

Optimisation of Reinforced Concrete Structures

Literature study

Guido Slobbe

Technische Universiteit Delft

OPTIMISATION OF REINFORCED CONCRETE STRUCTURES

LITERATURE STUDY

by

Guido Slobbe

in partial fulfilment of the requirements for the degree of

Master of Science

in Civil Engineering

at the Delft University of Technology,

Supervisor:	Prof. ir. R. Nijse	TU Delft
Thesis committee:	Dr. ir. drs. R. Braam,	TU Delft
	Dr. ir. J. L. Coenders,	TU Delft
	Ir. H. G. Krijgsman,	ABT

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

CONTENTS

Acronyms	v
1 Introduction	1
2 Structural optimisation	3
2.1 A short history of optimisation in general	4
2.2 What is optimisation?	5
2.3 Structural optimisation terminology	9
2.4 Structural optimisation classifications	12
2.5 Basic types of optimisation algorithms	14
3 Optimisation Algorithms	19
3.1 Brute force algorithms	20
3.2 Simulated Annealing	20
3.3 Cyclic coordinate search	21
3.4 Nelder-Mead simplex method	22
3.5 Genetic Algorithm (GA)	23
3.6 Particle Swarm Optimisation (PSO)	25
3.7 Ant Colony Optimisation (ACO)	31
3.8 Gradient based first order methods	32
3.9 Gradient (Hessian) based second order methods	33
3.10 Comparing algorithms	33
4 Object Function	35
4.1 Types of solution spaces	35
4.2 Score function	37
4.3 Penalty functions and constraints	38
5 Geometry Design	43
5.1 Method 1: Triangulation	44
5.2 Method 2: Building blocks (Lego's)	45

6	Reinforced Concrete Costs Analysis	49
6.1	Process	50
6.2	Concrete costs	51
6.3	Reinforcement costs	52
6.4	Formwork costs	54
6.5	Costs analysis model	57
7	Reinforced Concrete Structural Analysis	59
7.1	Design of reinforced concrete elements	60
7.2	Structural behaviour of elements	62
7.3	Detailing	64
7.4	Structural design methods	67
7.5	Comparing methods	73
	Bibliography	75
	List of Figures	77
	List of Tables	81
A	Examples	83
A.1	Example: Reinforcement optimisation with Grasshopper and GSA	83
A.2	Example: Behaviour of Particle Swarm Optimisation (PSO)	84

ACRONYMS

ACO	Ant Colony Optimisation	45
EC2	Eurocode 2: Design of concrete structures, NEN-EN 1992-1-1 [37]	61
EVO	Evolutionary Optimisation	23
FEM	Finite Element Method	14
GA	Genetic Algorithm.....	4
PSO	Particle Swarm Optimisation	84
SLS	Serviceability Limit State	60
SA	Simulated Annealing.....	77
STM	Strut-and-Tie Model	83
ULS	Ultimate Limit State	60

1

INTRODUCTION

This Master's thesis consists of two reports: the main report and a literature study. The main report focusses on the main research topic: the development of a structural optimisation methodology for reinforced concrete elements. This report, the literature study, contains the background and some alternatives on the discussed topics in the main report.

Performing structural optimisation on reinforced concrete requires research in both fields. The field of structural optimisation is wide, with applications in multiple disciplines of engineering and with lots of literature available. Most literature on this topic originates from the period 1990 - 2014. This can be traced back to the development of computational power, which is required to solve optimisation problems. The field of reinforced concrete engineering is rather different, but the origin and development of reinforced concrete has a surprisingly optimisation related goal. Reinforced concrete was developed by a French gardener for the purpose of strengthening his pottery. Further developments of the material are usually performed with goals like "increase strength" or "reduce costs". These goals can be directly linked to optimisations of some sort.

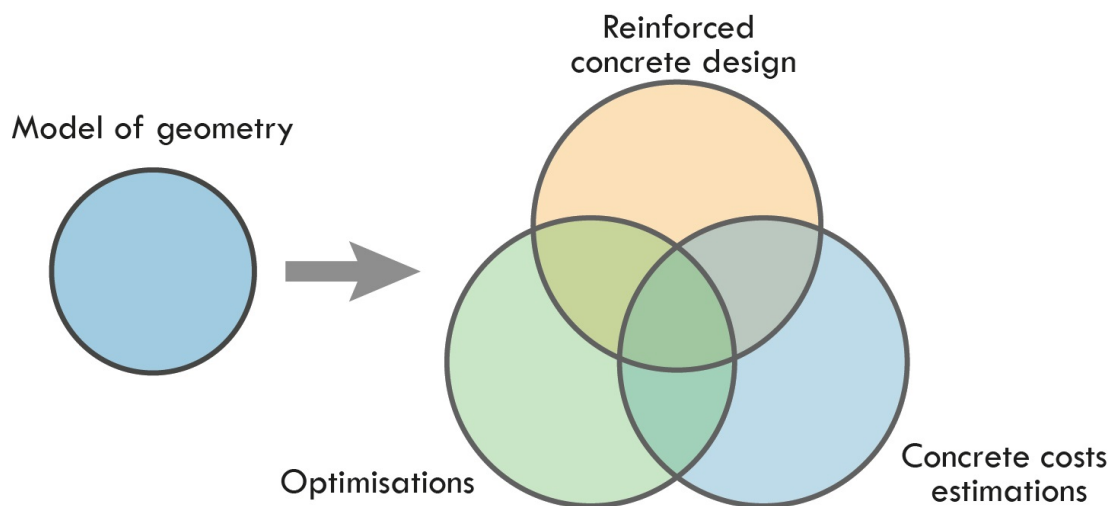


Figure 1.1: Major subjects in discussed in this literature study: modelling of structural (geometry), (structural) optimisation, costs estimations for reinforced concrete and structural engineering for reinforced concrete. The model, or simplification of the structure is required in all other subjects.

This literature study focusses on 4 topics: modelling of structures, structural optimisation, costs estimations and structural analysis of reinforced concrete. The second (optimisation), third (costs) and fourth (structure) subject follow logically from the research goal. Figure 1.1 displays the how these are linked together. The first subject, the model of the geometry, is required to model (or simplify) structural properties. In this Master's

Thesis, a truss model is used to estimate structural behaviour (including reinforcement), to create design variables for the optimisation process and partially to estimate costs.

In order to explain the field of structural optimisation, information is given on three topics. The first topic explains optimisation in general and the link between optimisation and structural optimisation. The second topic analyses and compares different optimisation algorithms and the third topic gives some background on object functions. Information on these three topics is required to be able to understand and select a good combination of techniques.

Structural optimisation requires information on the geometry of structures. The developed methodology works with truss based structures to model this geometry. Standardised truss "blocks" are used to represent "random" 3D structures in the analysis. There are several possible trussing methodologies which result in useful solutions. A system can be chosen based on certain requirements, like truss behaviour. Some knowledge on how trusses are used in the process is required to design standardised trusses to model geometry. Inaccurate models can result in inaccurate and/ or unreliable optimisations.

The objective of the developed methodology is to optimise the costs of a reinforced concrete structure. For this reason, information is required on the costs analysis of reinforced concrete. The goal is to develop a function to estimate the costs of options in the object function, based on the geometry model.

The final part of the literature study is on how to perform (a standardised) structural analysis and verification of a 3D reinforced concrete structure. The chapter on reinforced concrete engineering analyses several aspects and techniques in order to select a useful methodology for the verification of the safety of structures. To allow methodology to be applied in practice, the structural analysis has to fulfil the requirements set in Eurocode 2: Design of concrete structures, NEN-EN 1992-1-1 [37] (EC2).

The goal of the literature study is to research different techniques to optimise and analyse reinforced concrete engineering problems. The main requirement of this Master's Thesis is to create a structural optimisation methodology for reinforced concrete with costs minimisation as objective. Based on the subjects discussed in the literature study, knowledge is gathered on how to select options and what option to select.

2

STRUCTURAL OPTIMISATION

2.1	A short history of optimisation in general	4
2.2	What is optimisation?	5
2.2.1	Mathematical description of the structural optimisation problem	5
2.2.2	Common difficulties	7
2.3	Structural optimisation terminology	9
2.3.1	Design variable terminology	9
2.3.2	Optimisation algorithm terminology	10
2.3.3	Object function terminology	10
2.3.4	output terminology	11
2.4	Structural optimisation classifications	12
2.4.1	Sizing optimisation [2, 10]	13
2.4.2	Shape optimisation [2, 10]	13
2.4.3	Topology optimisation [2, 10]	13
2.4.4	Multi-material topology optimisation [7, 17]	13
2.5	Basic types of optimisation algorithms	14
2.5.1	Calculus based optimisation	15
2.5.2	Deterministic methods	15
2.5.3	Stochastic methods	15
2.5.4	Direct search methods	16
2.5.5	Nature inspired algorithms	16
2.5.6	Gradient based methods	17

The goal of this chapter is to study the background of optimisation and its structural application in particular. The content discusses some historical background, the basic principles and some possibilities of structural optimisation. Chapter 3 continues on the subject with a study on a number of optimisation algorithms. Appendix A contains a set of simple examples used by the author to improve understanding of the topic and it's applications.

Relevant questions in relation to the research questions which are discussed in this chapter are:

- How can a structural optimisation problem be expressed mathematically (Section 2.2.1)?
- What input information is required to be able to structurally optimise a reinforced concrete structure in terms of costs?
- What are possible processing strategies for optimisations (Section 2.4)?

- What are advantages and disadvantages of different optimisation techniques?

According to the dictionary [1], the definition of optimisation is: "To find the best compromise among several often conflicting requirements, as in engineering design". This definition outlines two pieces of knowledge. First, optimisation is generally considered as "The search for the best possible solution" and second: "The definition of the optimal solution is often based on a complex mix of requirements.". Usually, the optimal solution is defined in terms of: "The option x that results in the minimum (or maximum) score $f(x)$ while requirement(s) $g(x)$ is/ are satisfied."

2.1. A SHORT HISTORY OF OPTIMISATION IN GENERAL

The first known stories of optimisation date back as far as ancient Greece. In this era, mathematicians like Euclid, Zenodorus and Heron [2] attempted to solve mostly geometrical problems. Euclid, who lived as early as 300 BC, proved for example that a square is the rectangle with the greatest area given a total length of edges. These kind of optimisation problems found their use mostly in cartographic applications.

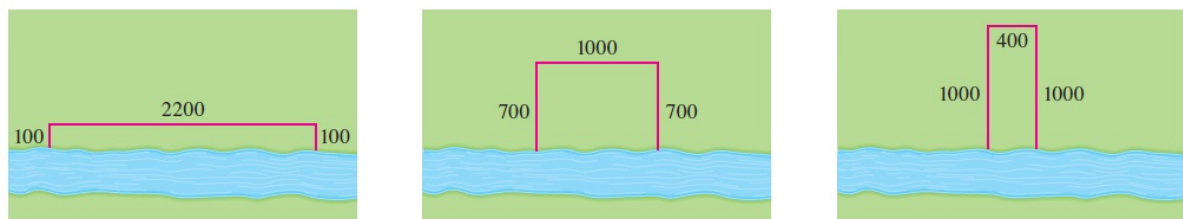


Figure 2.1: A classical rectangular (farm) area maximisation problem. "A farmer has n meters of fence, what is the largest rectangular area A that can be enclosed with this fence? Since all fence-lengths in this figure are equally long, it is simple to find that the centre figure has the largest surface. This problem based on and courtesy of Calculus, Early transcendentals [3].

Modern optimisation methods have their origin in the period around 1650-1750. In these days, people like Newton, Gauss, Fermat and Lagrange proposed calculus based and iterative formulas for finding minima or maxima. Especially differential calculus methods like for example minimisation and maximisation functions, Taylor expansions and Newton-Rapson [3] (which can be used for the approximation of roots or zeros of real valued functions) have been of great influence.

"The steepest-descent method is the simplest, one of the oldest, and probably the best known numerical method for unconstrained optimisation" [4]. This method was developed by Cauchy in 1847 and was one of the first gradient based methods. Gradient based optimisation methods take differential equations into account and can therefore be very effective to find minima and maxima.

During and shortly after World War 2, optimisation methodology advanced quickly [5]. Especially due to logistical (war-time) efforts, a number of techniques were developed. In this period, the Simplex method (Section 3.4) was developed by Dantzig (1947) and research was done on optimality conditions by Kuhn and Tucker (1951). The Simplex method was one of the first linear optimisation techniques and had a large influence on the field of logistics [6].

Developments in the field of optimisation over the past years are relatively synchronised with the development of computers [7]. The capability of computers to perform very large amounts of computations in a short period of time makes them useful for standardised iterative processes. Numerous optimisation algorithms are developed to be such procedures and can therefore be applied with computers effectively. Notable developments with interesting applications in the field of structural optimisation are among others: topology optimisation (Section 2.4), nature inspired algorithms and population based techniques. These techniques have been successfully applied in modern architecture to achieve optimised structures.

Many nature inspired and population based algorithms have been developed in the past 25 years. This can be explained by their requirement of a large computational power to run. Especially the Genetic Algorithm (GA) (which is discussed in Section 3.5) has been popular in structural optimisation. An example of a structural application of the GA can be seen in Figure 2.2. Another recent development is the Particle Swarm Optimi-



Figure 2.2: Application of structural optimisation on a space-truss. The goal of optimisation in this project: "Nationaal Militair Museum Soesterberg", was to minimise the weight of a 8000 element truss. The result was a 60 kg/m^3 roof instead of the 100 kg/m^3 which is usual for these kind of spans. The engineer in this project is ABT [8]. Courtesy: ABT [8]

sation (PSO) methodology (which is discussed in Section 3.6). This technique was developed by Eberhart and Kennedy [9] in 1995. PSO is an example of a nature inspired, population based algorithm that replicates the behaviour of birds or fish in nature. In this Master's thesis, this algorithm applied for the optimisation of reinforcement in concrete.

2.2. WHAT IS OPTIMISATION?

By analysing the definition of optimisation ("To find the best compromise among several often conflicting requirements, as in engineering design" [1]), it is possible to show a generalisation of the optimisation process. The goal of the process is described by "find the best". This implies that some definition of the "perfect" solution should exist (usually a minimum or maximum value of some "objective" function). The terms "compromise" and "conflicting requirements" indicate that it is usually impossible to find the "perfect" solution for all requirements, because they can be contradicting. Based on this information, any optimisation process requires a goal, requirements (limitations) and some process/ method that "searches" for "optimal" solutions.

Practical examples of optimisation can be found in many fields. Every day examples can, among others, be found in fields like: logistics, economics and engineering. Navigation software for example uses optimisation algorithms to determine the "best" route from location to destination (for example: see the "traveling-salesman problem in Figure 2.3). Economists continuously attempt to maximise profit and engineers try to improve their designs, for example by minimising the mass of a structure.

"Structural optimisation" is the technical term for the application of optimisation on structures. According to An Introduction to Structural Optimal Design, by [10]: "Structural optimisation is the subject of making and assemblage of materials sustain loads in the best way.". Usually, this translates in a minimisation of material, while stresses are kept below some level (which is partially conflicting due to Hooke's law).

2.2.1. MATHEMATICAL DESCRIPTION OF THE STRUCTURAL OPTIMISATION PROBLEM

According to literature, it is possible to describe optimisation as a mathematical model. Insight in such model can help to understand the basic principles of the problem. This standardised formulation shows the optimisation parameters (which are the variables that are optimised) and the constraint(s) of the system. In theory, it should be possible to solve the optimisation problem based on the information of this function.

A number of functions and variables are always present in a structural optimisation problem [10]. A structural optimisation problem (SO) can be described in as a minimisation or maximisation of a certain function. The goal of this function is to define a certain predefined "perfect score", which is a compromise between the score and the constraints. It depends on the optimisation algorithm (Chapter 3) what variables are tested

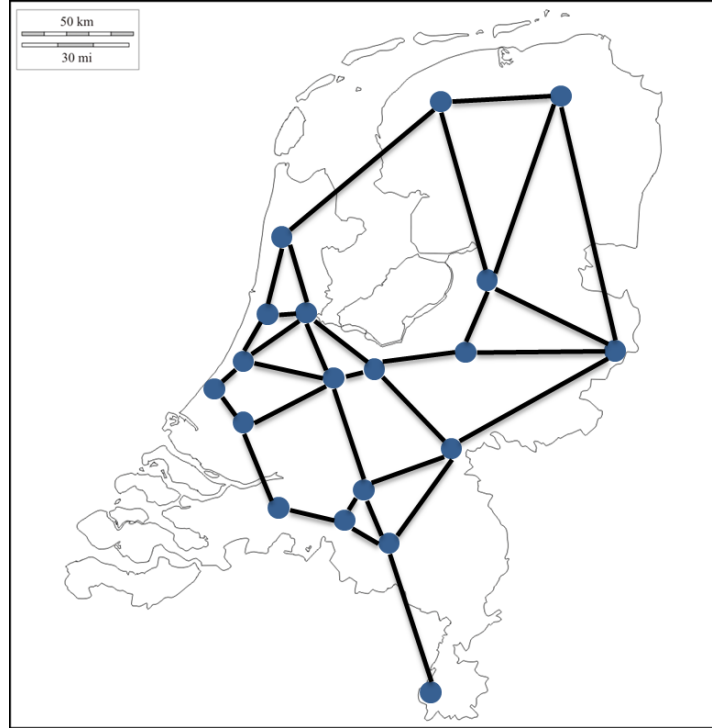


Figure 2.3: Impression of a traveling-salesman problem. The objective can be to visit all cities on the map with the shortest possible route. A possible restraint could be that all cities can only be visited once. Assuming that all cities can be used to start or finish, there are approximately 19 cities with each 2-4 roads, resulting in many possible solutions to the problem.

at what time (and therefore if the global optimum is being found). Each problem contains a set of input information, also known as the design variables. Functions and/ or variables can be limited by certain values, these limitations (i.e.: domain and range) are called “constraints”.

Generally, there are three questions to answer in order to define the optimisation problem:

- “What are the design variables?”: In this Master’s thesis, the design variables contain the stiffness of truss elements EA and therefore represent the amount of concrete or steel in a truss geometry.
- “What are the design constraints?”: The design constraints ensure the quality (safety) of the structure and limits the amount of possible solutions. In this Master’s Thesis, safety is tested by the Strut-and-Tie Model (STM) technique. One major limitation is that cross sections should be positive and below some maximum value.
- “How to determine the optimal solution?”: The optimal solution in case of costs optimisation is the one with the lowest costs which is still safe.

The mathematical formulation of this problem is of the form:

$$(S.O.) = \begin{cases} \text{minimise } f(\mathbf{x}, \mathbf{y}) \text{ with respect to } \mathbf{x} \text{ and } \mathbf{y} \\ \text{subjected to } \begin{cases} \text{behavioural constraint(s) on } \mathbf{y} \\ \text{design constraint(s) on } \mathbf{x} \\ \text{equilibrium constraint(s)} \end{cases} \end{cases} \quad (2.1)$$

Where:

- $f(\mathbf{x}, \mathbf{y})$ Is/ are the **objective function(s)** $f(\mathbf{x}, \mathbf{y}) = [f_1(\mathbf{x}, \mathbf{y}) \dots f_n(\mathbf{x}, \mathbf{y})]^T$. These functions are used to describe the design in terms of the variable(s) that needs to be optimised, the design variables. For example, the results(s) $f(\mathbf{x}, \mathbf{y})$ can represent the cost of some option.
- \mathbf{x} Are the **design variable(s)** $\mathbf{x} = [x_1 \dots x_n]^T$. The optimisation process searches for the values of \mathbf{x} resulting in the "best" or optimal solution. These design variables can i.e. describe the geometry, materials or reinforcement of a structure.
- \mathbf{y} Are the **state variable(s)** $\mathbf{y} = [y_1 \dots y_n]^T$. For a given design, \mathbf{y} is a function or vector that represents all constant properties of the problem, like the shape of the truss, the materials and structural limitations.

In case there are multiple object functions, it is usually harder to find a optimal solution. The design variables of such cases can result in different (even opposing) solutions for multiple object functions (for example: find the optimum value \mathbf{x} for the combined problem: minimise $f(x, y)$, with: $f_1(x, y) = y + (x - 1)^2$ and $f_2(x, y) = y - (x + 1)^2$). The result of such situations is usually a compromise between the multiple optima of individual object functions. To deal these kind of problems, the definition of a "Pareto optimum" [4] is introduced.

The definition of a Pareto optimum is described by "An introduction to Structural Optimisation" [10] as: "An allocation is defined as "Pareto efficient" or "Pareto optimal" when no further Pareto improvements can be made." Meaning that each change of the Pareto optimal variables (x^*, y^*) that makes at least one individual more "perfect" will also decrease the score of one or more variables.

The importance of the Pareto definition is shown if multiple object functions and design variables are present, each having a unique optimum solution. In such situations, optima can be obtained by a summation of all object functions. The "best" set of design variables in this situation is called the Pareto optimum.

$$\sum_{i=1}^n w_i * f_i(x, y) \quad (2.2)$$

Where:

- w_i Is the weight factor to indicate the importance of each object function. $w_i \geq 0$ and the sum of all weight factors is always equal to 1. $\sum_{i=1}^n w_i = 1$.
- $f_i(x, y)$ Is a object function.

2.2.2. COMMON DIFFICULTIES

There are many possible pitfalls when dealing with optimisations. A number of important pitfalls are discussed in this section. Avoidance of these difficulties can help to increase the success of optimisations in general.

DEFINE PROBLEM AS SIMPLE AS POSSIBLE.

A well known statement of Albert Einstein is: "Everything should be made as simple as possible, but not simpler." [11]. This statement also applies on optimisations in general. The simpler the structural optimisation, the more understandable the solution and therefore the lower the chance to create errors. Simpler models or object functions tend to be less computationally intensive. As a result, the algorithm can become faster and result in better solutions.

A warning has to be given on the topic simplification: models represent reality, if a model is simplified to the point that it is no longer realistic, the model becomes useless, regardless the results of an optimisation process.

THE "CONCEPT OF NO FREE LUNCH" [12].

The concept of no free lunch states that: "There is not one function to solve all problems". For optimisation techniques, this is relevant because this states that there is not one optimisation algorithm that works best

for all problems. It is common for problems to have multiple good solutions. Which one is best depends on the type problem. Understanding on different types of object functions, optimisation algorithm classes and the problem in general is required to select algorithms, functions and processes to solve certain problems.

DEALING WITH MODEL WEAKNESSES.

In sports, one way to win is by cheating. If the cheater wins a match, this can mean that someone wins a competition who is not necessarily the best. In some situations, optimisation algorithms tend to select winners (cheaters) that should not be selected for various reasons. Especially in structural design it is important to have some fail-safes to prevent unsafe options from being selected as winner.

A well known example of a model weakness in structural optimisations are penalised (unsafe) options that are selected due to a low overall score. For example: if the cheapest possible, but safe structure ($object = costs * penalty$) should be found for a problem, the model might select a very cheap, penalised solution on which the penalisation has a low influence ($object = 0 * penalty$). This situation indicates a too small influence of the penalisation on the problem. A possible solution is to perform ($object = costs + penalty$) as object function, or otherwise prevent penalised functions from being selected by some filter.

Dealing with model weaknesses can be done on several levels, the following list shows the preferred order of dealing with weaknesses:

1. Remove weaknesses from model.
2. Prevent weak options from being selected.
3. Ensure identification of weak options by model.
4. Ensure identification of weak options by user.

"GRAND-CANYON-PROBLEM"

Some object functions contain so called "Grand Canyon" problems. These are regions containing a global or local optimum, surrounded by a "impenetrable" infeasible region (as is displayed in Figure 2.4). If it cannot be shown that such situations cannot occur in a object function, it is possible to run into a local optimum due to this problem. "Grand Canyon" problems work in two directions: they can prevent algorithms from entering a (promising) solution space or they can prevent them from escaping.

Different algorithm classes have different methods to solve this problem. Population based systems are less sensitive for this problem. A population is usually spread over both sides of the "canyon". Also, they can have a functionality that replaces some bad individuals with random new ones at certain points. Other algorithms use a capability that allows sudden changes in parameters or have some build-in randomness or another solution.

A more difficult version of the "Grand Canyon" problem is the "Needle in a haystack" problem (Figure 2.4). The "Needle", or spike in such functions has such small influence on the rest of the solution space that it can hardly be detected. If this needle represents the global optimum, then it can be very hard to find.

TIME-INTENSIVE OPTIMISATIONS

The required time to run optimisation processes largely depends on four factors: amount of possibilities, computational intensity of object function(s), applied optimisation algorithm and the complexity of the solution space. Assuming that the fastest possible class of algorithm is applied on a problem, the speed can be increased by several means: reduce size and complexity of solution space, improve computational efficiency of code and use faster computers or computation techniques (like for example parallel computing).

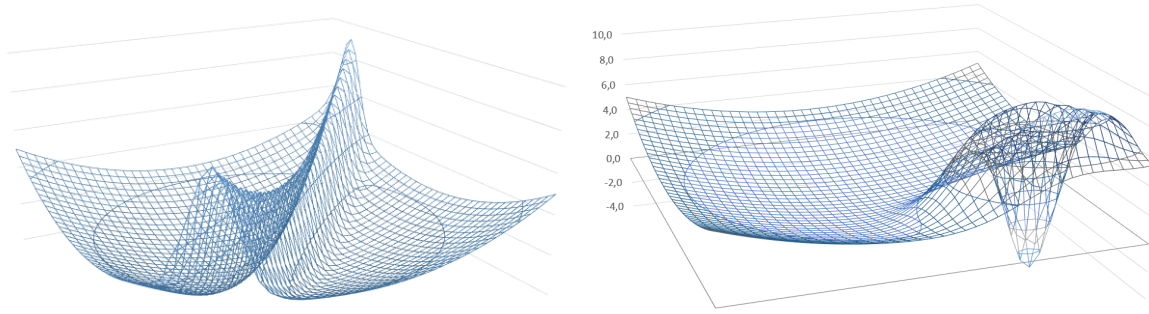


Figure 2.4: Representation of a "Grand Canyon" (left) and "Needle in a haystack" (right) problem. Some algorithms will be misled by these curves if they are not capable of crossing the "Canyon" or finding the "Needle". The global optimum of the left figure is at the bottom of the parabola left of the "mountain range". The global optimum of the right problem is at the bottom of the "Needle". The kind of problems are usually solved with stochastic and/ or population based algorithms.

2.3. STRUCTURAL OPTIMISATION TERMINOLOGY

Optimisation methodology is used in certain terms and definitions. As a part of the literature study, this section contains some important terms and definitions in the field of optimisation. The described terminology is related to one (or more) of the four categories: design variables, algorithm, object function (including fitness and constraint) and output.

2.3.1. DESIGN VARIABLE TERMINOLOGY

BOUNDEDNESS

"Proper bounds are necessary to avoid unrealistic solutions." [2]. For example: it is obvious that cross sections of beams are limited by a minimum and maximum cross section. Cross-sections can never be negative and are usually limited in size as well. Applying bounds can result in a serious reduction of the size of the optimisation problem due to less possibilities and therefore less options.

CONTINUOUS VARIABLES

"When design variables can have any numerical value within their allowable range, the problem is called a continuous-variable optimisation problem." [4].

DISCRETE VARIABLES

"A design variable is called discrete if its value must be selected from a given finite set of values. For example, a plate thickness must be one that is available commercially: $\frac{1}{8}$, $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{3}{4}$ and 1 inch, and so on. Similarly, structural members must be selected from a catalogue to reduce fabrication costs. Such variables must be treated as discrete in the standard formulation." [4]

DESIGN VARIABLES

Design variables are functions or vectors that describe the design, and which can be changed during optimisation. It may represent geometry of choice or material. When it describes geometry, it may relate to a sophisticated interpolation of shape or it may simply be the area of a bar, or the thickness of a sheet [4].

INTEGER VARIABLES

An integer variable, as the name implies, must have an integer value, for example, the number of crates to be shipped, the number of bolts used, and so on. Problems with such variables are so called discrete and integer programming problems. [4]

STATE VARIABLES

State variables are those parameters that are important for the process, but are no design variables. These can for example be material data, like the yield strength of certain elements.

STOCHASTIC VARIABLES

A Stochastic optimisation algorithm is an algorithm that includes random values at some point. These random values are called Stochastic variables. A stochastic variable can be any value within a given set (or domain) and can be either discrete or not.

2.3.2. OPTIMISATION ALGORITHM TERMINOLOGY

ITERATIVE METHODS

Iterative implies analysing several trial designs one after another until an acceptable design is obtained [4]. In computational mathematics, iterative methods are defined as: "An iterative method is a mathematical procedure that generates a sequence of improving approximate solutions for a class of problems."

TAYLOR SERIES EXPANSIONS

Some optimisation algorithms, especially the gradient based ones, use Taylor polynomials [13] their approximation of the optimum position. The mathematical description of the Taylor polynomial is:

$$P_n(x) = f(c) + (x - c)f'(c) + \frac{(x - c)^2}{2!}f''(c) + \dots + \frac{(x - c)^n}{n!}f^{(n)}(c) \quad (2.3)$$

Depending on the cut-off position, the numerical version of this polynomial also has an error. This error, or rest term, can be used to define the numerical accuracy of optimisation algorithms. If a polynomial is cut-off after $n = i$, the function is of the i^{th} order. The cut-off error can be described by:

$$R_n(x) = \frac{(x - c)^{n+1}}{(n + 1)!}f^{(n+1)}(\xi) \quad (2.4)$$

2.3.3. OBJECT FUNCTION TERMINOLOGY

CONSTRAINT

A constraint is a represents a limitation of the object function. A constraint can be expressed in multiple ways, for example by setting a domain for certain parameters or by applying penalty functions. In the context of this Master's Thesis, it will represent the structural safety penalisation and geometry limitation.

CONTINUOUS AND DISCONTINUOUS FUNCTION

In Calculus, early transcendentals [3], the definition of continuity is given by: "A function f is continuous at a number a if $\lim_{x \rightarrow a} f(x) = f(a)$. A function f is continuous on an interval if it is continuous at every number in the interval.". If object functions contain several continuous parts, for example f and g , then the result h is also continuous if: $h = f + g$, $h = f - g$, $h = c * g$, $f * g$, $\frac{f}{g}$ if $g \neq 0$.

It is potentially profitable to create continuous object functions, even if the applied optimisation algorithm can handle discontinuities. Continuous functions tend to be simpler to solve than their counterparts because they create less disturbance. Therefore, a continuous objective function is profitable for the speed (faster convergence), and reliability (higher probability for finding the global optimum) of the process.

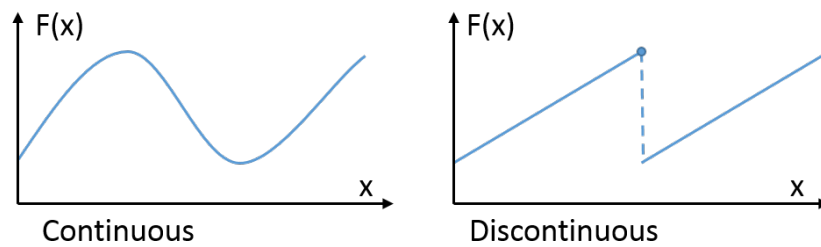


Figure 2.5: Example of a continuous (left) and discontinuous (right) function. The continuous function is usually easier to solve for optimisation algorithms, because they are not "mislead" by the objective function.

DETERMINISTIC AND STOCHASTIC PROBLEMS

Deterministic problems are methods that do not include stochastic variables. Stochastic problems are methods that do include stochastic variables.

OBJECTIVE FUNCTION

The object function is what defines the "goodness" of options. It contains both the optimisation goal and its constraints. The object function is usually of the shape $F(x, y)$, in which x is a parameter, vector or matrix containing the optimisation parameters and y contains the design variables.

SINGLE AND MULTI-OBJECTIVE PROBLEMS

Multi-objective problem is a problem with multiple objective functions. For example, if both the costs and weight of a structure should be minimised for some reason. Objectives in such situations can be contradicting, which makes a problem much more difficult. Also the balance between the different objectives should be taken into account.

SOLUTION SPACE

The solution space [6], or feasible region, is the set of all possible options of an optimisation problem that satisfy the problems constraints, potentially including inequalities, equalities and integer constraints. This is the initial set of candidate solutions to the problem, before the set of candidates has been narrowed down. In case of large multidimensional objective functions, it is often too complex to plot the solution space of a problem.

2.3.4. OUTPUT TERMINOLOGY

PERETO OPTIMALITY

"A design is Pareto optimal if there does not exist any other design that satisfies all of the objectives better." [10]. This definition coincides with the definition of the optimal solution ("To find the best compromise among several often conflicting requirements, as in engineering design" [1]), which states that the best solution is usually a compromise between several requirements.

UTOPIA POINT

A point in the multi objective solution space is called utopia point if the solution is optimal for all individual objectives [4].

LINEAR AND NON LINEAR RESPONSES

"A function f is linear if it satisfies $f(x_1 + x_2) = f(x_1) + f(x_2)$ and $f(\alpha x_1) = \alpha f(x_1)$ for every two points x_1 and x_2 in the domain and all α ." [14].

CONVEX AND NON-CONVEX RESPONSES

A convex response is defined as: "Any line connecting 2 points on the graph lies above it or on it." [2]. A non-convex function does not fulfil this requirement, it has "many local optima", non-linear constraint functions can result in non-convex feasible domains", "non-convex feasible domains can have multiple local boundary optima, even with linear objective functions", i.e. multi-modal functions.

SMOOTH OR NON-SMOOTH RESPONSES

Smooth functions are continuous and differentiable [3]. Non smooth functions are all functions that do not fulfil these requirements. All non smooth functions cannot be used in gradient based optimisation algorithms, since no gradient or continuity is available.

2.4. STRUCTURAL OPTIMISATION CLASSIFICATIONS

Structural optimisation in general can be classified into several strategies [6, 10]. These strategies differ mostly in the optimisation goal, type of input variables and output information. The mathematical part of the optimisation technique can still be solved according to various algorithms. Depending on the applied strategy, certain algorithms are preferred over other ones for various reasons. This principle is in accordance to the concept of "no free lunch" [12], which suggests that there is not one "magic" tool to solve all problems.

The given structural optimisation strategies are unique for their characteristic types of goals. The four main strategies are: sizing, shape, topology and material optimisation. In essence, these strategies are related or even overlapping with. The first three strategies are represented by Figure 2.6, which gives a schedule that indicates the type of structural optimisation. Material optimisation is usually related to one of these strategies.

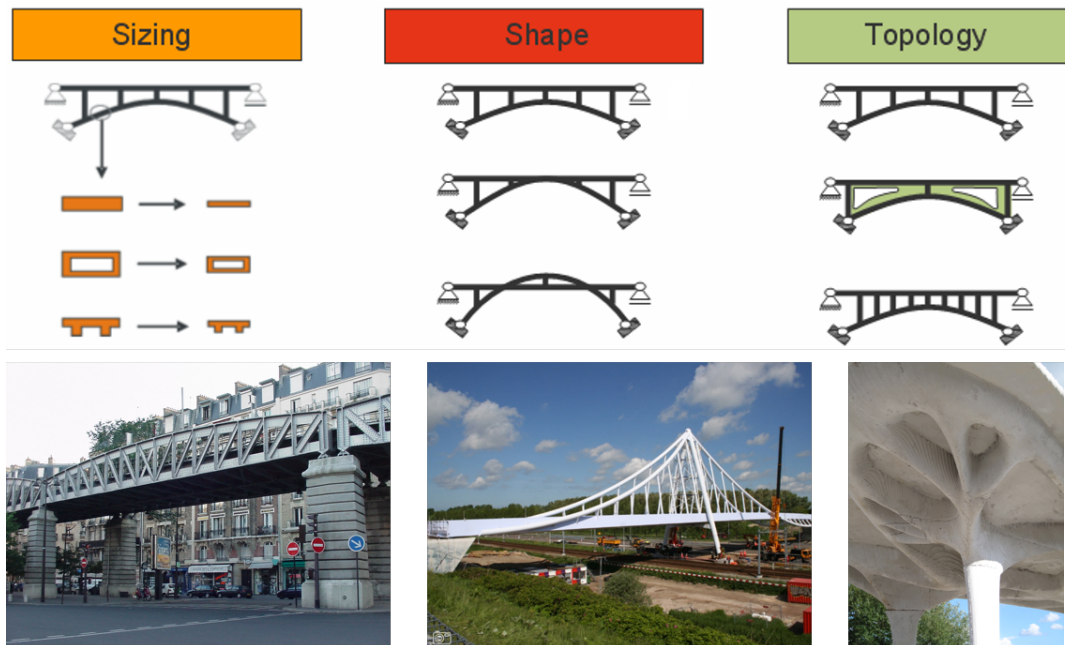


Figure 2.6: The three main structural optimisation strategies: sizing, shape and topology optimisation, courtesy: [15]. Sizing optimisation focuses on individual elements, shape optimisation attempts to adjust an initial solution and topology optimisation searches for the best possible shape within a given (boundary) volume.

2.4.1. SIZING OPTIMISATION [2, 10]

In sizing optimisation, the dimensions (or materials) of elements are the design variables. Since the geometry and boundary conditions (like loads and constraints) are already known, these can be used to define the constraints of the beam in terms of geometry, strength, deformation, etc. The object function is the goal of the optimisation, usually mass, stiffness or costs, but other goals could easily be defined. A sizing optimisation is used to optimise all individual elements of a structure.

Sizing optimisations are often applied in case of large steel structures with lots of repetition (trusses, space frames, etc.). The goal of sizing optimisation of trusses is usually to find the optimal cross-sections for the individual bar elements. The example: "Nationaal Militair Museum Soesterberg" in Figures 2.2 and 2.7 is an example of sizing optimisation by ABT [8].



Figure 2.7: Application of structural optimisation on a space-truss. The goal of optimisation in this project: "Nationaal Militair Museum Soesterberg", was to minimise the weight of a 8000 element truss. The result was a 60 kg/m^3 roof instead of the 100 kg/m^3 which is usual for these kind of spans. The engineer in this project is ABT [8]. Courtesy: ABT [8]

2.4.2. SHAPE OPTIMISATION [2, 10]

In a shape optimisation, structures are being transformed to find the optimal solution. For this reason, the design variables of a shape optimisation are the values x that transpose the original geometry of the structure. The constraints are similar to the ones for sizing optimisation. The only difference is that their definition is adjusted to the design variables. Geometrical constraints are described in terms of maximum transformations. The equilibrium of forces becomes more depending on the shape of the structure of each option. Shape optimisations are usually applied on "free formed" structures with a known basic shape. The goal could for example be to remove material to create a lighter structure.

The goal for a shape optimisation of a truss would be to find the optimal node locations for a certain truss with a given amount of nodes, bars and a certain bar-connectivity.

2.4.3. TOPOLOGY OPTIMISATION [2, 10]

Topology optimisations are based on the principle of removing material where it is not needed. To be able to work with a topology optimisation, the starting point is the design space of a structure. The design variables can then be described as a mesh of that design space. In this situation the constraints are still similar to sizing and shape optimisation. When it is found that an element is "useless" it is removed (or faded) from the equation until the optimal situation is found. Topology optimisations are often quite rough and serve as inspiration for the actual form of the structure.

2.4.4. MULTI-MATERIAL TOPOLOGY OPTIMISATION [7, 17]

In case of multi-material topology optimisation, a structure can be created from multiple materials. Usually, one material takes the tensile forces another material the compressive ones. In theory, this principle is perfectly suited for reinforced concrete structure, where the reinforcement takes the tension and the concrete the compression.



Figure 2.8: Left: Shape optimisation, transformation of initial truss into the optimal structural lay-out in terms of node locations. In this case: the British Museum in London, the structure was tested for more than just efficiency. Also architectural demands, like solar gain and acoustic performance where taken into account. The design variables can be traced back to the shape (curvature) and refinement of the truss structure [16].

Figure 2.9: Right: Attempted topology optimisation during the TU-Delft high-rise workshop 2013-2014. The goal of the optimisation was to find the lightest truss tube-structure for a high-rise building. The available design space was one cross section and a very dense truss surrounding the entire building. The design variable was a on-off switch for every bar in the given truss shape. The result was a very dense truss at the bottom of the building and a much opener truss at the top. Note: a very small number of bars was added later for architectural purposes.

Topology optimisation is not considered to be a structural design method, but a optimisation strategy. Due to the combination of Finite Element Method (FEM) and structural design which is required to use this technique. It is quite capable of giving an impression of the optimal design of structural shapes.

The main problem of a 3D multi-material optimisation is the processing time. Assuming a certain minimum accuracy for a problem, a three dimensional problem (R^3) has much more possibilities than a two dimensional one (R^2). Since the structural calculation and the amount of materials will also increase the intensity of the process, even a small and simple problem can consume a lot of computing power. According to Table 2.1, which is based on [17], the intensity for 3D topology optimisations is a factor 50 larger than a 2D optimisation. The explanation for the difference can be found in the amount of equations required to solve a 2D or a 3D option. Therefore, users should be careful when choosing a what method to use for a optimisation, 3D results may be more accurate, but take much longer to compute.

2d topology optimisation	3d topology optimisation
Mesh size: $l \times b$	Mesh size $l \times b \times h$
Finite element size: 8 nodes (quadratic)	Finite element size: 20 nodes (quadratic)
Stiffness matrix: 8×8	Finite element size: 20 nodes (quadratic)
Amount of finite element equations for 1 iteration of an object with 100×100 elements: $8.0 * 10^4$	Amount of finite element equations for 1 iteration of an object with $100 \times 100 \times 5$ elements: $1.2 * 10^6$

Table 2.1: Differences between 2D and 3D topology optimisation.

2.5. BASIC TYPES OF OPTIMISATION ALGORITHMS

Over time, many different algorithms for optimisation have been developed. Unfortunately, not one algorithm is the best option for all possible problems. Usually, the type of problem, the formulation of the object function, the amount and complexity of variables and the available computational power decide what algorithm is best for a situation. A common statement in the field of optimisation is "the concept of no free lunch" [18], which states that "no magic bullet can exist against all problematic features". Important motivations to chose one type of algorithm over another are:

- Computational efficiency (required time to run algorithm).

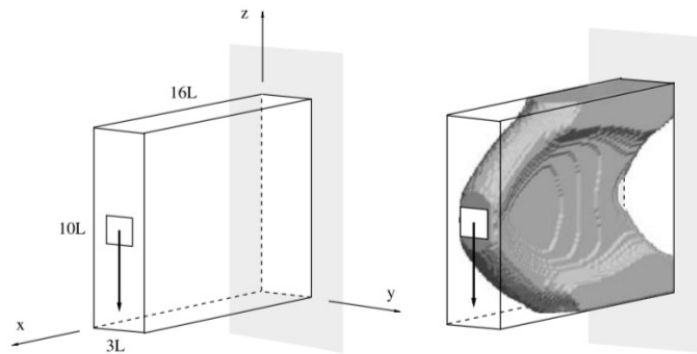


Figure 2.10: Example of three-dimensional cantilever beam. Courtesy: "Large scale topology optimization in 3D using parallel computing" [17].

- Accuracy of solution.
- Reliability (or robustness): does an algorithm find the global optimum within a known period?
- Simplicity: is the optimisation process understandable and usable?
- Possibility: is it realistically possible to apply the methodology?
- Availability (practical motivation): is the process known to a user or program?

Usually, the selection process of an optimisation algorithm runs in roughly two steps. First, the algorithm classification is selected, next the algorithm within this class itself. The rest of this section shows a number of optimisation algorithm classes and their basic properties. It is possible for optimisation algorithms to fit into multiple classifications.

2.5.1. CALCULUS BASED OPTIMISATION

This simple and elementary class of optimisation is based on casual calculus [3]. Application of calculus is usually the first attempt of engineers in optimisation [19]. In this strategy, object functions are solved analytically in order to find the exact solution space of a problem and therefore the exact solution. If it is simple and possible to find optima with this technique than this is preferred over alternatives. For larger or more complex problems, this is usually not the case.

A simple example of an application of a calculus based method is the classical rectangular (farm) area maximisation problem which was discussed in Figure 2.1. The solution (shown in Figure 2.11) to this problem is exact, undeniable and can clearly show the impact of alternative solutions. Application of alternative classes of optimisation algorithms give less clear, but still good results to the same problem.

2.5.2. DETERMINISTIC METHODS

"Deterministic methods find the global minimum by an exhaustive search over the set." [4]. This class of methods searches through the entire search space (for example by a pattern search) of a problem, resulting in a computationally very expensive, but guaranteed accurate method. Since the amount of possibilities in these kind of problems is known, the required number of steps is as well. Deterministic methods are usually applied in cases where it is absolutely required to find a global optimum and the problem is not too large. For example to scientifically prove the effectiveness of other methods in example optimisations.

2.5.3. STOCHASTIC METHODS

"Stochastic optimisation methods are optimisation methods that generate and use random variables" [4]. Stochastic methods have been developed as variations of the random search method (which literally searches

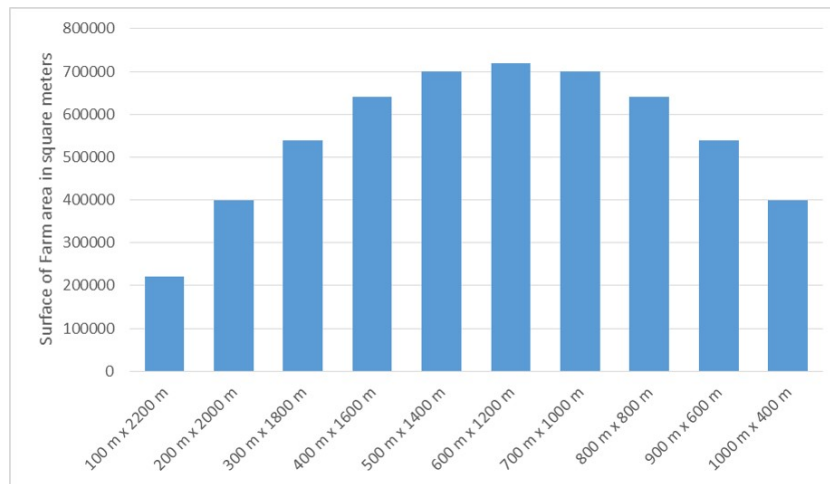


Figure 2.11: Solution to the rectangular farm area problem shown in Figure 2.1. The 600 x 1200 m solution results in the largest area. Because the continuous variables of this optimisation algorithm can be analysed analytically, it is possible to prove that this solution is the global optimum.

through the solution space randomly to find optimal solutions) with the goal of searching more effectively. Stochastic methods are generally much more efficient than deterministic methods, but do not give an absolute guarantee of finding the global optimum unless all possible options are attempted (which would make the method perform a similar computation as deterministic methods).

Stochastic methods usually search in two phases. First the algorithm searches for feasible regions and next, these regions are analysed in detail. Depending on the type of problem and method, algorithms may have trouble with discontinuous functions, run into local optima or become very slow.

Random numbers which are required in stochastic methods are usually computer generated. If stochastic methods are applied for scientific purposes, it can be required to reproduce the applied random numbers in order to reproduce experiments. In such cases, computer programs (like Matlab [20]) offer a "random seed" in order to apply similar random numbers for each run.

2.5.4. DIRECT SEARCH METHODS

The basic principle of many direct search method is to start on a random or estimated position (which depends on the amount of pre-knowledge), and then iteratively improve this position in order to find the optimum. Basic methods in this class are robust, but sensitive for local optima in case of non smooth or discontinuous object functions.

Direct search (or line search) methods do not use gradients to determine optima [4]. These methods are therefore useful in situations where gradients are difficult to find or if they result in local optima. As a consequence, direct search methods tend to be less efficient than gradient based methods. A well known method of this class is the Nelder-Mead-Simplex method.

2.5.5. NATURE INSPIRED ALGORITHMS

Nature inspired optimisation algorithms are classified as all methods which are based biological events. Examples of nature inspired algorithms are Simulated Annealing (SA), GA, PSO and Ant Colony Optimisation (ACO), which are respectively based on cooling of metal, evolution, swarm behaviour of fish or birds and the behaviour of ants.

Nature inspired search methods fall into the category of direct search methods [4]. A major advantage of nature inspired methods over the direct search ones is that some of them do not require continuous object functions to run. This class of algorithms can usually overcome multiple objectives at once, mixed design

variables, discontinuous non differentiable object functions. Nature inspired algorithms can and often will find global optima, but it is usually unknown when it is found [2].

The wide range of applications, their relative simplicity, efficiency and accuracy makes this class of algorithms relatively popular in engineering practice [18].



Figure 2.12: Particle swarm optimisation is a nature inspired optimisation algorithm based on the behaviour of fish schools. Imagine how individual fish search for food (goal), avoid predators (boundary) and influence the behaviour of their neighbours. Photo courtesy: David Doubilet.

2.5.6. GRADIENT BASED METHODS

Gradient based methods are more accurate than direct search methods. This class usually converges in a finite amount of steps. Gradient methods are applicable in case of continuous differentiable functions [4] that are not too complex to compute. Algorithms of this class are usually based on Taylor series [13]. First order methods only take gradient functions into account, second order techniques also apply Hessian functions, etc. Due to this Taylor based, numerical approach, methods can experience computational errors due to the "cut-off" of the Taylor sequence after a number of parts.

This type of optimisation is applied on smooth non-linear problems. Algorithms tend to converge in less steps, which makes them much more efficient. Unfortunately, gradient based methods may not be robust due to errors and can be mislead by certain gradient-functions. Still, if a gradient is available, this class of method outperforms the other discussed classes and are therefore preferable in such situations.

3

OPTIMISATION ALGORITHMS

3.1 Brute force algorithms.	20
3.2 Simulated Annealing	20
3.3 Cyclic coordinate search.	21
3.4 Nelder-Mead simplex method	22
3.5 Genetic Algorithm (GA)	23
3.5.1 Pseudocode for the Genetic Algorithm (GA).	24
3.6 Particle Swarm Optimisation (PSO)	25
3.6.1 PSO algorithm mathematics	26
3.6.2 Pseudocode for the Particle Swarm Optimisation (PSO) algorithm.	28
3.6.3 PSO algorithm properties	28
3.7 Ant Colony Optimisation (ACO)	31
3.7.1 Pseudocode for the Ant Colony Optimisation (ACO) algorithm.	32
3.8 Gradient based first order methods.	32
3.9 Gradient (Hessian) based second order methods	33
3.10 Comparing algorithms	33

There are many optimisation techniques that can serve as “engine” in optimisation processes. A number of such techniques are discussed in this chapter. Techniques can be divided into several classes. There are analytical (calculus based [3]) approaches and iterative (numerical) approaches. The analytical ones are exact, the numerical [13] are not and can experience numerical errors. Analytical optimisations are often unique problems that require a specific solution (like the one in Section 2.1). Numerical methods are used to estimate solutions of problems with standardised techniques. Structural optimisations are usually solved with standard numerical approaches because of their complexity and unique objective functions.

If numerical methods are applied to solve optimisation problems, then there are a number of effects that require attention. Important aspects are amongst others: solvability (can a method find a global optimum within a certain amount of time), reliability (what is the probability to find the global optimum?), efficiency (what time is required to find a sufficiently accurate solution) and sensitivity for errors (method diverges, numerical errors, etc.). To select specific algorithms for problems, it is possible to divide methods into classes. Based on these classes, it is possible to make a couple of statements [6]:

- When simple and possible, it is preferable to solve problems analytically. The analytical solution is precise and reliable.
- Simple 0^{th} order methods can often (for some methods, like the random ones, always) be solved with high accuracy, but they can be inefficient and/or unreliable.

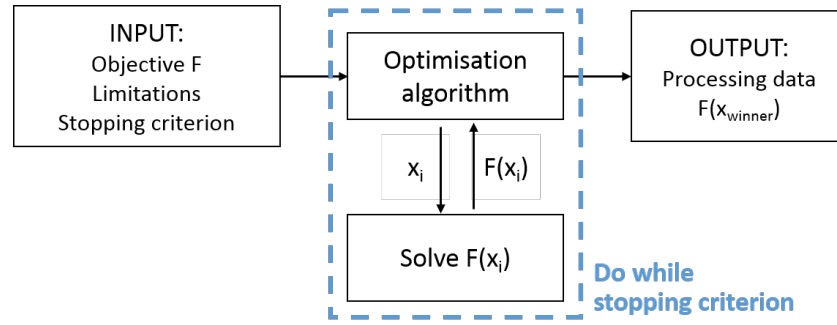


Figure 3.1: Basic diagram to show the location of an optimisation algorithm in the optimisation process.

- Line search methods are more effective for most problems than random methods, but require more computationally intensive algorithms that usually require less computations. Some line search methods can converge in local optima, especially in case of chaotic functions. Some algorithms, like the population based ones, solve this problem with swarm intelligence.
- Complex, higher order (i.e.: gradient based) methods are more efficient than 0^{th} order ones, but are not always solvable or precise. If functions are not differentiable, higher order methods cannot be used. Special attention should be paid to the accuracy and reliability of the solution due to the possibility of numerical errors.

3.1. BRUTE FORCE ALGORITHMS

Brute force (or random search [2]) methods are among the simplest analysis concepts in mathematical optimisation. The optima (within the given constraints) are computed by trying all (or at least a very large representative set of) options in a structured or random order. This process is very robust, since it will eventually consider all possible types of solutions and will therefore always find all local and global optima. A major disadvantage of trying all options is the computational intensity. For example: a simple problem with 3 parameters with 200 possible values would result in $200^3 = 8.000.000$ possible solutions. This number, and therefore the required time to solve a problem, increases exponentially with the amount of parameters.

Nevertheless, brute force optimisation can be useful for simplistic optimisations where computation time is not a problem. In case of more complex optimisations with lots of variables, the system can still be surprisingly useful, although it can be infeasible to run all options. Because the process has no problems with local minima or strange formulas, it will always find some answer. The "Needle in a haystack" problem (Section 2.2.2) is therefore not a problem for this algorithm. Brute force methods are usually a last resort, they will only be used when all other methods have failed, or the amount of options is relatively small. One application for brute force algorithms in science is to confirm the methodology more efficient algorithms. Running a complete brute force optimisation can confirm the results of other algorithms in such cases.

3.2. SIMULATED ANNEALING

Simulated Annealing (SA) [2] is a nature inspired random optimisation method based on the physical process annealing (heating and gradual cooling of metal/ glass to relieve internal stresses). For an annealing process to work effectively, it is required to let material cool slowly. If applied properly, the material achieves a minimum amount of residual stresses (which is considered to be optimal for metals). Translated to the SA algorithm, this indicates that sufficient time should be applied on analysing different "temperatures" or search areas, so the parameters "or material" can find best configuration. Areas with a large influence are given more time to cool than areas with a slow one. In other words: promising areas are analysed more extensively, so if a local optimum is found, the algorithm is likely to "scout" the area surrounding this point for more feasible options. Finally, a cooling piece of metal cannot suddenly increase its heat. The SA algorithm remembers its optima and only replaces these in case more feasible solutions are found.

A SA process consists of the following steps [7]:

- Step 1: Choose an initial temperature T_0 and a starting point x_0 to obtain $f(x_0)$. Initialise the iteration counter as $K = 0$, and another counter $k = 1$.
- Step 2: Randomly generate a new design variable x_k close to x_0 . If the point is infeasible, generate another random point until feasibility is satisfied. Generate a random number z uniformly distributed in $[1, 0]$ and $\Delta f = f(x_k) - f(x_0)$.
- Step 3: if $\Delta f \leq 0$, then take x_k as the new best point x_0 , set $f(x_0) = f(x_k)$ and go to step 4. Otherwise, calculate the probability density function $P(\Delta f) = \exp\left(\frac{\Delta f}{T_k}\right)$. If $z < P(\Delta f)$, then take x_k as the new best point x_0 and go to step 4. Otherwise, go to step 2.
- Step 4: If $k < L$, then set $k = k + 1$ and go to step 2. If $k > L$ and any of the stopping criteria is satisfied, then stop. Otherwise, go to step 5.
- Step 5: Set $K = K + 1$, $k = 1$; set $T_k = T_{k-1}$; go to step 2.

Simulated annealing has several characteristic properties. First, the acceptance of bad steps (temperature increase) is likely in the initial phase, but reduces at the end. Furthermore, simulated annealing is capable of escaping of local optima, which makes it a robust method which is a bit more efficient than the random methods, but can still be time consuming.

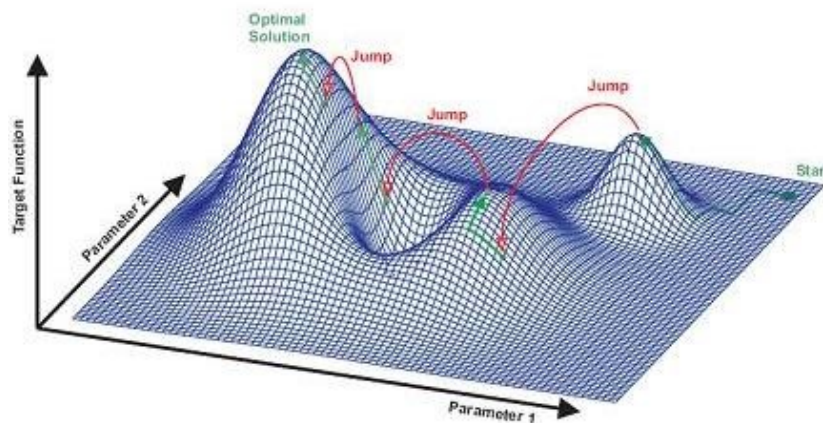


Figure 3.2: The Simulated Annealing (SA) algorithm. The red arrowed jumps represent the algorithm's ability to randomly search for new options. The green arrows represent the "cooling behaviour" of the material, used to "scout" the areas around solutions for more optimal ones. Courtesy: <http://www.frankfurt-consulting.de>.

3.3. CYCLIC COORDINATE SEARCH

Cyclic coordinate search [2] method(s) use analysis of their surroundings to continuously search for the most feasible search direction. This process is effective when the amount of design variables is limited and the solution space is continuous. The method uses partial minimisations in each design variable direction to achieve convergence. Unfortunately, this method can have slow convergence and is sensitive for local minima.

- Step 1: Select initial starting point $x_{initial}$ and find $f(x_{initial})$
- Step 2: Search for alternatives in each coordinate direction (design variable)
- Step 3: Perform single-variable optimisation along each direction s : $\min_a(f(x + as))$

- Step 4: Continue until convergence is achieved (go to step 2)

If this method becomes inefficient (/slow), the Powell's conjugate directions method [7] can be used. This method is similar to the cyclic coordinate search method, but adjusts the search directions to improve convergence. This variant is guaranteed to converge in n cycles for quadratic functions [6].

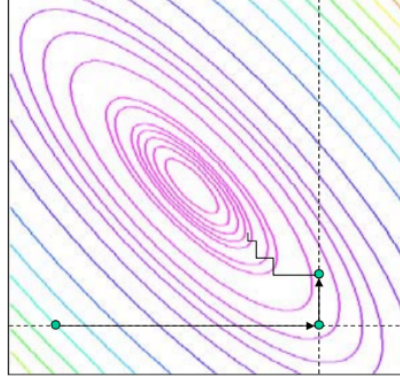


Figure 3.3: Optimisation for 2 design variables with cyclic coordinate search method. Every step, 1 dimension (or variable) is improved until the optimum is achieved. courtesy: [2]

3.4. NELDER-MEAD SIMPLEX METHOD

The Nelder-Mead simplex method [2] uses triangular shapes to move towards a minimum. The method uses several starting points, and then replaces the worst point for an improved one. This "improved" point is generated based on difference between the worst point and the average of the best points. As a result, the triangle is "flipped" over its best "axis" to generate a new triangle (this process is displayed in Figure 3.4). This process is repeated until convergence is achieved.

- Step 1: Determine values for $f(\mathbf{x})$ at $n + 1$ points in R^n and order them according to the values at the vertices: $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \dots \leq f(\mathbf{x}_n)$.
- Step 2: Calculate \mathbf{x}_0 , the centre of gravity of all points except \mathbf{x}_{n+1} .
- Step 3: Compute the reflected point $\mathbf{x}_r = \mathbf{x}_0 + \alpha(\mathbf{x}_0 - \mathbf{x}_{n+1})$ if the reflected point is better than the second worst, but not better than the best, i.e.: $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) \leq f(\mathbf{x}_n)$, then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1} = \mathbf{x}_r$ and go to step 1
- Step 4: if the reflected point is the best point so far, $f(\mathbf{x}_r) < f(\mathbf{x}_1)$ then compute the expanded point $\mathbf{x}_e = \mathbf{x}_0 + \gamma(\mathbf{x}_0 - \mathbf{x}_{n+1})$, if the expanded point is better than the reflected point $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1} = \mathbf{x}_e$ and go to step 1. Else, obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1} = \mathbf{x}_r$ and go to step 1. Else, go to step 5.
- Step 5: If $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$, compute contracted point $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_0 - \mathbf{x}_{n+1})$ if the contracted point is better than the worst point, i.e.: $\mathbf{x}_c < f(\mathbf{x}_{n+1})$ then obtain a new simplex by replacing the worst point $\mathbf{x}_{n+1} = \mathbf{x}_c$ and go to step 1. Else, go to step 6.
- Step 6: For all but the best point, replace the point with $\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_0 - \mathbf{x}_{n+1})$ for all $i \in 2, \dots, n + 1$ and go to step 1. Note, α, γ, ρ and σ are respectively the reflection, the expansion and the shrink coefficient. Standard values are $\alpha = 1, \gamma = 0.5, \rho = -0.5$ and $\sigma = 0.5$.

The Nelder-Mead simplex method is often seen as one of the more effective O^{th} order methods. The method has proven to be more effective than random search, cyclic search and simulated annealing [6]. The method can experience problems with discontinuous or chaotic objective functions and is therefore not guaranteed to converge in the global optimum.

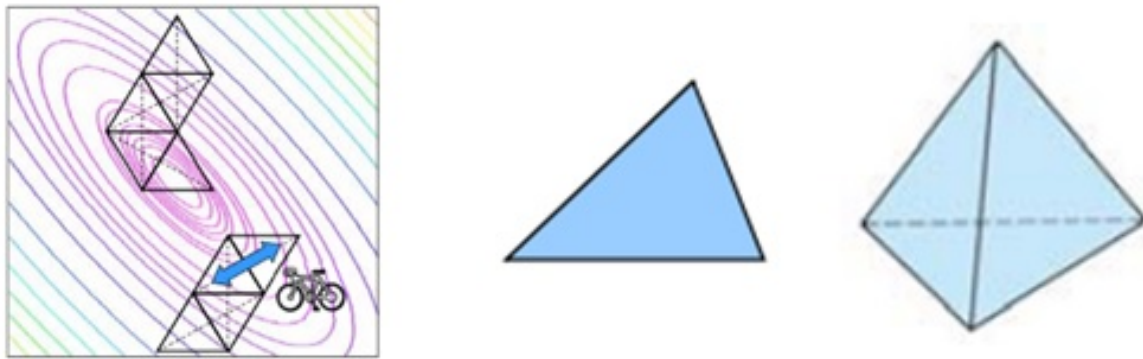


Figure 3.4: Optimisation for 2 design variables with Nelder-Mead-simplex method. Courtesy: [2].

3.5. GENETIC ALGORITHM (GA)

The Evolutionary Optimisation (EVO) method (also Genetic Algorithm (GA)), is based on the theory of evolution by Charles Darwin ("Survival of the fittest" [21]). In this method, design options are encoded in "chromosomal strings" (genes) that contain information from the input variables. In case of a GA, the strings are often (but not necessarily) build from a binary "DNA" string [2]. An algorithm can be used to translate this string into a structure (or specie) and therefore to solve the object function.

GA is a population based algorithm. To achieve optimisation, a set of options \mathbf{x} , "generation i ". This set of options is solved before the algorithm determines a next generation ($i + 1$). In some situations, parallel computing can be used to solve a generation simultaneously and save time. An advantage of the population based principle is that (discontinuous) "Grand Canyon" problems can be avoided because species are placed on both sides of the canyon.

The evolutionary theory states that only the "fittest" samples will survive to reproduce. A new "generation $i + 1$ " is created by combining the "surviving" subjects. This iterative process can be repeated until the stopping criteria (for example: maximum number of iterations, convergence, time limit, etc.) are met. An example of a GA is described by Principles of Optimal Design [6]:

- Step 1: Create an initial population \mathbf{x} . Each string has a certain length (number of variables) and each population has a number of strings. Strings can be created manually or with a certain randomness build in. A large and diverse population is more reliable, but will take longer to converge towards the optima. This can be explained by a larger range of possibilities (covering a bigger domain) and an increased number of options (which would take more time).
- Step 2: Evaluate the "fitness" of all the individuals $f(\mathbf{x})$. The fitness is determined by the result of the object function. Test the termination criteria to see if convergence is already reached. If not, continue with the new population, otherwise, the optimisation is completed.
- Step 3: Select individuals for reproduction. A larger selection results in slower, but more reliable convergence, a smaller one does exactly the opposite.
- Step 4: Create a new population, the selected individuals will reproduce (copy)/ combine /mutate (transform) to achieve this goal. In case of a slow object function, it can be useful to create a database with a history of solutions. In this way, solving strings twice might be prevented (which can save some processing time).
- Step 5: Return to step 2 for the next iteration.

The reproduction procedure of a GA normally has two options, combine and mutate. Figure 3.5 shows how they can be applied. In case of "combination", the best species are selected for reproduction, then the

generation is split at one random position in the string. A new generation is then created by combining the left half of strings with the right half of the other strings. The size of the new population is determined by the selection. The amount of options is equal to the permutation [22]: $\frac{n!}{(n-2)!}$, in which n is number of selected species and the number 2 indicates that the string is cut only once.

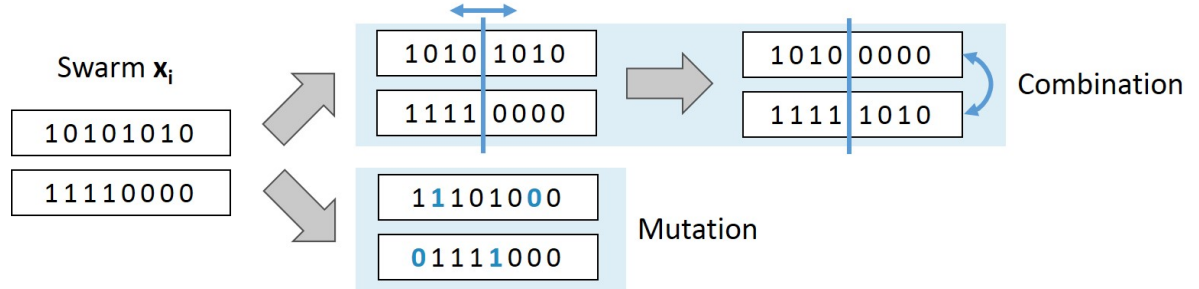


Figure 3.5: Genetic algorithm reproduction types: combination and mutation. Combination lets genes exchange parts of their string. Mutation changes random gene parts to their opposite value. Because only the strongest species survive, only the strong strings get to reproduce and therefore "evolve".

In case of a mutation, each specie is "mutated" at a random location. In this case a "1" becomes "0" and vice versa. The number of mutations per string can be varied by the designer. The consequence of more mutations per string would be slower, but more reliable convergence due assessment of more options.

Properties of GA are [7]: very robust (also work for discontinuous/ non-differentiable functions), global minimum can be found (although this can take an unknown amount of time while it cannot be verified that the result is the actual global optimum unless all options are assessed), many strategies are possible, time consuming. EVO is often applied in case of topology optimisations.

3.5.1. PSEUDOCODE FOR THE GENETIC ALGORITHM (GA).

Source: clever algorithms [23].

```

Data:  $Population_{size}, Problem_{size}, P_{crossover}, P_{mutation}$ 
Result:  $S_{best}$ 

1 Population  $\leftarrow$  InitializePopulation( $Population_{size}, Problem_{size}$ );
2 EvaluatePopulation(Population);
3  $S_{best} \leftarrow$  GetBestSolution(Population);
4 while StoppingCondition() do
5     Parents  $\leftarrow$  SelectParents(Population,  $Population_{size}$ );
6     Children  $\leftarrow \emptyset$ ;
7     foreach  $Parent_1, Parent_2 \in$  Parents do
8          $Child_1, Child_2 \leftarrow$  Crossover( $Parent_1, Parent_2, P_{crossover}$ );
9         Children  $\leftarrow$  Mutate( $Child_1, P_{mutation}$ );
10        Children  $\leftarrow$  Mutate( $Child_2, P_{mutation}$ );
11    end
12    EvaluatePopulation(Children);
13     $S_{best} \leftarrow$  GetBestSolution(Children);
14    Population  $\leftarrow$  Replace(Population, Children);
15 end
16 Return  $S_{best}$ ;

```

3.6. PARTICLE SWARM OPTIMISATION (PSO)

The original Particle Swarm Optimisation (PSO) algorithm(s) were created to simulate social behaviour of animals by Kennedy and Eberhart [9, 24]. PSO is a type of nature inspired algorithm that solves the global optimum and does not require a gradient function. The advantage of the PSO algorithm over a GA is its simplicity. GA algorithms require complex steps like mutations and/or crossovers, where PSO do not. The function is based upon "location" \mathbf{x}_i and "speed" \mathbf{v}_i . The function for the speed parameter shows strong similarities to the explicit numerical derivative "Euler Forwards" technique [13].

The inspiration of the algorithm is a swarm of (simple) animals, like a school of fish or a swarm of birds. Each animal in the "swarm" is a "particle" with limited knowledge. Only 5 types of knowledge are required to perform the optimisation process [4]:

1. Current location of variable (\mathbf{x}_i).
2. Current speed of variable (\mathbf{v}_i).
3. Fitness of particle ($f(\mathbf{x}_i)$).
4. Best location of particle (or personal) record (\mathbf{x}_{pr}).
5. Best location of overall (or swarm) record (\mathbf{x}_{sr}).

Swarm particles behave according "partice intelligence" and "swarm intelligence". This can be explained by a group (the swarm) of humans (the particles) trying to find the fastest route from A to B (considering the 5 types of knowledge as described above). The first iteration, each person (particle) would take it's own (random) route. After arrival, the fastest person (swarm leader) shares it's route with the other particles. The second iteration, each individual adjusts it's route according to it's own experience and the knowledge gained from the group. This process continues until the fastest route is found and all particle record routes become similar (convergence).

Unfortunately, it is not possible to conclude that the global optimum is found. The only ways to conclude this, is when all options are analysed or with some additional information (like the number of possibilities or the exact solution). In situations with a near endless amount of possibilities, it could always be possible to find "fitter" solutions than the current global optimum x_{ur} . Figure 3.6 gives an example of a 2 parameter optimisation ($f(x, y)$) in which a set of particles is searching for the lowest or highest point within the design space. Especially for non-smooth (discontinuous) and multi-modal (like sinusoidal) functions, it can be time-consuming to find the global optimum.

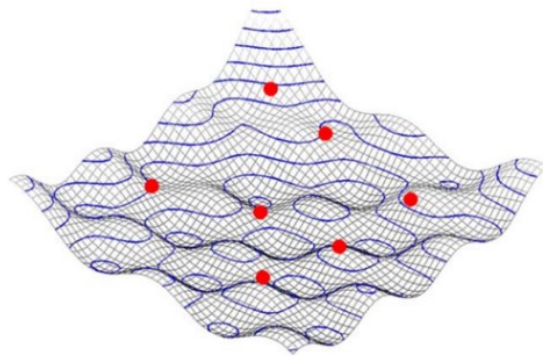


Figure 3.6: Particle swarm optimisation for a function $f(x, y)$. Each "ant" searches the minimum or maximum by taking their speed, personal best location and overall best location into account. Courtesy: Qirong Tang, Universitat Stuttgart.

Figure 3.7 shows the basic principle of a PSO algorithm. The dots and solid arrows display the influence four of the five types of knowledge (location, speed, particle record and swarm record). The fifth type: "fitness"

cannot be found in this figure. "Fitness" is represented by the solution to the object function for some option. A selection procedure to find the best solutions is used during every iteration to find particle and/or swarm record.

Particle Swarm Optimisation

$$\mathbf{v}_{i+1} = \mathbf{v}_i + c_1 r_1 (\mathbf{x}_{pr} - \mathbf{x}_i) + c_2 r_2 (\mathbf{x}_{sr} - \mathbf{x}_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{i+1}$$

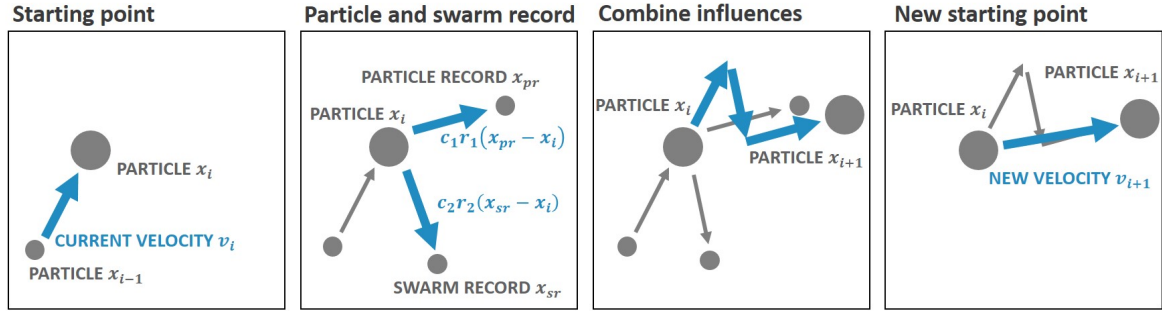


Figure 3.7: Behaviour of a PSO algorithm. the image shows, step by step how the optimisation algorithm combines the influences of current "velocity", particle record and swarm record to determine a "new velocity" and thus a next location. Mind that the images displays only one iteration of one particle in the PSO process. It is also important that the particle and swarm records have to be checked each iteration.

The behaviour of PSO over time can be described as sinusoidal. From the moment that the speed, particle and swarm record vectors point in opposing directions, the speed reduces (due to opposing vectors). During the next iterations, the velocity can shift "turns" towards the current optima. Eventually, this results in an effect similar to a pendulum slowing down at the lowest location after a couple of "swings" due to its momentum.

3.6.1. PSO ALGORITHM MATHEMATICS

Step-by-step PSO algorithm, based on "Introduction to optimum design" [4]:

- Step 0: Initialisation. Select the number of particles N_p , the influence factor for the personal record c_1 (usually between 0.5 and 2.0), the influence factor for the swarm record c_2 (usually between 0.5 and 2.0) and the stopping criteria, like if X_i and X_{i+1} become similar, then stop and the maximum number of iterations before the algorithm stops i_{max} . Set the initial velocity $\mathbf{v}_{i=1} = \mathbf{0}$ and the iteration counter $i = 1$.
- Step 1: Initial generation. Using a random procedure, generate N_p particles \mathbf{x}_i . The procedure should derive the initial generation within the boundaries \mathbf{x}_{min} and \mathbf{x}_{max} and solve the object function $f(\mathbf{x}_i)$. The final step is to determine the best solutions \mathbf{x}_{pr} (particle record) and \mathbf{x}_{sr} (swarm record). In the first iteration, the particle record is equal to the initial score of each particle.
- Step 2: Calculate velocities. The function for velocity contains three influences: initial velocity, personal record and swarm record. Calculate the velocity of each particle as:

$$\mathbf{v}_{i+1} = \mathbf{v}_i + c_1 r_1 (\mathbf{x}_{pr} - \mathbf{x}_i) + c_2 r_2 (\mathbf{x}_{sr} - \mathbf{x}_i) \quad (3.1)$$

Update the positions of the particles as:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_{i+1} \quad (3.2)$$

Check and enforce bounds on the particle positions:

$$\mathbf{x}_{min} \leq \mathbf{x}_{i+1} \leq \mathbf{x}_{max} \quad (3.3)$$

Note that if the velocity becomes small for the entire swarm, this is a sign of convergence. The factors r_1 and r_2 are used to apply some randomness to the solution. The higher the randomness, the larger the

chance of finding a global optimum. A consequence of a large randomness is however a more random and therefore (usually) slower algorithm.

- Step 3: Update the best solution. Calculate the costs function at all new points $f(\mathbf{x}_{i+1})$ and update \mathbf{x}_{pr} and \mathbf{x}_{sr} if $f(\mathbf{x}_{i+1}) \leq f(\mathbf{x}_{pr})$ or $f(\mathbf{x}_{i+1}) \leq f(\mathbf{x}_{sr})$. In the case where it takes some computational power to determine the fitness of a particle, it can be useful to create a database of particles and their solutions. In case a particle already exists, the fitness is copied from the database in order to save time. All new solutions are added to the database.
- Step 4: Stopping criterion. Check for convergence of the iterative process. If a stopping criterion is satisfied (i.e., $k = k_{max}$ or convergence), stop. Otherwise, set $k = k + 1$ and go to Step 2.

The applied parameters in the PSO algorithm are:

N_p	is the swarm size (number of species) $N_1 \dots N_n \dots N_p$
c_1	is a parameter describing the influence of the local record of ant N_n
c_2	is a parameter describing the influence of the global record of all ants
i_{max}	is the maximum number of iterations
\mathbf{v}_i	is the "current" speed of generation i
i	is the iteration counter $1, 2, i, \dots, i_{max}$
\mathbf{x}_i	is the "current" location of generation i
\mathbf{x}_{min}	is the lower bound value of \mathbf{x}
\mathbf{x}_{max}	is the upper bound value of \mathbf{x}
$f(\mathbf{x}_i)$	is the object function.
\mathbf{x}_{pr}	is the location of the particle (/local) record for each ant i
\mathbf{x}_{sr}	is the location of the swarm (/global) record for all ants
\mathbf{v}_{i+1}	is the new speed of generation $i + 1$
r_1	is a parameter which introduces a randomness to avoid local convergence
r_2	is a parameter which introduces a randomness to avoid local convergence

In order of finishing the explanation of the optimisation algorithm, the flowchart in Figure 3.8 is created. The flowchart shows steps that have to be taken to realise the optimisation process. Note that the algorithm described is one for a minimisation process. In a maximisation algorithm, the personal and swarm records would search for the largest value instead of the smallest one.

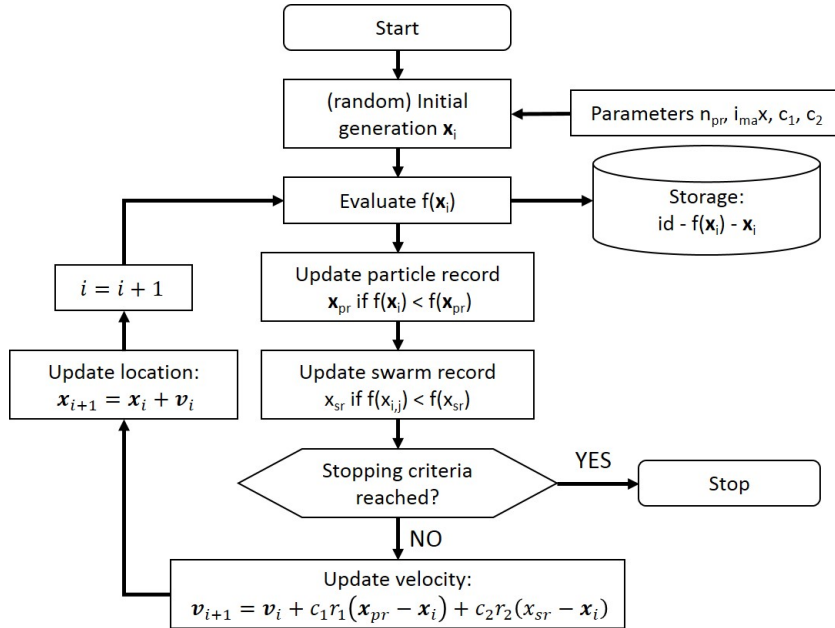


Figure 3.8: Flow chart of a PSO algorithm. Noticeable parts are among others the evaluation of the objective function $f(\mathbf{x}_i)$ for option \mathbf{x}_i and the set of rules used to determine the next set of solutions that are to be evaluated. Mind that this flowchart is slightly different from the description in this section, which is explained by the overlapping parts of the initialisation phase and the iterative phase.

3.6.2. PSEUDOCODE FOR THE PARTICLE SWARM OPTIMISATION (PSO) ALGORITHM.

Source: Introduction to optimum design [4].

```

Data:  $SwarmSize, x_{min}, x_{max}, c_1, c_2, r_1, r_2$ 
Result:  $F_{sr}, X_{sr}$ 

1  $iteration = 0;$ 
2 Generate swarm of random solutions  $x_i, i = 1, 2, \dots, SwarmSize;$ 
3 Check domain  $x_i \in [x_{min}, x_{max}]$ ;
4 foreach Particle in swarm do
5      $x_{pr} = x_i;$ 
6      $v_i = 0;$ 
7     Evaluate fitness  $f(x_i);$ 
8     Store  $f(x_i)$  in database;
9 end
10 while stopping condition() do
11     Select swarm leader  $S_{sr};$ 
12     Update particle velocity  $v_{i+1} = v_i + c_1 r_1 (x_{pr} - x_i) + c_2 r_2 (x_{sr} - x_i);$ 
13     Update particle position  $x_{i+1} = x_i + v_{i+1};$ 
14     Check domain  $x_i \in [x_{min}, x_{max}]$ ;
15     foreach Particle in swarm do
16         if  $x_i$  in database then
17             Retrieve  $f(x_i);$ 
18         else
19             Evaluate fitness  $f(x_i);$ 
20             Store  $f(x_i)$  in database;
21         end
22         If  $f(x_{pr}) > f(x_i)$  Update particle record  $x_{pr} = x_i;$ 
23     end
24      $iteration = iteration + 1;$ 
25     update stopping condition();
26 end
27 Return  $f_{wr};$ 

```

3.6.3. PSO ALGORITHM PROPERTIES

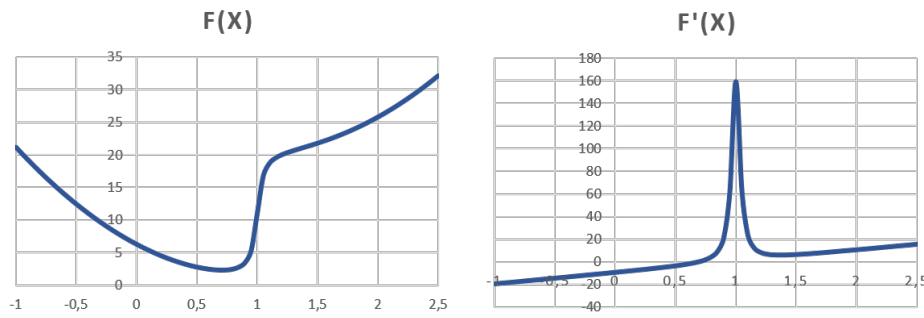
In the main report of this Master's Thesis, PSO is chosen as the preferred algorithm for optimisation. Some experimentation has been performed in order to understand the behaviour of a PSO algorithm in practice. The behavioural study is a useful tool to help users understand what constants to select for the optimisation process to achieve the required behaviour. If the factors N_p, c_1, c_2, r_1 and r_2 are chosen correctly, then this results in a fast and efficient algorithm. Appendix A.2 contains an overview of some experimentation with a PSO algorithm in case of a simple object function and a small amount of design variables.

The object function chosen for this the analysis is similar to the penalty function in Section 4.3. This function is:

$$f(x) = 1 + 5(x - 1)^2 + \frac{20}{\pi} \tan^{-1}(25(x - 1)) \quad (3.4)$$

ANALYSIS OF GLOBAL OPTIMUM WITHOUT OPTIMISATION

The first step of the process is to determine the global optimum. This is required to verify the accuracy of results found in the analysis. As can be seen in the plot of the function and it's derivative, the global minimum is near $x = 0.8$.

Figure 3.9: Function $f(x)$ and it's derivative.

A more exact computation results in an analysis of the differential equation to find minima and maxima for $f'(x) = 0$. Given Figure 3.9, there is only one minimum in the range $x \in [-1; 2,5]$.

$$f'(x) = 10(x-1) + \frac{20}{\pi} \frac{1}{(25(x-1))^2 + 1} = 0 \quad (3.5)$$

A simplification of this function results in:

$$-625x^3 + 1875x^2 - 1876x + 610,08 = 0 \quad (3.6)$$

In order to solve this equation, the Newton Rapson method [13] is applied. This results in:

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)} \quad (3.7)$$

with:

$$g(x) = -625x^3 + 1875x^2 - 1876x + 610,08 \quad (3.8)$$

$$g'(x) = -1875x^2 + 3750x - 1876 \quad (3.9)$$

After some iterations, convergence is achieved in $x = 0.70761$, which coincides with the global minimum of the objective function $f(x)$. Part of the Newton-rapson computation is shown in Table 3.1.

Iteration i	1	2	3	4	5	6
input x_i	0,80000	0,65901	0,70104	0,707468	0,70761	0,70761
$g(x_i)$	-10,716	9,206	1,108	0,023	1,11E-05	2,59E-12
$g'(x_i)$	-219,0	-168,6	-161,5	-161,3	-161,3	-161,3

Table 3.1: Results of the first 6 iterations of the Newton Rapson method. Convergence is found at the fifth iteration. The value $x_6 = 0.70761$ is the global minimum of $f(x)$.

BEHAVIOURAL ANALYSIS RESULTS

In case of a large number of particles N_p , the initial diversity of the swarm is relatively large. A large number of particles may result in slower iterations (n times a certain equation takes less time than $n + +$ times the same equation). The larger diversity results a larger probability on finding a "fitter" solution in the first iteration. An advantage of a large diversity is the larger probability of finding the global optimum. If a group of particles

finds a local optimum, the rest of the swarm can still find the global one. Figure 3.6 shows an example of a "fitness" landscape with some local and a global solution. The effect of a larger population can be clearly seen in this figure. A larger (spread) swarm would have a larger probability of finding the optimum.

Figures 3.10, 3.11 and 3.12 demonstrate the influence of factors c_1 and c_2 . Literature states that the value of these factors has to be between 0,5 and 2,0. The images show what will happen in case this is not done. In case of a small c-factor, step sizes get small but accurate. This accuracy comes with a price. More steps have to be taken to achieve the optimum and the method is therefore slower than one with a higher c. In case the c-factor is too large, particles start moving too fast. In this situation, the a particle shows a sinusoidal behaviour with an increasing amplitude. The result is high, or even diverging particles, resulting in inaccurate solutions of the algorithm. In case the factor $c_i * r_i > 1$, particles will tend to move past their optima because of larger step sizes. This results in a more random behaviour of the algorithm.

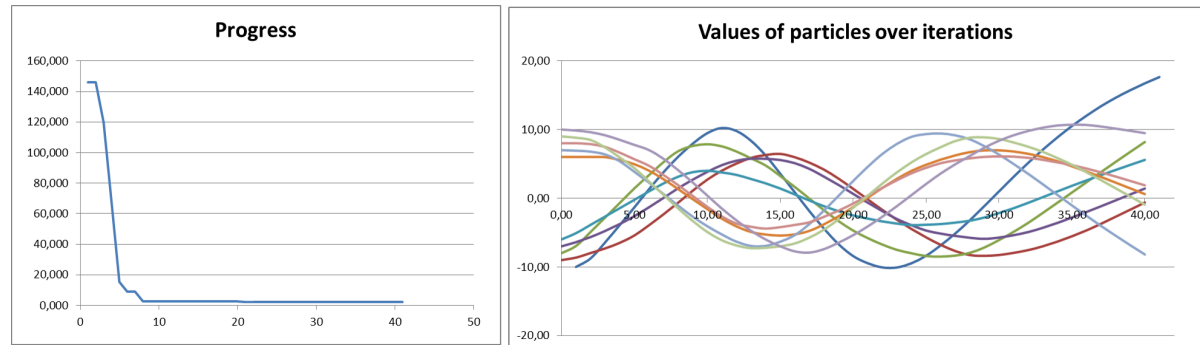


Figure 3.10: Particle progress for $c_1 = c_2 = 0,1$. A low value for c_1 and c_2 can result in accuracy, but slow convergence.

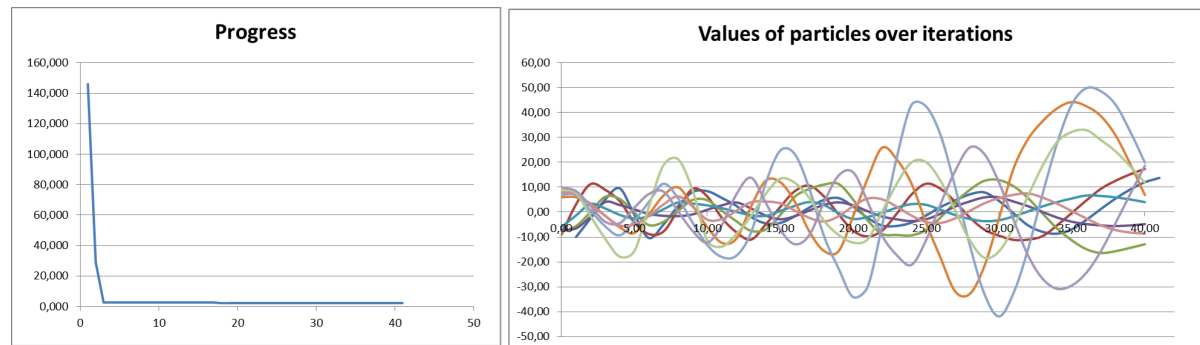


Figure 3.11: Particle progress for $c_1 = c_2 = 1$. A normal value for c_1 and c_2 should result in reasonable accuracy and convergence.

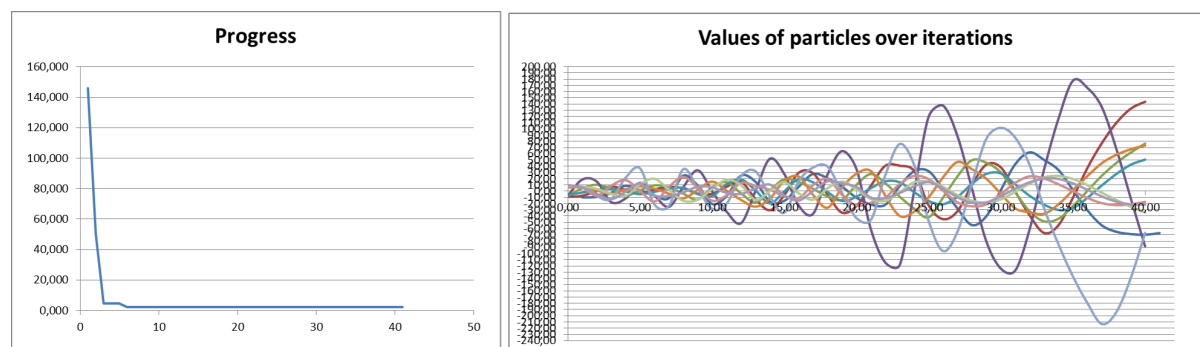


Figure 3.12: Particle progress for $c_1 = c_2 = 2$. A high value for c_1 and c_2 can result in low accuracy and divergence of parameters. Note that particles tend to behave much more random in this situation.

The function shown in Figures 3.10, 3.11 and 3.12 is the result of a 50 step iteration process with 10 ants and one parameter. the object function used for this test is the combined quadratic-arctangent penalty function shown in section 4.3.

The final parameters of a PSO algorithm are r_1 and r_2 . The main purpose of these parameters is to include randomness in the optimisation. Randomness can be a useful tool to escape from local optima and increase the probability of finding the global optimum. The larger the influence of r_1 and r_2 , the more an algorithm will represent a brute force algorithm (see 3.1). As a consequence, a more random algorithm is usually slower in convergence. The word "usually" is used in this statement because a random optimisation technique has some chance on finding the optimum based on a "lucky shot".

3.7. ANT COLONY OPTIMISATION (ACO)

Ant Colony Optimisation (ACO), which was invented in 1992 by Marco Dorigo, which performed a Ph.D. Thesis on the subject [25]. The algorithm is based on the behaviour of ants in nature, especially on their "food-searching-behaviour". The best known application of this algorithm is the so called "traveling salesman problem" [18]. This problem can be solved quite directly with this algorithm because inspiration of the system is very similar.

ACO is part of the Discrete, stochastic, nature inspired and swarm intelligence classes of algorithms. ACO is closely related to PSO, which is also a nature inspired swarm class algorithm. This means that the algorithm can find global optima for discontinuous and multi-modal problems, but the required amount of time to solve a problem is unknown.

The description of ACO is done best with its natural inspiration. The process starts with a number of ants that "wander" from their colony into the environment. Once an ant locates a food-source, it returns to the colony while leaving a trail of pheromones. The purpose of the pheromones is to mark the route for future ants. Over time, this marker evaporates and therefore gets weaker. When new ants leave the colony, they have a larger probability of following the paths which are marked strongest with pheromones (due to short routes or more visits). This way, the shortest route to a food source gets more popular and therefore even more pheromones. Due to the evaporation process, only the shortest routes will "survive" because ants have a higher probability of choosing this route, leaving other routes relatively unvisited. After a while the algorithm will focus on a global optimum solution.

The amount of pheromone that ant i assigns to option j is defined by:

$$\tau_{i,j} \leftarrow (1 - \rho) * \tau_{i,j} + \sum_{k=1}^m \Delta \tau_{i,j}^k \quad (3.10)$$

The probability of ant i to pick option j as (partial) solution for the problem is computed by the probability function. This function is described by:

$$P_{i,j} \leftarrow \frac{\tau_{i,j}^\alpha \eta_{i,j}^\beta}{\sum_{k=1}^c (\tau_{i,j}^\alpha \eta_{i,j}^\beta)} \quad (3.11)$$

in which:

i	Ant number in colony
j	design variable number
m	Number of ants
k	Current ant number
$\tau_{i,j}$	Pheromone of component
ρ	Decay factor
$\sum_{k=1}^m \Delta \tau_{i,j}^k$	$1/S_{cost}$

$P_{i,j}$	Probability of selection of component
α	Heuristic coefficient
β	History coefficient
c	Set of usable components
$\eta_{i,j}^k$	Contribution of overall score to selecting component

3.7.1. PSEUDOCODE FOR THE ANT COLONY OPTIMISATION (ACO) ALGORITHM.

Source: Clever algorithms [23].

```

Data: ProblemSize, PopulationSize, m,  $\rho$ ,  $\alpha$ ,  $\beta$ 
Result:  $P_{best}$ 

1 Create heuristic solution  $P_{best}$ ;
2 Costs  $P_{best, costs}$  of  $P_{best}$ ;
3 Initialize pheromone;
4 while stopping condition() do
5   Determine candidates  $S$ ;
6   for  $i = 1$  to  $m$  do
7     Probabilistic stepwise construction;
8     Analyse costs  $S_{i, Costs}$ ;
9     if  $S_{i, Costs} \leq P_{best}$  then
10       $P_{best, costs} = S_{i, Costs}$ ;
11       $P_{best} = S_i$ ;
12    end
13  Candidates;
14 end
15 Decay of pheromone;
16 foreach Solution in candidates do
17   update pheromone;
18 end
19 end
20 Return  $P_{best}$ ;

```

3.8. GRADIENT BASED FIRST ORDER METHODS

First order methods are usually recognisable for their capability to use the gradient of the object function. When the gradient of a function is known, it becomes easier to find the optimum, since it can now be known what direction to search for. First order methods, like the “Steepest decent method [6]” are usually based on numerical algorithms (often in the form of Taylor polynomials [3]) to estimate the optimum function. 1st order methods in general are more powerful, but less robust than 0th order methods. If not used carefully, they can diverge or suffer from numerical errors. The steepest descent method is described below:

- Step 1: start with a (random) arbitrary \mathbf{x}_1
- Step 2: Set first search direction $\mathbf{d}_1 = -\nabla \mathbf{f}_1$
- Step 3: Line search to find the next point: $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$
- Step 4: Update search direction: $\mathbf{d}_{i+1} = -\nabla \mathbf{f}_{i+1}$
- Step 5: Repeat the process from step 3 on until the stopping criterion (convergence) is reached.

3.9. GRADIENT (HESSIAN) BASED SECOND ORDER METHODS

The second order Newton method [6] is considered to be more powerful than the first order methods. In this method, the second order derivative is also used in a second order Taylor series. This function is capable of finding the minimum value of a quadratic function in 1 step, but if the Hessian function H is not positive, divergence will occur. The results of this method are very efficient, but not robust. According to the literature, robustness-problems might be avoided via certain additions to the method.

Local approximation: 2^{nd} order Taylor polynomial:

$$f(\mathbf{x} + \Delta\mathbf{x}) \cong f(\mathbf{x}) + (\nabla f)^T + \frac{1}{2} \Delta\mathbf{x}^T H \Delta\mathbf{x} = \tilde{f}(\Delta\mathbf{x}) \quad (3.12)$$

in which:

The gradient [13] function is:

$$\nabla f = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \dots \\ \partial f / \partial x_n \end{bmatrix}$$

The Hessian matrix [13] is:

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

If the step-size is derived from the equation above, this results in: $\nabla \tilde{f}(\Delta\mathbf{x}) = \mathbf{0} \Rightarrow \nabla f + H\Delta\mathbf{x} = \mathbf{0} \Rightarrow \Delta\mathbf{x} = -H_{-1}\nabla f$. Summarised, this method can be displayed in the following steps:

- Step 1: Step-size: $\mathbf{s}_k = -H_{-1}\nabla f$
- Step 2: Update: $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
- Step 3: Check for divergence and/ or numerical errors (as is usual in numerical functions)

3.10. COMPARING ALGORITHMS

The selection of an optimisation algorithm depends on the type of problem (concept of "No free lunch" [12]). Especially the amount of design variables, the type of solution space (continuous/ discontinuous, smooth/ chaotic), the computational intensity of the object function and the required time and accuracy can be influential in the selection process. In some cases, it is possible to design functions such that they compliment the properties of the selected optimisation algorithms.

Table 3.2 summarises a number of properties that can be useful when selecting an algorithm. The properties in this table can be used to exclude classes of optimisation algorithms. In general, gradient based order methods are faster, but result in complex, time-consuming object functions. Stochastic methods are very robust, but usually very slow and cannot guarantee that the global optimum is found before all options are analysed.

Algorithm	Primary type	Solution space	Population based	Stability	Efficiency
Brute Force	Stochastic	Discontinuous	No	Very robust	$n < 10$
Simulated Annealing SA	Stochastic	Discontinuous	No	Very robust	$n < 10$
Cyclic coordinate search	Direct search	Continuous	No	Local optima	$10 < n < 50$
Nelder-Mead simplex method	Direct search	Continuous	No	Local optima	$10 < n < 50$
Genetic Algorithm GA	Nature inspired	Discontinuous	Yes	Robust	$50 < n < 500$
Particle Swarm Optimisation PSO	Nature inspired	Discontinuous	Yes	Robust	$50 < n < 500$
Ant Colony Optimisation ACO	Nature inspired	Discontinuous	Yes	Robust	$50 < n < 500$
First order methods	Gradient based	Continuous	No	Not robust	$n > 1000$
Second order methods	Gradient based	Continuous	No	Not robust	$n > 1000$

Table 3.2: Comparison of optimisation algorithms.

Figure 3.13 shows a number of algorithm classifications. Because classifications sort algorithms on certain properties, they can be useful for a comparison. In the main report, [PSO](#) is selected as preferred optimisation algorithm. A motivation is given in the main report.

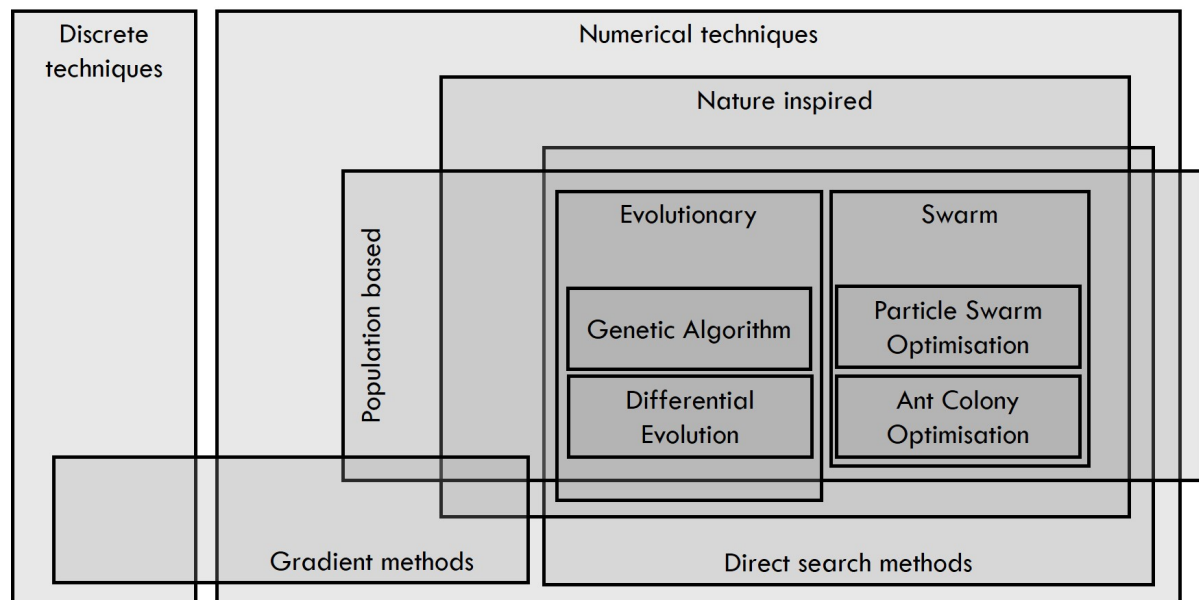


Figure 3.13: relation of algorithm to other optimisation techniques

4

OBJECT FUNCTION

4.1	Types of solution spaces	35
4.2	Score function	37
4.2.1	Common structural optimisation goals	37
4.3	Penalty functions and constraints	38
4.3.1	Logarithmic or log barrier [4] penalty function	38
4.3.2	Quadratic or non-linear [7] penalty function	39
4.3.3	Unit-step or multiple segment [7] penalty function	39
4.3.4	Arctangent penalty function	40
4.3.5	Combined arctangent + quadratic penalty function	40

The goal of optimisation processes is to find the design variable(s) which result in the highest or lowest value of the object function. The object function is what describes the "goodness" of options. Mathematically, this means: "Find $x_{optimum}$ resulting in the minimum value of the object function $f(x_{optimum}) = \min(f(x))$ ". The influence of the type or shape of functions is significant. Some optimisation algorithms run more efficient with certain types of functions than others. Especially subjects like: smoothness, continuity and differentiability are important factors. For example: numerous algorithms require derivatives of object functions to run, such algorithms are impossible if the derivative cannot be found.

Object functions usually contain two parts: score and penalty. The score defines the ultimate goal of the process, for example: "find the fastest route from point A to point B ". The penalty function contains the (negative) reaction to certain constraints. A simple example of a penalty function is: "if your speed exceeds a certain level, you have to wait x minutes". In these examples, moving too fast will costs time and will therefore make certain options less feasible. The balance of between score and penalisation can influence the global optimum and have to be treated carefully. In some situations, a small penalty is no problem, in others (like the case of structural safety), penalty is not allowed at all.

The goal of this chapter is to improve knowledge of object functions in general in order to select and/ or construct efficient algorithms and functions. The first section explains the influence of solution spaces of object functions on optimisation algorithms. The other sections describe the basic principles of score and penalty functions, which are respectively used to determine the goodness and to penalise options that do not fit restraints

4.1. TYPES OF SOLUTION SPACES

In practice, there are many types of solution spaces for object functions. Certain algorithms work better with certain solution spaces. For this reason, it is important to understand the shape of the solution space to solve

a problem. In order to give simple examples of the solution space, this section focusses on problems with 1 or 2 parameters. An increase in parameters increases complexity of the solution space, but the comparison between types of solutions is similar.

For the selection of certain classes of optimisation algorithms, it is important to know if object functions are continuous or discontinuous, smooth or multi-modal and if gradients are easily determined or not. Changing parameters can significantly influence the score. For example: if only one element in a structure is made slightly stronger, they usually has only a small influence on the solution, but if the material is changed, the entire solution can be different.

The basic types of object functions are displayed in Figure 4.1. Smooth (parabolic) functions are the best possible situation. Smooth functions have only one optimal area for each parameter. The gradient of a smooth function can usually be determined with relative ease. Multi-modal functions are "the next best" situation. Although there are multiple (local) optima, the gradient function is still solvable. Sinusoidal functions are an example of multi-modal situations.

Chaotic and discontinuous functions are harder to solve. Gradients of these functions can be misleading or hard to find. The discontinuity can also force algorithms into local optima, for example due to the effect of "Grand Canyon" effects. Flat functions can disable the progress of optimisation algorithms due to their constant value. If processes cannot determine what direction to go to, they tend to work as random search algorithms or assume that the optimum is found.

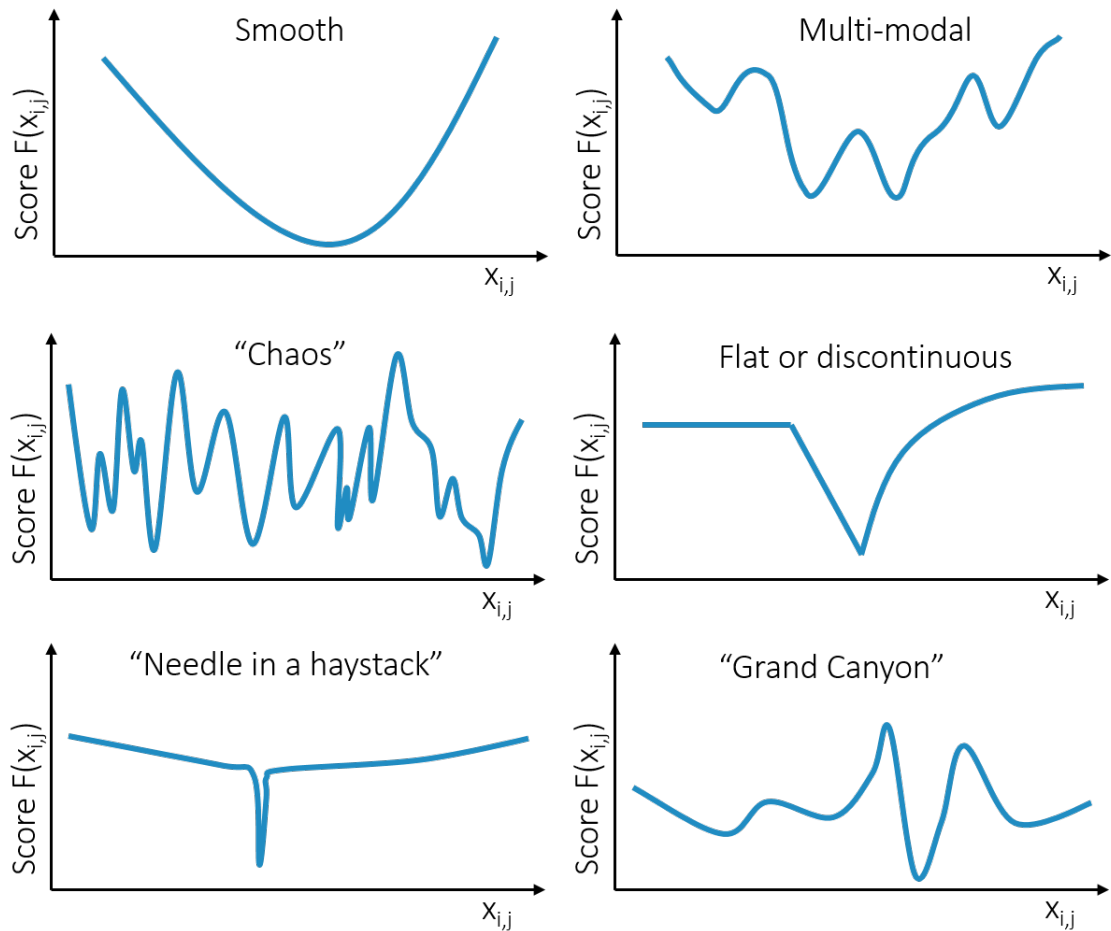


Figure 4.1: Different types of object function solution spaces.

One very hard function to solve is the "Needle-in-a-haystack" problem[18]. These are relatively flat functions with one "spike" at a random location (see Figure 4.1) which represent an optimum. The problem with such

functions is that "spikes" are hard to find and/or identify. Thus it is often unknown whether or not they exist, where they are and if they represent an optimal situation. Solution spaces that contain "Needle-in-a-haystack" problem can be misled by their gradient and functions have trouble noticing "spike" optima due to their small size.

4.2. SCORE FUNCTION

In Optimisations, object functions determine the "fitness" (or quality) of certain options. The object function generally contains two parts: the score and the penalty. The configuration of score and penalty depends on the chosen optimisation goal and functions [4]. Common configurations in practice are: $F_{object} = F_{score} + F_{penalty}$ and $F_{object} = F_{score} * F_{penalty}$.

The score-function is determined by the optimisation goal. There is a large variety of possible goals for objective functions. Because the shape of the score function depends on its goal, it is considered to be hard to determine a standardised score function. It makes more sense to analyse the principals of objective function analysis and design them such that they perform well in optimisation algorithms. This can be achieved by designing smooth, continuous functions with proper boundary values.

It can be useful to influence the output of the score functions such that it is within a certain range. By controlling the score, it is partially possible to manage the interaction between the score and penalty function. A proper score-function should at least answer some basic questions. Three such questions that, according to "Principles of Optimal design" of Papalambros [6] need to be answered are:

- How are designs, and especially their differences described?
- What is the criterion for "best" design?
- What are the available means?

These questions describe the usual challenges in the definition of optimisation problems. score-functions should be a relatively accurate model of the problem. There should be a clear definition of the "best" (or optimal) score and functions can only be described by available information.

4.2.1. COMMON STRUCTURAL OPTIMISATION GOALS

Structural optimisation can be performed on numerous goals, but is mostly applied to minimise costs or to improve the safety of a structure within certain limits. Common goals are:

- Economical: minimise costs or maximise profit
- Weight: safe costs by removing material or make a structure float in water.
- Geometrical: improve safety, optimise use of floorspace.
- Risk: improve safety, reduce construction risks or improve reliability.
- Time: Reduce construction time

For this Master's Thesis, it is chosen to work with costs minimisation of reinforced concrete structures. A major argument for this objective is that costs minimisation is usually the purpose of optimisations. Many optimisations with other goals are applied to achieve this goal. In case of reinforced concrete optimisation, costs are a more stable optimisation goal compared to weight and/ or geometry. A weight or geometrical optimisation would not necessarily result in the cheapest solution due to the mix of concrete and reinforcement. This problem is discussed in Chapter 6.

4.3. PENALTY FUNCTIONS AND CONSTRAINTS

Penalty functions represent the constraints of the object function. For example: "for $f(x) = x^2$, find the value $x_{optimum}$ resulting in the lowest $f(x) = f_{min}$ while $x_{min} < x < x_{max}$ ". In this situation the limitation $x_{min} < x < x_{max}$ can be achieved by both a constraint or a penalty function. A constraint would prevent function $f(x)$ from attempting options out of $x_{min} < x < x_{max}$. A penalisation would ensure that "out-of-range" options are not feasible.

Penalisations and/or constraints can be applied on several levels in a problem. It is advised to design constraints and penalisations in the same picking order as safety regulation strategies. Problem solving according to safety regulations is usually applied in the following order:

1. Prevent problem from occurring (no measures requires)
2. Solve problem at source (apply limitations to prevent situation(s) from occurring: $x_{min} < x < x_{max}$)
3. Prevent problem at location of "pain" (apply smooth penalty functions that work for every x)
4. Limit "pain" at location (apply non-smooth penalty functions that do not work for every x)

By applying this principle to penalty functions, a design strategy is presented with several advantages. Preventing problems from occurring results in a more reliable model since boundary condition problems do not occur. Giving a range to design parameters limits the amount of options, resulting in a faster process. Preventing problems at location results in smooth penalty functions that still show the direction of the best solution. The final measure works, but potentially disrupts the model by non-existent or infinite values at penalty locations.

It is important that the influence of the score and penalty function onto each other is carefully designed. Large influences of one function can undermine the influence of the others, resulting in solutions based on score *or* penalty instead of score *and* penalty. This effect can be illustrated by a penalty function that dominates the object function. Optima of such functions are found at the minimum value of the penalty function. This effect influences the optimisation such that the objective, which is described by the score function, is neglected. In case of structural optimisation, this can result in a strong (very safe), but expensive solution due to overdimensioning of elements.

In general, it is advised to design object functions such that penalty functions have most influence when penalisations are required, but are minimal when they do not. The shape of penalisation functions influences the effectiveness of optimisation algorithms similarly to score functions. Therefore, the penalisations work best if they are smooth, continuous and differentiable.

In the situation that a penalty function *is* required, there are a number of options. If possible, it is more profitable to prevent a penalty situation from happening. For example, if an optimisation variable has to be within a certain range $x_{min} < x < x_{max}$, it is preferred not to analyse options that are "out-of-range". Another advantage of this principle is the limitation of the amount of possible solutions and so a computational improvement. Especially direct constraints on the design parameters are suited for this principle.

If a penalty function is required, then there are numerous possibilities. It can be useful to design custom penalty-functions to achieve project related behaviour for certain situations. A number of penalty types are discussed in the rest of this paragraph.

4.3.1. LOGARITHMIC OR LOG BARRIER [4] PENALTY FUNCTION

A logarithmic penalty function [7] (see Figure 4.2) is a useful "barrier" penalty function. The real value of a logarithm will grow towards infinity from some infeasibility point. Real valued solutions for this function do not exist for values larger than the infeasibility point. This has some consequences for the use of this type of functions in algorithms. Optimisation algorithms have to be able to detect the "barrier" before it is reached. Therefore, the application of logarithmic functions limits the step-size and therefore speed the

optimisation process. If for some reason the infeasibility point is passed, than the non-existing points of the penalty function result in a crash of the algorithm.

The nature of this function ensures that any option that does not fulfil the requirements is penalised with an extreme penalty. The main advantage of this function is also its disadvantage. In case of infinite values, optimisation algorithms tend to give extreme results. Algorithms will therefore avoid such points. To avoid such situations, algorithms take small steps in order to detect and react the slope of a logarithmic function before the infeasibility point is reached. A consequence of this technique is that the first iteration should never create a penalty to avoid infinity results. This system can be problematic in case of "Grand-canyon" problems, because algorithms can hardly cross the Canyon.

$$y = a - \log(x - b) \quad (4.1)$$

Smooth function?	yes.
Solvable for every x?	singularity after penalty value.
Penalty application speed:	yes (exponential).
Lowest penalty direction:	lowest penalty at $x = 0$, $x \in \langle 0, \text{inf} \rangle$.
Penalty type:	4: limit pain.

4.3.2. QUADRATIC OR NON-LINEAR [7] PENALTY FUNCTION

Quadratic penalty functions (see Figure 4.2) are relative simple solutions. These functions tent to penalise all functions the further they become from a certain value. The advantage of this principle is that optimisation algorithms tend to search for these points. A disadvantage is that the relatively slow penalty may allow values above certain points while this is not allowed. In some situations, it can be useful to combine this function with other penalty functions in order to create the lowest penalty score on a location of choice. Quadratic functions are especially useful in case all functions are penalised and if penalisation of options is acceptable.

$$y = a + b * (x - d) + c * (x - d)^2 \quad (4.2)$$

Smooth function?	yes.
Solvable for every x?	yes.
Penalty application speed:	no, penalties increase for larger and smaller x.
Lowest penalty direction:	yes.
Penalty type:	3: solve at location.

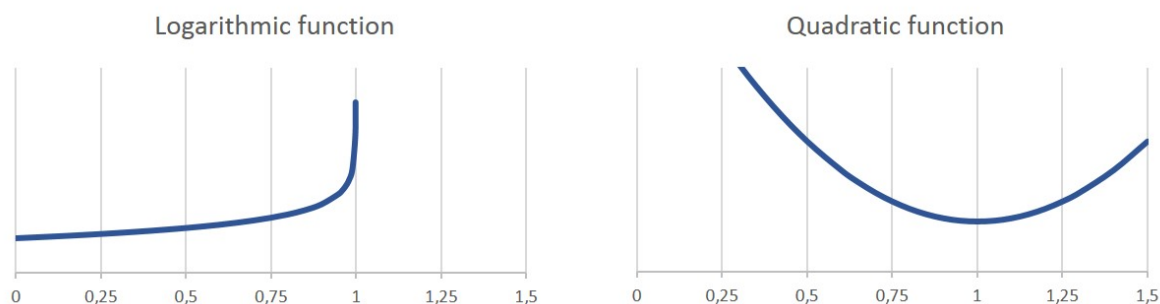


Figure 4.2: Logarithmic (left) and quadratic (right) penalty functions

4.3.3. UNIT-STEP OR MULTIPLE SEGMENT [7] PENALTY FUNCTION

The unit-step function [26] (see Figure 4.3) is a direct translation of the required effect of a penalisation. This function can be designed to have no effect when no penalty is required, but can have extreme effects if

this is. Unfortunately, a unit-step function is not a smooth function and contains primarily constant values. This results in a major disadvantage. Optimisation algorithms can hardly determine the location x where a function is penalised. This can be explained by the derivative of the penalty function. The derivative of a unit step function is zero for all values of x , with a infinitely small pulse located at the step. For this reason, no information on the location of the step can be found in other functions of x then the step itself.

$$\begin{cases} x \leq a & y = b \\ x > a & y = c \end{cases} \quad (4.3)$$

Smooth function?	no.
Solvable for every x ?	yes.
Penalty application speed:	yes (infinite speed).
Lowest penalty direction:	no, constant values.
Penalty type:	4: limit pain.

4.3.4. ARCTANGENT PENALTY FUNCTION

A arctangent [3] shaped penalty function (see Figure 4.3) looks much like a unit-step function. A major difference between these two functions is the "smoothness" of the arctangent function. This smoothness partially solves the optimisation algorithms inability to find the location of the penalty. The solution can also create a disadvantage. This type of functions has no constant values at the beginning and end. Due to this effect, an optimisation algorithm may consider options with a very low penalty "fitter" while this is not necessarily the case. The steepness and amplitude of arctangent function should therefore be designed carefully.

$$y = \left(a + \frac{b}{2}\right) + \frac{b}{\pi} * \tan^{-1}(c(x-d)) \quad (4.4)$$

Smooth function?	yes.
Solvable for every x ?	yes.
Penalty application speed:	yes.
Lowest penalty direction:	lowest penalty at $x = 0$, $x \in (0, \text{inf})$.
Penalty type:	3: solve at location.

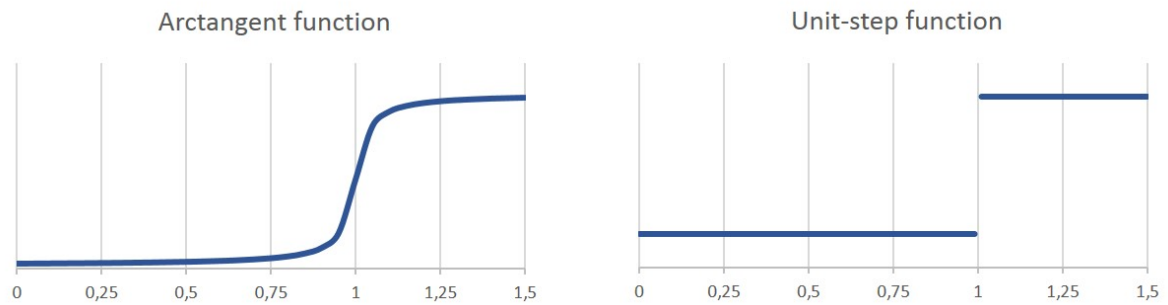


Figure 4.3: Arctangent (left) and unit-step (right) penalty functions

4.3.5. COMBINED ARCTANGENT + QUADRATIC PENALTY FUNCTION

For some penalty functions, it is advantageous to combine several functions. For example, if a normal force unity check is performed for a structural element, the unity check $\left(\frac{\text{designstress}}{\text{limitstress}}\right)$ may never be higher than a certain value, but optimal usage of the material is normally found just below this point. To achieve this effect, a combination of a quadratic and arctangent function can be made (see Figure 4.4). This function avoids the disadvantage of an arctangent to search for the lowest penalty while maintaining the influence of

the arctangent function to create infeasibility near the penalty point. In case of the situation in Figure 4.4, the best values are found near a unity check of 0,9, although the characteristics of the function can be designed to fit some given requirements.

$$y = (a + b * (x - d) + c * (x - d)^2) + \left(\left(e + \frac{f}{2} \right) + \frac{f}{\pi} * \tan^{-1}(g(x - d)) \right) \quad (4.5)$$

Smooth function?	yes.
Solvable for every x?	yes.
Penalty application speed:	yes (arctangent effect).
Lowest penalty direction:	yes (quadratic effect).
Penalty type:	3: solve at location.

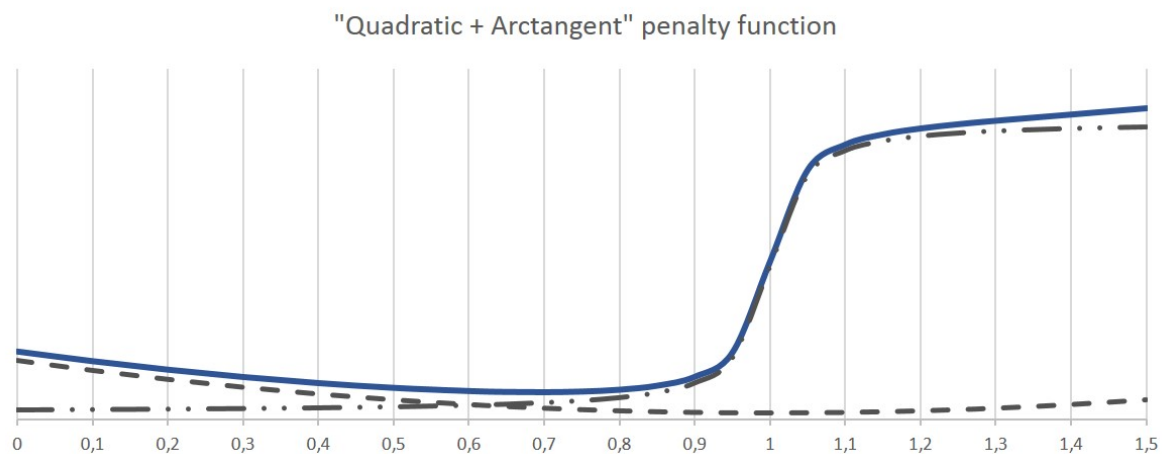


Figure 4.4: Combined arctangent + quadratic penalty function

5

GEOMETRY DESIGN

5.1 Method 1: Triangulation	44
5.2 Method 2: Building blocks (Lego's)	45

The main report uses truss models to represent reinforced concrete geometry. There are multiple techniques capable of systematically designing such truss models. This chapter is used to present a number of such techniques. Eventually, users can decide what technique to use, or to design a truss by themselves. The resulting truss is allowed if it fulfils certain requirements (which are described in the main report). These requirements are shown in Figure 5.1. Major requirements include: Strut-and-Tie Model (STM) requirements, geometrical limitations, and truss configuration requirements.

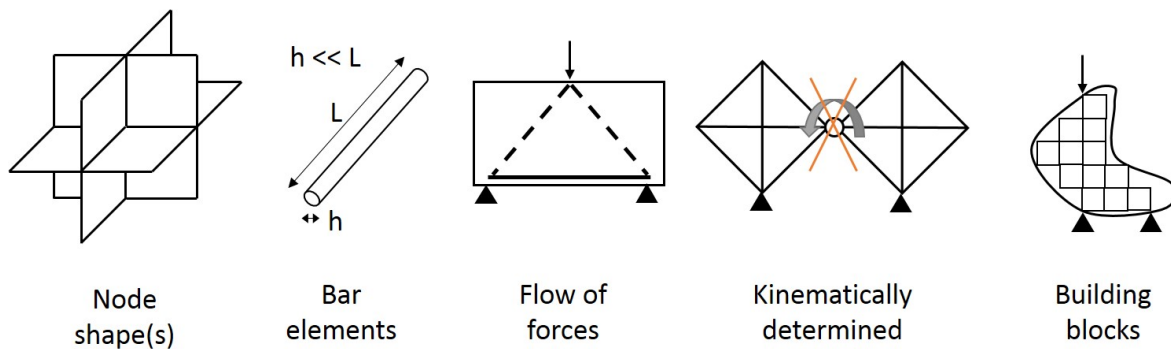


Figure 5.1: Limiting conditions for STM truss design: node shape (due to Eurocode 2: Design of concrete structures, NEN-EN 1992-1-1 [37] (EC2) analysis), bar elements (due to truss methodology and volume limitations), flow of forces (the truss should be capable of representing the flow of forces), kinematics (truss should not be able to move to ensure matrix displacement analysis stability) and building blocks (modular and systematic approach).

The type of optimisation parameters is influenced by the design of the truss model. There are basically two optimisation approaches. The first approach generates a standardised truss from a structure, resulting in a shape optimisation (Figure 2.6). In this case, the parametric design of the shape gives the input parameters. The amount of bars in the truss, and their locations, might change during the process. This approach allows optimisation of the structural shape. The other approach generates a sort of 3D mesh inside a given shape. In this approach represents a sizing optimisation. In this case, the amount of parameters is equal to the amount of bar elements. This approach specialises in the optimisation of reinforcement in a constant shape.

For structural design of a constant shape, the costs optimisation can be reduced to "optimising the reinforcement within a certain shape". Reducing the amount of reinforcement (given a constant complexity of that reinforcement) in a constant shape is identical to reducing the costs of such structure. This can be stated

because the amount of concrete and the shape of the formwork are constant and will therefore have no influence on the costs of the element. The only major costs parameter left is reinforcement.

The quality of a **STM** depends for a large part on the chosen truss model. The shape and stiffness of the truss determines the force distribution. Therefore, the shape of the truss is important to determine the reinforcement and quality of a solution. This chapter describes two possible approaches (see Figure 2.6): Triangulation and building blocks. The design parameter of the optimisation process is assumed to be the axial stiffness EA , resulting in one optimisation parameter per bar element in a truss.

5.1. METHOD 1: TRIANGULATION

The first considered method to design a truss in a 3D shape is called "triangulation". A **STM** optimisation based on triangulation searches for the optimal shape of a truss. A set of node locations (each within the given shape) generates a triangular mesh over the nodes. If this is executed correctly, the output can be considered to be a truss. The truss can be analysed, the object function can then be solved and the next iteration can be started to find the optimum. The definition of triangulation can be described as:

"Triangulation is the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by two adjacent triangles. It was proved in 1925 that every surface has a triangulation, but it might require an infinite number of triangles and the proof is difficult." [27]

As can be seen, this definition is for surfaces and polygons. To create a truss for a 3D structure, a triangulation existing out of tetrahedrons can be made (Figure 5.3). To create this kind of truss, a so called "Delaunay triangulation" can be adjusted to handle 3D shapes. This can be done according to [28]. A major advantage of this triangulation type is it's availability in 3d modelling programmes like "Rhinceros" [29], "Grasshopper" [30] and mathematical programmes like "Matlab" [20]. With the help of these programmes, it is relatively simple to create a set of lines to represent the truss structure.

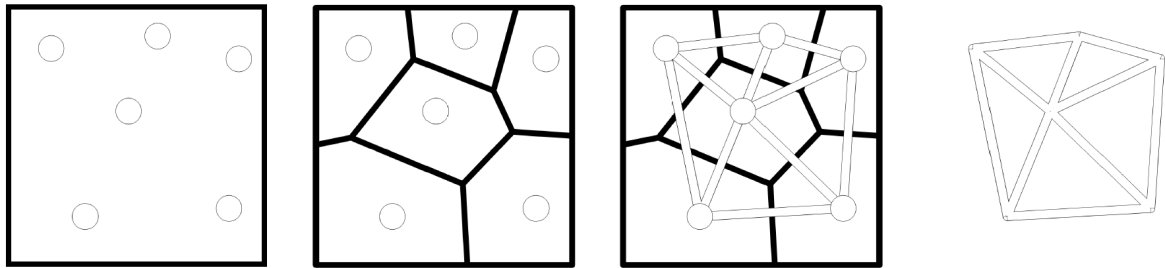


Figure 5.2: Representation of truss building with a Delaunay algorithm. Nodes are connected in two steps. First, a Voronoi diagram is drawn to analyse what areas are closest to what nodes. Next, all nodal areas that "share a border" are connected linearly.

The following definition shows the capability of a Delaunay triangulation to deal with multi dimensional triangulations:

"For a set \mathbf{P} of points in the (d -dimensional) Euclidean space, a Delaunay triangulation $DT(\mathbf{P})$ such that no point in \mathbf{P} is inside the circumhypersphere of any simplex in $DT(\mathbf{P})$ it is known that there exists a unique Delaunay triangulation for \mathbf{P} , if \mathbf{P} is a set of points in general position; that is, there exists no k -flat containing $k+2$ points nor a k -sphere containing $k+3$ points, for $1 \leq k \leq d-1$ (e.g., for a set of points in 3 ; no three points are on a line, no four on a plane, no four are on a circle, and no five on a sphere)." [31]

The application of a triangulation based truss in a **STM** can be very useful. For this situation, the node locations, which are described within the design space, can be used to define the truss. A change in node locations may lead to a different truss and so an alternative solution. Based on the deformation method (as is described in section 7.4.2), the normal forces of the truss can be determined. By adjusting the (smallest) strut cross-sections with the angle of the bars, the concrete stress can be kept at a relatively constant value (considering small deformations). This holds true because the assumed small deformation of the truss reduces the force-stiffness influence, while the cross section and normal force share a linear relationship (cross

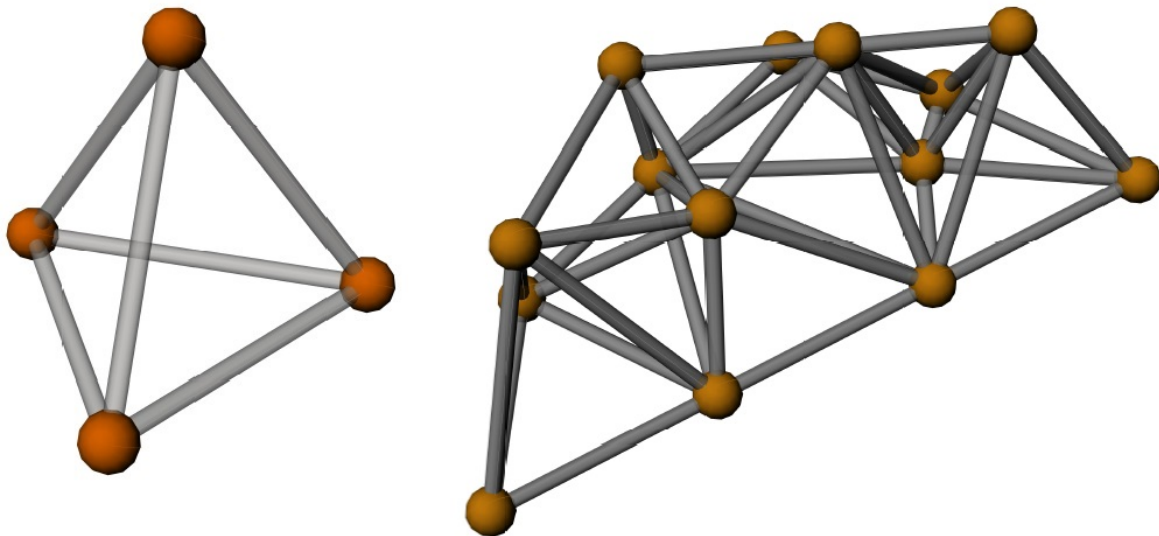


Figure 5.3: Example of a tetrahedron, for the triangulation, the shape of the tetrahedron can change, but it will still have 4 nodes with 6 lines each. (left: 1 tetrahedron, right: random mesh example build from tetrahedrons)

section grows with force).

ADVANTAGES AND DISADVANTAGES:

- Advantage: Truss shapes generated with the Delaunay system tend to avoid narrow angles. Narrow can angles create overlapping struts in the [STM](#) and are therefore not preferable.
- Advantage: Node locations can be a very efficient parameter to determine the optimal truss shape. It reduces the optimisation to a well known "road search" problem, which can be solved by for example an Ant Colony Optimisation ([ACO](#)) [6].
- Advantage: Given a sufficiently refined triangulation, practically every shape is approachable.
- Advantage and disadvantage: This method puts a truss through the effective part of the structure. Due to this effect, not all of the structure is checked, ineffective parts are left out.
- Disadvantage: Each node is unique in its connectivity and angles. A separate analysis might be required for each node to verify its strength.
- Disadvantage: A Delaunay triangulation can create a complex truss with different angles for each element. This can result in complex reinforcement shapes. These shapes are not preferable due to increased workloads of construction workers for more complex reinforcement shapes.
- Disadvantage: The changing truss shape makes it difficult to apply self weight forces onto the structure.
- Disadvantage: Optimisation methods, like the [ACO](#), are not practical with a changing set of optimisation parameters. Unfortunately, the preferred technique for this method would be to analyse the simpler trusses (containing less nodes) at first, and somehow increase the amount of nodes if required.
- Disadvantage: "For a set of points on the same line, there is no Delaunay triangulation." [31]. Not all output of a Delaunay triangulation represents a workable truss structure, some measures may be required to ensure triangulations to be a workable truss.

5.2. METHOD 2: BUILDING BLOCKS (LEGO'S)

An alternative technique to generate 3D truss models can be developed with building blocks. By creating a standard block and filling a structure with such blocks, it is possible to model many different geometries. A

building block based design can avoid the main problems (chaotic trusses and difficulties with nodal analysis of generated nodes) of the Delaunay triangulation.

A building block based truss generation solves the problems in the triangulation technique. Standardised building blocks can be designed to result in straight reinforcement bars and planar truss nodes. A building block can look like Figure 5.4. This building block is limited by constant angles of its elements, resulting in a realistic and relatively simple reinforcement design.

A building block based truss can be generated by splitting a design space into box-shaped pieces, filling each of them with a building block to create a truss structure (Figures 5.6 and 5.7) and then filter to remove all bars and nodes outside of the given geometry.

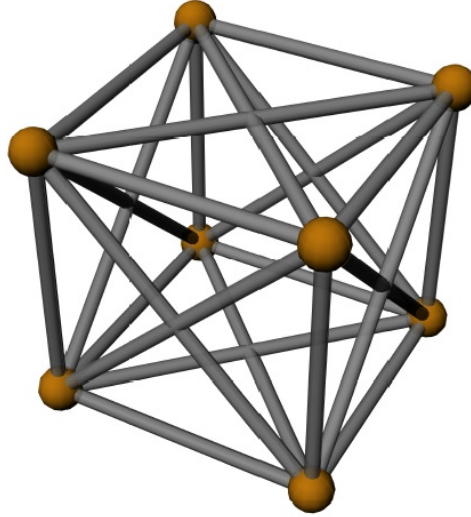


Figure 5.4: example of mesh pattern

The optimisation parameters of this building-block based approach is the axial stiffness (EA of the truss elements). By increasing or decreasing the values of individual bars, this method can effectively use the force stiffness relationship to optimise the structure. In case of *STM* optimisation, this is a major advantage, because the amount struts can be maximised while the tie cross-sections (and so the amount of reinforcement) can be minimised. The result should be a compression based design where loads are forced to flow as direct between two points as possible within a given shape. This will be verified in the main report of this Master's Thesis.

The creation of a truss with standard building blocks starts with the creation of a certain 3D element (Figure 5.6). Next a set of nodes with constant (Cartesian coordinate based) δx -, Δy - and Δz - distances is drawn within the shape. The δx -, Δy - and Δz - distances are the truss design parameter used to define the refinement of the truss. Too small elements can result in a computationally more intensive optimisation due to an increased amount of nodes and lines. Note that a reduction in building block size results in an exponential increase of building blocks, which can influence the computational intensity of the model. Larger elements can reduce the complexity, but may create inaccuracies. Parts of a structure may be neglected if building blocks are too large, resulting in unrealistic structures.

Spherical elements are used to analyse what nodes are closest to each node. By drawing spherical elements with a radius of $R = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ over each node, it is possible to find all truss bars required for the model. Figure 5.5 demonstrates this process for a 2D shape.

To create the connecting lines, all nodes within the spheres are connected to the central node. This process is repeated for each node to achieve a complete truss. The set-up with spheres has two complications. First this method connects every node twice (node 1 - node 2 and node 2 - node 1), this can be solved by building in a filter that removes all duplicates. Second, the shape of the sphere describes what nodes are connected. In this case, only nodes on the x-y, y-z and x-z plane are connected. This is done for two reasons: to reduce

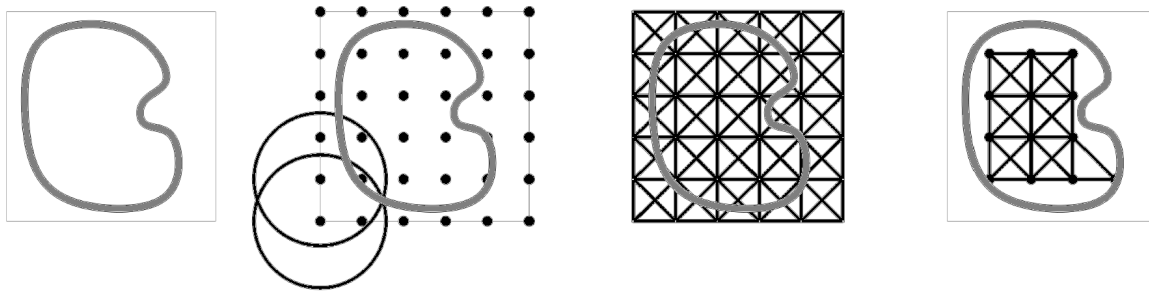


Figure 5.5: Left: design a shape, this can be a 3D object; second: create a set of nodes inside the shape, node distances $\Delta x, \Delta y$ and Δz are constants. Then draw a spherical element centred in each node; third: find all bar positions based on node connectivity between the centre node and all other nodes within each element (filter for duplicates). Right: remove all elements outside or intersecting with the design-shape and check the structure for errors and if necessary adjust.

the amount of possible reinforcement angles in order to simplify the structure and to reduce the amount of elements and so the computational intensity of the process. The consequence of this choice and system may be that some optima will be excluded because they would require curved reinforcement or reinforcement in different angles than the ones available.

When all nodes are connected, the truss is completed. It is now possible to define all nodes and lines and continue to the next step. Figures 5.6 and 5.7 show a possible result of this method. Note that a more refined truss would be a little more capable of taking the difference in height of the beam into account. The constant node distances are however perfect for including the self weight force of the structure. In case of a triangulated structure this would be much more difficult due to changing triangle shapes and the fact that such a mesh would not necessarily take the entire structure into account.

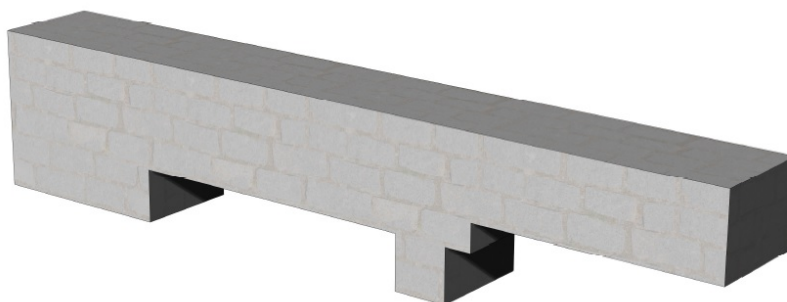


Figure 5.6: example shape of a beam

ADVANTAGES AND DISADVANTAGES:

- Advantage: Constant truss shape results in relative simple reinforcement shapes due to a limited amount of possible angles and/ or directions of ties.
- Advantage: The optimisation parameter for this situation is "cross section". This technique takes the force-stiffness relation of the statically indeterminate truss into account and uses it as an advantage.
- Advantage: The constant truss results in some very basic nodes that can all be solved similarly.
- Advantage: The constant truss shape makes it simple to apply self weight forces on all nodes, resulting in a more realistic normal force diagram.
- Advantage and disadvantage: This method puts a truss through the entire structure. Due to this effect, the entire structure is checked, also the ineffective parts of the structure.

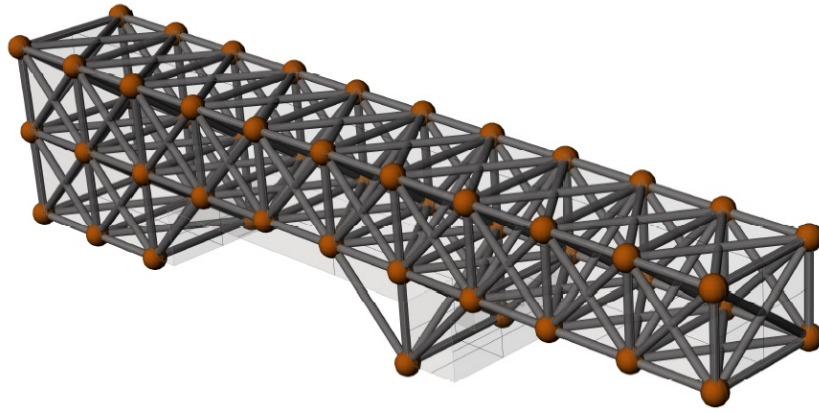


Figure 5.7: Example shape of meshed beam

- Disadvantage: The basic truss shape results in more nodes and elements compared to the triangulation technique. This approach will be slightly more intensive to compute.
- Disadvantage: Truss may have some difficulties to fill slender objects or strange shapes if the block size is not sufficiently small. This may result in "un-trussed" parts of the structure that may not be analysed.

6

REINFORCED CONCRETE COSTS ANALYSIS

6.1	Process	50
6.2	Concrete costs	51
6.3	Reinforcement costs	52
6.4	Formwork costs	54
6.4.1	Repetition factor	54
6.4.2	Formwork types	55
6.5	Costs analysis model	57

The first step in a structural optimisation process is to define the "optimisation goal". The "optimisation goal" depends on the target of a project. Many subjects can be used as optimisation goal, for example: lightest, strongest, thinnest, least deformed, etc. From an economical point of view, the goal of optimisations is "saving costs" or "find the most economical". This goal can usually be achieved in multiple ways. A combination of these two statements can result in the following structural optimisation goal: *"Find most economical structural design that fulfils the safety requirements."*

In case of steel structures, the economical solution depends mostly on two factors. The amount of steel defines the material costs and the amount of detailing determines the workload (welding, bolting, ...). Steel structures can be optimised in terms of costs by reducing the mass (which leads to less material costs), the amount of connections or by simplification of the structure (reduction of workload, mass production, ...).

In case of reinforced concrete structures, the most economical situation depends on more factors. In this case, the costs depend materials and labour required to for concrete, reinforcement and formwork. In case of reinforced concrete, a lighter structure with more reinforcement *can* be more expensive than a heavier one with less reinforcement (and vice-versa). For this reason, and because there are usually much more design parameters, reinforced concrete optimisations are considered to be more complex than similar steel ones.

A number of questions needs to be answered in order to perform a structural optimisation of a reinforced concrete structure:

- What is the purpose of the (costs) optimisation? How and why is it going to be used and what level of detail is required to solve the problem with sufficient accuracy?
- In what design stage is the optimisation process most effective? What can be learned from the optimisation process and when do designers require results?
- What method should be used to estimate costs of options and what accuracy is required to select the most economical solution?

- How can the costs of the concrete, reinforcement and formwork be estimated?
- Is it possible to compare resulting costs estimations with similar ones from practice to validate the accuracy of the model?

6.1. PROCESS

The purpose of the costs analysis is specified in the problem statement and research goals. The costs analysis model in this chapter is "score" part of the object function. The optimisation algorithm uses this model to determine the most economical option (given a set of structurally safe solutions). The optimal design and the local optima that are found can be used to determine the most economical design. The motivation for costs as objective is that in theory, this would give more realistic results because design decisions are similar to the ones made in practice.

The maximal effectiveness of the optimisation process can be achieved in early design stages. According to the MacLeamy curve [32], it is still relatively simple to change the design in this stage (see Figure 6.1). The larger design freedom in early stages results in less constraints and therefore more design possibilities. Insights from early stage optimisation processes can easily be applied in the design to achieve maximum effectiveness.

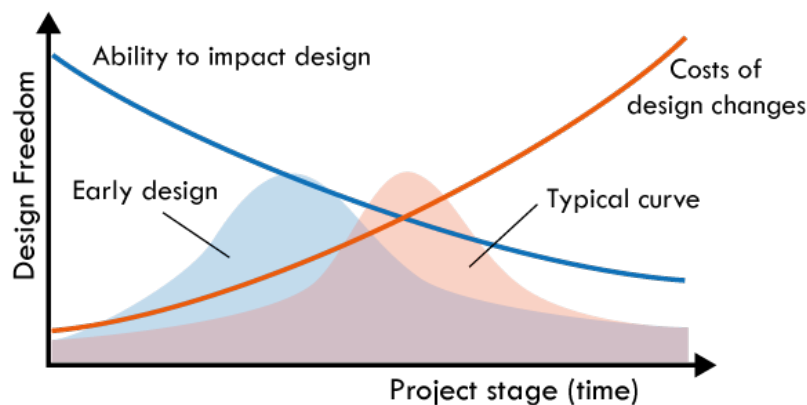


Figure 6.1: Relation between level of specification and influence on design, based on the MacLeamy curve [32]. This curves suggests that the design freedom is largest at the beginning of the process. Because optimisations attempt can help to make efficient design choices, they are most influential at the beginning of a process.

To be able to determine the costs of a reinforced concrete structure, the model described in the lecture notes of the TU Delft course "CT4170 Construction Technology of Civil Engineering projects" [33] is used. This method is specialised for civil engineering structures. For this paragraph, the assumption is made that the cost calculation for a civil engineering structure and a building can be conducted similarly, since the method describes sufficient parameters that are about reinforced concrete in general.

While estimating the costs of a certain reinforced concrete structure, several uncertainties must be kept in mind. The cost estimations in this Master's thesis consider "random" reinforced concrete structural elements, this means that only the structure is taken into account. Major parameters in costs estimation are repetition, location of the structure or structural behaviour.

The model performs a cost *estimation*, the focus is on impact of certain variables on the costs. Furthermore, costs estimations may be inaccurate, but the comparison between the different options can still be used for a comparison between options. Due to all mentioned uncertainties, it is possible that designers prefer certain options over the ones with little costs. Contractors may choose for different (more expensive) solutions to avoid or minimise risks or for many other reasons.

6.2. CONCRETE COSTS

Concrete costs depend on several parameters, especially the mix design of the concrete (i.e.: cement, sand, gravel and water) and the labour required for transportation, pouring and finishing are influential. Two models are suggested to compute costs of concrete. The first method uses a list to link the concrete strength class to the costs, the second technique uses the mix design (i.e. 1 cement : 2 sand : 3 gravel, water/cement factor 0.7) and is therefore more complex (and sometimes more accurate). It is important to create a list with mix designs for different types of concrete and current unit costs for the materials that are used.

The significance of labour costs is large because materials require processing and transportation to site. the further away the site, the more expensive the transportation. Higher labour costs will increase the costs for pouring and finishing of the concrete. The process used to estimate concrete costs is displayed in Figure 6.2.

CONCRETE MATERIAL COSTS: OPTION 1

The costs of the material concrete itself can be determined in two ways. The first way is to use the data from the website <http://www.bouwkosten.nl> (anno 2014) to determine the concrete costs by strength class. The costs can then simply be read from a table. The results are reasonably accurate for concrete construction in the Netherlands. The complete data chart to determine the concrete properties and costs looks like Table 6.1: concrete properties and costs per concrete strength class [34]. Mind that the costs values should be kept up to date. The unconfirmed values are left open in this case. A reasonably reliable estimation of unknown costs can be made by interpolation (or extrapolation) of the confirmed costs.

Problems with this method arise when costs are required for different environmental classes or specific concrete compositions. For such situations, it is advised to use the second method for determination of the concrete costs.

Type:	Class:	f_{ck} N/mm ²	f_{cm} N/mm ²	f_{cd} N/mm ²	f_{ctm} N/mm ²	$F_{ctk,0.05}$ N/mm ²	f_{ctd} N/mm ²	E_{cm} N/mm ²	E_{cr} N/mm ²	Costs €/m ³
C12/15	XC1 S3	12	20	8.0	1.57	1.10	0.73	27085	15477	84.50
C20/25	XC2 S3	20	28	13.3	2.21	1.55	1.03	30000	17143	84.50
C25/30		25	33	16.7	2.56	1.80	1.20	31500	11800	
C28/35	XC2 S3	28	36	18.7	2.77	1.94	1.29	32308	18462	91.20
C30/37		30	38	20.0	2.90	2.03	1.35	32800	18743	
C35/45	XC2 S3	35	43	23.3	3.21	2.25	1.50	34100	19486	98.70
C40/50		40	48	26.7	3.51	2.46	1.64	35200	20114	
C45/55	XC2 S3	45	53	30.0	3.80	2.66	1.77	36300	20743	118.00
C50/60		50	58	33.0	4.07	2.85	1.90	37300	21314	
C53/65	XC2 S3	53	61	35.3	4.16	2.91	1.94	37846	21626	138.00
C55/67		55	63	36.7	4.21	2.95	1.97	38200	21829	
C60/75		60	68	40.0	4.35	3.05	2.03	39100	22343	
C70/85		70	78	46.7	4.61	3.23	2.15	40700	23257	
C80/95		80	88	53.3	4.84	3.39	2.26	42200	24114	
C90/105		90	98	60.0	5.04	3.53	2.35	43600	24914	

Table 6.1: Concrete properties and costs [34]. Only data given by the source is placed into the model.

CONCRETE MATERIAL COSTS, OPTION 2

A more reliable and less location and concrete composition dependent method to determine the concrete costs is to derive the concrete costs from its actual composition. The main problem is that the exact concrete composition is often unknown until the final design (or execution) stages. The main advantage is that local material cost data can be applied to give a more accurate estimation of the costs. Such material data is usually available in practice and is therefore workable. Special types of concrete, like self-compacting concrete can be estimated more accurately with this method.

$$\epsilon_{material} = \epsilon_{sand} + \epsilon_{gravel} + \epsilon_{cement} + \epsilon_{water} + \epsilon_{additives} \quad (6.1)$$

TRANSPORTATION TO SITE

When the concrete mixture is created, it should be transported to the construction site. A simple method to estimate the transportation costs is to use some basic costs (i.e.: loading/ unloading of the concrete truck) and some distance related costs (i.e.: fuel, salary). For the estimation of the transportation costs, a basic costs of €42, – and a distance related costs of 0,2€/km is applied.

$$\epsilon_{transportation} = \epsilon_{basic} + distance * \epsilon_{distance} \quad (6.2)$$

POURING OF CONCRETE

The final step is the pouring and finishing of the concrete. The main factor in this case is the workload. To be able to estimate the workload, one should look at the complexity of the job. For example, mass concrete requires less effort per cubic metre than concrete in slender, detailed structures. The resulting costs can be described as a workload times a certain complexity factor times the unit costs of the workload. The workload for pouring and finishing 1 m³ of concrete is estimated to be 1 hour.

$$\epsilon_{pouring} = workload * complexity * \epsilon_{manhour} \quad (6.3)$$

TOTAL COSTS

The total costs of 1 €/m³ of concrete in place are estimated by combining the previous values. The flow chart in Figure 6.2 shows the link between the parameters and the concrete costs. Note that the main parameters are material costs, workload and complexity.

$$\epsilon_{concrete} = \epsilon_{material} + \epsilon_{transportation} + \epsilon_{pouring} \quad (6.4)$$

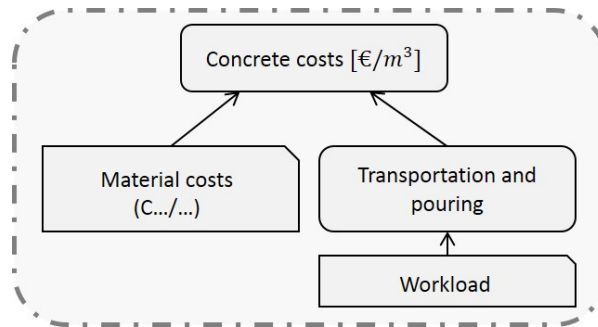


Figure 6.2: Determining the costs of the concrete volume.

6.3. REINFORCEMENT COSTS

Reinforcement costs are determined by a combination of the factors: material, transportation, cutting, bending and placing. Important parameters are the (estimated) bar diameter, the total mass and the complexity of reinforcement configurations. These parameters determine the material costs and the workload required to cut, bend and place the reinforcement. The complexity of the reinforcement has some influence onto the workload required to put the reinforcement in place. The process is described by Figure 6.3.

MATERIAL COSTS

The material costs of the reinforcement depends on the amount of material ordered and the bar diameter. To determine the material costs of the reinforcement, the data from <http://www.bouwkosten.nl> (anno 2014) was used. Mind that reinforcement steel prices depend heavily on the market values for steel on a location and can develop over time. The data is displayed in Table 6.2: reinforcement steel material cost in €/kg for bars, source: <http://www.bouwkosten.nl>. It is assumed that transportation is included in the given price levels.

Bar diameter in mm	0 - 200 kg	200 - 500 kg	500 - 1000 kg	1000 - 5000 kg	5000 - 10000 kg	10000 - 25000 kg	25000 - 50000 kg
8	1.43	1.13	0.93	0.73	0.73	0.70	0.69
10	1.39	1.09	0.89	0.74	0.69	0.66	0.65
12	1.37	1.07	0.87	0.72	0.67	0.64	0.63
16	1.36	1.06	0.86	0.71	0.66	0.63	0.62
20	1.36	1.06	0.86	0.71	0.66	0.63	0.62
25	1.36	1.06	0.86	0.71	0.66	0.63	0.62
32	1.39	1.06	0.89	0.74	0.69	0.66	0.65
40	1.41	1.11	0.91	0.76	0.71	0.68	0.67

Table 6.2: Reinforcement steel material costs in €/kg for bars, source: <http://www.bouwkosten.nl>

CUTTING BENDING AND PLACING

Cutting, bending and placing reinforcement takes time. Depending on the reinforcement percentage and the complexity of the reinforcement, the workload can be determined. This workload can then be used to estimate the costs for the cutting, bending and placing of the reinforcement in terms of €/kg.

$$\epsilon_{cut,bend,place} = workload * complexity * \epsilon_{manhour} \quad (6.5)$$

TOTAL COSTS

The total costs of the reinforcement can be estimated by combining the previous values. The flow chart in Figure 6.3 shows the link between the parameters and the reinforcement costs. Note that the main parameters are material costs, workload and complexity.

$$\epsilon_{rebar,unit} = \epsilon_{material} + \epsilon_{cut,bend,place} \quad (6.6)$$

$$\epsilon_{rebar} = \rho * \gamma_{steel} * \epsilon_{rebar,unit} \quad (6.7)$$

where:

ρ

is the reinforcement percentage in [%]

γ_{steel}

is the mass - density relation of steel, $\gamma_{steel} = 8150 \text{ kg/m}^3$

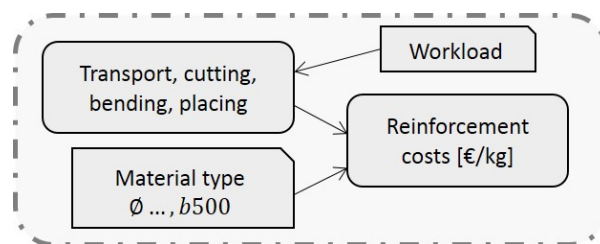


Figure 6.3: Determining the costs of reinforcement.

6.4. FORMWORK COSTS

Determining the costs for formwork and falsework is more complex and less precise than finding the reinforcement or concrete costs. The falsework costs depend on the location and shape of a structure. For this Master's thesis, the falsework is assumed to be out of scope because it completely depends on the surrounding structure and location. Note that the costs of the falsework are potentially very influential.

Due to many different formwork-systems, there are lots of options. Each of the options can be a feasible and/or preferable solution for certain conditions. To deal with the large amount of options, it is possible to create a table with properties for the individual formwork-systems. Such table has to contain information on the costs and possibilities of formwork systems. The next step is to combine information from the gathered data with the project information. Finally, it is possible to select the price of the most feasible solution. The process is shown in Figure 6.4.

To determine the costs of a square meter of formwork in a certain situation, a number of factors should be taken into account. The investment to buy the formwork(system), the investment to maintain the formwork and the workload to place and remove the formwork. When the system is used multiple times, this can have a significant effect on the formwork. The formwork can be bought once and then used a certain amount of times. These factors would lead to the following formula [33]:

$$\frac{\text{costs}}{m^2} = \frac{I_{\text{investment}}}{N_{\text{repetition}}} + \frac{\text{manhours}}{m^2} \quad (6.8)$$

where:

$I_{\text{investment}}$	the investment to buy and maintain the formwork times the amount of times it should be replaced or maintained when operational.
$N_{\text{repetition}}$	The amount of times the system is used during construction.
$\frac{\text{manhours}}{m^2}$	The workload for a square meter times the salary of the worker.

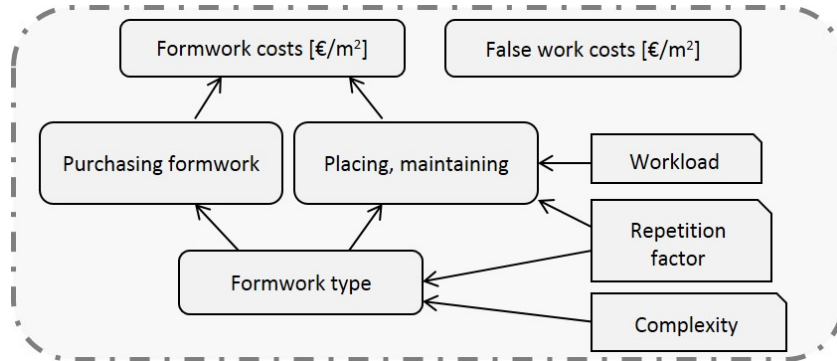


Figure 6.4: Determining the formwork costs.

6.4.1. REPETITION FACTOR

As is shown in Equation 6.8, there are some advantages for the formwork costs when it can be used several times, instead of having to buy a new formwork every time, the system can just be maintained and re-used. Another advantage of this repetition can be seen in the work load. People tend to work faster if actions have to be repeated. This effect is called the “Learning effect [33]”, the learning effect can be described with the following formula (also see Figure 6.5):

$$D_n = \phi^{\log_2(n)} * D_1 \quad (6.9)$$

where:

ϕ	Repetition factor, usually between 0.65 and 0.95
n	Amount of products
D_1	Average workload in case of multiple series of n elements

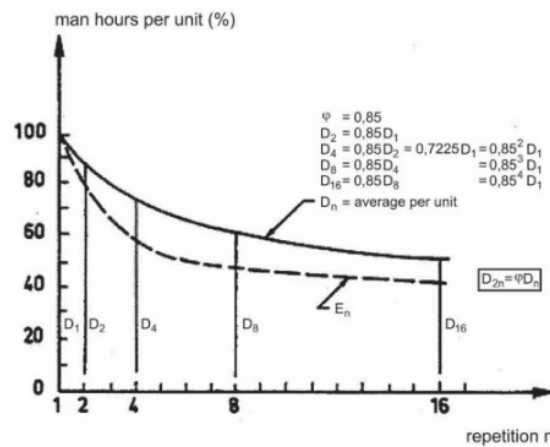


Figure 6.5: Learning cycle effect, courtesy of [33]

Mind that the repetition factor will change when the repetition process is disturbed. It will not completely return to 100% at the slightest disturbance, but the efficiency will drop significantly. Application of the repetition factor can also be applied onto the concrete and reinforcement costs per object, but is less effective in these situations, since formwork can be reused within a single object.

6.4.2. FORMWORK TYPES

Table 6.3 shows a list with several formwork types, their costs and capabilities. Because this table does not contain all systems, there exists a probability that the chosen formwork system is not the most efficient one for a given situation.

Type:	Purchase costs €/m ²	Workload per unit hrs/m ²	Maintenance costs €/m ²	Maintenance frequency [-]	Complexity
Traditional	25	2	13	8	straight
System	200	1	100	100	Straight
Pre-assembled	1750	0.2	875	1000	Straight
Traditional curved	400	2	200	8	Curved
EPS foam	220	2	220	1	Double curved
Composite curved	1600	1	800	100	Double curved

Table 6.3: formwork types and costs

When all considered formwork types are plotted in graphs and the repetition is known. It is possible to read the formwork costs and type out of these graphs. Figures 6.6, 6.7 and 6.8 show the application of the given formwork types for straight, curved and double curved formwork types. In these plots, the total size of the structure is not taken into account. This can result in some inaccuracies in the results.

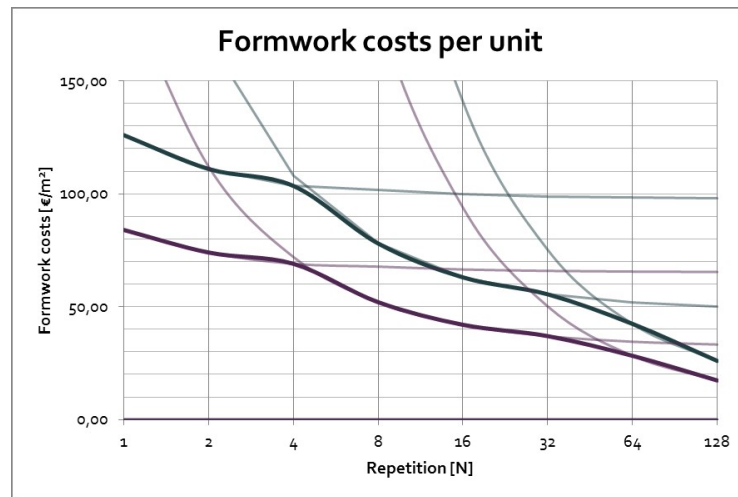


Figure 6.6: Formwork costs for non-curved formwork systems

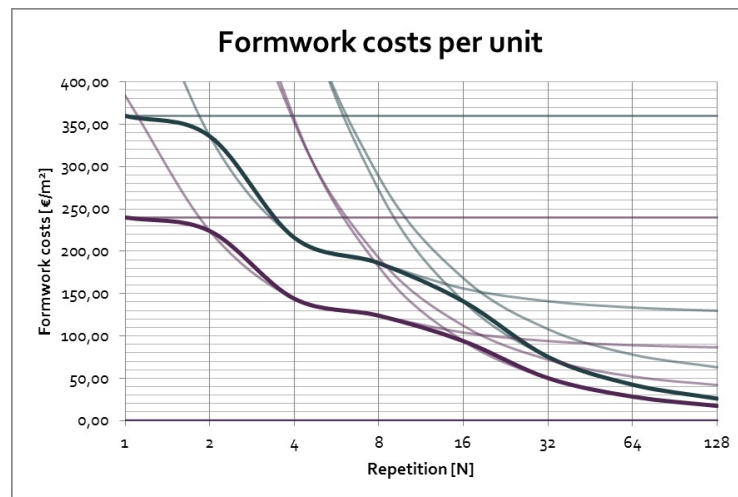


Figure 6.7: Formwork costs for curved formwork systems

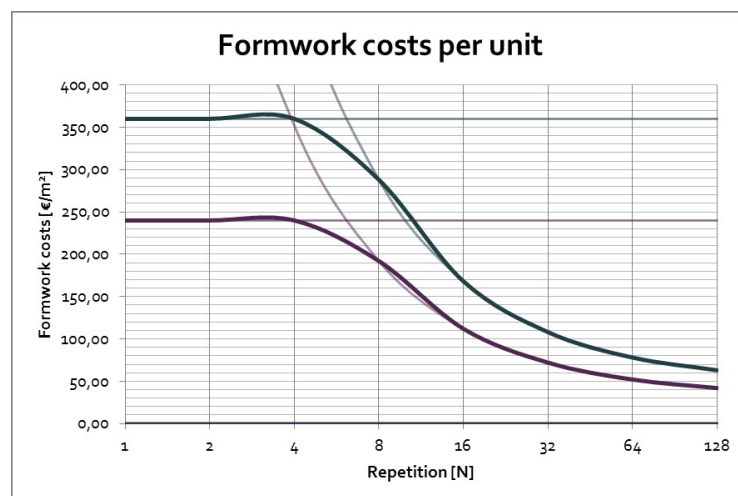


Figure 6.8: Formwork costs for double-curved formwork systems

6.5. COSTS ANALYSIS MODEL

In the former paragraphs, all relevant structural aspects of the reinforced concrete costs analysis were discussed. The next step is to combine the theory into one model to be able to have a specific and consisted concrete costs model. By using this model, one can find the specific costs of the reinforced concrete, split in concrete, reinforcement and formwork costs. One can analyse where the costs come from, which is essential if one is executing a structural optimisation process.

Figure 6.9 shows the entire process of the costs analysis. Mind that the output information is a price for the entire structure, or for just 1 m^3 of concrete. The input information can be reduced to the amounts of concrete, reinforcement and formwork surface, combined with the man-hour costs, material costs and the repetition factor of the object.

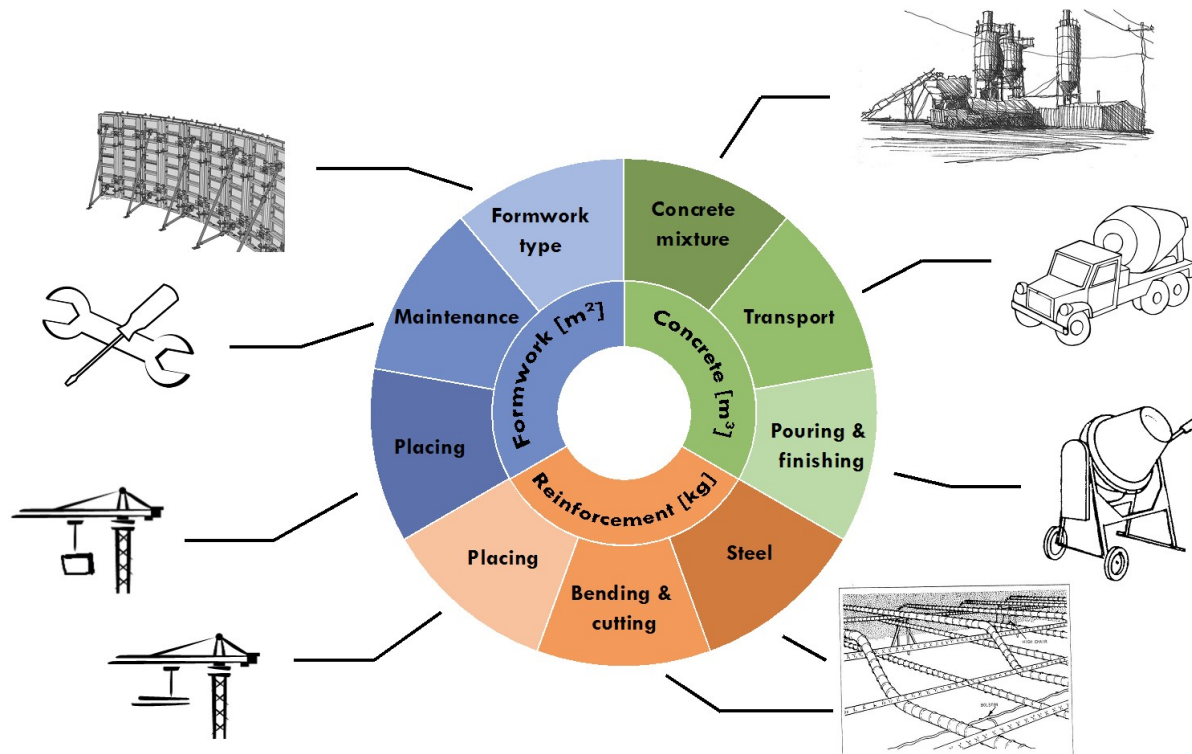


Figure 6.9: Costs analysis of reinforced concrete.

7

REINFORCED CONCRETE STRUCTURAL ANALYSIS

7.1 Design of reinforced concrete elements	60
7.1.1 Geometry specification	60
7.1.2 Strength requirements	61
7.1.3 Deformation limit's	61
7.1.4 Cracking behaviour of reinforced concrete	62
7.2 Structural behaviour of elements	62
7.2.1 The link between structure and structural element	63
7.2.2 Required behaviour of a reinforced concrete object	63
7.3 Detailing	64
7.3.1 The "golden design rules"	65
7.3.2 Design issues	65
7.4 Structural design methods	67
7.4.1 Method 1: Regular Eurocode calculations	67
7.4.2 Method 2: Strut and Tie Modelling	67
7.4.3 Method 3: Stringer Panel Models	69
7.4.4 Method 4: FEM, "Reinforcement design in solid concrete"	70
7.4.5 Method 5: FEM, "Modelling reinforcement"	71
7.4.6 Method 6: Case based reasoning (non-structural)	73
7.5 Comparing methods	73

This chapter is used for the literature study on 3D reinforced concrete analysis methodologies. The main goal of this chapter is to gain knowledge on how to engineer a 3D reinforced concrete structures. This research should help answering the research question on reinforced concrete analysis methods. A number of questions are set for this chapter:

- What aspects are important for the design of a safe and effective 3D reinforced concrete object (Section 7.1)?
- What is the link between the structure and the object of the optimisation (Section 7.2)
- What is the influence of reinforcement detailing on the design (Section 7.3)?
- What design strategies are useful for the design of a 3D reinforced concrete structure, and which ones are suited for a future structural optimisation process (Section 7.4 and 7.5)?

The research in this chapter is used for the development of a structural analysis model. Know-how on structural engineering methodology is considered important to complete this goals. The knowledge of this chapter is combined with know-how in the previous chapters on costs analysis and optimisation techniques to create an assessment model to compare options.

7.1. DESIGN OF REINFORCED CONCRETE ELEMENTS

The first step in the design of a “random” 3D reinforced concrete object, is to determine what design aspects should be considered. In case of reinforced concrete, these are usually geometry, strength, deformations and cracking criteria. The geometrical criteria represent the limits of the shape and location of objects. What design space may be used for the structure and what are the limitations? The strength criteria are required for the safety of the design. The object should be designed to be able to withstand the loading of a given design situation. During this Master’s Thesis, this is assumed to be the Ultimate Limit State (ULS) (future versions might include other load cases, such as the Serviceability Limit State (SLS)). For the sake of safety, it is preferred if a structure warns before it fails. If a structure cracks first, yields next and fails next, than brittle failure can be controlled. To prevent cracking from being harmful to the structure, some cracking limitations can be set. The final main structural criterion is the limitation of deformations. Deformation criteria are mainly set to ensure aesthetics, comfort and durability of a structure.

If a structural engineer is familiar with the limits (constraints) of a structure, the design process can be started. The first design step is to look at the required behaviour of the design. possible requirements could involve questions like: "What forces should the structure handle?" or "What size can the structure be?". If the required behaviour of a design is analysed, it is possible to search for the input information and state what output information is required.

7.1.1. GEOMETRY SPECIFICATION

The first design aspect of a reinforced concrete object is the geometry of the structure. Since a concrete element is just a part of the total structure, a specific geometrical description is required. The available design space is often limited by demands like: aesthetics, doors/ windows, mechanics (installations), etc. To deal with these kind of demands, a design space should be defined (see Figure 7.1 for an example), existing of a design domain (the space within which the structure must be defined) and a void domain (the space within the structure which cannot be used for the structure) [10]. Within the given design domain, the object must

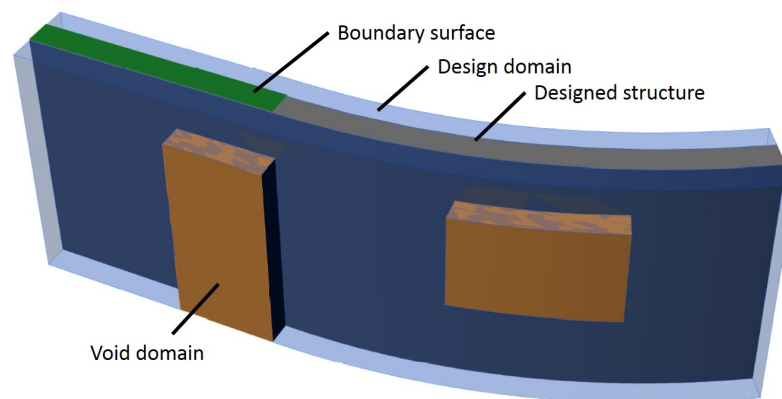


Figure 7.1: Definition of "Design domain", "Void domain" and "boundary surface" for the design of a wall structure.

be constructed. Another condition is required to ensure that the structure will be connected with its surrounding (structural) elements. These surfaces, where the structure is connected to the object will be defined as Boundary surfaces. This is basically the section where designed structural element ends and another one begins. These surfaces can hold boundary conditions in terms of external forces (or stresses) or required stiffness (deformation limits).

7.1.2. STRENGTH REQUIREMENTS

There exist a number of requirements to ensure safety in structural designs. One such requirement is that the strength of a structure is sufficient to fulfil the loading requirements. How the strength of a structure is checked depends on the method that is applied. Most strength-related methods check whether or not the stresses (following from normal, moment and shear forces in all directions) stay within their limits. These limits usually depend on material and design choices. Mind that in a 3D concrete structure, materials loaded equally in compression in all directions can be far stronger than materials loaded differently from each angle. Because concrete is a brittle material, the "von Mises yield criterion" does not hold for tensile stresses. Three-dimensional stresses in brittle materials can be checked with for example the "Mohr-Coulomb criterion", which is related to the "von Mises yield criterion" [35]:

$$\sigma_{vgl} = \frac{1}{\sqrt{2}} * \sqrt{(\sigma_{xx} - \sigma_{yy})^2 + (\sigma_{yy} - \sigma_{zz})^2 + (\sigma_{zz} - \sigma_{xx})^2 + 6\sigma_{xy}^2 + 6\sigma_{yz}^2 + 6\sigma_{zx}^2} \quad (7.1)$$

In case of reinforced concrete, there is another requirement to ensure the safety structures. To prevent brittle failure, a structure should warn before it fails [36]. This effect is achieved by designing a concrete structure to crack first and fail later ($M_{cr} < M_{Rd}$). In case of a simply supported beam, the structure should crack in the concrete tensile zone first, than the reinforcement should yield before finally the concrete compressive zone may (not) fail, resulting in collapse of the beam. To handle these requirements, the amount of reinforcement in a structure should be designed with care. Specific demands are created for the minimum and maximum percentage of reinforcement to ensure the described behaviour of cracking first, then deforming due to yielding reinforcement and finally failing.

In situations where reinforced concrete structures are optimised on costs, the strength requirements can often be described as constraints (See Chapter 4.3). If a strength requirement is not met, this translates into a certain "penalty". The "penalty" is applied in order to prevent options from being chosen as the "fittest" (or optimum).

7.1.3. DEFORMATION LIMIT'S

Concrete deformations are often difficult to find. Due to the "less predictable" modulus of elasticity of concrete [37] compared to steel, the deformations are highly changeable. To analyse deformations, the cracking behaviour of the structure should also be analysed. If a structure is cracked, the modulus of elasticity lowers and the deformations will increase. Analysis of deformations are often performed with $M - \kappa$ diagrams to find the material properties and the usual equations (i.e.: "forget-me-not" equations [38]) to estimate the deformations.

Deformations of concrete can also be created for other reasons. Imposed deformation effects [39], like creep, shrinkage and/or temperature loading can result in cracks or deformation of concrete. Because the effects of these kinds of deformations are usually rather small and do not contribute to the main subject of this Master's thesis, the effects of imposed deformations are neglected.

Deformations (or rotations) in concrete can be seen as a structural limit for multiple reasons. They can be limited to ensure comfort, functioning of the structure, structural behaviour, safety, to avoid cracking of concrete, etc. The Eurocode 2: Design of concrete structures, NEN-EN 1992-1-1 [37] (EC2) specifically states deformation limits for different kinds of situations. Unfortunately, calculations of the deformations of reinforced concrete do not always represent reality. Especially when higher concrete strengths than designed are applied, the deformation-behaviour of a structure might change, potentially resulting in different forces and therefore deformations in the structure (especially clamped connections can be sensitive for these kind of changes).

In this Master's thesis, the main reason why deformations should be limited are:

- Ensuring structural behaviour. For example: a model of a clamped connection should behave similar to the model, clamped connections should not deform like hinged connections and vice versa. The main reason is that forces in a (statically indeterminate) structure will change with the with the stiffness of that structure, resulting in a unreliable model.

- Deformations and cracking of the concrete are linked. Cracking of concrete lowers the modulus of elasticity of concrete, enlarging deformations which in turn increase the cracking of concrete. Large deformations can potentially result in unacceptable cracks in the reinforced concrete. The control of the concrete cracking will help to limit deformations.
- Ensuring comfort/ functionality/ safety of the structure. For example: machines, façades or other parts of the structure could be damaged by deformations of the structure. Meaning that deformations should be limited to ensure the quality of the structure.

To ensure the limits described above, a couple of deformation limits can be set. In the first place, EC2 states that $w_{max} = w_{tot,k} - w_c \leq 0.004 * l_{eff}$. in which $w_{tot,k}$ is the deformation of the structure in the SLS, w_c is the camber of the structure and l_{eff} is the effective length of the part of the structure. The factor 0.004 can be changed for specific purposes (like the ones described above).

7.1.4. CRACKING BEHAVIOUR OF REINFORCED CONCRETE

“The principal idea of designing reinforced concrete is to have the concrete resisting the compressive forces and the reinforcement the tensile forces. Steel stresses actually start to develop after the concrete is cracked. “Cracking” is therefore a phenomenon that is typical for the behaviour of concrete and is not a problem at all, unless crack widths are too large so that there is risk of corrosion of the reinforcement steel” [36].

This text from the lecture notes of the TU Delft course CTB2220, “Dictaat CTB2220/3335 Gewapend beton”, states that cracking of reinforcement is generally not a big issue. The main problem arises when the durability of the structure is reduced due to corrosion of the reinforcement steel. To check the cracking of the concrete, it is possible to use the formulas of the EC2. To determine the crack width criterion, knowledge is required on the environmental class and the required behaviour (in some cases, cracking is not acceptable due to i.e.: water tightness) of the concrete.

The cracking of the concrete is influenced by a number of factors. First the loading of the concrete should be known in the SLS. Then, a check is required to find out in what cracking stage the concrete is. These can be: uncracked stage, crack formation stage, stabilised cracking stage, steel bars only and finally, the yielding of reinforcement. Finally, the formulas from the EC2 can be used to check the size of the cracks of the structure.

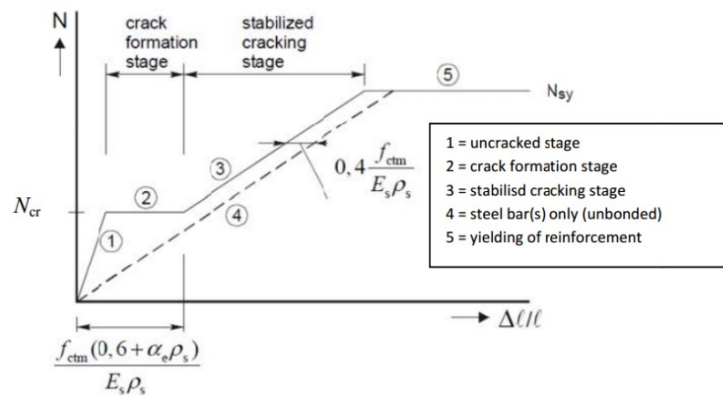


Figure 7.2: Cracking behaviour of a reinforced concrete tensile member, courtesy: TU Delft [36]

7.2. STRUCTURAL BEHAVIOUR OF ELEMENTS

This Master's thesis is about the structural optimisation of reinforced concrete objects. This means that not the entire structure, but just a part of it is reviewed. This assumption has certain effects on the calculations that are being made. Some of them can have a large influence on the design of the structure as a whole. The assumed starting point is that the (general) shape of the structure and the governing forces are already known.

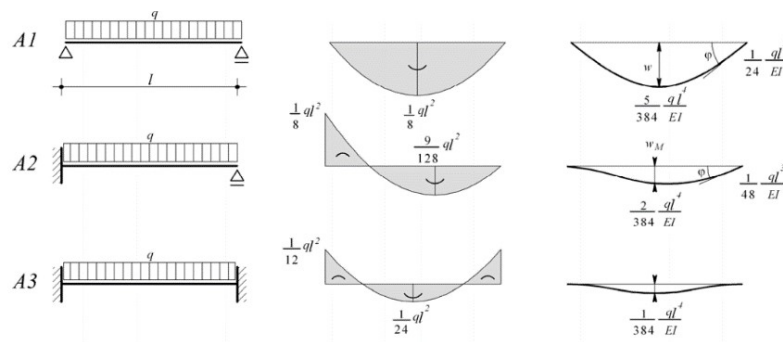


Figure 7.3: "Forget me not's", courtesy: [40]

The shape/ dimensions of structural element is going to be determined with a structural optimisation. This section discusses the issues that have to be accounted for in the process.

7.2.1. THE LINK BETWEEN STRUCTURE AND STRUCTURAL ELEMENT

The more global structural design can have a large influence on the design of individual parts of a structure. As can be seen in Figure 7.3, forces and deformations can change due to different boundary conditions. Optimising a structure itself to minimise the amount of forces and so the required amount of material can make a large difference in the design. One can decide where to attract more forces and where not to. Since the starting point of this Master's thesis is at the point where the structure and its forces are already known, this subject is out of scope. It can however be quite interesting to keep these kind of effects in mind when designing a proper structure.

When an object is produced more than once, the effects of repetition can be taken into account. Repetition can have effects in materials (like formwork) that can be re-used, processes that become more efficient (less man hours required for the n^{th} product then for the 1st one) and bulk materials which can be cheaper when bought in large amounts. If possible, it can really save costs to standardise objects to make sure repetition is maximised [33]. More information on the cost effects of repetition can be found in Section 6.4.1.

Finally, the location of the object within the structure can have some influence on the costs of object. Especially when lots of false work is required to be able to create the structure, or when objects have to be placed at difficult locations, this can have influence in the construction process. When secondary costs have to be made to be able to create a structure, this is often not very efficient. The solution for this problem is often strongly depending on the object and its location.

7.2.2. REQUIRED BEHAVIOUR OF A REINFORCED CONCRETE OBJECT

When one is designing a prefabricated reinforced concrete connection [41], a number of properties should be taken into account. A prefabricated reinforced concrete connection is of course not completely equal to a reinforced concrete element, but the requirements for properties are quite similar. In "Reader CIE4281 Designing and Understanding Precast Concrete Structures in Buildings" [41] it is described what these properties should be:

- Standardisation (mass production)
- Simplicity (beneficial for costs)
- Tensile capacity (reinforcement tensile capacity, anchor lengths, detailing)
- Ductility (No brittle failure, large deformations before failure)
- Movements (required stiffness, free form capacity)

- Fire resistance (Structural boundary conditions)
- Durability (concrete quality, cover, fatigue, cracking)
- Aesthetics (shape related boundary conditions)

A number of these properties are described in previous chapters (standardisation, simplicity, tensile capacity, ductility, movements, durability) and others are not (fire resistance and aesthetics). Fire resistance and aesthetics are not considered to be out of scope for this Master's thesis.

An important aspect in the structural design of a reinforced concrete structure, is whether or not the required behaviour is being realised. The modelled behaviour of the structure has to be similar to the actual one. For example, a clamped connection has to behave as if it one, otherwise the model is inaccurate, which is a potential cause of failure. A method to check whether the model is correct or not, is to check the force equilibrium to find the forces in the structure and then make sure that no excessive deformations occur.

To perform structural optimisation, a number of properties and conditions have to be taken into account. Unfortunately, it is difficult to make sure all properties are included. There are often selection criteria excluded from optimisation processes. This results in inaccuracies in the results of the process. The output of the structural optimisation is just an advise (not to decide), preferably in terms of “make sure parameter x looks like y”. *The goal of the optimisation process should be to guide the design and not to be the design.*

7.3. DETAILING

The relationship between structural detailing, durability and design is considered important. Sufficient quality of detailing of a reinforced concrete structure improves the durability (and therefore the quality) of the design. In this section, a number of design issues and rules are discussed in an attempt to include detailing of the reinforcement in the structural optimisation process.

In the current design process, detailing of reinforcement is placed at the end of the process (see Figure 7.4). In many situations, problems arise when the beam has to be detailed. In the first stages of design, al strength, deformation and cracking checks could have suggested that the beam was ok, but when it turns out that for example the reinforcement does not fit into the structure, the design may still be unacceptable.

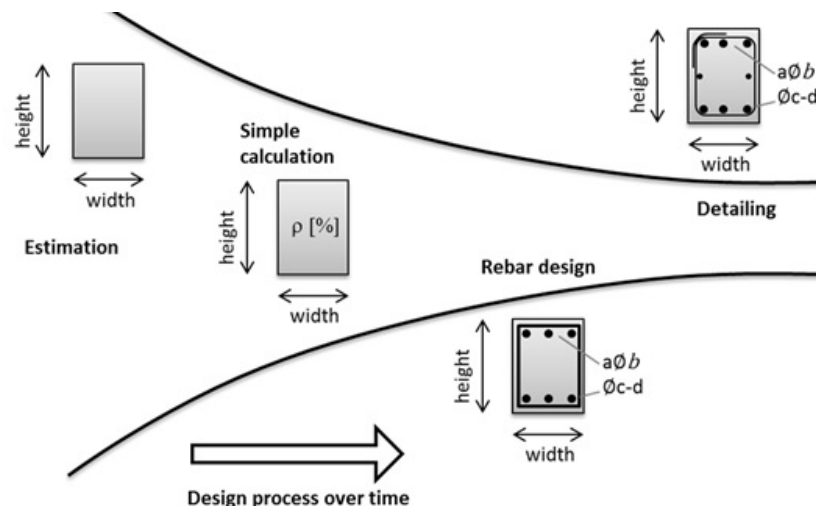


Figure 7.4: The design process of a reinforced concrete beam. Mind the level of detailing for each stage (working from a rough design to a detailed design)

Problems in concrete detailing can often be solved easily. Most problems arise due to the “inflexibility” at the end of the design process, giving the engineer less space to change the dimensions of the structure. To avoid these kind of problems, one should try to think of detailing in the earlier stages of the design (see Figure 6.1). In this chapter, some basic “rules” for the detailing of reinforced concrete structures are formulated.

7.3.1. THE "GOLDEN DESIGN RULES"

When one is pouring and finishing concrete, some structures are easier than other structures. It is possible to define a set of rules to ensure the workability of the concrete (and so improve the quality). The three “golden design rules” for reinforced concrete structures, according to the book “Manual for detailing reinforced concrete structures to EC2 [42]” are:

Rule 1: Concrete must be cast into its final position across an essentially vertical path.

horizontal paths must be avoided. This must be taken into consideration in the design drawings for reinforcing bar arrangements. Concrete aggregates should be able to pass the reinforcement bars in their way quickly.

Rule 2: The vibrator must be able to reach the bottom reinforcement.

With 65 mm spacing, a standard 50 mm vibrator will be able to reach the bottom reinforcement. Wider spacing should always be applied to be sure compacting the concrete is possible. (note: 65 mm spacing refers to the open space between reinforcement bars and not the distance the centre points of the bars)

Rule 3: Concrete consistency must be in keeping with the reinforcement arrangement. As a general rule the concrete slump should be no smaller than 60 mm.

Unless the reinforcement is arranged very openly and spaciouly or powerful vibration methods are used, overly dry concrete is characterised by the following:

- The required strength can be reached in test specimen (but not on site) with a lower proportion of cement. The real strength of the concrete can be lower
- Since the control specimens can be compacted with no difficulty, the laboratory trials will furnish good information
- In-situ placement and a good surrounding of the reinforcement will be difficult to achieve and the loose consistency will lower actual on-site strength

7.3.2. DESIGN ISSUES

In this section, a number of concrete detailing design issues are discussed. The goal of the discussion is to define a set of “rules” that should help to design a structure. The source for these points of attention is, just like the golden design rules, “Manual for detailing reinforced concrete structures to EC2 [42]”. Other sources are the EC2, and some basic books on reinforced concrete design [36, 43].

CONCRETE COVER

“The concrete cover is the distance between the surface of the reinforcement closest to the nearest concrete surface (including ties and stirrups and surface reinforcement where relevant) and the nearest concrete surface [42]”

The concrete cover is meant to protect the reinforcement from corrosion, make sure the reinforcement has enough bond strength and gives adequate fire resistance. The concrete cover can be determined according to EC2 Chapter 4.4.1. If the concrete cover is being determined, the required environmental class in which the concrete is designed should be known. One can then simply read the required concrete cover from tables.

Special attention should be paid to the corners of the structures. Because reinforcement bars need to be bend to realise corners, this creates the situation where the concrete cover in corners is much larger than normal [37]. In some situations, especially when bar diameters larger than $\varnothing > 12$ mm are applied, these distances can become quite large. One should take the higher chance of damage into account in these situations. A possible solution could be “smoothening” the corners to prevent this kind of damage.

PERMISSIBLE MANDREL DIAMETERS FOR BEND BARS (EC2, 8.3)

Reinforcement bars often have to be bend to connect to other bars, or to fit inside a structure. The mandrel diameter should fulfil a certain set of rules. The minimum mandrel diameter for bends, hooks and loops is \varnothing

Tabel 8.1N — Minimumdoordiameter om schade aan de wapening te vermijden

a) voor staven en draden

Staafdiameter	Minimumdoordiameter voor ombuigingen, haken en haarspelden (zie figuur 8.1)
$\phi \leq 16 \text{ mm}$	4ϕ
$\phi > 16 \text{ mm}$	7ϕ

b) voor gelaste gebogen wapening en wapeningsnetten gebogen na het lassen



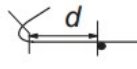

Minimumdoordiameter	
 of 	 of 
5ϕ	$d \geq 3\phi$: 5ϕ $d < 3\phi$ of lassen binnen de buigzone: 20ϕ
OPMERKING De doordiameter mag voor lassen binnen de buigzone zijn verminderd tot 5ϕ , als het lassen is uitgevoerd in overeenstemming met bijlage B van prEN ISO 17660.	

Figure 7.5: Mandrel diameters according to EC2, courtesy: [37]

4 for bar diameters $\phi \leq 16 \text{ mm}$ and is $\phi 7$ for bar diameters $\phi > 16 \text{ mm}$. The length of the bar bend the end of the bend should be larger than $\phi 5$ to be allowed to take the bend into account as part of the reinforcement. For further rules considering the bending of reinforcement bars, one should take a look at EC2 Chapter 8.3.

BUNDLED BARS

When bar spacing's are getting too small, one of the possible solutions is to bundle reinforcement bars to increase this spacing. When one bundles reinforcement bars, there are also two main disadvantages. The first one is the higher difficulty with the anchoring lengths, the other one is that the more complex design makes construction more complex and therefore more costly.

Bundling of reinforcement bars is not always allowed. To deal with this issue, certain rules can be applied. As a rule, no more than three bars can be bundled together (to make sure concrete can still surround and bond to the reinforcement bars as much as possible). The equivalent diameter of a set of bundled reinforcement bars cannot be larger than 55 mm. the equivalent bar diameter can be found with the formula $\phi_n = \phi \sqrt{n_b}$, where n_b is the number of bars and ϕ is the diameter of each individual bar.

SURFACE REINFORCEMENT

For bars with diameters over 32 mm, it might be useful to apply surface reinforcement to control the cracking of the concrete or to control the spalling of the concrete. Surface reinforcement should consist of a welded-wire mesh of narrow bars. The bars applied in the surface reinforcement can also be used in the structural calculations for the shear and moment force capacity of the structure.

ANCHORAGE LENGTHS

Not all reinforcement can be applied in one piece, therefore it is possible to anchor reinforcement bars to each other to create a continuous system. In the EC2, specific rules for the anchorage lengths are given, especially the concrete strength (required for the load transfer) and the bar diameter are important to be able to find the required anchorage length of a reinforcement bar. In practice, locations with lots of anchors can have a very dense reinforcement mesh. Therefore, a smart placement of the reinforcement can improve the quality of the design.

7.4. STRUCTURAL DESIGN METHODS

To find out what structural engineering method is suited to use in the structural optimisation process, it is required to search for different engineering methods. Each method has its own advantages and disadvantages. In the final section of this chapter, it will be discussed what methods are useful for optimisation and what methods are probably not.

When a reinforced concrete object is designed, the structural engineering method can be tested for a couple of subjects. The method should be uniform, suitable for optimisation process and capable to design reinforcement in a random 3D concrete shape. The results of the method should be reliable and understandable. Paragraphs 7.4.1 to 7.4.6 will describe several structural engineering methods.

7.4.1. METHOD 1: REGULAR EUROCODE CALCULATIONS

Regular design methods are used to describe/ model a simplified 3d structure. The structure will be described in terms of columns, beams, diagonals, slabs and walls. The structure is checked according to the regular design codes. First the dimensions of the structure have to be estimated, then the reinforcement is designed and the structural design is completed. The optimisation parameters can be values like material types, reinforcement design and geometry. Normal elements in this method are: "columns", "beams", "plates" and "slabs". Problems can arise when structures are designed that are difficult to simplify into elements (i.e: in case of a 3D volume). The data generated with this method is normally sufficiently clear, making this method quite accurate and workable.

THE METHOD

This method can be different for each problem. The list below describes the steps that should be taken to design a proper structure. Mind that not all special structures can be modelled correctly with this model. EC2 has to be taken into account to find further specifications. Especially oddly shaped objects are harder to model with this method.

Usually, the design procedure contains the following steps:

- Assume dimensions for the geometry of the structure and determine the model that is to be used to design the structure (column, beam, slab, wall, etc.)
- Determine the loading and load cases for the model. Then calculate the forces to design the structure for all load combinations.
- Check the (combination of) Normal, shear and moment forces in the structure in the ULS and design the reinforcement to fit the requirements.
- If required, check the structure for the secondary effects, like punching shear, torsion, etc. If it is required, additional reinforcement or concrete should be applied.
- Check the deformations and cracking behaviour of the structure in the SLS. If required, the design should be changed.

7.4.2. METHOD 2: STRUT AND TIE MODELLING

Strut-and-Tie Model (STM) are widely used to design i.e.: shear-reinforcement or detailing. The method is allowed in the EC2 and is a relatively simple and uniform method to design structures. In this method, struts are used to describe compression areas. Ties are used to design the tension areas in the concrete. This means that a good STM shows similarities to the stress trajectories (which describe the compression and tension stresses) of the structure. In the tension areas, a reinforcement can be designed and in the compression areas, the compressive stress can be checked. The system works for most (well described) situations and can be applied on 3D shapes as well [36].

+/-	Advantages and disadvantages:
+	Results are simple, robust and reasonably accurate
+	Widely used and accepted in practice to design 1d and 2d structures. This generally means that it is reliable, effective and described in the codes.
-	Does not describe how to design 3d reinforcement. The result is that the engineer should try to fit simplified 1- or 2-dimensional engineering in a three-dimensional structure.
-	Not all shapes are possible, the method can only be applied when a structure can be simplified to the existing design models (generally 1d beams and 2d plates).
-	different shapes can give different approaches. Resulting in a not so uniform method that can be difficult to model correctly.

Table 7.1: Advantages and disadvantages of method 1

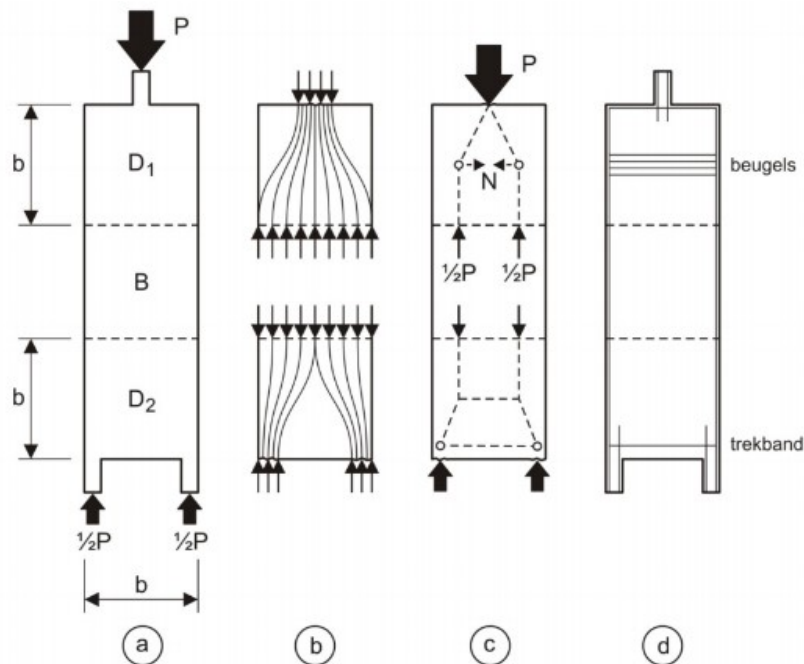


Figure 7.6: Detailing in a wall using STM. Courtesy: [36]

THE METHOD

Determining the reinforcement in a D-region requires the following steps to be taken:

- Determine the size and location of the D-regions. The disturbed length is about the size of the zone in which the load should be spread.
- Determine the boundary conditions in the border areas of the D-regions. The forces at the border areas are partially determined by the global distribution of forces in the structure.
- Visualise the flow of stresses through the D-region. This can greatly assist the engineer to create a good model to fit the stress paths in the region.
- Develop a strut and tie model to represent the flow of forces. Note that it can be beneficial to also take the location of supports and nodes into account.
- Calculate the forces in the struts, check the stresses in the compressive and nodal zones and determine the reinforcement in the tensile zones. Also check whether the designed compressive zones and reinforcement fit within the design area.
- Design and check the anchorages for the reinforcement.

- Provide additional crack control reinforcement if required.

After all steps are taken to design the reinforcement in the D-region. It is advised to synchronise/ check the results with those of the B-regions. The anchorages of the reinforcement, the deformations and cracking of the concrete are not taken into account in this method. Extra steps (6. and 7.) are required to ensure the quality of the design.

+/-	Advantages and disadvantages:
+	In principle, secondary effects like punching shear should not be a problem since the strut and tie model (when conducted correctly) would take them into account.
+	The method is uniform, reliable and relatively simple for all shapes.
+	The method is easy to apply onto complex three-dimensional structures.
-	Basic shape of structure required before reinforcement can be designed, the method is used to determine the reinforcement in the structure.
-	Does not tell anything about brittle failure, stability or cracking of concrete. Additional calculations should be made to check these properties.

Table 7.2: Advantages and disadvantages of method 2 (STM)

7.4.3. METHOD 3: STRINGER PANEL MODELS

The stringer panel method has been developed by the University of Denmark, Politecnico di Milano and the TU Delft [44]. The method, which is represented by Figure 7.7, is based on stringers and panels. Stringers are assumed to carry normal forces while panels carry the shear forces. In some cases, panels can carry normal forces as well [45].

The concept of stringer panels is based on reinforced concrete walls. It is common for walls to have concentrations of reinforcement near their edges. Stringers can make full use of this property by using relatively large lever arms for the reinforcement.

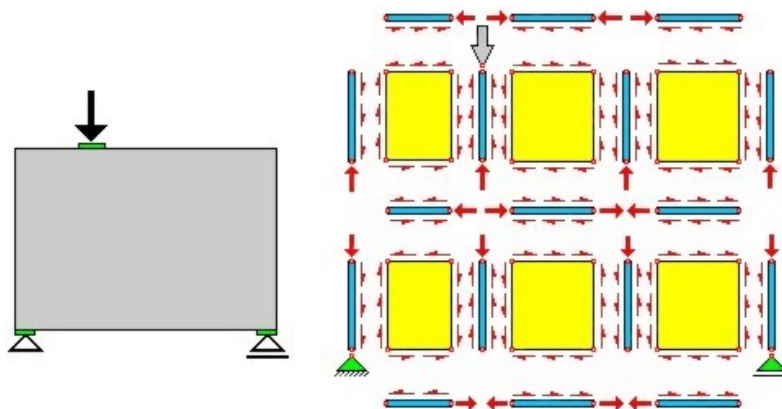


Figure 7.7: Stringer panel method, the stringers carry normal forces and the panels the shear (and normal) forces. The stringer panel method is developed for the analysis of walls. Stringers can easily be "replaced" by reinforcement bars. Courtesy: Hoogenboom [45]

The stringer panel concept is somewhere between the Finite Element Method (FEM) and Strut-and-Tie Model (STM) method. Differences between stringer panels and the STM technique are among others:

- Stringer panels tend to have a less complicated shape composed of rectangles compared to the truss shapes required for a STM.
- Stringer panels represent the geometry of a structure while a STM represents the flow of forces through a structure. Some STM cannot be used for multiple load cases due to this effect.

- Engineers have much more design freedom in case of **STM**. This is both positive due to the design freedom and negative due to the larger difficulty in finding the best options. There are usually just a few logical models in case of stringer panels.

THE METHOD

The described method generally uses the next steps to determine the stresses and the reinforcement in the given structure:

- Choose stringer panel lay-out
- Establish load cases and combinations
- Perform a linear analysis with the stringer panel method
- Select reinforcement based on this analysis

+ / -	Advantages and disadvantages:
+	The method is uniform for all shapes, the input that are required can be found in the boundary conditions and the design space.
+	The simplicity of the method results in a computationally efficient technique.
+ / -	Stringers represent reinforcement. This can be positive because lever arms will be relatively large due to the position of stringers on the edges of the structure. Non-stringer positions of reinforcement can be somewhat difficult to place.
-	This model is developed for wall shaped structure. It can be difficult to apply this technique on other shapes.
-	The method is not described in the codes and can therefore be complex to apply in practice.

Table 7.3: Advantages and disadvantages of method 3

7.4.4. METHOD 4: FEM, "REINFORCEMENT DESIGN IN SOLID CONCRETE"

In the paper "Computation of reinforcement for solid concrete [46]", a set of reinforcement design rules is proposed for **FEM** containing volume elements. The proposed system could be used to determine the reinforcement in a random concrete volume object. This method is basically a finite element method for reinforced concrete volume elements, capable of designing reinforcement. For a more specific description, one should look into the paper.

THE METHOD

The described method generally uses the next steps to determine the stresses and the reinforcement in the given structure:

- Define the geometry and boundary conditions of the object. Also make sure loads are given in the **SLS** and **ULS**.
- Describe the stress criterion (Mohr, von Mises, Tresca, etc.), in other words, when does the reinforcement steel yield (f_{yd}) and when does the concrete crack (f_{ct}, f_{ck}).
- Equilibrium of forces, make use of the equilibrium equations of reinforced concrete in a cracked stage to describe the forces in the structure. Determine the stresses and find out what the optimal reinforcement ratio for each direction is. To be able to work and check the stresses in the cube, the principle stresses will have to be found first.
- The designed structure can now be checked in the **ULS** for strength and the **SLS** for cracking.

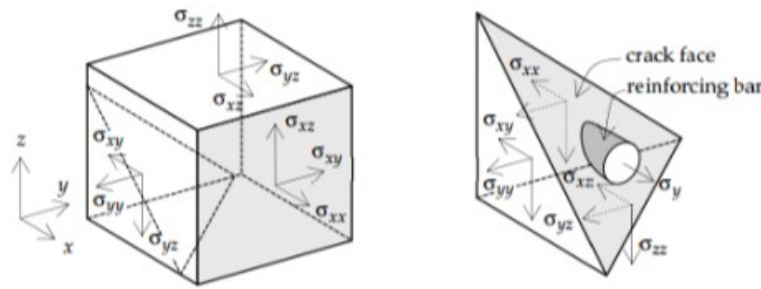


Figure 7.8: Stresses in small cube and cracked cube part. courtesy: [46]

Equilibrium equations for a finite element of this method [46]:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_{xx} - \rho_x f_{yd} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} - \rho_y f_{yd} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} - \rho_z f_{yd} \end{bmatrix} * \begin{bmatrix} \cos \alpha \\ \cos \beta \\ \cos \gamma \end{bmatrix} \quad (7.2)$$

+/-	Advantages and disadvantages:
+	In principle, secondary effects like punching shear should not be a problem since this model (when conducted correctly) would take them into account automatically.
+	The method is uniform for all shapes, the input that are required can be found in the boundary conditions and the design space.
-	The method can result in a mesh of concrete cubes with each a specific reinforcement ratio. One should learn to manage these ratios to find a realistic reinforcement design.
-	The large amount of data that is generated with this method could result in a very large processing time.
-	The method is not described in the codes and can therefore be complex to apply in practice.

Table 7.4: Advantages and disadvantages of method 4

7.4.5. METHOD 5: FEM, "MODELLING REINFORCEMENT"

Finite Element Method (FEM) where already described twice in the past few sections. Finite element analysis is potentially a very capable method to design a random reinforced concrete structure. Former sections already described a FEM analysis used in a topology optimisation to find the right place for the reinforcement and a FEM analysis specifically designed to find the reinforcement ratio in elements. There are numerous variants capable to design reinforced concrete objects.

One method that would specifically interesting for reinforcement design is non-linear “smeared cracking [47]” system. This method describes basic finite elements that display a certain elastic behaviour to describe cracks in the concrete. When also the reinforcement is modelled in the model, a quite accurate simulation of reality could be made. Mind that smeared cracking spreads cracks over the entire structure, resulting in some danger that the model overestimates reality.

THE METHOD

The effect of cracking is spread over the area that belongs to an integration point. Advantage: smeared cracks can occur anywhere in the mesh in any direction. This can be modelled with a mesh element that looks like Figure 7.9.

A crack is initiated when the principal tensile stress σ_1 exceeds the value of the tensile strength f_t [48]. The direction of the crack is perpendicular to the direction of the principal tensile stress. After crack initiation,

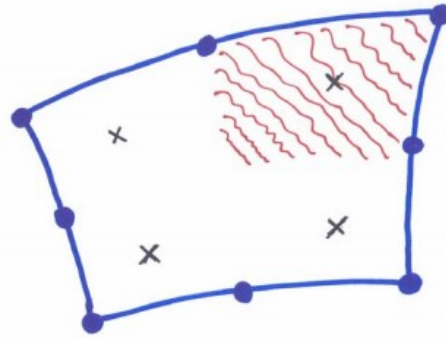


Figure 7.9: Quadratic finite element with smeared cracking, courtesy: [47].

the principal stress follows a tension-softening stress-strain diagram. Three material parameters (same as for discrete crack):

- tensile strength f_t
- fracture energy G_f
- shape of the softening diagram

Additional parameter: crack band width h over which the crack is smeared out.

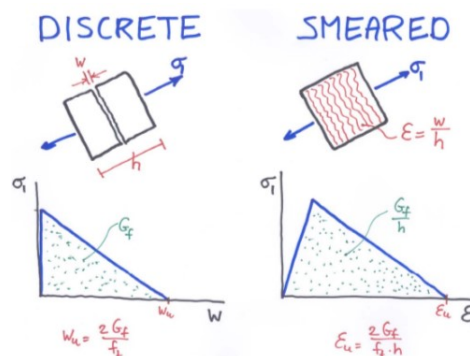


Figure 7.10: Stress strain relationship, courtesy: [47].

When the method is applied correctly in a finite element program. The results, in case of a deep beam element would look like Figure 7.11. The results can be used to check the strength, deformation and cracking behaviour of a reinforced concrete object. With this method, it is still quite simple to model reinforcement into the concrete model. The reinforcement can be modelled as a separate (different) finite element that behaves like steel.

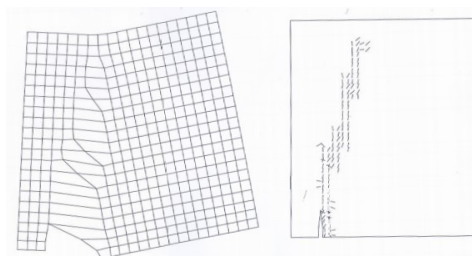


Figure 7.11: Resulting cracking behaviour, courtesy: [47].

+ / –	Advantages and disadvantages:
+	In principle, secondary effects like punching shear should not be a problem since this model (when conducted correctly) would take them into account automatically.
+	The method is uniform for all shapes, the input that are required can be found in the boundary conditions and the design space.
+	Cracking behaviour is taken into account with this method.
–	The method can be computationally intensive, this method might be a bit overkill for the required optimisation process.
–	Smeared cracking spreads cracks over the structure, this might lead to a overestimation of the strength of the structure.
–	The method is not described in the codes and can therefore be complex to apply in practice.

Table 7.5: Advantages and disadvantages of method 5

7.4.6. METHOD 6: CASE BASED REASONING (NON-STRUCTURAL)

Reasoning can often be described as a process. One starts from scratch and then, with the help generalised rules, one can come to conclusions. In case based reasoning, the primary knowledge is “harvested” from previous experiences, with the advantage of learning from previous situations. The definition of this method can be described as:

“In CBR, new solutions are generated not by chaining, but by retrieving the most relevant cases from memory and adapting them to fit new situations.” [49]

In practice, most logical structural options are already applied. To be able to work with this method, lots of historical projects should be put into a database. The engineer can then specify the structure that is to be designed and find all similar ones. The historically successful structures can then be used (copied and then changed) for the current project. The unsuccessful ones can be used to learn from experiences that should not reoccur.

Since this principle is out of the scope of this Master’s thesis (its not a structural engineering subject), it will not be addressed in further analysis. The system can however be very useful for larger/ older companies with a large history of projects. With this method they can effectively learn from their previous mistakes and successes.

+ / –	Advantages and disadvantages:
+	Very effective for standard structures, a standard design can just be copied, transformed and/or combined to fit into the surroundings. Some standardisation can be applied to make it easy for a building to fit on its specific location.
+	This method saves time and money due to the repetition factor. If a structure is being build multiple times, normally, the later structures will experience less errors and faster construction.
–	Unique structures will not be able to use this technique and structures are often quite unique for their situation, so some engineering will always have to be done.
–	If this method gets too successful, the build environment could become filled with copied structures. People tend to find such an environment not very interesting.
–	If new problems arise with the original design, this could potentially affect all copied structures.

Table 7.6: Advantages and disadvantages of method 6

7.5. COMPARING METHODS

The final step of this chapter is to compare and select the methods that are most suited for this Master’s thesis. The methods (see Section 7.4) should be judged on several parameters that are required for the optimisation of a reinforced concrete three-dimensional structure. Finally, a selection of the most promising methods can be made.

When a three-dimensional reinforced concrete structure is optimised, what properties would be required to make it successful? The method should be at least capable to make a complete analysis of a three-dimensional reinforced concrete structure, giving the properties “3D capable” and “complete analysis possible”. Furthermore, it would be preferable if the method is computationally not too intensive, and applicable for as many shapes as possible, “simplicity” and “uniform method”. Furthermore, the results of the method should be allowed for practical usage and should be reasonably accurate.

The list below describes all selection criteria. If a method fulfils these requirements, than it should be able to work for optimisation processes.

- 3D capable (make sure all shapes are possible).
- Complete analysis possible (strength, deformations and cracking can all be checked).
- Simplicity (simple methods improve understanding, reduce errors and may boost computational speed).
- Uniform method (to ensure a maximum number of options are possible with one technique).
- Method is described in EC2 (or other design code).
- Accuracy.

A comparison of options on these properties suggests that the STM is the preferred method. This suggestion is motivated by a number of properties. Design according to EC2 techniques requires different methods for different problems. This reduces the simplicity and uniformity of the technique. The stringer panel method is very simple and uniform, but it is developed for wall shaped structures. The technique is interesting, but requires adjustments for applications on random 3D structures and is not clearly described in EC2.

FEM techniques are computationally more complicated than STM techniques. Method 4 ("Design in solid concrete") is capable of estimating reinforcement, but the technique seems to be experimental. Method 5 (non linear analysis with modelled reinforcement) is computationally very intensive, but also very accurate. Reinforcement has to be modelled for all options, which is very complex.

The STM appears to have all properties of the list above. It can be used for 3D shapes (with 3D trusses) and, given a sufficiently detailed truss, it offers a complete linear analysis of loading effects. The technique is simple to apply and can be standardised by using "building-block-based" trusses. The technique is described in EC2 and is usually a little conservative because material between the truss and the concrete tensile strength are usually neglected.

Assumption: The Strut-and-Tie Model (STM) technique is selected as preferred method for the structural optimisation process.

BIBLIOGRAPHY

- [1] W. Collins *et al.*, *Collins english dictionary - complete & unabridged 10th edition*, (2014).
- [2] F. van Keulen and M. Langenaar, *Wb1440 engineering optimization*, (2013), lecture slides.
- [3] J. Stewart, *Calculus, early transcendentals* (Cengage learning inc., 2011).
- [4] J. Arora, *Introduction to optimum design* (Elsevier, 2012).
- [5] H. Wolkowicz, *Optimization: Theory, algorithms, applications*, Presentation (2006).
- [6] P. Papalambros and D. Wilde, *Principles of Optimal Design - Modeling and Computation* (Cambridge University Press, 2000).
- [7] S. Burns, *Recent Advances in Optimal Structural Design* (ASCE, Urbana, 2002).
- [8] ABT-bv, *Begroting model*, (2014), internal document, Dutch.
- [9] R. Kennedy, J. and Eberhart, *Particle swarm optimization*, *Proceedings of IEEE International Conference on Neural Networks IV*, pages: 1942 t/m 1948 (1995).
- [10] P. W. Christensen and A. Klarbring, *An Introduction to Structural Optimization* (Springer, 2009).
- [11] A. Einstein, *Everything should be made as simple as possible, but not simpler*, (1950).
- [12] W. Wolpert, D.H. and Macready, *No Free Lunch Theorems for Search*, *Technical Report SFI-TR-95-02-010*, Tech. Rep. (Santa Fe Institute, 1995).
- [13] C. Vuik, P. van Beek, F. Vermolen, and J. van Kan, *Numerieke Methoden voor Differentiaalvergelijkingen* (VSSD, 2006) dutch.
- [14] D. Lay, *Linear algebra and its applications* (Pearson, 2005).
- [15] Michi, *Design variables of structural optimization problems*. Website (2010).
- [16] Foster and Partners, *British museum court*, (2012).
- [17] T. Borrvall and P. J., *Large scale topology optimization in 3D using parallel computing*, Tech. Rep. (Division of Mechanics, Mechanical engineering systems, Linköping university, 2000).
- [18] C. Raymond, *Nature-Inspired Algorithms for Optimisation* (Springer, 2009).
- [19] W. Spillers and K. MacBain, *Structural Optimization* (Springer, 2009).
- [20] Mathworks, ed., *Matlab Primer R2014b* (Mathworks, 2014).
- [21] C. Darwin, *The origin of species* (New York P.F. Collier, 1909).
- [22] A. Simone, *An Introduction to the Analysis of Slender Structures* (TU Delft, 2007).
- [23] J. Brownlee, *Clever Algorithms: Nature-Inspired programming recipes* (lulu.com, 2011).
- [24] Y. Shi and R. Eberhart, *A modified particle swarm optimizer*, IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 69 (1998).
- [25] M. Dorigo and T. Stutzle, *Ant Colony Optimization* (MIT Press, 2004).

- [26] W. Boyce and R. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, 9th ed. (Wiley, 2008).
- [27] G. Francis and J. Weeks, *Conway's zip proof*, Amer. Math. Monthly , 393 (1999).
- [28] M. Siqueira, *An Introduction to Algorithms for Constructing Conforming Delaunay Tetrahedrizations*, Tech. Rep. (University of Pennsylvania, 2003).
- [29] R. McNeel and Associates, eds., *Rhinoceros 5 User's Guide For Windows* (Robert McNeel and Associates, 2014) 3d Computer Graphics.
- [30] A. Payne and R. Issa, *Grasshopper Primer, Second Edition* (LIFT Architects and Robert McNeel and Associates, 2009) visual Programming.
- [31] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computation Geometry, Algorithms and Applications* (Springer Berlin Heidelberg, 2008).
- [32] B. Paulson, *Designing to reduce construction costs*, *Journal of The Construction Division* (1976).
- [33] A. van der Horst, *CT4170 Construction Technology of Civil Engineering Projects - Lecture notes*, 3rd ed. (TU Delft, 2013).
- [34] *GTB 2010 grafieken en tabellen voor beton*, Tech. Rep. (Betonvereniging, 2010) dutch.
- [35] C. Hartsuijker and H. Welleman, *Mechanics of Structures CT4145/CT2031 Module: Introduction Into Continuum Mechanics* (TU Delft, 2008) dutch.
- [36] J. Walraven, *Dictaat CTB2220/3335 Gewapend beton* (TU Delft, 2013) dutch.
- [37] *Eurocode 2: Design of concrete structures - part 1-1: General rules and rules for buildings*, Tech. Rep. (Nederlands Normalisatie Instituut, 2005).
- [38] C. Hartsuijker and H. Welleman, *Mechanica, statisch onbepaalde constructies en bezwijkanalyse* (Academic Service, 2004) dutch.
- [39] K. van Breugel, C. Walraven, J.C. and van der Veen, and R. Braam, *Concrete Structures under Imposed Thermal and Shrinkage Deformations, Theory and Practice*, edited by K. van Breugel and E. Koenders (TU, 2013).
- [40] J. Gerrits, *Bk2000 rekenvoorbeeld*, Faculty of Architecture, TU Delft (2010), bijlage 2: Mechanica "Vergeet-me-nietje".
- [41] D. Hordijk, R. Nijse, and S. Pasterkamp, *Reader CIE4281 Designing and Understanding Precast Concrete Structures in Buildings* (TU Delft, 2012).
- [42] J. Calavera, *Manual for detailing reinforced concrete structures to EC2* (CRC Press, 2011).
- [43] C. Braam and P. Lagendijk, *Constructie leer gewapend beton* (Cement en Beton, 2010) dutch.
- [44] SPanCad, *Stringer-panel method*, Internet (1999).
- [45] P. Hoogenboom, *Discrete Elements and Nonlinearity in Design of Structural Concrete Walls*, *Ph.D. thesis*, TU Delft (1998).
- [46] P. Hoogenboom and A. de Boer, *Computation of reinforcement for solid concrete*, (2008) pp. 247–272.
- [47] J. Rots and M. Hendriks, *Cie5148 computational modeling of structures, lesson: smeared cracks and reinforcement*, (2013), lecture at TU Delft, course CIE5148.
- [48] H. Welleman, *Structural Mechanics 4 CIE3109 Module: Work and Energy Methods* (TU D, 2014) dutch.
- [49] D. Leake, *Case-Based Reasoning: Experiences, Lessons and Future Directions*, Tech. Rep. (AAAI Press/ MIT Press, 1996).

LIST OF FIGURES

1.1	Major subjects in discussed in this literature study.	1
2.1	Rectangular area maximisation with limited fencing (Problem based on Calculus [3])	4
2.2	Application of structural optimisation on a space-truss.	5
2.3	Impression of a traveling-salesman problem.	6
2.4	Grand Canyon and even more difficult "Needle in a haystack" problem.	9
2.5	Example of a continuous (left) and discontinuous (right) function.	11
2.6	Structural optimisation strategies	12
2.7	Application of structural optimisation on a space-truss.	13
2.8	Shape optimisation	14
2.9	Topology optimisation	14
2.10	Example of three-dimensional cantilever beam.	15
2.11	Solution to the rectangular farm area problem	16
2.12	Particle swarm optimisation is a nature inspired optimisation algorithm based on the behaviour of fish schools	17
3.1	Basic diagram to show the location of an optimisation algorithm in the optimisation process.	20
3.2	Simulated Annealing (SA)	21
3.3	cyclic coordinate search method	22
3.4	Nelder-Mead-simplex method	23
3.5	Genetic algorithm, mutation and combination	24
3.6	Particle swarm optimisation. Courtesy: Qirong Tang, Universitat Stuttgart.	25
3.7	Representation of the Particle Swarm Optimisation (PSO) algorithm	26
3.8	PSO flow chart	27
3.9	Function $f(x)$ and it's derivative.	29
3.10	Particle progress for $c_1 = c_2 = 0, 1$	30
3.11	Particle progress for $c_1 = c_2 = 1$	30
3.12	Particle progress for $c_1 = c_2 = 2$	30
3.13	relation of algorithm to other optimisation techniques	34

4.1	Different types of object function solution spaces.	36
4.2	Logarithmic and quadratic penalty functions	39
4.3	Arctangent and unit-step penalty functions	40
4.4	Combined arctangent + quadratic penalty function	41
5.1	Limiting conditions for STM truss design	43
5.2	Representation of truss building with a Delaunay algorithm.	44
5.3	Tetrahedron	45
5.4	example of mesh pattern	46
5.5	truss realisation	47
5.6	example shape of a beam	47
5.7	Example shape of meshed beam	48
6.1	Relation specification - design influence	50
6.2	Determining the costs of the concrete volume.	52
6.3	Determining the costs of reinforcement.	53
6.4	Determining the formwork costs.	54
6.5	Learning cycle effect	55
6.6	Formwork costs for non-curved formwork systems	56
6.7	Formwork costs for curved formwork systems	56
6.8	Formwork costs for double-curved formwork systems	56
6.9	Costs analysis	57
7.1	Geometry limitations	60
7.2	Cracking behaviour	62
7.3	Forget me not's	63
7.4	Design-detailing	64
7.5	Mandrel diameters according to EC2	66
7.6	Detailing in a wall using STM	68
7.7	Stringer panel method.	69
7.8	Stresses in small cube and cracked cube part.	71
7.9	Quadratic finite element with smeared cracking.	72
7.10	Stress strain relationship	72
7.11	Resulting cracking behaviour,	72

A.1 Reinforcement optimisation with Grasshopper and GSA.	83
A.2 Particle progress for $c1=c2=0.1$	84
A.3 Particle progress for $c1=c2=1$	84
A.4 Particle progress for $c1=c2=2$	84

LIST OF TABLES

2.1	Differences between 2D and 3D topology optimisation	14
3.1	Results of the first 6 iterations of the Newton Rapson method. Convergence is found at the fifth iteration. The value $x_6 = 0.70761$ is the global minimum of $f(x)$	29
3.2	Comparison of optimisation algorithms.	34
6.1	Concrete properties and costs [34]. Only data given by the source is placed into the model. . . .	51
6.2	Reinforcement steel material costs in €/kg for bars, source: http://www.bouwkosten.nl	53
6.3	formwork types and costs	55
7.1	Advantages and disadvantages of method 1	68
7.2	Advantages and disadvantages of method 2 (STM)	69
7.3	Advantages and disadvantages of method 3	70
7.4	Advantages and disadvantages of method 4	71
7.5	Advantages and disadvantages of method 5	73
7.6	Advantages and disadvantages of method 6	73

A

EXAMPLES

A.1. EXAMPLE: REINFORCEMENT OPTIMISATION WITH GRASSHOPPER AND GSA

It is possible, but computationally very intensive, to use the developed truss modelling tools to generate a Strut-and-Tie Model (STM) and test this model in Oasys GSA. An exemplary solution to such problem is shown in Figure A.1.

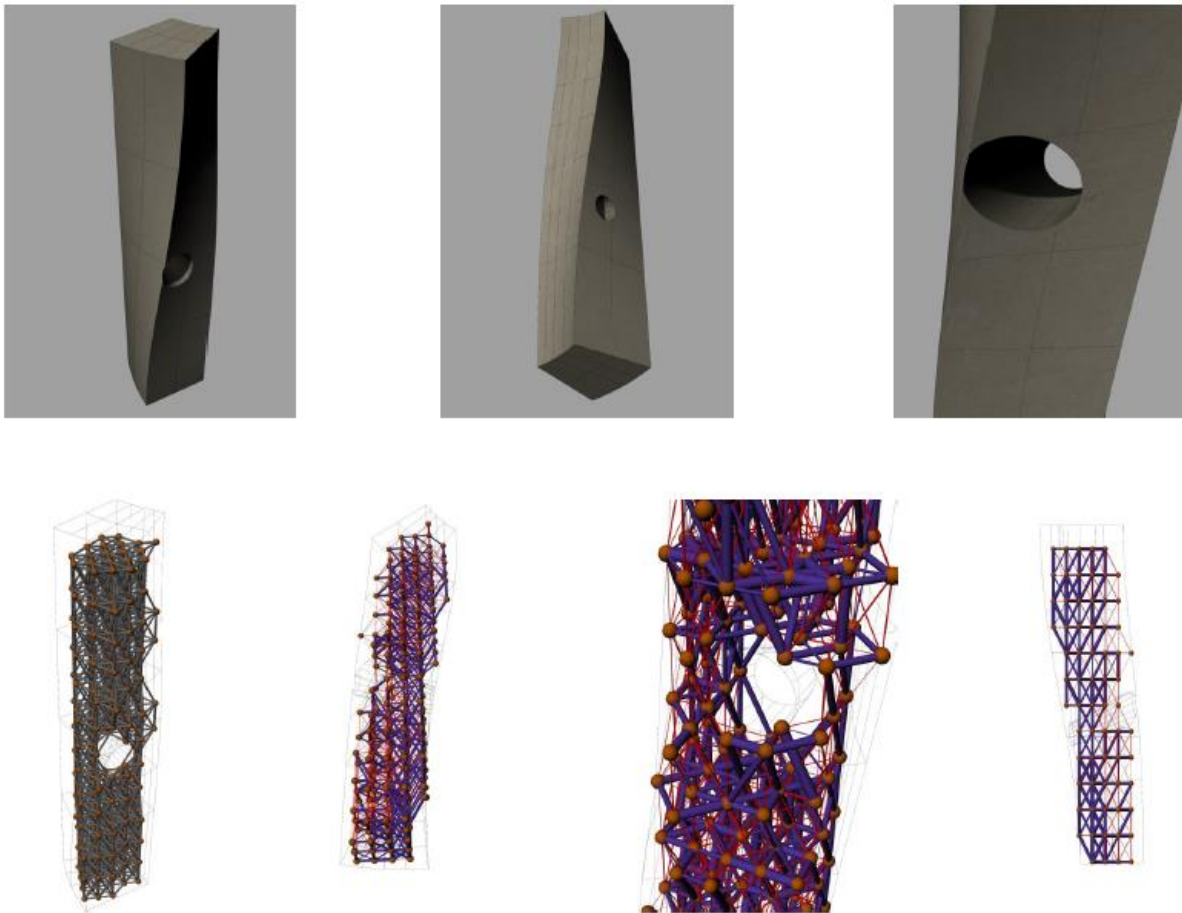


Figure A.1: Reinforcement optimisation with Grasshopper and Oasys GSA, linked by Geometry Gym.

A.2. EXAMPLE: BEHAVIOUR OF PARTICLE SWARM OPTIMISATION (PSO)

The information in this example is based on the examination of properties in Section 3.6.3. The goal of the experiment is to gain some insight in the behavioural properties of PSO algorithms. Specifically the influence of the amount of particles, and the impact of constants c_1 and c_2 are interesting. Figures A.2, A.3 and A.4 show the results of the behavioural study. The Excel sheets used to compute output are shown on the following pages. These sheets contain results for the situation $c_1 = c_2 = 0.1$, using the standard PSO formulas and the given objective function.

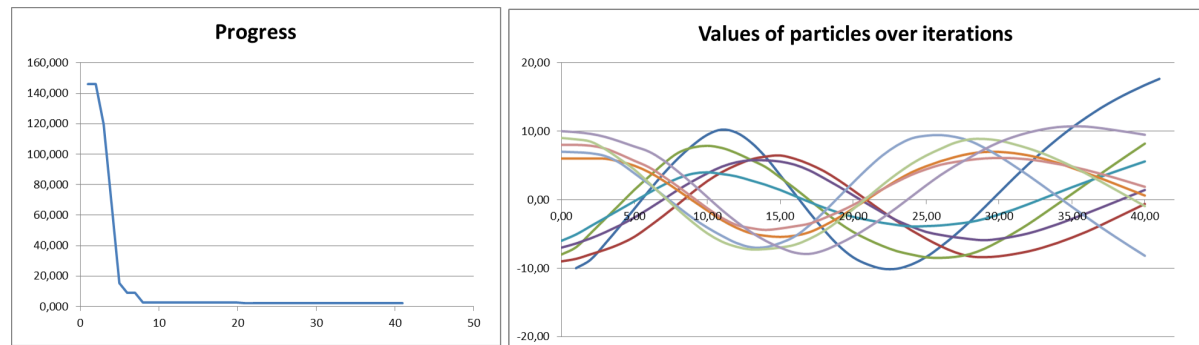


Figure A.2: Particle progress for $c_1=c_2=0.1$. A low value for c_1 and c_2 can result in accuracy, but slow convergence.

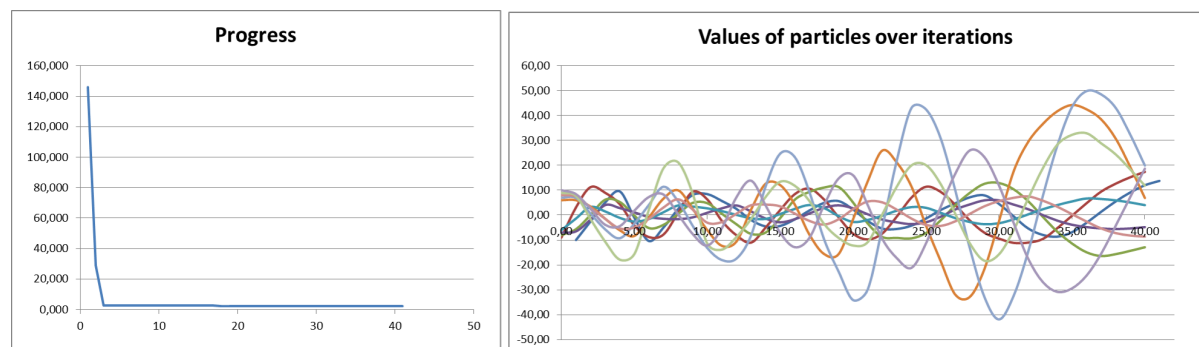


Figure A.3: Particle progress for $c_1=c_2=1$. A normal value for c_1 and c_2 should result in reasonable accuracy and convergence.

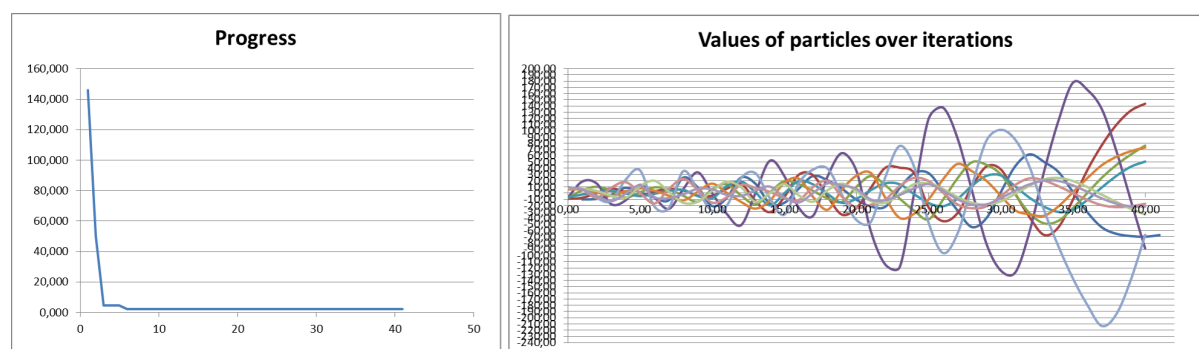


Figure A.4: Particle progress for $c_1=c_2=2$. A high value for c_1 and c_2 can result in low accuracy and divergence of parameters. Note that particles tend to behave much more random in this situation.

generation:	0	1	2	3	4	5	6	7
x1,i	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
x2,i	2,00	1,98	1,87	1,75	1,59	1,42	1,25	1,07
x3,i	3,00	2,93	2,79	2,49	2,15	1,80	1,42	1,02
x4,i	4,00	3,73	3,31	2,75	2,05	1,33	0,60	-0,14
x5,i	5,00	4,83	4,52	3,96	3,26	2,49	1,71	0,88
x6,i	6,00	5,60	4,79	3,71	2,57	1,32	0,05	-1,21
x7,i	7,00	6,81	6,47	6,02	5,54	4,70	3,56	2,41
x8,i	8,00	7,67	7,22	6,56	5,90	5,21	4,44	3,44
x9,i	9,00	8,53	7,36	6,07	4,74	3,17	1,54	-0,16
x10,i	10,00	9,83	9,60	8,83	7,55	5,76	3,95	2,08
f1,i	11,000	11,000	11,000	11,000	11,000	11,000	11,000	10,326
f2,i	25,745	25,566	24,511	23,449	22,280	21,273	20,290	17,817
f3,i	40,873	39,502	36,868	31,961	27,369	23,872	21,261	13,394
f4,i	65,915	58,233	47,642	36,119	26,306	20,766	2,446	7,695
f5,i	100,936	94,117	82,742	64,599	46,335	31,980	23,161	3,119
f6,i	145,949	126,616	92,776	57,547	33,145	20,724	5,800	25,574
f7,i	200,958	189,866	170,738	146,776	123,990	89,293	53,664	30,711
f8,i	265,964	243,687	214,223	175,745	140,895	109,734	80,222	50,737
f9,i	340,968	304,463	223,432	149,349	90,896	44,534	21,991	7,920
f10,i	425,972	410,703	390,537	327,879	235,275	134,062	64,540	26,641
xpr1,i	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
xpr2,i	2,00	1,98	1,87	1,75	1,59	1,42	1,25	1,07
xpr3,i	3,00	2,93	2,79	2,49	2,15	1,80	1,42	1,02
xpr4,i	4,00	3,73	3,31	2,75	2,05	1,33	0,60	0,60
xpr5,i	5,00	4,83	4,52	3,96	3,26	2,49	1,71	0,88
xpr6,i	6,00	5,60	4,79	3,71	2,57	1,32	0,05	0,05
xpr7,i	7,00	6,81	6,47	6,02	5,54	4,70	3,56	2,41
xpr8,i	8,00	7,67	7,22	6,56	5,90	5,21	4,44	3,44
xpr9,i	9,00	8,53	7,36	6,07	4,74	3,17	1,54	-0,16
xpr10,i	10,00	9,83	9,60	8,83	7,55	5,76	3,95	2,08
Fpr1,i	11,000	11,000	11,000	11,000	11,000	11,000	11,000	10,326
Fpr2,i	25,745	25,566	24,511	23,449	22,280	21,273	20,290	17,817
Fpr3,i	40,873	39,502	36,868	31,961	27,369	23,872	21,261	13,394
Fpr4,i	65,915	58,233	47,642	36,119	26,306	20,766	2,446	2,446
Fpr5,i	100,936	94,117	82,742	64,599	46,335	31,980	23,161	3,119
Fpr6,i	145,949	126,616	92,776	57,547	33,145	20,724	5,800	5,800
Fpr7,i	200,958	189,866	170,738	146,776	123,990	89,293	53,664	30,711
Fpr8,i	265,964	243,687	214,223	175,745	140,895	109,734	80,222	50,737
Fpr9,i	340,968	304,463	223,432	149,349	90,896	44,534	21,991	7,920
Fpr10,i	425,972	410,703	390,537	327,879	235,275	134,062	64,540	26,641
Xwr,i	1,00	1,00	1,00	1,00	1,00	1,00	0,60	0,60
Fwr,i	11,000	11,000	11,000	11,000	11,000	11,000	2,446	2,446
v1,i	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
v2,i	0,00	-0,02	-0,11	-0,13	-0,16	-0,17	-0,17	-0,18
v3,i	0,00	-0,07	-0,14	-0,30	-0,34	-0,35	-0,38	-0,40
v4,i	0,00	-0,27	-0,42	-0,57	-0,69	-0,73	-0,73	-0,73
v5,i	0,00	-0,17	-0,31	-0,56	-0,70	-0,76	-0,78	-0,83
v6,i	0,00	-0,40	-0,81	-1,08	-1,14	-1,25	-1,27	-1,26
v3,i	0,00	-0,19	-0,34	-0,46	-0,48	-0,84	-1,14	-1,15
v4,i	0,00	-0,33	-0,46	-0,65	-0,67	-0,68	-0,77	-1,00
v5,i	0,00	-0,47	-1,17	-1,30	-1,33	-1,57	-1,63	-1,70
v6,i	0,00	-0,17	-0,23	-0,76	-1,29	-1,79	-1,80	-1,87
c1	0,1	0,10	0,10	0,10	0,09	0,09	0,09	0,09
c2	0,1	0,10	0,10	0,10	0,09	0,09	0,09	0,09
	40							

generation:	8	9	10	11	12	13	14	15
x1,i	0,95	0,91	0,87	0,82	0,77	0,72	0,66	0,61
x2,i	0,50	0,26	0,05	-0,13	-0,26	-0,33	-0,33	-0,25
x3,i	0,51	0,10	-0,29	-0,64	-0,90	-1,02	-1,01	-0,85
x4,i	0,01	-0,60	-1,17	-1,63	-1,94	-2,02	-1,90	-1,54
x5,i	1,24	0,53	-0,19	-0,85	-1,47	-1,86	-2,05	-2,12
x6,i	-0,25	-1,18	-2,04	-2,77	-3,08	-3,03	-2,54	-1,78
x7,i	-1,36	-2,51	-3,56	-4,23	-4,44	-4,12	-3,31	-2,44
x8,i	-3,10	-4,51	-5,71	-6,82	-7,06	-6,57	-5,69	-4,39
x9,i	2,12	0,56	-0,98	-2,31	-3,52	-4,19	-4,56	-4,56
x10,i	1,14	-0,56	-2,21	-3,71	-5,09	-5,83	-5,94	-5,64
f1,i	5,342	3,768	2,981	2,565	2,361	2,294	2,321	2,405
f2,i	2,764	4,052	5,818	7,620	9,096	10,096	10,062	9,078
f3,i	2,717	5,320	9,469	14,634	19,275	21,468	21,392	18,197
f4,i	6,167	13,906	24,711	35,596	44,179	46,593	43,267	33,454
f5,i	20,243	2,660	8,249	18,266	31,603	41,893	47,698	49,885
f6,i	9,019	24,936	47,318	72,126	84,313	82,345	63,725	39,680
f7,i	28,991	62,596	105,110	137,648	149,126	132,013	93,985	60,107
f8,i	85,122	152,623	226,053	306,932	325,781	287,762	224,539	146,300
f9,i	27,028	2,530	20,772	55,893	103,100	135,831	155,883	155,828
f10,i	19,382	13,379	52,672	111,875	186,296	234,338	241,720	221,422
xpr1,i	0,95	0,91	0,87	0,82	0,77	0,72	0,72	0,72
xpr2,i	0,76	0,76	0,76	0,76	0,76	0,76	0,76	0,76
xpr3,i	0,51	0,51	0,51	0,51	0,51	0,51	0,51	0,51
xpr4,i	0,68	0,68	0,68	0,68	0,68	0,68	0,68	0,68
xpr5,i	1,24	0,53	0,53	0,53	0,53	0,53	0,53	0,53
xpr6,i	0,72	0,72	0,72	0,72	0,72	0,72	0,72	0,72
xpr7,i	-0,09	-0,09	-0,09	-0,09	-0,09	-0,09	-0,09	-0,09
xpr8,i	0,58	0,58	0,58	0,58	0,58	0,58	0,58	0,58
xpr9,i	2,12	0,56	0,56	0,56	0,56	0,56	0,56	0,56
xpr10,i	1,14	-0,56	-0,56	-0,56	-0,56	-0,56	-0,56	-0,56
Fpr1,i	5,342	3,768	2,981	2,565	2,361	2,294	2,294	2,294
Fpr2,i	2,341	2,341	2,341	2,341	2,341	2,341	2,341	2,341
Fpr3,i	2,717	2,717	2,717	2,717	2,717	2,717	2,717	2,717
Fpr4,i	2,303	2,303	2,303	2,303	2,303	2,303	2,303	2,303
Fpr5,i	20,243	2,660	2,660	2,660	2,660	2,660	2,660	2,660
Fpr6,i	2,294	2,294	2,294	2,294	2,294	2,294	2,294	2,294
Fpr7,i	7,160	7,160	7,160	7,160	7,160	7,160	7,160	7,160
Fpr8,i	2,494	2,494	2,494	2,494	2,494	2,494	2,494	2,494
Fpr9,i	27,028	2,530	2,530	2,530	2,530	2,530	2,530	2,530
Fpr10,i	19,382	13,379	13,379	13,379	13,379	13,379	13,379	13,379
Xwr,i	0,72	0,72	0,72	0,72	0,72	0,72	0,72	0,72
Fwr,i	2,294	2,294	2,294	2,294	2,294	2,294	2,294	2,294
v1,i	-0,03	-0,04	-0,04	-0,05	-0,05	-0,05	-0,05	-0,05
v2,i	-0,26	-0,23	-0,22	-0,18	-0,13	-0,08	0,00	0,08
v3,i	-0,42	-0,41	-0,39	-0,36	-0,26	-0,11	0,00	0,17
v4,i	-0,67	-0,61	-0,58	-0,45	-0,31	-0,08	0,11	0,36
v5,i	-0,70	-0,71	-0,71	-0,66	-0,62	-0,39	-0,20	-0,07
v6,i	-0,97	-0,93	-0,86	-0,73	-0,31	0,05	0,49	0,76
v3,i	-1,27	-1,15	-1,05	-0,66	-0,22	0,32	0,81	0,87
v4,i	-1,76	-1,41	-1,20	-1,11	-0,24	0,49	0,89	1,30
v5,i	-1,47	-1,55	-1,55	-1,33	-1,21	-0,67	-0,37	0,00
v6,i	-1,69	-1,71	-1,65	-1,50	-1,38	-0,74	-0,11	0,30
c1	0,0825	0,08	0,08	0,08	0,07	0,07	0,07	0,07
c2	0,0825	0,08	0,08	0,08	0,07	0,07	0,07	0,07

generation:	16	17	18	19	20	21	22	23
x1,i	0,59	0,55	0,51	0,49	0,47	0,47	0,48	0,50
x2,i	0,48	0,65	0,82	0,99	1,15	1,30	1,42	1,51
x3,i	-0,61	-0,44	-0,23	0,05	0,40	0,78	1,16	1,51
x4,i	-0,34	0,40	1,15	1,88	2,55	3,12	3,53	3,79
x5,i	-2,93	-2,30	-1,56	-0,68	0,25	1,19	2,11	2,97
x6,i	-1,69	-0,84	0,13	1,14	2,10	2,99	3,71	4,36
x7,i	-1,89	-0,87	0,20	1,27	2,34	3,30	4,03	4,69
x8,i	-2,88	-1,66	-0,27	1,17	2,57	3,87	5,02	5,94
x9,i	-4,52	-3,65	-2,52	-1,26	0,02	1,32	2,58	3,74
x10,i	-2,63	-0,98	0,76	2,50	4,18	5,64	6,87	7,86
f1,i	2,458	2,583	2,706	2,805	2,872	2,871	2,859	2,769
f2,i	2,847	2,344	2,538	9,028	19,494	20,617	21,259	21,808
f3,i	14,113	11,536	8,798	5,765	3,207	2,391	19,518	21,820
f4,i	10,167	3,235	19,444	24,552	32,850	43,334	52,824	59,912
f5,i	78,203	55,664	33,835	15,228	4,163	19,881	26,938	40,296
f6,i	37,141	18,040	5,081	19,328	26,831	40,622	57,721	77,465
f7,i	42,715	18,583	4,520	20,453	29,781	47,283	66,856	88,927
f8,i	76,295	36,414	9,247	19,639	33,104	62,150	101,758	143,172
f9,i	153,272	109,025	62,895	26,731	6,056	20,708	33,271	58,408
f10,i	66,838	20,655	2,342	32,022	71,616	128,704	193,044	256,389
xpr1,i	0,69	0,69	0,69	0,69	0,69	0,69	0,69	0,69
xpr2,i	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73
xpr3,i	0,74	0,74	0,74	0,74	0,74	0,74	0,74	0,74
xpr4,i	0,86	0,86	0,86	0,86	0,86	0,86	0,86	0,86
xpr5,i	0,58	0,58	0,58	0,58	0,58	0,58	0,58	0,58
xpr6,i	0,49	0,49	0,49	0,49	0,49	0,49	0,49	0,49
xpr7,i	0,02	0,02	0,20	0,20	0,20	0,20	0,20	0,20
xpr8,i	-0,37	-0,37	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27
xpr9,i	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08
xpr10,i	-0,28	-0,28	0,76	0,76	0,76	0,76	0,76	0,76
Fpr1,i	2,296	2,296	2,296	2,296	2,296	2,296	2,296	2,296
Fpr2,i	2,303	2,303	2,303	2,303	2,303	2,303	2,303	2,303
Fpr3,i	2,313	2,313	2,313	2,313	2,313	2,313	2,313	2,313
Fpr4,i	2,905	2,905	2,905	2,905	2,905	2,905	2,905	2,905
Fpr5,i	2,500	2,500	2,500	2,500	2,500	2,500	2,500	2,500
Fpr6,i	2,818	2,818	2,818	2,818	2,818	2,818	2,818	2,818
Fpr7,i	6,104	6,104	4,520	4,520	4,520	4,520	4,520	4,520
Fpr8,i	10,515	10,515	9,247	9,247	9,247	9,247	9,247	9,247
Fpr9,i	5,489	5,489	5,489	5,489	5,489	5,489	5,489	5,489
Fpr10,i	9,389	9,389	2,342	2,342	2,342	2,342	2,342	2,342
Xwr,i	0,69	0,69	0,69	0,69	0,69	0,69	0,69	0,69
Fwr,i	2,296	2,296	2,296	2,296	2,296	2,296	2,296	2,296
v1,i	-0,05	-0,04	-0,03	-0,02	-0,02	0,00	0,00	0,02
v2,i	0,16	0,17	0,17	0,17	0,17	0,15	0,11	0,09
v3,i	0,09	0,17	0,21	0,28	0,35	0,38	0,37	0,36
v4,i	0,64	0,74	0,75	0,73	0,67	0,57	0,41	0,27
v5,i	0,48	0,62	0,75	0,88	0,93	0,94	0,92	0,86
v6,i	0,76	0,85	0,97	1,01	0,96	0,89	0,73	0,65
v3,i	0,90	1,02	1,07	1,07	1,07	0,96	0,73	0,66
v4,i	1,03	1,22	1,39	1,44	1,40	1,31	1,15	0,92
v5,i	0,45	0,87	1,13	1,25	1,28	1,30	1,26	1,16
v6,i	1,57	1,65	1,74	1,73	1,69	1,46	1,22	1,00
c1	0,0625	0,06	0,06	0,06	0,05	0,05	0,05	0,05
c2	0,0625	0,06	0,06	0,06	0,05	0,05	0,05	0,05

generation:	24	25	26	27	28	29	30	31
x1,i	0,57	0,63	0,68	0,75	0,81	0,86	0,92	0,97
x2,i	1,23	1,36	1,48	1,57	1,64	1,67	1,68	1,66
x3,i	1,78	2,01	2,17	2,32	2,42	2,46	2,46	2,42
x4,i	1,16	1,73	2,25	2,71	3,13	3,51	3,83	4,04
x5,i	2,86	3,43	3,91	4,28	4,49	4,62	4,69	4,61
x6,i	3,99	4,31	4,54	4,65	4,65	4,54	4,35	4,05
x7,i	5,11	5,56	5,83	5,93	5,96	5,82	5,46	4,98
x8,i	5,20	5,42	5,53	5,39	5,15	4,85	4,51	4,03
x9,i	7,34	8,02	8,49	8,86	8,80	8,48	8,04	7,45
x10,i	9,06	9,18	8,85	8,29	7,42	6,38	5,32	4,16
f1,i	2,516	2,378	2,300	2,316	2,481	2,911	3,935	6,709
f2,i	20,200	20,960	21,631	22,189	22,625	22,878	22,937	22,794
f3,i	23,707	25,824	27,684	29,564	30,921	31,548	31,415	30,885
f4,i	19,570	23,304	28,639	35,398	43,604	52,361	60,845	67,045
f5,i	38,070	50,421	63,138	74,621	81,948	86,501	89,029	86,178
f6,i	65,533	75,641	83,647	87,597	87,689	83,631	77,019	67,551
f7,i	105,514	124,838	137,412	142,450	143,832	137,209	120,543	100,099
f8,i	109,204	118,743	123,451	117,471	107,185	95,017	82,540	66,936
f9,i	221,797	267,107	301,205	329,534	325,327	300,443	268,661	229,123
f10,i	345,909	355,294	329,271	286,725	227,010	165,652	114,331	70,732
xpr1,i	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71
xpr2,i	0,64	0,64	0,64	0,64	0,64	0,64	0,64	0,64
xpr3,i	0,64	0,64	0,64	0,64	0,64	0,64	0,64	0,64
xpr4,i	0,58	0,58	0,58	0,58	0,58	0,58	0,58	0,58
xpr5,i	0,85	0,85	0,85	0,85	0,85	0,85	0,85	0,85
xpr6,i	0,22	0,22	0,22	0,22	0,22	0,22	0,22	0,22
xpr7,i	0,17	0,17	0,17	0,17	0,17	0,17	0,17	0,17
xpr8,i	0,45	0,45	0,45	0,45	0,45	0,45	0,45	0,45
xpr9,i	0,83	0,83	0,83	0,83	0,83	0,83	0,83	0,83
xpr10,i	-0,68	-0,68	-0,68	-0,68	-0,68	-0,68	-0,68	-0,68
Fpr1,i	2,294	2,294	2,294	2,294	2,294	2,294	2,294	2,294
Fpr2,i	2,359	2,359	2,359	2,359	2,359	2,359	2,359	2,359
Fpr3,i	2,344	2,344	2,344	2,344	2,344	2,344	2,344	2,344
Fpr4,i	2,491	2,491	2,491	2,491	2,491	2,491	2,491	2,491
Fpr5,i	2,814	2,814	2,814	2,814	2,814	2,814	2,814	2,814
Fpr6,i	4,391	4,391	4,391	4,391	4,391	4,391	4,391	4,391
Fpr7,i	4,739	4,739	4,739	4,739	4,739	4,739	4,739	4,739
Fpr8,i	2,991	2,991	2,991	2,991	2,991	2,991	2,991	2,991
Fpr9,i	2,645	2,645	2,645	2,645	2,645	2,645	2,645	2,645
Fpr10,i	15,286	15,286	15,286	15,286	15,286	15,286	15,286	15,286
Xwr,i	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71
Fwr,i	2,294	2,294	2,294	2,294	2,294	2,294	2,294	2,294
v1,i	0,05	0,06	0,06	0,06	0,06	0,06	0,05	0,05
v2,i	0,14	0,13	0,12	0,09	0,06	0,04	0,01	-0,02
v3,i	0,26	0,23	0,17	0,15	0,10	0,04	-0,01	-0,04
v4,i	0,58	0,57	0,52	0,45	0,43	0,38	0,32	0,21
v5,i	0,63	0,57	0,48	0,37	0,22	0,13	0,07	-0,08
v6,i	0,50	0,32	0,23	0,11	0,00	-0,11	-0,19	-0,30
v3,i	0,52	0,45	0,27	0,10	0,03	-0,14	-0,36	-0,48
v4,i	0,53	0,22	0,11	-0,13	-0,24	-0,30	-0,34	-0,48
v5,i	0,74	0,68	0,47	0,37	-0,05	-0,33	-0,44	-0,59
v6,i	0,64	0,12	-0,32	-0,56	-0,87	-1,04	-1,06	-1,17
c1	0,0425	0,04	0,04	0,04	0,03	0,03	0,03	0,03
c2	0,0425	0,04	0,04	0,04	0,03	0,03	0,03	0,03

generation:	32	33	34	35	36	37	38	39
x1,i	0,79	0,83	0,86	0,90	0,93	0,95	0,98	1,00
x2,i	1,40	1,33	1,24	1,15	1,06	0,96	0,86	0,76
x3,i	1,50	1,13	0,76	0,39	0,01	-0,35	-0,69	-1,04
x4,i	4,24	4,09	3,85	3,56	3,26	2,91	2,56	2,19
x5,i	3,29	3,00	2,65	2,28	1,89	1,48	1,07	0,65
x6,i	5,32	5,23	5,08	4,84	4,51	4,14	3,76	3,37
x7,i	2,21	1,13	0,04	-1,06	-2,14	-3,19	-4,25	-5,28
x8,i	6,12	5,13	4,13	3,10	2,06	1,00	-0,07	-1,13
x9,i	9,24	9,07	8,75	8,39	7,87	7,23	6,56	5,83
x10,i	2,06	0,36	-1,34	-3,01	-4,63	-6,19	-7,70	-9,18
f1,i	2,409	2,591	2,898	3,383	4,171	5,506	7,825	11,152
f2,i	21,179	20,750	20,232	19,473	17,155	5,925	2,848	2,333
f3,i	21,734	19,221	2,339	3,297	6,114	10,236	15,512	21,875
f4,i	73,424	68,696	61,429	53,776	46,330	39,141	32,935	27,918
f5,i	47,198	40,843	34,534	28,985	24,643	21,618	17,614	2,334
f6,i	114,444	110,444	104,033	94,607	82,533	70,074	58,868	48,906
f7,i	28,123	19,216	5,898	22,269	50,402	89,047	138,780	198,543
f8,i	152,085	106,187	70,020	42,915	26,337	10,348	6,929	23,704
f9,i	360,654	346,293	321,443	293,996	257,008	215,124	175,459	137,704
f10,i	26,404	3,450	28,548	81,309	159,511	259,633	379,553	519,310
xpr1,i	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71
xpr2,i	0,70	0,70	0,70	0,70	0,70	0,70	0,70	0,70
xpr3,i	0,61	0,61	0,76	0,76	0,76	0,76	0,76	0,76
xpr4,i	0,56	0,56	0,56	0,56	0,56	0,56	0,56	0,56
xpr5,i	0,58	0,58	0,58	0,58	0,58	0,58	0,58	0,65
xpr6,i	0,77	0,77	0,77	0,77	0,77	0,77	0,77	0,77
xpr7,i	-0,33	-0,33	0,04	0,04	0,04	0,04	0,04	0,04
xpr8,i	0,88	0,88	0,88	0,88	0,88	0,88	0,88	0,88
xpr9,i	0,64	0,64	0,64	0,64	0,64	0,64	0,64	0,64
xpr10,i	0,38	0,38	0,38	0,38	0,38	0,38	0,38	0,38
Fpr1,i	2,293	2,293	2,293	2,293	2,293	2,293	2,293	2,293
Fpr2,i	2,293	2,293	2,293	2,293	2,293	2,293	2,293	2,293
Fpr3,i	2,422	2,422	2,339	2,339	2,339	2,339	2,339	2,339
Fpr4,i	2,549	2,549	2,549	2,549	2,549	2,549	2,549	2,549
Fpr5,i	2,477	2,477	2,477	2,477	2,477	2,477	2,477	2,334
Fpr6,i	2,359	2,359	2,359	2,359	2,359	2,359	2,359	2,359
Fpr7,i	10,048	10,048	5,898	5,898	5,898	5,898	5,898	5,898
Fpr8,i	3,117	3,117	3,117	3,117	3,117	3,117	3,117	3,117
Fpr9,i	2,361	2,361	2,361	2,361	2,361	2,361	2,361	2,361
Fpr10,i	3,335	3,335	3,335	3,335	3,335	3,335	3,335	3,335
Xwr,i	0,71	0,71	0,71	0,71	0,71	0,71	0,71	0,71
Fwr,i	2,293	2,293	2,293	2,293	2,293	2,293	2,293	2,293
v1,i	0,04	0,04	0,04	0,03	0,03	0,03	0,03	0,02
v2,i	-0,06	-0,08	-0,09	-0,09	-0,09	-0,10	-0,10	-0,10
v3,i	-0,34	-0,37	-0,37	-0,37	-0,37	-0,36	-0,35	-0,34
v4,i	-0,10	-0,15	-0,24	-0,28	-0,31	-0,34	-0,36	-0,36
v5,i	-0,21	-0,30	-0,34	-0,38	-0,39	-0,41	-0,41	-0,42
v6,i	-0,02	-0,09	-0,15	-0,24	-0,33	-0,37	-0,38	-0,39
v3,i	-1,03	-1,08	-1,09	-1,09	-1,08	-1,05	-1,05	-1,04
v4,i	-0,83	-0,99	-1,00	-1,03	-1,04	-1,06	-1,06	-1,06
v5,i	-0,05	-0,18	-0,31	-0,36	-0,52	-0,64	-0,67	-0,73
v6,i	-1,68	-1,70	-1,70	-1,66	-1,62	-1,56	-1,51	-1,48
c1	0,0225	0,02	0,02	0,02	0,01	0,01	0,01	0,01
c2	0,0225	0,02	0,02	0,02	0,01	0,01	0,01	0,01