body modes, obviously, the above result follows:

$$\Lambda_{im|s} = 0 \text{ because } \lambda_{(i)} = 0 \quad \text{for} \quad i, m \in RBM \tag{8.108}$$

Applying the rigid body test and pre-multiplying (8.107) by another rigid mode $\boldsymbol{\varphi}_n$ again gives:

$$\begin{aligned} \boldsymbol{\varphi}_n \cdot \left(\Lambda_{im|s} \, \boldsymbol{\phi}_i + \lambda_{(i)} a_{mi} \, \boldsymbol{\phi}_i \right) &= 0; \quad i \in EGM \text{ and } m, n \in RBM \\ \left(\Lambda_{im|s} + \lambda_{(i)} a_{mi} \right) \delta_{in} &= 0 \\ \Lambda_{mn|s} + \lambda_{(n)} a_{mn} &= \Lambda_{mn|s} = 0 \end{aligned} \tag{8.109}$$

As a conclusion we state that the rigid body subset $\Lambda_{mn|s}$ of components of $\mathbf{K}_{,s}$ must identically vanish such that the rigid body condition (8.106) is satisfied allowing the rigid body test (8.104) to be successful. Analogously, this result must also apply to numerical approximations of analytical derivatives.

8.5.6.1 Inconsistent numerical derivatives of **K**

A simple numerical derivative $\Delta\mathbf{K}$ is determined by the difference of two matrices evaluated for two distinct values s and s + Δs of the design variable. All entities evaluated at s + Δs will be denoted by an upper bar:

$$\begin{aligned} \Delta\mathbf{K} &= \frac{\overline{\mathbf{K}}(s+\Delta s)-\mathbf{K}(s)}{\Delta s} \\ \therefore\quad \Delta\mathbf{K}\,\Delta s &= \overline{\mathbf{K}}(s+\Delta s)-\mathbf{K}(s)=\overline{\Lambda}_{ij}\overline{\boldsymbol{\phi}}_i\otimes\overline{\boldsymbol{\phi}}_j-\Lambda_{ij}\boldsymbol{\phi}_i\otimes\boldsymbol{\phi}_j;\quad i,j\in EGM \end{aligned} \tag{8.110}$$

Analogously defining the difference of base vectors and matrix components as

$$\overline{\boldsymbol{\phi}}_i=\boldsymbol{\phi}_i+\Delta\boldsymbol{\phi}_i \quad\text{and}\quad \Delta\boldsymbol{\phi}_i=\Delta a_{ik}\boldsymbol{\phi}_k;\quad \overline{\Lambda}_{ij}=\Lambda_{ij}+\Delta\Lambda_{ij} \tag{8.111}$$

we can proceed:

$$\begin{aligned} \Delta\mathbf{K}\,\Delta s &= \overline{\Lambda}_{ij}\left(\boldsymbol{\phi}_i\otimes\boldsymbol{\phi}_j+\Delta\boldsymbol{\phi}_i\otimes\boldsymbol{\phi}_j+\boldsymbol{\phi}_i\otimes\Delta\boldsymbol{\phi}_j+\Delta\boldsymbol{\phi}_i\otimes\Delta\boldsymbol{\phi}_j\right)-\Lambda_{ij}\boldsymbol{\phi}_i\otimes\boldsymbol{\phi}_j;\quad i,j,k,l\in EGM \\ &= \left(\Delta\Lambda_{ij}+\overline{\Lambda}_{ki}\Delta a_{kj}+\overline{\Lambda}_{kj}\Delta a_{ki}+\overline{\Lambda}_{kl}\Delta a_{ki}\Delta a_{lj}\right)\boldsymbol{\phi}_i\otimes\boldsymbol{\phi}_j \end{aligned} \tag{8.112}$$

Finally, the co-variant components of $\Delta\mathbf{K}$ are given as

$$\begin{aligned} \Delta K_{ij} &= \frac{1}{\Delta s}\left(\Delta\Lambda_{ij}+\overline{\Lambda}_{ki}\Delta a_{kj}+\overline{\Lambda}_{kj}\Delta a_{ki}+\overline{\Lambda}_{kl}\Delta a_{ki}\Delta a_{lj}\right);\quad i,j,k,l\in EGM \\ &= \frac{1}{\Delta s}\left(\Delta\lambda_{(i)}\delta_{ij}+\overline{\lambda}_{(i)}\Delta a_{ij}+\overline{\lambda}_{(j)}\Delta a_{ji}+\overline{\lambda}_{(k)}\Delta a_{ki}\Delta a_{kj}\right) \end{aligned} \tag{8.113}$$

The inherent change of basis while generating the difference of matrices, obviously, forces to fail the rigid body condition. Inspecting the rigid body subset of ΔK_{mn} we realize:

$$\Delta K_{mn}=\frac{1}{\Delta s}\overline{\lambda}_{(k)}\Delta a_{km}\Delta a_{kn}\neq 0;\qquad m,n\in RBM;\ k\in DFM \tag{8.114}$$

because of the non-zero eigenvalues of deformation modes in the perturbed configuration s + Δs.

Further inspection tells that the problem arises for rigid body rotations only. As rigid body translation modes are preserved by a design change Δs the difference of related mode vectors vanishes:

$$\begin{gathered} \Delta\boldsymbol{\phi}_m=0\quad\text{and}\quad \Delta a_{mk}=0; \\ \text{for}\ k\in EGM\quad\text{and}\quad \boldsymbol{\phi}_m\ \text{rigid body translation} \\ \text{and}\ \Delta K_{mn}=0\quad\text{for}\quad \boldsymbol{\phi}_m\ \text{or}\ \boldsymbol{\phi}_n\ \text{rigid body translations} \end{gathered} \tag{8.115}$$

8.5.6.2 Consistent numerical derivative

The simple numerical derivative $\Delta\mathbf{K}$ is modified as consistent numerical derivative $\Delta_c\mathbf{K}$ which passes the rigid body test. As shown above it is at least necessary to nullify the rigid body subspace of $\Delta\mathbf{K}$. That is simply done by, first, adding additional components to the rigid body subspace [44]

$$\Delta_C\mathbf{K} = \Delta\mathbf{K} + C_{mn}\boldsymbol{\varphi}_m \otimes \boldsymbol{\varphi}_n ; \qquad m,n \in RBM \tag{8.116}$$

and, then, applying the rigid body test for rigid body rotation modes $\boldsymbol{\varphi}_r$, $\boldsymbol{\varphi}_s$ to identify spurious matrix components:

$$\boldsymbol{\varphi}_r \cdot \Delta_C\mathbf{K} \cdot \boldsymbol{\varphi}_s = \boldsymbol{\varphi}_r \cdot \left(\Delta\mathbf{K} + C_{mn}\boldsymbol{\varphi}_m \otimes \boldsymbol{\varphi}_n\right) \cdot \boldsymbol{\varphi}_s = 0 ; \qquad m,n,r,s \in RBM \tag{8.117}$$

Comparing with (8.114) the correction terms are readily identified as:

$$C_{mn} = -\frac{1}{\Delta s}\,\bar{\lambda}_{(k)}\Delta a_{km}\Delta a_{kn} \qquad m,n \in RBM ; k \in DFM \tag{8.118}$$

Because, typically, $\Delta\mathbf{K}$ is given in Cartesian type basis rather than modal decomposition the correction matrix components are identified by:

$$C_{mn} = -\boldsymbol{\varphi}_m \cdot \Delta\mathbf{K} \cdot \boldsymbol{\varphi}_n \tag{8.119}$$

In the most general case of three dimensional structural analysis and three independent rigid body rotations not more than six matrix components C_{mn} must be determined to evaluate the consistent numerical derivative $\Delta_c\mathbf{K}$. The numerical effort can be neglected compared to twice the evaluation of $\mathbf{K}$ at s and s + Δs.

Necessary prerequisite of the suggested correction procedure is the knowledge of rigid body rotation modes. For stiffness matrices in structural analysis, however, these modes can easily by anticipated by simple geometric inspection. There is no problem to identify rigid rotational motion vectors $\mathbf{r}_i$ around all spatial axes in terms of the chosen degrees of freedom, e.g. nodal displacement and rotation components for shell models. As easily as those rotational motions can be detected, however, these are in general neither orthogonal among each other nor normalized. Prior to apply (8.119) the rigid rotation modes have be extracted from the rigid rotation motion vectors $\mathbf{r}_i$, e.g. by applying Gram-Schmidt othogonalization.

An alternative technique to generate an “exact” numerical derivative $\mathbf{K}_{,s}$ is reported by Olhoff et al. [49] and van Keulen [50].

8.5.7 Shape update and sensitivity weighting

The steepest descent direction field **d** for the shape update is the negative gradient of the shape gardient $df_\Gamma/d\mathbf{x}$; refer to (8.27) for the continuous case. It holds:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \alpha\, \mathbf{d}\, \Delta t \tag{8.120}$$

Eventually, the factor α is determined by a line search. The descent direction might be discretized in terms of the discrete descent direction $\mathbf{d}_s$ of the design parameters **s** applying the mapping (8.45):

$$\mathbf{d} = \mathbf{B}\, \mathbf{d}_s \tag{8.121}$$

Analogously to (8.120), the update formula of the design parameters is

$$\mathbf{s}(t+\Delta t) = \mathbf{s}(t) + \alpha_s\, \mathbf{d}_s\, \Delta t \tag{8.122}$$

again with a line search factor α_s.

The parameter descent direction $\mathbf{d}_s$ is determined from the discrete design derivative (8.92) where we identify the steepest descent direction field **d** and apply (8.121):

$$\frac{dF}{d\mathbf{s}} = \int_\Gamma \mathbf{B}^T \frac{df_\Gamma}{d\mathbf{x}}\, d\Gamma = -\int_\Gamma \mathbf{B}^T \mathbf{d}\, d\Gamma = -\int_\Gamma \mathbf{B}^T \mathbf{B}\, \mathbf{d}_s\, d\Gamma \tag{8.123}$$

Finally, the descent design update direction $\mathbf{d}_s$ is identified as the discrete version of the negative design gradient $df_\Gamma/d\mathbf{s}$ which gives:

$$\frac{dF}{d\mathbf{s}} = \int_\Gamma \mathbf{B}^T \mathbf{B} \frac{df_\Gamma}{d\mathbf{s}}\, d\Gamma = \int_\Gamma \mathbf{B}^T \mathbf{B}\, d\Gamma \frac{df_\Gamma}{d\mathbf{s}} \tag{8.124}$$

From that point of view the discrete design gradient (which is the negative design update direction $\mathbf{d}_s$) might be understood as to be generated from the discrete design derivative by weighting with the inverse of the weighting matrix **M** [12]:

$$\begin{aligned} \mathbf{M} &= \int_\Gamma \mathbf{B}^T \mathbf{B}\, d\Gamma \\ \frac{df_\Gamma}{d\mathbf{s}} &= \mathbf{M}^{-1} \frac{dF}{d\mathbf{s}} \end{aligned} \tag{8.125}$$

Alternatively, a lumped matrix can be used with the diagonal entries M_i which, of course, is more efficient to be inverted. That pays off for shape functions with large, overlapping support, e.g. NURBS and IGA [14]:

$$\begin{aligned} \mathbf{M} &= \mathrm{diag}[M_i] \\ M_i &= \int_D B_i\, dD \end{aligned} \tag{8.126}$$

Discrete derivatives dF/d**s** and, eventually, d**G**/d**s**, d**H**/d**s** of the objective function F and the set of constraints **C** = {**G**, **H**}, respectively, have to be weighted as indicated to be consistent with the equivalent continuous problem and to give discretization independent descent directions. Directly applying the discrete derivatives as descent direction will lead to discretization depended iteration histories and eventually to different local solutions.

In practice the procedure is not necessary if the integration domains Γ_i assigned to the design parameters s_i are (at least approximately) of the same size. That is typically the case if on regular parameter grids the same shape functions of equal support are used. As well, it is not necessary for global minimization problems which will converge to the proper result anyway.

The procedure must definitely be applied in the case of IGA based shape optimization. The open NURBS used as shape functions have the mentioned disadvantageous properties. IGA shape optimization of nonconvex problems is heavily dependent on proper sensitivity weighting.

8.5.7.1 Example

Consider a rectangular plate modeled by two patches taking bilinear shape functions, Fig. (). The design parameters $\{s_1, s_2, s_3\}$ are the vertical coordinates of the control nodes at the upper edge. The objective function is the area A of the plate.

The continuous formulation:

$$\text{objective}: \quad \Phi = \int_\Omega d\Omega = 2a \int dy$$

$$\text{objective density}: \quad f_\Omega = \frac{df_\Gamma}{dz} = 1$$

$$\text{variation}: \quad \delta\Phi = \frac{d\Phi}{dz}\delta z = \int_\Gamma \frac{df_\Gamma}{dz}\delta z \, d\Gamma$$

$$\text{shape gradient}: \quad \mathbf{v}\cdot\mathbf{n} = \frac{df_\Gamma}{dz} = 1$$

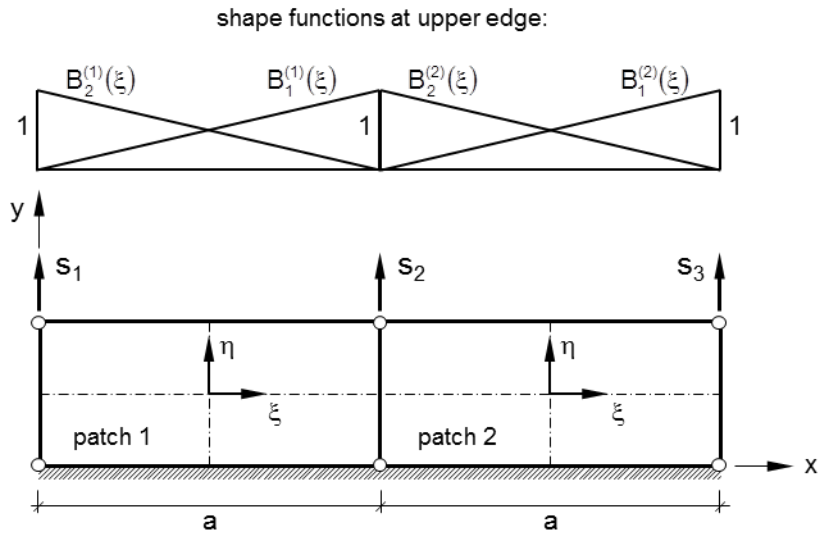


Fig. 8.29: Shape optimization of rectangular plate, upper edge to modified.

The discrete problem formulation:

$$\text{discrete ojective}: \quad F = \tfrac{1}{2} a \left(s_1 + 2 s_2 + s_3 \right)$$

$$\text{discrete design derivative}: \quad \frac{dF}{d\mathbf{s}} = \frac{a}{2} \begin{Bmatrix} 1 \\ 2 \\ 1 \end{Bmatrix}$$

Obviously, the discrete design derivatives do not represent the continuous shape gradient, Fig. (8.30).

Weighting of the design derivatives:

The two linear shape functions at the top edge of each element in natural coordinates:

$$B_1 = \tfrac{1}{2}(1+\xi)$$

$$B_2 = \tfrac{1}{2}(1-\xi)$$

Interpolation of coordinates and determinant of the Jacobian:

$$\text{element } 1: \quad x = B_2 \cdot 0 + B_1 \cdot a; \quad \det J = \frac{dx}{d\xi} = \frac{a}{2}$$

$$\text{element } 2: \quad x = B_1 \cdot 2a + B_2 \cdot a; \quad \det J = \frac{dx}{d\xi} = \frac{a}{2}$$

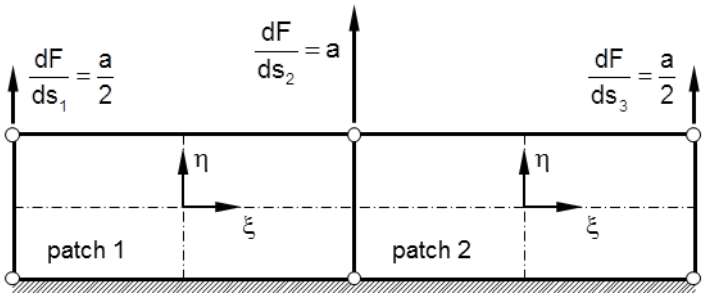


Fig. 8.30: Shape optimization of rectangular plate, discrete design derivatives.

Generation of the weighting matrix **M** by element wise integration at the upper edges, the head indices in brackets refer to the element:

$$M_{11} = \int_{\xi^{(1)}=-1}^{1} B_2^{(1)} B_2^{(1)} \det J \, d\xi^{(1)} = \tfrac{1}{3} a$$

$$M_{12} = \int_{\xi^{(1)}=-1}^{1} B_2^{(1)} B_1^{(1)} \det J \, d\xi'^{(1)} = \tfrac{1}{6} a$$

$$M_{13} = \int_{\xi^{(1)}=-1}^{1} B_2^{(1)} B_1^{(2)} \det J \, d\xi^{(1)} = 0$$

$$M_{22} = \int_{\xi^{(1)}=-1}^{1} B_1^{(1)} B_1^{(1)} \det J \, d\xi'^{(1)} + \int_{\xi^{(2)}=-1}^{1} B_2^{(2)} B_2^{(2)} \det J \, d\xi'^{(2)} = \tfrac{2}{3} a$$

$$M_{23} = \int_{\xi^{(2)}=-1}^{1} B_2^{(2)} B_1^{(2)} \det J \, d\xi'^{(2)} = \tfrac{1}{6} a$$

Weighting of the derivatives:

$$\mathbf{M} = a \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & \frac{1}{6} & \frac{1}{3} \end{bmatrix}$$

$$\frac{df_\Gamma}{d\mathbf{s}} = \mathbf{M}^{-1} \frac{dF}{d\mathbf{s}} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$$

The corrected derivatives represent the continuous shape gradient at the respective nodes. The corrected derivatives might be used for consistent shape update where the upper edge is moved parallel in vertical direction.

Alternative diagonal weighting matrix:

$$M_1 = \int_{\xi^{(1)}=-1}^{1} B_2^{(1)} \det J \, d\xi^{(1)} = \tfrac{1}{2} a$$

$$M_2 = \int_{\xi^{(1)}=-1}^{1} B_1^{(1)} \det J \, d\xi^{(1)} + \int_{\xi^{(2)}=-1}^{1} B_2^{(2)} \det J \, d\xi^{(2)} = a \; ; \qquad \mathbf{M}_{\text{diag}} = \text{diag}\left[\tfrac{1}{2} a, a, \tfrac{1}{2} a\right]$$

$$M_3 = \int_{\xi^{(2)}=-1}^{1} B_1^{(2)} \det J \, d\xi^{(2)} = \tfrac{1}{2} a$$

Weighting of the derivatives:

$$\frac{\mathrm{df}_\Gamma}{\mathrm{d}\mathbf{s}} = \mathbf{M}_{\mathrm{diag}}^{-1}\,\frac{\mathrm{dF}}{\mathrm{d}\mathbf{s}} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$$

Again, we get the proper information.

8.6 Examples for the CAD-based shape optimization

The following examples focus on the geometric aspects of the modeling of CAD-based shape optimization problems. They are not intended to completely report all details.

8.6.1 Shape optimization of a shell – multi patch model and continuity

The example demonstrates the use of continuity conditions of multipatch design models. It is inspired by the famous shell by E. Saarinen on the campus of the Massachusetts Institute of Technology, USA, built in 1955. The original shape is created as a truncated spatial triangle of an ideal sphere which rests on the three vertices. As a consequence the free edges had to be reinforced by edge beams to improve the overall stiffness of the structure. The shell has been made from reinforced concrete. It is about 9 cm thick, spans across 48m and is 15m high. The depth of the edge beams varies with a maximum of about 50cm at the supports. The question is for the optimal shape of an alternative shell of improved stiffness but without an edge beam.

The geometry is modeled by 6 cubic Bézier-patches which constitute the design model. G_1-continuity is enforced by superimposed continuity elements which connect the neighbor control nodes of adjacent patches, Fig. 8.31. Note the center node where 6 design patches meet. At this node there is a non-regular point of the surfaces. Consequently, the related continuity element is non-regular as well. Consequently – and because of symmetry – that continuity element remains plane throughout the optimization. Concerning symmetry the vertical coordinates of 6 control nodes have been assigned as design parameters and the remaining nodes are directly linked. Additionally, 5 more parameters have been assigned as discrete values to model the continuous thickness variation of the shell again respecting symmetry. All together there are 11 discrete design parameters used for optimization. As objective function the strain energy has been chosen. Constraints have been defined to limit the maximum height and the structural volume. The nodes and elements of the analysis model ar generated from the design model by interpolation. Linear elasticity is assumed and Reissner/Mindlin degenerated shell elements applied. Fig. 8.32 shows the initial spherical shape and the optimal shape which is characterized by the negative curvature at the free edges which gives the additional necessary stiffness. Refer to Fig. 8.31 for the optimal spatial position of the continuity elements controlling the linear relations between adjacent control nodes.

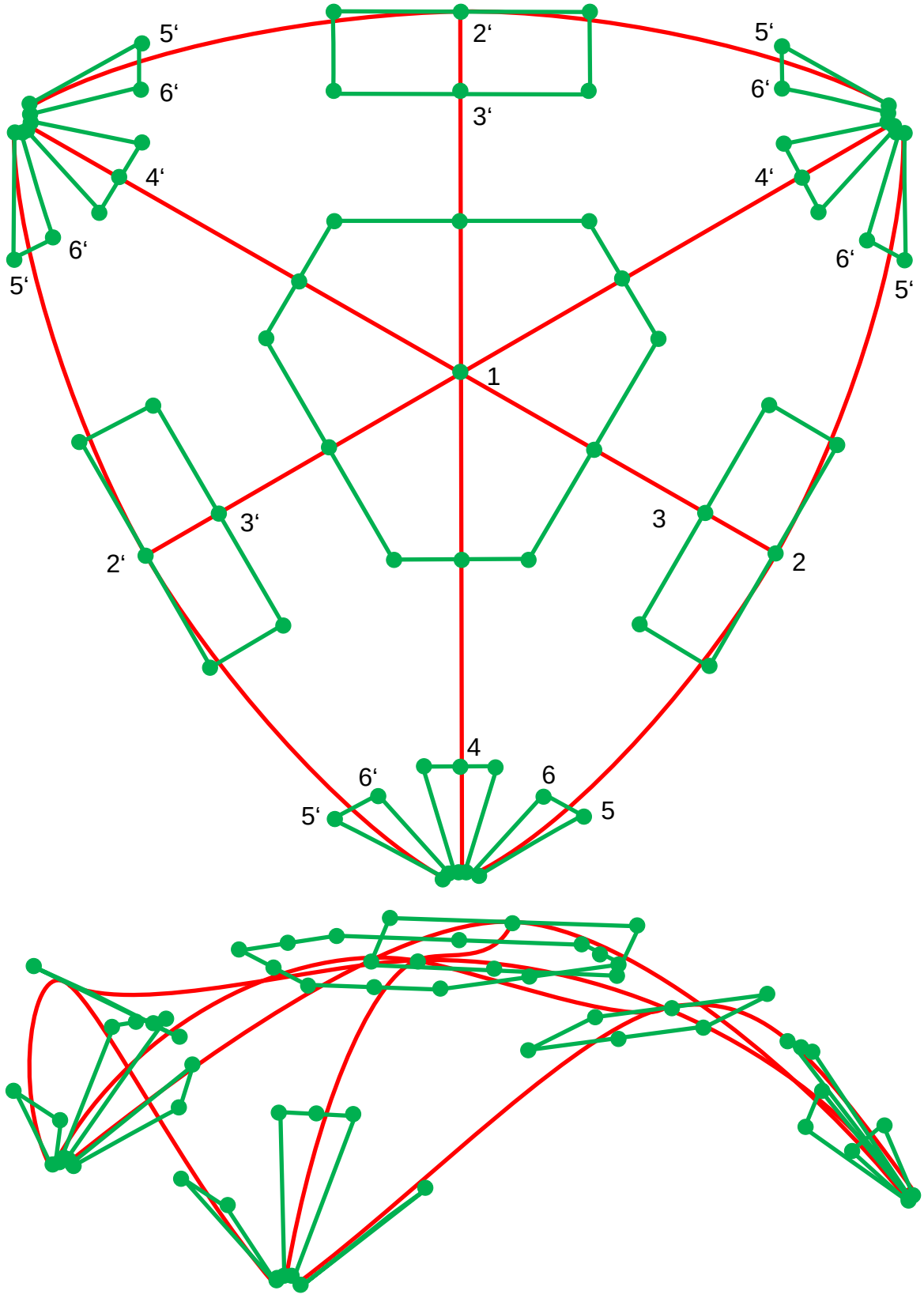


Fig. 8.31: Design model of free form shell: 4 Bézier patches; 6 variable control nodes, symmetry conditions: node (·)’ directly linked to node (·); continuity elements; ground plan (top); isometric view of the optimized model (bottom)

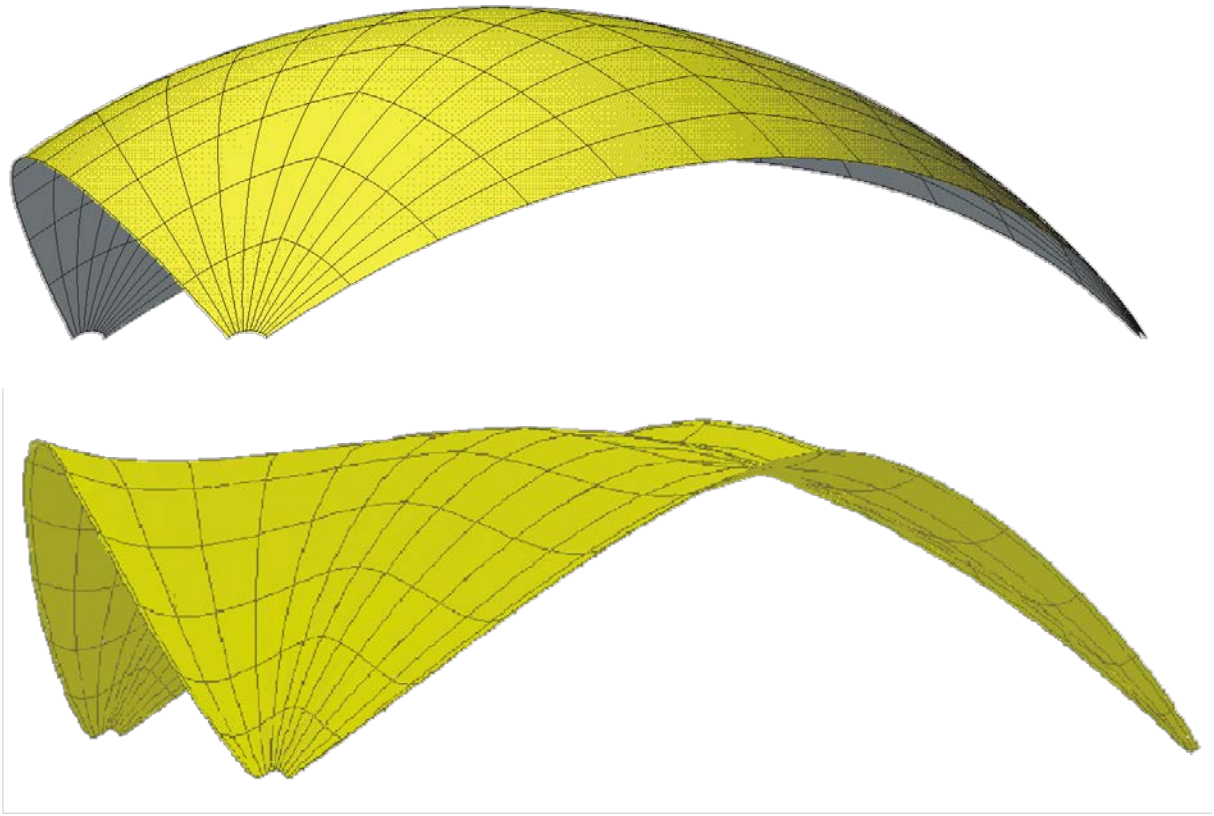


Fig. 8.32: Free form shell; generated shape and finite elements of the analysis model; initial shape (top); optimal shape (bottom)

8.6.2 IGA shape optimization of a plate - sensitivity weighting

That example is a variation of the example presented in chap. 8.5.7.1 designed to demonstrate the effect of sensitivity weighting in the special context of the isogeometric analysis (IGA), Fig. 8.33. The design parameters are chosen to be the vertical coordinates of the displayed seven control nodes. The analysis model is not displayed. It is derived from the same patch by k-refinement, inserting as many additional nodes and degrees of freedom as necessary to determine converged solutions of the state equations. Obviously, design and analysis model share the same geometry which is an advantageous property of IGA. Regarding the data flow and mappings between design and shape model there is no basic difference between IGA and standard techniques. Again, the objective is to minimize the area with the optimum of zero area. Both weighted and unweighted gradients lead to this global optimum. Clearly, this optimal solution is trivial. However, the example is useful to illustrate the difference between weighted and unweighted sensitivities and the corresponding design update. The plate is discretized with quadratic B-splines in length direction and linear in height direction. For the first optimization step the nodal sensitivities, which represent the discrete design derivatives, sare computed and depicted in Fig. 8.33 (b). It can be seen clearly that the discrete nodal sensitivities are not constant, which represents the specific discretization by open B-spline shape functions. As a result, if based on the discrete design derivative, the design update is not constant as well, see Fig. 8.33 (c). It is interesting to note that, here, the distribution of design derivatives corresponds to the distribution of consistent nodal forces representing a distributed constant load. This is due to

the fact that consistent nodal forces are obtained by integrating the product of load function and shape function over the domain. For a constant load, this simplifies to multiplying the load with the integral of the shape function over the domain, which is identical to the weighting factors of matrix **M** as defined in 8.126. Fig. 8.33 (d) shows the design update with the weighted nodal sensitivities (representing the discrete design gradient) which is consistent with the constant, analytical design gradient [14].

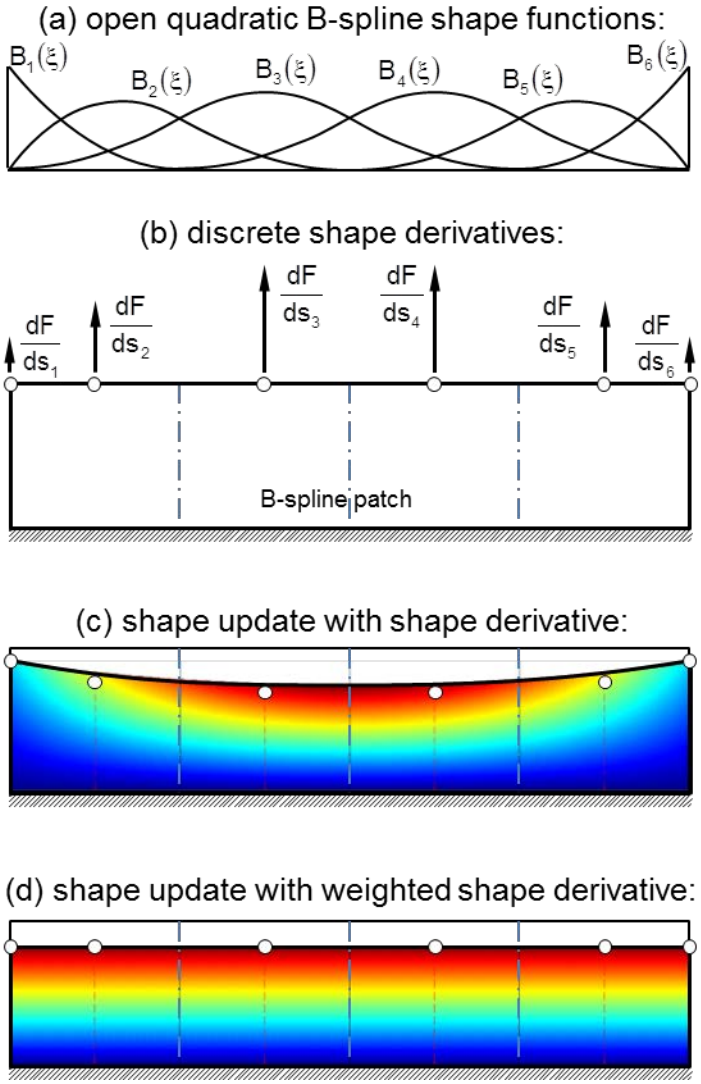


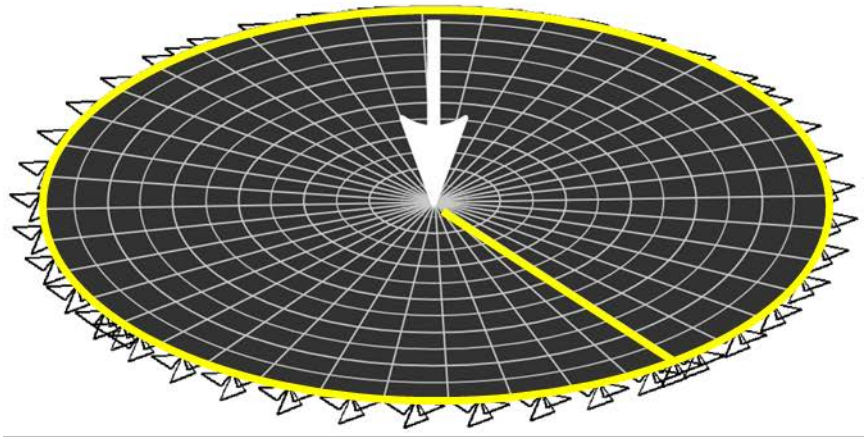
Fig. 8.33: Shape optimization of a plate with the isogeometric analysis

8.6.3 IGA shape optimization of a spatial shell - sensitivity weighting

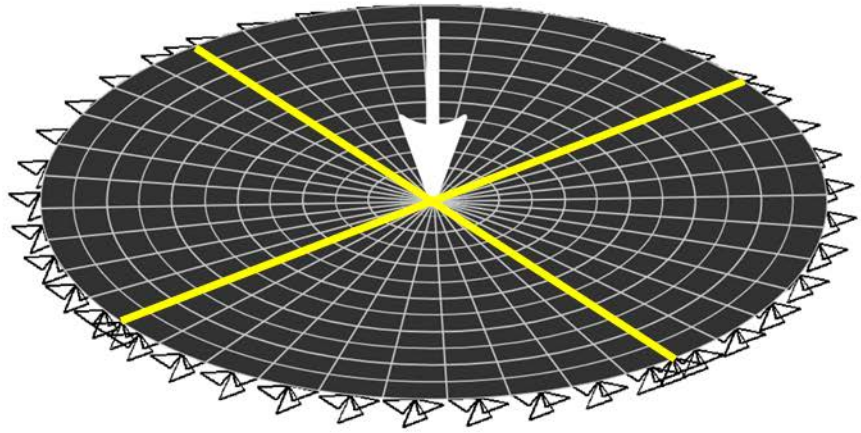
As a second example, we investigate a more complex, non-convex problem, where sensitivity weighting affects the final design by avoiding parametrization-dependent local minima [14]. The problem set up is shown in Fig. 8.34. An initially plane, circular plate, simply supported and with a single load at the center, is to be optimized with respect to stiffness by minimizing the strain energy. The plate is loaded by a concentrated vertical downward at the center. A IGA Kirchoff shell element is used for analysis. Linear elasticity is assumed. The geometry is modeled as a surface of revolution by two alternatives: (i) as an rotational symmetric period spline, and (ii) composed of four patches with four lines of G^0-continuity at at 0°, 90°, 180°, and 270°.

At these locations, G^1-continuity is enforced by applying appropriate coupling techniques, e.g. by bending strips. Bending strips are equivalent to the continuity elements as discussed before. As a consequence the support of parameters on the G^0-lines is enlarged compared to others inside the domain. Straight forwardly, the distribution of unweighted nodal objective sensitivities is not rotational symmetric as compared to the periodic model. Weighting again gives rotational symmetric distributed nodal values, again constituting the difference between discrete design derivative and discrete design gradient. Both, weighted and non-weighted sensitivities generate valid descend directions and can be used for numerical optimization. For the present non-convex problem, however, the consequence for the further optimization evolution is dramatic: Shape update with the rotationally symmetric sensitivities will converge to a rotationally symmetric optimal shape, whereas, in contrast, applying the unweighted, non-rotational sensitivities of model (ii) the optimization may converge to a local optimum with beads along the G^0-lines, Fig. 8.35. Both solutions are valid global or local optimal results. Note, that for the periodic model both, weighted and non-weighted sensitivities are rotational symmetric and lead to the same, rotationally symmetric optimal shape.

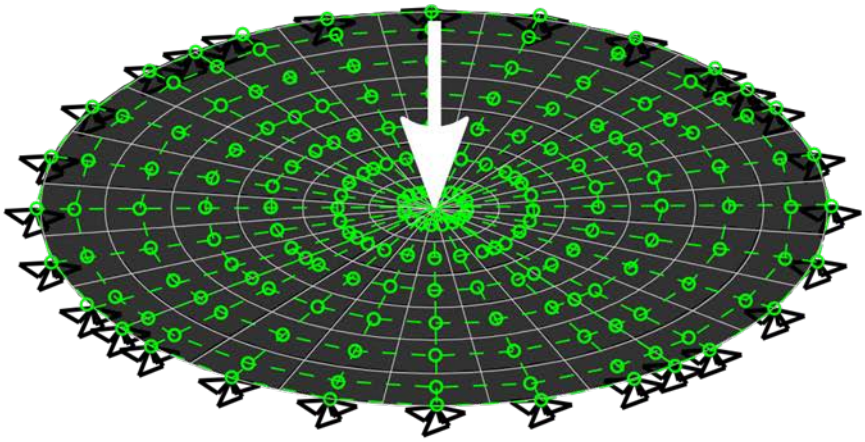
The IGA is very sensitive to the effect of non-weighted sensitivity information because of the eventually great differences of support and influence of the individual shape functions which are related to the design parameters. The same effect occurs for every CAD-based design control technique when there is a big difference of the influences of the design parameters. In engineering practice, there are many approximating modeling decisions to be taken. All of them trigger the optima of the discrete model which have to be critically discussed. Always, local minima are well accepted. Therefore, the weighting of sensitivities might be seen as a pragmatic tool to quickly explore the design space for alternative solutions. The mathematical insight into the basic differences of design derivative and gradient and their discrete equivalents is always very helpful to learn about the problem at hand and to find proper optimal solutions. In principle, numerical optimization must be understood as a tool for design exploration rather than for automation of standard engineering tasks. Shape optimization is dominated by modeling decisions, procedural standardization is impossible.



(a) Periodic model

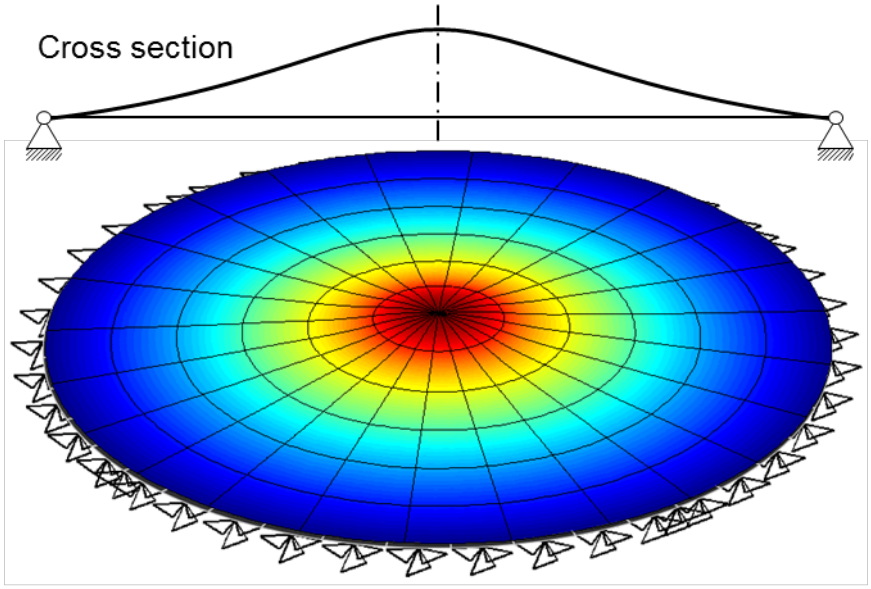


(b) Multipatch model with G1-continuous coupled patches

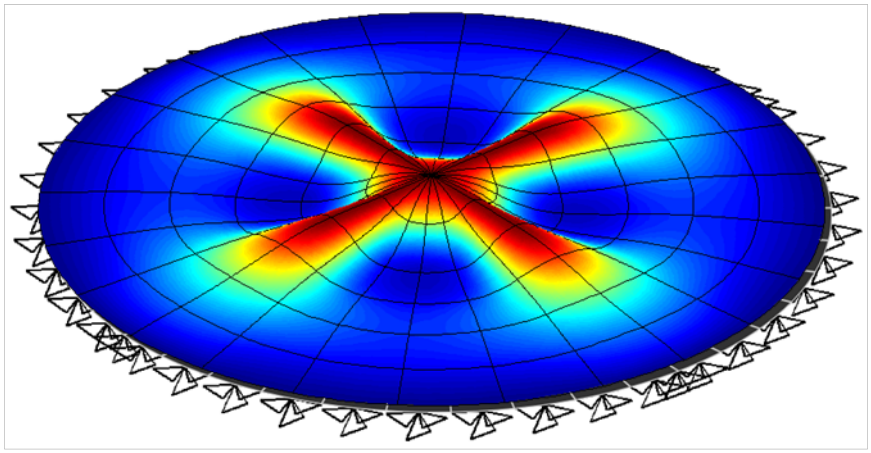


(c) Control nodes for model (b)

Fig. 8.34: IGA shell optimization; Alternative design models.



(a) Global optimum with rotational symmetric sensitivity data; perodic model; multipatch model with weighted sensitivity



(b) Local optimum with non-rotational symmetric sensitivity data, multipatch model with non-weighted sensitivity

Fig. 8.35: IGA shell optimization; Global and local minima.

8.6.4 Shape optimization of a connecting rod - Move directions

Refer to the design model given in Fig. 8.36. Considering symmetry it is the top half of a plane connecting rod made of an elastic metal material. The structure is loaded by a pressure load at the inner ring as shown. Linear elastic plane stress is assumed. There are four design patches of the Coons' type to set up the design model. Topologically rectangular Coons patches are defined by linearly blended individual edge functions. Refer to the four patches and the patch edges at the top of the rod. From left to right there are a linear, two cubic Bézier splines and a circle used as individual edge functions. The five bottom edges are modeled from left to right as linear, linear, circle, circle, linear. All "through-height" edges are chosen as linear functions. G^1-continuity of the outer contour is guaranteed by proper linking of control nodes and a G^1-continuity element between patch 2 and patch 3. Consequently, 4 design parameters remain for the shape optimization of the outer contour of the rod. The first shape parameter controls the thickness of the rod's handle to the left. The parameter is related to a vertical move direction and the three control nodes at the top edge from the left are linked together. Parameter two controls the movement of the tangent continuity element in the inclined direction as indicated in Fig. 8.36. Parameter three controls the rotation of the continuity element by additional relative movement of the first and third node of the continuity element, Fig. 8.36. The radius of the outer circular part to right is controlled by parameter four. Consequently, the top node at the y-axis, together with the Bézier-node to the left are moved vertically and linked to the horizontal movement of the most right control node on the x-axis. The individual shape modes and stress sensitivity coefficients assigned to a unit variation of each parameter are displayed in Fig. 8.37. Volume is chosen as objective function and von Mises stresses are controlled at the nodes of the analysis grid not to exceed the elastic limits. Alternatively, sequential quadratic programming (SQP) and the method of moving asymptotes (MMA) are chosen as optimizers. The optimization converges fast and smoothly for both cases in less than 10 iteration steps, Fig. 8.38. The stress constraints at the flange of the inner ring and at the left support restrict the optimal design.

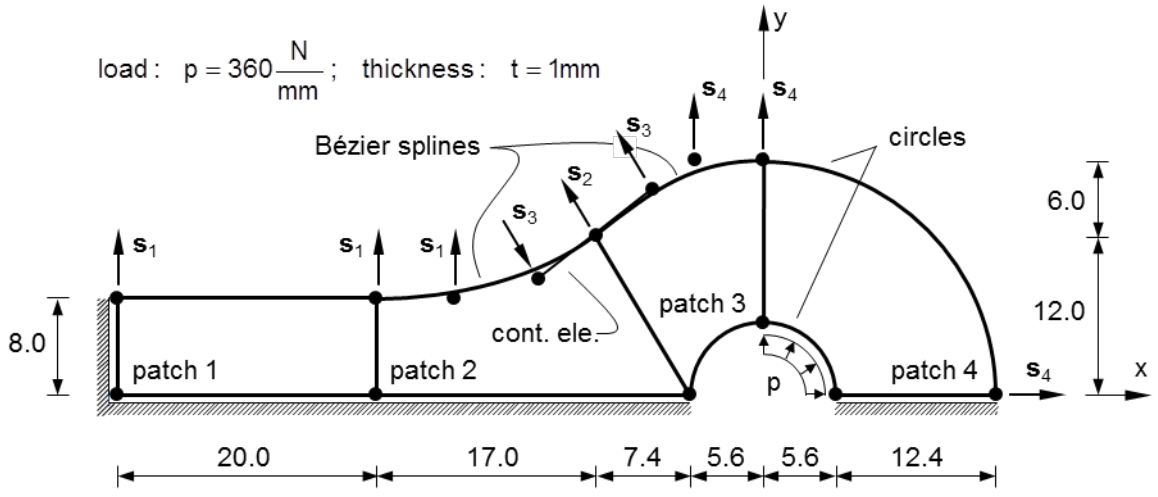


Fig. 8.36: Connecting rod; design model: four linked design parameters.

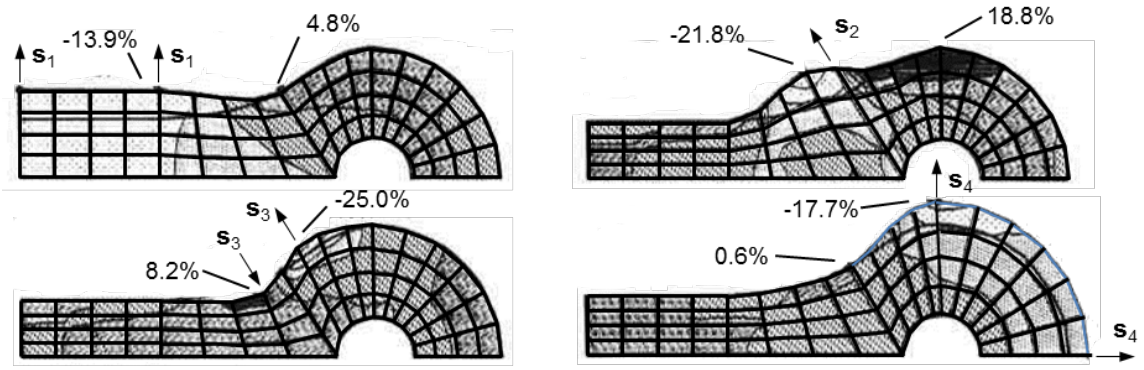


Fig. 8.37: Connecting rod; design model: four linked design parameters. Von Mises stress sensitivity coefficients.

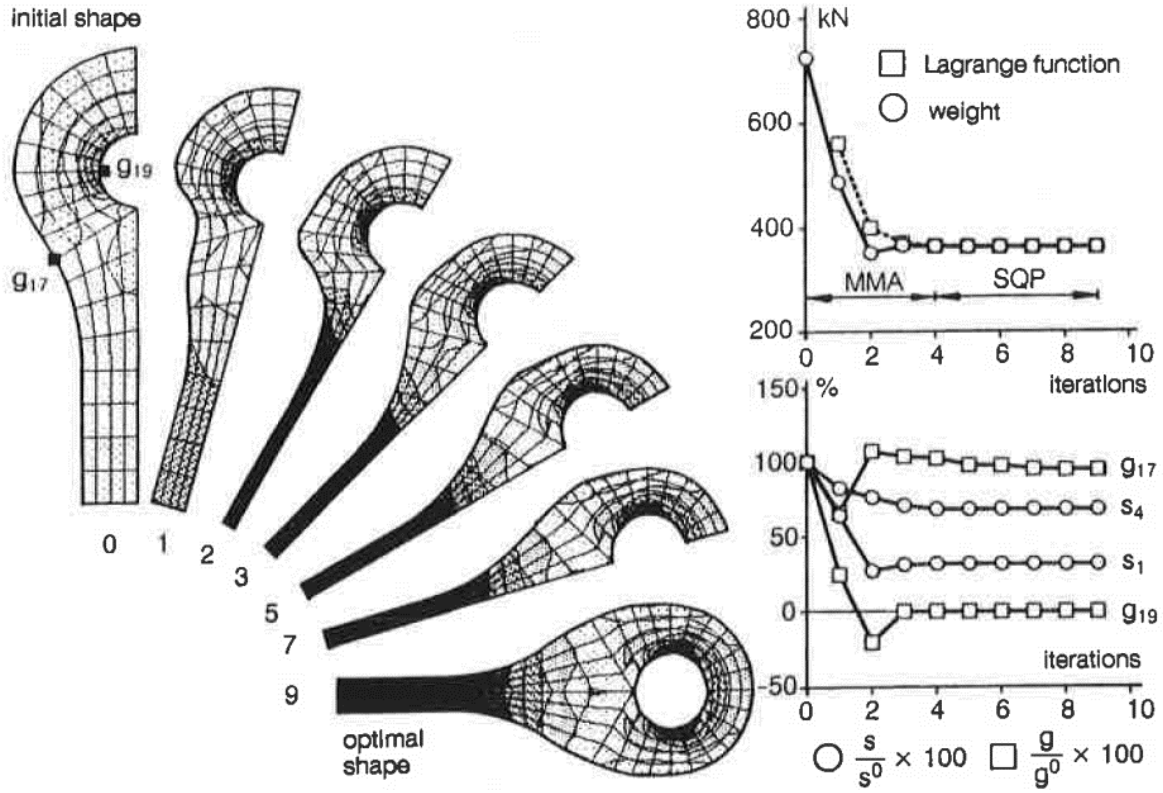


Fig. 8.38: Connecting rod; Iteration history (MMA and SQP)

8.7 References

[1] S. Adriaenssens, P. Block, D. Veenendaal, C. Williams, Shell structures for architecture: Form finding and optimization, 2014.

[2] I. Mungan, J.F. Abel, Fifty Years of Progress for Shell and Spatial Structures - In celebration of the 50th Anniversary Jubilee of the IASS (1959-2009), in, International Association for Shell and Spatial Structures (IASS), Madrid, Spain, 2011, pp. 500.

[3] F. Otto, Zugbeanspruchte Konstruktionen, Band 1 + 2, Ullstein Verlag,, Frankfurt, Berlin, 1962.

[4] K. Linkwitz, H.J. Schek, Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen, Ingenieurarchiv, 40 (1971) 145-158.

[5] E. Haug, G.H. Powell, Finite element analysis of non-linear membrane structures, in: Proc. IASS Pacific Symposium – Part II on Tension Structures and Space Frame, Tokyo and Kyoto, 1971.

[6] J.H. Argyris, T. Angelopoulos, B. Bichat, A general method for the shape finding of lightweight tension structures, Computer Methods in Applied Mechanics and Engineering, 3 (1974) 135-149.

[7] O.C. Zienkiewicz, J.S. Campbell, Optimum structural design, in: R.H. Gallagher, O.C. Zienkiewicz (Eds.) Shape Optimization and Sequential Linear Programming, Wiley, London, 1973.

[8] M.H. Imam, THREE-DIMENSIONAL SHAPE OPTIMIZATION, International Journal for Numerical Methods in Engineering, 18 (1982) 661-673.

[9] V. Braibant, C. Fleury, Shape optimal design using B-splines, Computer Methods in Applied Mechanics and Engineering, 44 (1984) 247-267.

[10] K.-U. Bletzinger, S. Kimmich, E. Ramm, Efficient modeling in shape optimal design, Computing Systems in Engineering, 2 (1991) 483-495.

[11] J.A. Cottrell, T.J. Hughes, Y. Bazilevs, Isogeometric analysis: toward integration of CAD and FEA, John Wiley & Sons, 2009.

[12] Z.P. Wang, S. Turteltaub, Isogeometric shape optimization for quasi-static processes, International Journal for Numerical Methods in Engineering, 104 (2015) 347-371.

[13] D. Fußeder, B. Simeon, A.V. Vuong, Fundamental aspects of shape optimization in the context of isogeometric analysis, Computer Methods in Applied Mechanics and Engineering, 286 (2015) 313-331.

[14] J. Kiendl, R. Schmidt, R. Wüchner, K.U. Bletzinger, Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting, Computer Methods in Applied Mechanics and Engineering, 274 (2014) 148-167.

[15] A.P. Nagy, Isogeometric Design Optimisation. Technische Universiteit Delft, Delft, The Netherlands, 2011.

[16] W.A. Wall, M.A. Frenzel, C. Cyron, Isogeometric structural shape optimization, Computer Methods in Applied Mechanics and Engineering, 197 (2008) 2976-2988.

[17] H. Azegami, A solution to domain optimization problems (in Japanese), Transactions of the Japan Society of Mechanical Engineers, Part A, 60 (1994) 1479-1486.

[18] E.J. Haug, K.K. Choi, V. Komkov, Design Sensitivity Analysis of Structural Systems, Academic Press, 1986.

[19] C. Le, T. Bruns, D. Tortorelli, A gradient-based, parameter-free approach to shape optimization, Computer Methods in Applied Mechanics and Engineering, 200 (2011) 985-996.

[20] M. Hojjat, E. Stavropoulou, K.-U. Bletzinger, The vertex morphing method for node-based shape optimization, Computer Methods in Applied Mechanics and Engineering, 268 (2014) 494-513.

[21] O. Sigmund, Morphology-based black and white filters for topology optimization, Structural and Multidisciplinary Optimization, 33 (2007) 401-424.

[22] O. Pironneau, Optimal Shape Design for Elliptic Systems, Springer, Berlin, Heidelberg, 1984.

[23] B. Mohammadi, O. Pironneau, Shape optimization in fluid mechanics, Annual Review of Fluid Mechanics, 36 (2004) 255–279.

[24] B. Mohammadi, O. Pironneau, Applied Shape Optimization for Fluids, Oxford University Press, 2010.

[25] A. Jameson, Optimum aerodynamic design using control theory, Computational Fluid Dynamics Review, 3 (1995) 495–528.

[26] A. Jameson, J.C. Vassberg, Studies of alternative numerical optimization methods applied to the brachistochrone problem, Computational Fluid Dynamics, 9 (2000) 281–296.

[27] A. Jameson, Aerodynamic shape optimization using the adjoint method, in: B. Von Karman Institute (Ed.) Lecture at the Von Karman Institute, Brussels, Belgium, 2003.

[28] A. Stück, T. Rung, Adjoint RANS with filtered shape derivatives for hydrodynamic optimisation, Computers and Fluids, 47 (2011) 22-32.

[29] J. Sokolowski, J.-P. Zolésio, Introduction to shape optimization: Shape sensitivity analysis. , Springer, Berlin, Heidelberg, 1992.

[30] R.M. Lewis, Numerical computation of sensitivities and the adjoint approach, in: J.e.a. Boorggaard (Ed.) Computational Methods for Optimal Design and Control, Birkhäuser Boston, 1998, pp. 285-302.

[31] D.I. Papadimitriou, K.C. Giannakoglou, A continuous adjoint method with objective function derivatives based on boundary integrals, for inviscid and viscous flows, Computers & Fluids, 36 (2007) 325-341.
[32] C. Othmer, A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows, International Journal for Numerical Methods in Fluids, 58 (2008) 861-877.
[33] Z. Lyu, J.R.R.A. Martins, Aerodynamic shape optimization of an adaptive morphing trailing-edge wing, Journal of Aircraft, 52 (2015) 1951-1970.
[34] J. Hadamard, Mémoire sur le probleme d'analyse relatif a l'équilibre des plaques élastiques, in: Mémoire des savants étrangers, 33, 1907, -Œuvres de Jacques Hadamard, pp. 515-641.
[35] H. Azegami, K. Takeuchi, A smoothing method for shape optimization: Traction method using the Robin condition, International Journal of Computational Methods, 3 (2006) 21-33.
[36] E. Catmull, J. Clark, RECURSIVELY GENERATED B-SPLINE SURFACES ON ARBITRARY TOPOLOGICAL MESHES, CAD Computer Aided Design, 10 (1978) 350-355.
[37] L. Kobbelt, A variational approach to subdivision, Computer Aided Geometric Design, 13 (1996) 743-761.
[38] K.-U. Bletzinger, A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape, Structural and Multidisciplinary Optimization, 49 (2014) 873-895.
[39] V. Braibant, C. Fleury, Shape optimal design-A CAD-oriented formulation, Engineering with Computers, 1 (1986) 193-204.
[40] K.-U. Bletzinger, E. Ramm, A general finite element approach to the form finding of tensile structures by the updated reference strategy, International Journal of Space Structures, 14 (1999) 131-145.
[41] E. Stavropoulou, M. Hojjat, K.U. Bletzinger, In-plane mesh regularization for node-based shape optimization problems, Computer Methods in Applied Mechanics and Engineering, 275 (2014) 39-54.
[42] M. Firl, R. Wüchner, K.-U. Bletzinger, Regularization of shape optimization problems using FE-based parametrization, Structural and Multidisciplinary Optimization, 47 (2013) 507-521.
[43] K.-U. Bletzinger, M. Firl, J. Linhard, R. Wüchner, Optimal shapes of mechanically motivated surfaces, Computer methods in applied mechanics and engineering, 199 (2010) 324-333.
[44] K.-U. Bletzinger, M. Firl, F. Daoud, Approximation of derivatives in semi-analytical structural optimization, Computers & Structures, 86 (2008) 1404-1416.
[45] H. Eschenauer, P.U. Post, M. Bremicker, Application of the optimization procedure SAPOP on the layout of structural components, Bauingenieur Berlin, 63 (1988) 515-526.
[46] C. De Boor, A Practical Guide to Splines, 3 ed., Springer, New York, 1985.
[47] L. Piegl, W. Tiller, The NURBS book, 2 ed., Springer, Berlin, 1997.
[48] F. van Keulen, R.T. Haftka, N.H. Kim, Review of options for structural design sensitivity analysis. Part 1: Linear systems, Computer Methods in Applied Mechanics and Engineering, 194 (2005) 3213-3243.
[49] N. Olhoff, J. Rasmussen, E. Lund, A Method of “Exact” Numerical differentiation for error elimination in finite-element-based semi-analytical shape sensitivity analyses*, Mechanics of Structures and Machines, 21 (1993) 1-66.
[50] F. Van Keulen, H. De Boer, Rigorous improvement of semi-analytical design sensitivities by exact differentiation of rigid body motions, International Journal for Numerical Methods in Engineering, 42 (1998) 71-91.

8.8 Further references related to shape optimal design

Arnout S, Firl M, Bletzinger K-U (2012) Parameter free shape and thickness optimization considering stress Response. Structural and Multidisciplinary Optimization 45:801-814.

Bletzinger K-U, Kimmich S, Ramm E (1991) Efficient modeling in shape optimal design. Computing Systems in Engineering 2:483-495.

Bletzinger K-U, Wüchner R, Daoud F, Camprubí N (2005) Computational methods for form finding and optimization of shells and membranes. Computer Methods in Applied Mechanics and Engineering 194:3438-3452.

Bletzinger K-U, Firl M, Daoud F (2008) Approximation of derivatives in semi-analytical structural optimization. Computers and Structures 86:1404-1416.

Bletzinger K-U, Firl M, Linhard J, R. Wüchner (2010) Optimal shapes of mechanically motivated surfaces. Computer Methods in Applied Mechanics and Engineering 199:324-333.

Bletzinger K-U (2014) A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape, Structural and Multidisciplinary Optimization 49:873-895.

de Boer A, van der Schoot M, Bijl H (2007) Mesh deformation based on radial basis function interpolation. Computers and Structures 85:784–795.

Bourdin B (2001) Filters in topology optimization. International Journal for Numerical Methods in Engineering 50(9):2143–2158

Braibant V, Fleury C (1986) Shape optimal design: A CAD-oriented formulation. Engineering with Computers 1(4):193-204.

Bruns TE, Tortorelli DA (2001) Topology optimization of nonlinear elastic structures and compliant mechanisms. Computer Methods in Applied Mechanics and Engineering 190(26-27):3443–3459

Cho S, Ha SH (2009) Isogeometric shape design optimization: exact geometry and enhanced sensitivity. Structural and Multidisciplinary Optimization 38:53–70.

Clausen P, Pedersen C (2006) Non-parametric Large Scale Structural Optimization for Industrial Applications. Proc. ECCM-2006, Lisbon, Portugal.

Firl M, Bletzinger K-U (2012) Shape optimization of thin walled structures governed by geometrically nonlinear mechanics. Computer Methods in Applied Mechanics and Engineering 237:107-117.

Firl M, Wüchner R, Bletzinger K-U (2013) Regularization of shape optimization problems using FE-based Parameterization. Structural and Multidisciplinary Optimization 47:507-521.

Ha SH, Choi KK, Cho S (2010) Numerical method for shape optimization using T-spline based isogeometric method. Structural and Multidisciplinary Optimization 42:417–428.

Hojjat M, Stavropoulou E, Bletzinger K-U (2014) The vertex morphing method for node-based shape optimization. Computer Methods in Applied Mechanics and Engineering 268:494-513.

Kiendl J (2011) Isogeometric Analysis and Shape Optimal Design of Shell Sructures. PhD thesis, Technische Universtität München.

Kiendl J, Schmidt R, Wüchner R, Bletzinger K-U (2014) Isogeometric Shape Optimization of Shell Structures. Computer Methods in Applied Mechanics and Engineering, 274:148-167.

Kim NH, Choi KK, Botkin ME (2002) Numerical method for shape optimization using meshfree method. Structural and Multidisciplinary Optimization 24:418-429.

Litke N, Levin A, Schröder P (2001) Trimming for subdivision surfaces. Computer Aided Geometric Design 18:463-481.

Seo YD, Kim HJ, Youn SK (2010) Shape optimization and its extension to topological design based on isogeometric analysis. International Journal of Solids and Structures 47:1618-1640.

Scherer M, Denzer R, Steinmann P (2010) A fictitious energy approach for shape optimization. International Journal for Numerical Methods in Engineering 82:269-302.

Schmidt R (2013) Trimming, Mapping and optimization in Isogeometric Analysis of Shell Structures. PhD thesis, Technische Universtität München.

Sigmund O, Maute K (2012) Sensitivity filtering from a continuum mechanics perspective. Structural and Multidisciplinary Optimization 46:471-475.

Sokolowski J, Zolésio J.-P. (1992) Introduction to shape optimization: shape sensitivity analysis. Springer.

Stavropoulou E, Hojjat M, Bletzinger K-U (2014) In-plane mesh regularization for parameter-free shape optimization problems. Computer Methods in Applied Mechanics and Engineering, 275:39-54.

Zorin D, Schröder P, Sweldens W (1996) Interpolating subdivision for meshes with arbitrary topology. In: Computer Graphics (SIGGRAPH ’96 Proceedings), 189–192.

9 Topology Optimization

This chapter will be supported by selected papers.

10 Form Finding of Membrane Structures

This chapter will be supported by selected papers.

11 References

All of the following references are available at the library of Technische Universität München.

Engineering related:

[1] Arora J. *Introduction to optimum design*. Amsterdam, Elsevier/Academic Press, 2004.
[2] Arora J. *Optimization of structural and mechanical systems*. New Jersey, World Scientific, 2007.
[3] Baier H, Seßelberg C, Specht B. *Optimierung in der Strukturmechanik*. Braunschweig, Veweg, 1994.
[4] Bendsøe M, Sigmund O. *Topology optimization, theory methods and applications*. Berlin, Springer, 2004.
[5] Bletzinger K.-U. *Formoptimierung von Flächentragwerken*. Dissertation, Universität Stuttgart, 1990.
[6] Eschenauer, H (Ed.). *Multicriteria design optimization – procedures and applications*. Berlin, Springer, 1990.
[7] Haftka R, Gürdal Z. *Elements of structural optimization*. Dordrecht, Kluwer, 1992.
[8] Harzheim L. *Strukturoptimierung – Grundlagen und Anwendungen*. Frankfurt a. M., Deutsch, 2008.
[9] Hörnlein, H (Ed.). *Software systems for structural optimization*. Basel, Birkhäuser, 1993.
[10] Hernández S, Fonán AN. *Practical applications of design optimization*. Southampton, WIT Press, 2002.
[11] Kirsch, U. *Structural optimization – fundamentals and applications*. Berlin (u.a.), Springer, 1993.
[12] Klein B. *Leichtbau-Konstruktion*. Wiesbaden, Vieweg & Teubner, 2009.
[13] Marti K, Gröger D. *Stochastische Strukturoptimierung von Stab- und Balkentragwerken*. Berlin, Springer, 2006.
[14] Schumacher A. *Optimierung mechanischer Strukturen – Grundlagen und industrielle Anwendungen*. Berlin, Springer, 2013.
[15] Spiller WR, McBain KM. *Structural optimization*. Dordrecht, Springer, 2009.
[16] Wiedemann J. *Leichtbau*. Berlin, Springer, 2007.

Mathematics related:

[1] Alt, W. Nichtlineare Optimierung – eine Einführung in Theorie, Verfahren und Anwendungen. Braunschweig, Vieweg, 2011.
[2] Borgwardt, KH. *Aufgabensammlung und Klausurtrainer zur Optimierung für die Bachelorausbildung in mathe-matischen Studiengängen*. Wiesbaden, Vieweg & Teubner, 2010.
[3] Burkard R, Zimmermann U. *Einführung in die mathematische Optimierung*. Berlin, Springer, 2012.
[4] Haslinger J, Mäkinen R. Introduction to shape optimization – theory, approximation and computation. Philadelphia, SIAM, 2003.
[5] Luenberger, D, Ye Y. *Linear and nonlinear programming*. New York, Springer, 2008.
[6] Ulbrich M, Ulbrich S. *Nichtlineare Optimierung*. Basel, Birkhäuser, 2012.

Selection of publications related to optimization and form-finding by the Chair of Structural Analysis (Lehrstuhl für Statik):

[1] K.-U. Bletzinger, Extended method of moving asymptotes based on second-order information, Structural Optimization, 5 (1993) 175-183.
[2] K.-U. Bletzinger, E. Ramm, Form finding of shells by structural optimization, Engineering with Computers, 9 (1993) 27-35.
[3] E. Ramm, K.-U. Bletzinger, R. Reitinger, Shape optimization of shell structures, Journal of the International Association for Shell and Spatial Structures, 34 (1993) 103-121.
[4] E. Ramm, K.-U. Bletzinger, R. Reitinger, K. Maute, Challenge of structural optimization, in: International Conference on Computational Structures Technology - Proceedings, 1994, pp. 27-52.
[5] R. Reitinger, K.-U. Bletzinger, E. Ramm, Shape optimization of buckling sensitive structures, Computing Systems in Engineering, 5 (1994) 65-75.
[6] K.-U. Bletzinger, K. Maute, Towards generalized shape and topology optimization, Engineering Optimization, 29 (1997) 201-216.
[7] K.-U. Bletzinger, Novel form optimization concepts of shell and membrane structures, in: European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2000, 2000.
[8] K.-U. Bletzinger, E. Ramm, Structural optimization and form finding of light weight structures, Computers and Structures, 79 (2001) 2053-2062.

[9] N. Camprubí, M. Bischoff, K.-U. Bletzinger, Shape optimization of shells and locking, Computers and Structures, 82 (2004) 2551-2561.
[10] N. Camprubí, F. Daoud, K.-U. Bletzinger, Design smoothing in shape optimization of shells, in: ECCOMAS 2004 - European Congress on Computational Methods in Applied Sciences and Engineering, 2004.
[11] M. Pagitz, K.-U. Bletzinger, Shape optimization of a bow, Structural and Multidisciplinary Optimization, 28 (2004) 73-76.
[12] K.-U. Bletzinger, R. Wüchner, F. Daoud, N. Camprubí, Computational methods for form finding and optimization of shells and membranes, Computer Methods in Applied Mechanics and Engineering, 194 (2005) 3438-3452.
[13] F. Daoud, N. Camprubí, K.-U. Bletzinger, Filtering and regularization techniques in shape optimization with CAD-free parametrization, in: 3rd M.I.T. Conference on Computational Fluid and Solid Mechanics, 2005, pp. 1231-1235.
[14] F. Daoud, F. Jurecka, K.-U. Bletzinger, Filtering and regularization shape optimization techniques for preliminary design, in: Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 2005, pp. 5111-5118.
[15] K.-U. Bletzinger, M. Firl, F. Daoud, Approximation of derivatives in semi-analytical structural optimization, Computers and Structures, 86 (2008) 1404-1416.
[16] M. Fischer, M. Firl, H. Masching, K.-U. Bletzinger, Optimization of nonlinear structures based on object-oriented parallel programming, in: Proceedings of the 7th International Conference on Engineering Computational Technology, 2010.
[17] M. Hojjat, E. Stavropoulou, T. Gallinger, U. Israel, R. Wüchner, K.-U. Bletzinger, Fluid-structure interaction in the context of shape optimization and computational wind engineering, in: Lecture Notes in Computational Science and Engineering, 2010, pp. 351-381.
[18] F. Dieringer, R. WüChner, K.-U. Bletzinger, Advanced cutting pattern generation - Consideration of structural requirements in the optimization process, in: Structural Membranes 2011 - 5th International Conference on Textile Composites and Inflatable Structures, 2011, pp. 84-91.
[19] H. Masching, M. Fischer, M. Firl, K.-U. Bletzinger, Finite element based structural optimization using object-oriented parallel programming, Civil-Comp Proceedings, 95 (2011).
[20] S. Arnout, M. Firl, K.-U. Bletzinger, Parameter free shape and thickness optimisation considering stress response, Structural and Multidisciplinary Optimization, 45 (2012) 801-814.
[21] M. Firl, K.-U. Bletzinger, Shape optimization of thin walled structures governed by geometrically nonlinear mechanics, Computer Methods in Applied Mechanics and Engineering, 237-240 (2012) 107-117.
[22] M. Fischer, H. Masching, K.-U. Bletzinger, Efficient design of large lightweight composite structures: A parameter free optimization approach, in: 12th AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2012.
[23] M. Fischer, H. Masching, M. Firl, K.-U. Bletzinger, Design of lightweight composite structures: A parameter free structural optimization approach, in: Key Engineering Materials, 2012, pp. 1391-1396.
[24] J. Degroote, M. Hojjat, E. Stavropoulou, R. Wüchner, K.-U. Bletzinger, Partitioned solution of an unsteady adjoint for strongly coupled fluid-structure interactions and application to parameter identification of a one-dimensional problem, Structural and Multidisciplinary Optimization, 47 (2013) 77-94.
[25] M. Firl, R. Wüchner, K.-U. Bletzinger, Regularization of shape optimization problems using FE-based parametrization, Structural and Multidisciplinary Optimization, 47 (2013) 507-521.
[26] B. Yang, K.-U. Bletzinger, Q. Zhang, Z. Zhou, Frame structural sizing and topological optimization via a parallel implementation of a modified particle Swarm algorithm, KSCE Journal of Civil Engineering, 17 (2013) 1359-1370.
[27] K.-U. Bletzinger, A consistent frame for sensitivity filtering and the vertex assigned morphing of optimal shape, Structural and Multidisciplinary Optimization, 49 (2014) 873-895.
[28] M. Hojjat, E. Stavropoulou, K.-U. Bletzinger, The Vertex Morphing method for node-based shape optimization, Computer Methods in Applied Mechanics and Engineering, 268 (2014) 494-513.
[29] J. Kiendl, R. Schmidt, R. Wüchner, K.-U. Bletzinger, Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting, Computer Methods in Applied Mechanics and Engineering, 274 (2014) 148-167.
[30] D. Markus, A. Kelder, R. Wüchner, K.-U. Bletzinger, Lift force optimization of gravity base offshore foundations using CFD, in: Proceedings of the International Offshore and Polar Engineering Conference, 2014, pp. 624-631.
[31] E. Stavropoulou, M. Hojjat, K.-U. Bletzinger, In-plane mesh regularization for node-based shape optimization problems, Computer Methods in Applied Mechanics and Engineering, 275 (2014) 39-54.

Dissertations related to optimization and form-finding by the Chair of Structural Analysis (Lehrstuhl für Statik):

[1] Kai-Uwe Bletzinger, *Formoptimierung von Flächentragwerken*, Universität Stuttgart, 1990.
[2] Natalia Camprubí, *Design and Analysis in Shape Optimization of Shells*, Technische Universität München, 2004.
[3] Fernaß Daoud, *Formoptimierung von Freiformschalen - Mathematische Algorithmen und Filtertechniken*, Technische Universität München, 2005.
[4] Alexander Hörmann, *Ermittlung optimierter Stabwerkmodelle auf Basis des Kraftflusses als Anwendung plattformunabhängiger Prozesskopplung*, Technische Universität München, 2006.
[5] Roland Wüchner, *Mechanik und Numerik der Formfindung und Fluid-Struktur-Interaktion von Membrantragwerken*, Technische Universität München, 2006.
[6] Florian Jurecka, *Robust Design Optimization Based on Metamodeling Techniques*, Technische Universität München, 2007.
[7] Johannes Linhard, *Numerisch-mechanische Betrachtung des Entwurfsprozesses von Membrantragwerken*, Technische Universität München, 2009.
[8] Bin Yang, Modified Particle Swarm Optimizers and their Application to Robust Design and Structural Optimization, Technische Universität München, 2009.
[9] Matthias Firl, *Optimal Shape Design of Shell Structures*, Technische Universität München, 2010.
[10] Josef Kiendl, *Isogeometric Analysis and Shape Optimal Design of Shell Structures*, Technische Universität München, 2011.
[11] Robert Schmidt, *Trimming, Mapping, and Optimization in Isogeometric Analysis of Shell Structures*, Technische Universität München, 2013.
[12] Michael Fischer, *Finite Element Based Simulation, Design and Control of Piezoelectric and Lightweight Smart Structures*, Technische Universität München, 2013.
[13] Falko Dieringer, *Numerical methods for the design and analysis of tensile structures*, Technische Universität München, 2014.
[14] Ute Israel, *Optimierung in der Fluid-Struktur-Interaktion: Sensitivitätsanalyse für die Formoptimierung auf der Grundlage des partitionierten Verfahrens*, Technische Universität München, 2014.
[15] Electra Stavropoulou, *Sensitivity analysis and regularization for the shape optimization of coupled problems*, Technische Universität München, 2014.
[16] Majid Hojjat, *Node-based parametrization for shape optimal design*, Technische Universität München, 2014.
[17] Daniel Markus, *Numerical and experimental modeling for shape optimization of offshore structures*, Technische Universität München, 2014.