Application of a Gradient-Based Algorithm to Structural Optimization

by

Pierre Ghisbain

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN CIVIL AND ENVIRONMENTAL ENGINEERING AT THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2009

Signature of Author: [illegible]

Department of Civil and Environmental Engineering
January 26, 2009

Certified by: [illegible]

Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: [illegible]

Daniele Veneziano
Chairman, Departmental Committee for Graduate Students

Application of a Gradient-Based Algorithm to Structural Optimization

by

Pierre Ghisbain

Submitted to the Department of Civil and Environmental Engineering on January 28, 2009 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Civil and Environmental Engineering at the Massachusetts Institute of Technology.

ABSTRACT

Optimization methods have shown to be efficient at improving structural design, but their use is limited in the engineering practice by the difficulty of adapting state-of-the-art algorithms to particular engineering problems. This study proposes the use of a robust gradient-based algorithm, whose adaptation to a variety of design problems is more straightforward. The algorithm was first applied to truss geometry and beam shape optimization, both forming part of the increasingly popular class of structural form-finding problems. The results showed that the gradient-based method is an appropriate tool for defining shapes in structures. The robustness of the algorithm was verified, as a series of structural configurations were treated with similar efficiency. The gradient-based method was also applied to a more traditional structural design problem through the optimization of a steel girder, resulting in a hybrid scheme featuring a truss stiffener. Throughout the study, emphasis was laid on the practical computer implementation of the gradient-based algorithm in interaction with structural analysis tools.

Thesis Supervisor: Jerome J. Connor
Title: Professor of Civil and Environmental Engineering

ACKNOWLEDGMENTS

My thanks go first to Professor Jerome J. Connor, whose kind supervision and technical advice have guided me since my arrival at MIT. His invaluable experience has been an endless source of supportive suggestions. I wish the MEng class of 2008 much success. They have been an intelligent and entertaining group of people to work with. My best wishes also accompany the MIT Steel Bridge Team, whose project has been an exciting opportunity to apply the concepts developed in this thesis. I am grateful to my parents and to my great uncle Jean for their interest and support in my studies. I also owe much thanks to the Robert Guenassia Fellowship and to the Jean Gaillard Memorial Fund for their additional financial support.

Contents

Chapter 1

Structural Optimization Overview

Reducing costs while meeting performance standards is a common challenge in structural design. Engineers typically rely on experience and standardized design procedures to make their structures more efficient. Though not widely used in the structural engineering practice, more systematic methods based on mathematical algorithms and grouped under the generic name of *Structural Optimization* are available to help designing efficient structures.

This first chapter is a general introduction to structural optimization, emphasizing the reasons that motivated the further study of a particular algorithm. The formulation of structural optimization problems in mathematical terms is first presented. A general solution strategy is introduced, and several methods are detailed. Past research works are summarized, stressing the difficulties of applying optimization in the practice of structural engineering and leading to a study proposal.

1.1 Structural Optimization Problems

Optimization is a vast field of mathematics whose theory is still actively being developed. But when applied to structural engineering, it is essentially regarded as a tool helpful to the engineer willing to design more efficient structures. The traditional gap between mathematics and engineering must be bridged in order to use the optimization theory to solve actual design problems. This is done through appropriate formulation of the structural engineering problems, which are written as mathematical expressions that can be handled by optimization algorithms.

1.1.1 Mathematical Formulation

Mathematicians have divided the field of optimization into several problem categories, each type of problem being solved by applying specific strategies. Structural systems often have nonlinear properties, and all structures are subject to physical constraints. Therefore, it is somehow natural to resort to the branch of mathematics referred to as *nonlinear constrained optimization.* The general nonlinear constrained optimization problem can be stated as follows:

$$\begin{array}{rl} \text{find} & \underline{x} \\ \text{to minimize} & f(\underline{x}) \\ \text{subject to} & \underline{g}(\underline{x}) = 0 \\ \text{and} & \underline{h}(\underline{x}) \leq 0 \end{array}$$

Optimization Variables

$\underline{x} = (x_1 , x_2 \ldots x_n)$ is a set of variables whose values are modified during the optimization process. Each variable x_i can be binary, discrete or continuous.

Objective Function

$f(\underline{x})$ is a scalar function of the optimization variables. The goal of the optimization process is to minimize the value of f by adjusting the variables composing $\underline{x}$.

Constraints

$$\underline{g}(\underline{x}) = \begin{pmatrix} g_1(\underline{x}) \\ g_2(\underline{x}) \\ \vdots \\ g_p(\underline{x}) \end{pmatrix} \quad \text{and} \quad \underline{h}(\underline{x}) = \begin{pmatrix} h_1(\underline{x}) \\ h_2(\underline{x}) \\ \vdots \\ h_q(\underline{x}) \end{pmatrix} \quad \text{are vector functions of } \underline{x}.$$

A set of optimization variables $\underline{x}$ is acceptable if $\begin{cases} g_i(\underline{x}) = 0 & \text{for} \quad i = 1 \ldots p \\ h_j(\underline{x}) \leq 0 & \text{for} \quad j = 1 \ldots q \end{cases}$

1.1.2 Structural Engineering Formulation

Various mathematical methods, referred to as *algorithms*, have been developed to solve the generic problem presented in section 1.1.1. The principle of structural optimization is to express a structural engineering problem in the generic mathematical form and to solve it using one of the available algorithms. This section presents what the variables, objective functions and constraints can be in structural engineering.

Optimization Variables

The variables considered in structural design optimization can be any feature of the structure being optimized. When solving a problem, the choice of the optimization algorithm greatly depends on the type of variables involved. In particular, it is important to distinguish binary, discrete and continuous variables.

Binary variables are mostly used in connection design and topology optimization (see section 1.3.2). Examples of such variables include: presence/absence of a bracing member in a building frame, pinned/rigid connection at a joint.

Discrete variables are typically used to count structural elements and to represent the properties of structural members available in standard sizes (see section 1.3.1). Examples of such variables include: wide flange section assigned to a beam, number of columns in a building, number of bolts forming a connection.

Continuous variables can represent geometrical features of structures and are also used in structural member sizing. Examples of such variables include: column-to-column distance in a building, location of a truss node, thickness of a shear wall.

Variables of different categories can be combined to better represent engineering problems, although the resolution is usually simpler and quicker when a single type of variable is involved. Limiting the number of variables is also key to the convergence speed of the optimization algorithm. However, with computers capabilities increasing fast, there is a tendency to use more variables in optimization problems, thus reducing the number of assumptions to be made.

Objective Function

The objective function shall represent the goal of the optimization process. Except for some very high performance structures, good engineering design is a balance between performance and cost, making structural optimization a multi-objective problem. Two strategies are employed to end up with a single objective function, which is necessary to implement efficient optimization algorithms. A first approach is to use weighting factors to build a single composite function out of several objectives. This method has been used, in particular, to take into account both technical and architectural considerations in conceptual design problems (Merello, 2006). The potential applications of this approach seem limited, as composite objective functions are somehow arbitrary. The other strategy is to select a single optimization objective to be minimized and to express all other objectives as constraints to be satisfied. This approach is natural in the majority of actual engineering problems. A typical scenario is to minimize the weight of a structure considering the maximum allowable deflection as a constraint.

Purely technical objectives such as weight or stiffness have been less used in recent works, as minimizing the overall cost of structures is what the industry is interested in. However, cost estimation in terms of the optimization variables is often problematic. Sustainability objectives, such as minimizing the total embodied energy, may become more important in structural optimization.

Constraints

Two categories of constraints are distinguished in structural optimization, depending on what they apply to.

A first type of constraints are directly applied to the optimization variables. Such constraints are used, in particular, to set the boundaries of the continuous design parameters. Constraints can also relate several variables. For example, if two variables are used to represent the outer diameter and the wall thickness of a steel pipe, a constraint must impose that the wall be thinner than half of the diameter at all times.

The other category of constraints applies to the structure being optimized. Deflection criteria and maximum allowable stresses are typical example of such constraints, often imposed by the construction codes. Structural analysis is required to check whether these constraints are respected in a particular design.

1.2 Solution Strategy

The structural optimization problems introduced in section 1.1 can be solved using a very diverse range of methods. This section presents the key concepts common to all structural optimization techniques and governing their implementation on computers. The main classes of solution strategies are then distinguished.

1.2.1 Algorithmic Approach

A variety of methods have been developed to solve the optimization problems in their mathematical form. These methods, grouped under the name of *optimization algorithms*, are able to minimize an objective function by adjusting variables while satisfying constraints. In structural optimization, the variables, objective and constraints represent physical properties of the structure being optimized. Since the algorithms deal exclusively with the mathematical form of the problem, they are interfaced with computer models representing the physical structure. The model is used to perform structural analyses requested by the optimization algorithm. The exact interaction scheme between the algorithm and the analysis tool depends optimization method, but a general flow chart is represented on figure 1.1.

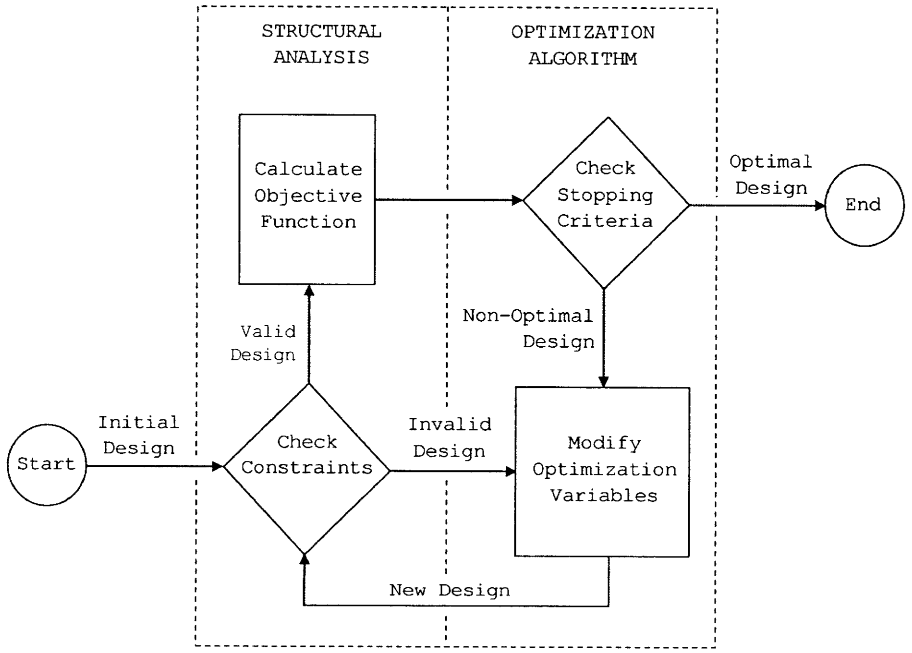


Figure 1.1: General Structural Optimization Flow Chart

The structural optimization process is iterative. The algorithm generates a design by assigning values to the optimization variables. After being updated with the new values of the variables, the structural model is used to perform analysis. The quality of the current design is characterized by the value of the objective function and the state of the constraints, both obtained from the analysis results. These results are taken into account by the algorithm to generate new designs, as long as the termination criteria are not met.

Each iteration involves algorithmic steps (variables modification and stopping criteria check) and structural analysis steps (constraints check and objective function calculation). Analysis steps are typically much longer than algorithmic steps . Quick structural analysis, using efficient programs and simple models, is key to the speed of the overall process. Optimization time is also reduced by choosing an algorithm that converges quickly towards the optimal design, thus requiring fewer iterations and time-consuming analysis steps.

In this study, the Matlab® computing environment was used to run optimization algorithms, available through built-in optimization toolboxes. An algorithm is seen as a black box and interacts with two pieces of Matlab® code (.m files) created by the user to define the objective and constraints of the problem (figure 1.2). The optimization variables (`x`) are handled by the algorithm. The external functions (`objective.m`, `constraints.m`) are called by the algorithm whenever it needs to know the value of the objective function (`f`) or the state of the constraints (`c,ceq`) for a given design. All details can be found in the *Optimization Toolbox™ 3 User's Guide* (The MathWorks™, 2007b).

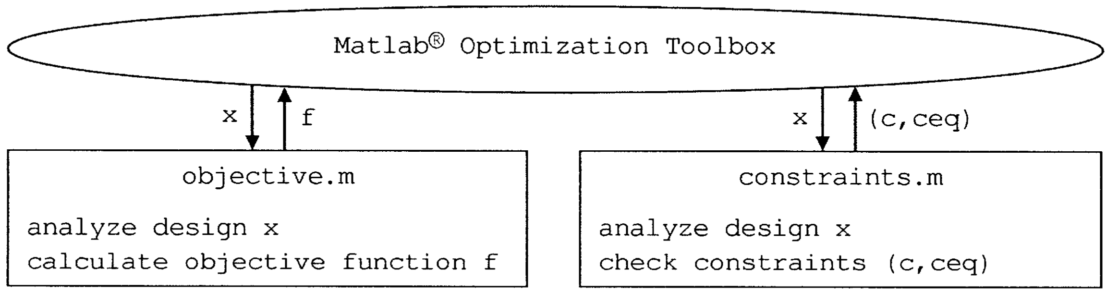

Matlab® Optimization Toolbox
x
f
x
(c,ceq)
objective.m
analyze design x
calculate objective function f
constraints.m
analyze design x
check constraints (c,ceq)

Figure 1.2: Matlab® Optimization Toolbox Flow Chart

1.2.2 Optimization Algorithms

While engineers focus on applying existing algorithms to solve physical problems, the mathematics of optimization is still being developed. Dozens of algorithms are available to solve the nonlinear constrained optimization problems considered in structural engineering. The three groups of algorithms distinguished in this section do not constitute a formal classification of the optimization methods. This is more of a quick introduction to different types of algorithms, whose applicability and implementation differ in the context of structural optimization. Moreover, the few algorithms mentioned here are far from being an exhaustive list of the available optimization methods.

Gradient-Based Algorithms

Gradient-based algorithms seek to modify the optimization variables that have the greatest effect on the objective function. The concept of gradient is used to determine the influence of each variable on the value of the objective function. Since the gradient cannot be explicitly calculated in most cases, the algorithm estimates it by slightly changing the value of each variable and measuring the subsequent effect on the objective function. If the constraints allow it, the change that had the greatest effect on decreasing the value of the objective function is amplified to generate a new design and finish the iteration. The process is repeated until a termination criterion is met. Initially developed to deal with continuous variables, gradient-based algorithms can be adapted to handle discrete parameters as well, though with a loss of efficiency. Algorithms are also modified to prevent them from converging towards local optimums. A great advantage of the gradient-based algorithms is their inherent self-adaptivity. At each iteration, the optimization variables are adjusted with an appropriate magnitude, based on the value of the gradient.

Search Algorithms

As opposed to gradient-based methods, search algorithms generate new designs at the beginning of each iteration. Starting with a single current design, a series of new potential designs is generated by modifying one or several optimization variables. These changes are more or less arbitrary depending on the algorithm, but a certain degree of randomness is often involved. Then, the value of the objective function is calculated and compliance with the constraints is checked for each new design. The acceptable design with the lowest objective function value is selected as the new current design, finishing the iteration. The process is repeated until a termination criterion is met. With no gradient involved, search algorithms are well-suited to handle discrete variables, which are very common in structural optimization. However, implementing these algorithms is not always easy, as parameters governing the generation of the new designs need to be adapted to each particular problem for the process to converge properly. The convergence of a search algorithm is also influenced by the initial design considered. Figure 1.3 shows optimization results obtained with the same algorithm but using different initial conditions. Examples of search algorithms that have been applied to structural optimization include Simulated Annealing (Kost and Baumann, 2001), Pattern Search (Baldock et al., 2005), Tabu Search (Kargahi et al., 2006) and Big Bang/Big Crunch Optimization (Camp, 2007). More details about these algorithms can be found in the references cited.

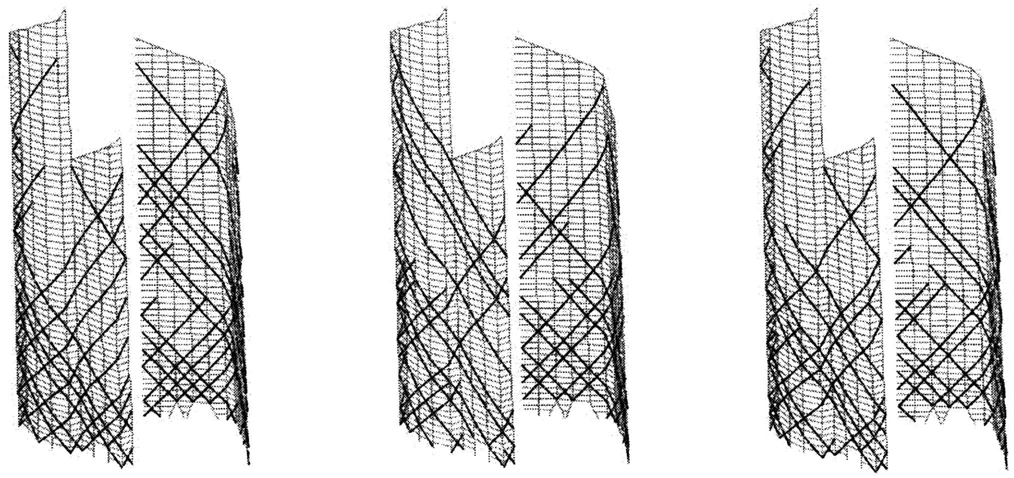


Figure 1.3: Facade Bracing Optimization by Pattern Search (Baldock et al., 2005)

Evolutionary Algorithms

Evolutionary algorithms include the popular genetic algorithms, which have been applied in many fields of science and engineering to solve optimization problems. Like the search methods previously described, genetic algorithms generate a series of new designs at each iteration. But instead of keeping a single current design, genetic algorithms maintain a population of designs that evolves through the optimization process. The evolution of the design population is inspired by Darwin's *survival of the fittest* theory and is governed by mathematical methods seeking to mimic genetics, each optimization variable representing a gene. At each iteration, parents are selected among the best designs of the current population. The values of their optimization variables are mixed to generate children designs, and random changes are also applied to prevent early convergence of the population. These operations are described as genetic crossovers and mutations. The best children are added to the population, while old and less fit designs are removed from it. The process is repeated until a homogeneous design population is obtained.

The general genetic algorithm pseudo-code is as follows:

```
Generate initial population
Evaluate each individual fitness
Repeat until termination criterion is met
      Pick parents among the best individuals (selection)
      Generate children by mixing the parents properties (crossover)
      Applies random changes to the children properties (mutation)
      Update population with best children
```

The implementation difficulties mentioned for search algorithms are also encountered with genetic algorithms. The behavior of genetic algorithms is determined by parameters governing the selection, crossover and mutation processes, which need to be adjusted for each particular optimization case. Nevertheless, genetic algorithms have been applied to a variety of structural optimization problems, such as truss members sizing (Rajeev and Krishnamoorthy, 1992), truss geometry definition (Kost, 2003), steel frame members sizing (Foley et al., 2007) and shear wall placement in building frames (van de Lindt and Dao, 2007).

1.3 State-of-the-Art Review

Even though the use of optimization is still limited in structural engineering practice, a great deal of research has been carried out in this field. An article by Cohn and Dinovitzer (2004) summarizes a review of 500 published structural optimization examples. All types of structures have been treated, and a variety of optimization algorithms have been used. Other research works have focused on developing algorithms for specific use in structural optimization. This section presents the main research trends and the potential applications of structural optimization.

1.3.1 Standard Sizing

The primary potential use of optimization in structural engineering is probably to size the elements composing a structure. It is not always easy to understand the contribution of each particular member to the overall performance of a large structure, and optimization can help designing a system meeting a given performance criterion at a minimal cost. Sizing the members of a steel frame is a common and repetitive task in structural engineering practice, which explains why many attempts to automate and optimize the process have been made.

Steel members are most often made of standard steel shapes. The variables representing the steel shapes in the optimization process are therefore discrete. Arora (2000) presented 8 algorithms dealing with discrete variables and successfully applied to structural member sizing. Many examples of frames and trusses member sizing have been published.

Rigid frames have been optimized using a range of algorithms, such as Simulated Annealing (Balling, 1991), Ant Colony Optimization (Camp et al., 2004), Tabu Search (Kargahi and Anderson, 2006a) and Genetic Algorithms (Alimoradi et al., 2007). Several algorithms have been used for truss member sizing as well, such as Genetic Algorithms (Rajeev and Krishnamoorthy, 1992), Outer-Approximation/Equality-Relaxation (Šilih and Kravanja, 2003), Tabu Search (Kargahi et al., 2006) and Big Bang/Big Crunch optimization (Camp, 2007).

1.3.2 Topology Definition

The topology of a structure is defined by the arrangement of its constituting elements. To optimize the topology is to find the combination of structural elements forming the most efficient structure. Because of its higher level of abstraction, structural topology optimization is a relatively recent research topic and has little application in the industry. However, recent developments in topology optimization have been driven by the growing popularity of form-finding architecture.

Improving bracing schemes in steel frames is a potential application of topology optimization. Bracing is a traditional bone of contention between architects and engineers, and to minimize the number of braces and their effect on the appearance of the structure is interesting. Bracing schemes have been optimized using continuum-based optimization (Mijar et al., 1998), in which frames are fully braced by a fictitious continuum that is gradually removed to end up with a discrete bracing system (figure 1.4). More classical algorithms, working by addition and removal of bracing elements, have also been used. Baldock et al. (2005) used a Pattern Search algorithm to optimize the topology of a braced facade on a freeform building. Similar studies were carried out on shear wall placement (van de Lindt and Dao, 2007).

Topology optimization has also been used to define the shape of full structures. Reasonable results were obtained for sparse structural systems, such as truss bridges and transmission towers (Rahmatalla and Swan, 2003). As form-finding is becoming increasingly popular, further developments in topology optimization are expected.

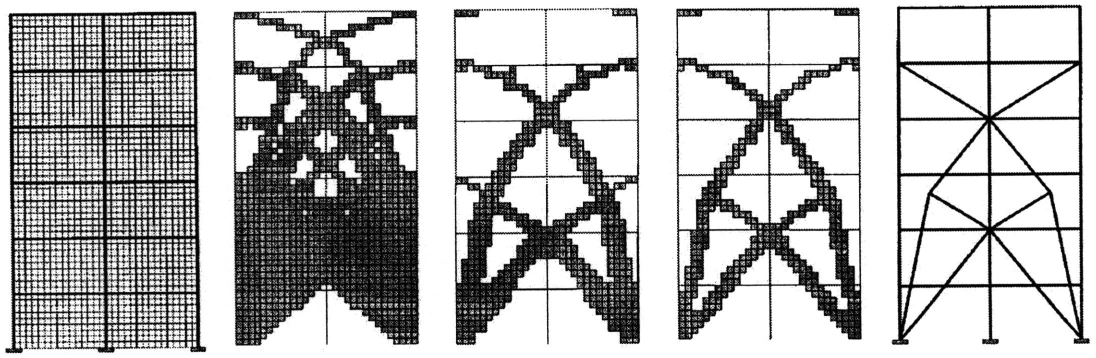


Figure 1.4: Continuum Bracing Topology Optimization (Mijar et al., 1998)

1.3.3 Structural Member Design

The previously described works aim at increasing the overall performance of structures. Optimization has also been implemented at a lower level to improve the design of individual structural members.

Slender members, such as beams and columns, can be optimized in two ways. Transverse optimization improves the design of the cross-section, while longitudinal optimization varies the cross-sectional properties over the length of the member to end up with a more efficient material distribution.

Steel structural members, and wide flanges shapes in particular, are available in standard sizes of constant cross-sections. Longitudinal optimization of steel members has little practical application, as it is usually not considered economically worthwhile to fabricate steel members of varying cross-sections and designed for a specific structure and loading. Transverse optimization has more potential applications, such as the design of buckling-resistant cross-sections to be used in high-performance columns (Liu et al., 2004).

Optimization of concrete members is of greater interest, as the use of formworks makes it relatively easy to fabricate optimized concrete members. Depending on the loading, appropriate cross-sections can be designed and varied over the length of the member to better distribute the material. Some very practical applications have been proposed, such as the design of an optimized box girder for concrete bridges (Cohn and Lounis, 1994). In reinforced concrete structures, the cross-section of every member needs to be designed and the reinforcement layout can be optimized to minimize the cost and simplify fabrication (Balling and Yao, 1997).

1.3.4 Algorithms Development

The bulk of the published works on structural optimization present the application of existing algorithms to structural engineering problems. In parallel, some have focused on modifying the algorithms and developing new methods in order to provide more efficient tools for structural optimization.

Topology optimization problems are highly nonlinear, as the addition or removal of a single member can greatly modify the behavior of a structure. This property makes it difficult to implement traditional optimization algorithms, whose efficiency typically decreases with the degree of nonlinearity. The concept of topological derivatives (Mróz and Bojczuk, 2000) have been used to adapt the optimization algorithms to topology problems, while the analysis method presented by Kirsch and Papalambros (2000) allows quick calculation of the effect of a topological modification on the overall performance of a structure. Both of these promising tools have been successfully applied to the topology optimization of trusses.

Structural optimization is an iterative process and can require advanced analysis at every step, making a huge number of computations necessary. Managing the CPU time is a key issue in implementing optimization on computers. Park et al. (2006) presented a method to better distribute genetic algorithms on a PC cluster and demonstrated its efficiency on frame and truss optimization problems.

1.4 Conclusions and Study Proposal

This quick review shows that optimization is applicable to structural design. As a proof of feasibility, optimization methods have been applied to virtually all types of structures and successfully solved a variety of engineering problems. Structural optimization techniques are constantly being improved at both the mathematical and the implementation levels.

In spite of its potential, optimization is still not widely used in the practice of structural engineering. This lack of application can be partly explained by the fact that typical optimization objectives are not fully relevant in the structural engineering industry. For example, construction speed is more critical than materials cost in most building projects, making weight minimization methods less interesting. Another reason why optimization is rarely used by structural engineering firms is that its implementation on a particular project can be problematic. The most efficient optimization methods require a great deal of adjustment to each particular problem, making their use tedious for structural engineers.

Using more robust algorithms might be a better approach to structural optimization. Compared to evolutionary methods, gradient-based algorithms require very few adjustments to be fully operational on a given optimization problem. Such algorithms are usually slower, but convergence speed is not necessarily a critical issue if the optimization method is easy to implement.

The following study evaluates the applicability of a gradient-based algorithm to different types of structural engineering problems. Since they are relatively simple to analyze, trusses have often been used as examples in the development of structural optimization methods. The geometrical optimization of trusses is proposed as a first implementation of the gradient-based algorithm in chapter 2. The algorithm is then applied at a lower level to optimize single structural members with the design of several beam shapes in chapter 3. To finish, an actual design problem is considered. A hybrid structure combining beam and truss properties is optimized in chapter 4.

Chapter 2

Truss Geometry Optimization

Trusses are popular examples in structural optimization. Optimization methods are implemented on structures by running algorithms in interaction with structural analysis tools. By working on trusses when developing an optimization technique, one can focus on the algorithmic aspects since this analysis is relatively straightforward and robust for this type of structure. Many truss optimization examples are available in the literature, and two categories of problem constitute the majority of the published works. Theoretical studies have been carried out on optimizing truss topologies, while other works have focused on the much more practical problem of optimal member sizing.

The geometry of a truss is characterized by the locations of its nodes. It can be seen as an intermediate level between the topology and the constitutive members. In a traditional truss design process, the geometry would be adjusted after defining the topology and before sizing the members. Geometry optimization has not been as popular as topology definition or member sizing, and fewer examples are available. The growing popularity of form-finding architecture may increase the interest for geometry optimization. Therefore, truss geometry optimization was selected as a first application example for the gradient-based algorithm considered in this study. A matrix analysis method was used as the engine of a quick truss analysis tool programmed to interact with the optimization algorithm. The resulting optimization program was applied to the geometry optimization of various truss configurations.

2.1 Truss Matrix Analysis

The proposed gradient-based optimization algorithm needs to interact with a structural analysis tool in order to optimize trusses. Since algorithmic optimization methods are iterative, structural analysis is repeated many times throughout the optimization process. These successive analyses represent most of the overall time needed to reach the optimal solution. For the optimization process to be reasonably fast, it is important to couple the algorithm to a structural analysis tool that is efficient at solving the type of structure considered. Trusses are quickly analyzed using stiffness matrices, provided that a simple linear behavior can be assumed. This section presents the main steps for solving trusses by matrix analysis and proposes a convenient way of using this method in the context of optimization.

2.1.1 Linear Truss Model

Each truss member contributes to the rigidity of the full structure. The behavior of an individual truss member is governed by several force-displacement relationships summarized in a matricial form called *stiffness matrix.*

Models of different complexities can be used to derive the stiffness matrix of a truss member. In this study, a simple linear model is considered. Nonlinear terms affect the way trusses deform under loading, but they do not change the solutions to the geometry optimization problems considered here. Typically, a truss being optimized for stiffness converges towards the same optimal geometry with linear and nonlinear models, even though the magnitudes of the deflections differ slightly. Nonlinear effects are therefore not considered, allowing for faster structural analysis and a reduced optimization time.

2.1.2 Three-Dimensional Truss Member Stiffness Matrix

Considering a linear model, the force-displacement relationship for a three-dimensional truss member is expressed as a 6-by-6 stiffness matrix. The derivation of the stiffness matrix is detailed in appendix A, and its final expression is given below.

The truss member (figure 2.1) is limited by two nodes (A, B). Each node has 3 degrees of freedom corresponding to the orthogonal directions $[x,\ y,\ z]$. The nodal displacements in these directions are noted as $[u,\ v,\ w]$ respectively. The truss member is subject to externally-applied loads and to the actions of the other members connected to its nodes. The nodal forces are noted as $[F_x,\ F_y,\ F_z]$. The stiffness matrix relates the nodal displacements to the nodal forces and depends on the geometry and properties of the truss member.

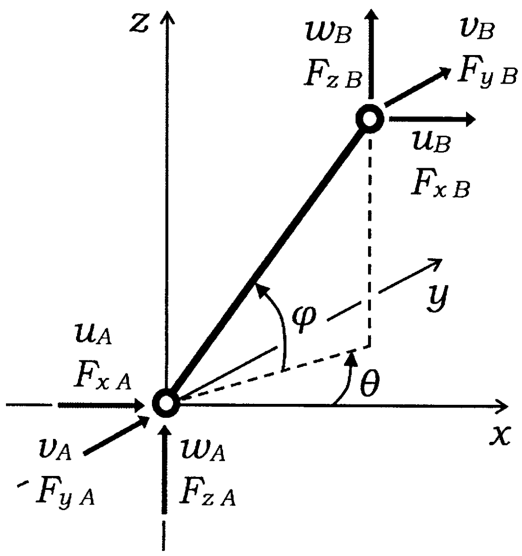


Figure 2.1: Truss Member

For a truss member of length L, cross-sectional area A, modulus of elasticity E and whose orientation is described by the angles θ and φ as shown on figure 2.1, the force-displacement relationship is expressed as:

$$\underline{F} = \underline{K}\,\underline{U} \qquad \text{with the stiffness matrix} \qquad \underline{K} = \frac{EA}{L}\,\underline{G}\,\underline{G}^{\mathrm{T}}$$

$$\text{where} \quad \underline{F} = \begin{pmatrix} F_{\mathrm{xA}} \\ F_{\mathrm{yA}} \\ F_{\mathrm{zA}} \\ F_{\mathrm{xB}} \\ F_{\mathrm{yB}} \\ F_{\mathrm{zB}} \end{pmatrix} \qquad \underline{U} = \begin{pmatrix} u_{\mathrm{A}} \\ v_{\mathrm{A}} \\ w_{\mathrm{A}} \\ u_{\mathrm{B}} \\ v_{\mathrm{B}} \\ w_{\mathrm{B}} \end{pmatrix} \qquad \underline{G} = \begin{pmatrix} \cos\theta\cos\varphi \\ \sin\theta\cos\varphi \\ \sin\varphi \\ -\cos\theta\cos\varphi \\ -\sin\theta\cos\varphi \\ -\sin\varphi \end{pmatrix}$$

The stiffness matrix of every constitutive member is calculated as the first step of the matrix analysis process of a truss.

2.1.3 Full Truss Solution

Full Truss Stiffness Matrix

The stiffness matrix of every member of the truss is built as described in section 2.1.2. These matrices are combined to represent the full truss as a large stiffness matrix, using the following term-by-term matrix addition:

$$\overset{\text{Truss Member 1}}{\begin{pmatrix} \underline{F}_{\mathrm{A}} \\ \underline{F}_{\mathrm{B}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(1)}_{\mathrm{AA}} & \underline{K}^{(1)}_{\mathrm{AB}} \\ \underline{K}^{(1)}_{\mathrm{BA}} & \underline{K}^{(1)}_{\mathrm{BB}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{A}} \\ \underline{U}_{\mathrm{B}} \end{pmatrix}} \qquad \overset{\text{Truss Member 2}}{\begin{pmatrix} \underline{F}_{\mathrm{B}} \\ \underline{F}_{\mathrm{C}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(2)}_{\mathrm{BB}} & \underline{K}^{(2)}_{\mathrm{BC}} \\ \underline{K}^{(2)}_{\mathrm{CB}} & \underline{K}^{(2)}_{\mathrm{CC}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{B}} \\ \underline{U}_{\mathrm{C}} \end{pmatrix}}$$

Truss Members 1 and 2

$$\begin{pmatrix} \underline{F}_{\mathrm{A}} \\ \underline{F}_{\mathrm{B}} \\ \underline{F}_{\mathrm{C}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(1)}_{\mathrm{AA}} & \underline{K}^{(1)}_{\mathrm{AB}} & \underline{0} \\ \underline{K}^{(1)}_{\mathrm{BA}} & \underline{K}^{(1)}_{\mathrm{BB}} + \underline{K}^{(2)}_{\mathrm{BB}} & \underline{K}^{(2)}_{\mathrm{BC}} \\ \underline{0} & \underline{K}^{(2)}_{\mathrm{CB}} & \underline{K}^{(2)}_{\mathrm{CC}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{A}} \\ \underline{U}_{\mathrm{B}} \\ \underline{U}_{\mathrm{C}} \end{pmatrix}$$

For a truss with N nodes, the size of the full stiffness matrix is $3N$-by-$3N$. It relates the displacements of the nodes to the applied loads and reaction forces acting on the truss. The stiffness matrix of the full truss cannot be used is this initial form and needs to be rearranged and reduced to solve the truss analysis problem.

Rearrangement and Reduction

Three categories of degree of freedom are distinguished, as introduced by the example shown of figure 2.2.

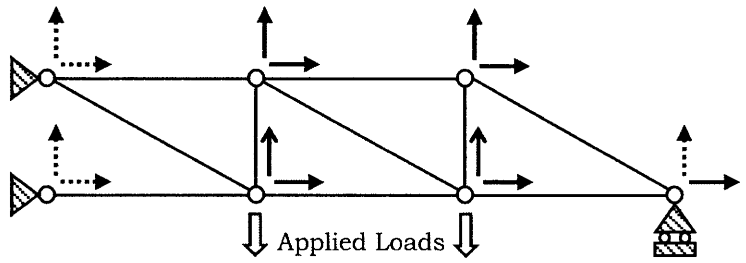


Figure 2.2: Truss DOFs Distinction

Symbol	Generic Name	External Force	Displacement
⟶	Control DOF	Applied	Free
—▶	Unconstrained DOF	None	Free
····▶	Fixed DOF	Reaction	None

The stiffness matrix of the full truss is rearranged by grouping the degrees of freedom into the 3 proposed categories. The non-zero external forces and the free degrees of freedom are combined to form the column vectors $\underline{P}$, $\underline{R}$, $\underline{U}_\mathrm{C}$ and $\underline{U}_\mathrm{U}$ as follows:

Degrees of Freedom	External Forces	Displacements
Control DOFs	$\underline{P}$	$\underline{U}_\mathrm{C}$
Unconstrained DOFs	$\underline{0}$	$\underline{U}_\mathrm{U}$
Fixed DOFs	$\underline{R}$	$\underline{0}$

The external forces $\underline{P}$ are applied at the control degrees of freedom, whose displacements are noted as $\underline{U}_\mathrm{C}$. There is no force (forces = $\underline{0}$) acting on the unconstrained degrees of freedoms, whose displacements are noted as $\underline{U}_\mathrm{U}$. Reactions $\underline{R}$ occur at the fixed degrees of freedom, whose displacements are $\underline{0}$. The force-displacement relationship for the full truss is now written as:

$$\begin{pmatrix} \underline{P} \\ \underline{0} \\ \underline{R} \end{pmatrix} = \begin{pmatrix} \underline{K}_\mathrm{PC} & \underline{K}_\mathrm{PU} & \underline{K}_\mathrm{P0} \\ \underline{K}_\mathrm{0C} & \underline{K}_\mathrm{0U} & \underline{K}_\mathrm{00} \\ \underline{K}_\mathrm{RC} & \underline{K}_\mathrm{RU} & \underline{K}_\mathrm{R0} \end{pmatrix} \begin{pmatrix} \underline{U}_\mathrm{C} \\ \underline{U}_\mathrm{U} \\ \underline{0} \end{pmatrix} \tag{2.1}$$

The externally applied forces $\underline{P}$ are known, and the goal is to solve for $\underline{U}_\mathrm{C}$, $\underline{U}_\mathrm{U}$ and $\underline{R}$. By using the lines of (2.1) as 3 equations, the displacements $\underline{U}_\mathrm{C}$ and $\underline{U}_\mathrm{U}$ and the reactions $\underline{R}$ can be expressed as functions of the applied loads $\underline{P}$:

$$\underline{U}_\mathrm{C} = \underline{F}_\mathrm{C}\,\underline{P} \qquad \underline{U}_\mathrm{U} = \underline{F}_\mathrm{U}\,\underline{P} \qquad \underline{R} = \underline{E}\,\underline{P} \tag{2.2}$$

$$\begin{aligned} \text{Control DOFs Flexibility Matrix: } \underline{F}_\mathrm{C} &= (\underline{K}_\mathrm{PC} - \underline{K}_\mathrm{PU}\,\underline{K}_\mathrm{0U}^{-1}\,\underline{K}_\mathrm{0C})^{-1} \\ \text{Unconstrained DOFs Flexibility Matrix: } \underline{F}_\mathrm{U} &= -\underline{K}_\mathrm{0U}^{-1}\,\underline{K}_\mathrm{0C}\,\underline{F}_\mathrm{C} \\ \text{Force Equilibrium Matrix: } \underline{E} &= \underline{K}_\mathrm{RC}\,\underline{F}_\mathrm{C} + \underline{K}_\mathrm{RU}\,\underline{F}_\mathrm{U} \end{aligned}$$

In the following, the three matrices defined in (2.2) are referred to as *intermediate matrices*, as they are just tools to solve (2.1) but have little physical meaning. The stiffness matrix of a full truss can be very large, so the calculation of $\underline{F}_\mathrm{C}$, $\underline{F}_\mathrm{U}$ and $\underline{E}$ require many operations. When implementing optimization on computers, not all three of these matrices are always needed. For example, if a cantilever truss-beam with a point load applied at the end is being optimized to limit the deflection of the free extremity, only $\underline{F}_\mathrm{C}$ is needed since one only wants to calculate the displacement of a control degree of freedom.

2.2 Truss Optimization Program

Truss geometry optimization was implemented in Matlab®. A built-in optimization toolbox was used to run the gradient-based algorithm, and a program was developed to define and analyze trusses in interaction with the optimization tool. Trusses were analyzed using the stiffness matrix method presented in section 2.1. The strategy for implementing structural optimization on computers was introduced in section 1.2.1 and more details on the Matlab® optimization tool can be found in the *Optimization Toolbox™ 3 User's Guide* (The MathWorks™, 2007b).

2.2.1 Schematic Diagram

Figure 2.3 shows the group of Matlab® functions used to optimize trusses and the way they interact. Each rectangle represents a function, that is, a piece of code contained in a separate file. The arrows represent arguments being passed between functions. The layout is top-down, meaning that a function called during the execution of another function is represented below the latter function. More clarifications about the schematic diagram can be found in appendix C. The functions `objective`, `constraints` and `output` interact directly with the optimization toolbox, while all other functions form the truss analysis program.

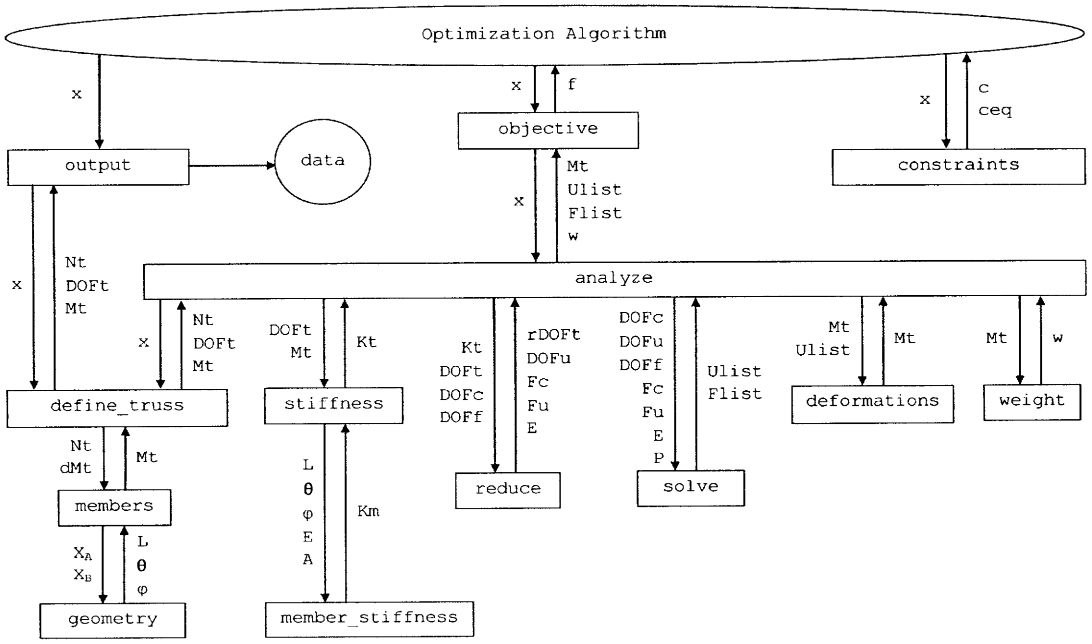


Figure 2.3: Truss Optimization Program Schematic Diagram

2.2.2 Functions Description

The operations carried out by the functions represented on the schematic diagram (figure 2.3 p.27) are described in this section. The code is available in appendix D.

Optimization Objective Function - `objective`

The objective function (`objective`) is required to use the Matlab® optimization toolbox (see figure 1.2 p.13). It is called by the optimization algorithm whenever it needs to evaluate the quality of a design scenario. The algorithm sends the current values of the set of optimization variables (`x`) to the objective function. The objective function transmits these variables to the truss analysis function (`analyze`) which returns the results of the analysis: list of truss nodal displacements (`Ulist`) and reaction forces (`Flist`), total weight of the truss (`w`) and various data about the truss members (`Mt`). Using these data, the objective function evaluates the value (`f`) of the current truss, and the way this is done depends on the goal of the optimization process. For example, if a cantilever truss-beam is being optimized for stiffness, the objective value is the deflection at the free extremity. This value is returned to the optimization algorithm, whose goal is to minimize (`f`) by adjusting the values in (`x`).

Optimization Constraints Function - `constraints`

The constraints function (`constraints`) is required to use the Matlab® optimization toolbox (see figure 1.2 p.13). It is called by the optimization algorithm whenever it needs to check whether a design scenario is acceptable or not. It is used to define the boundaries of the optimization variables and the criteria that the optimal truss must meet. The function receives the values of the optimization variables (`x`) from the optimization algorithm and returns two series of numbers (`c, ceq`) calculated from the optimization variables. The design scenario is acceptable if all numbers in the first list (`c`) are negative and all numbers in the second list (`ceq`) are zero. For example, if the truss has to meet a deflection criterion, the constraints function (`constraints`) calls the analysis function (`analyze`), finds the maximum deflection from the deflection list (`Ulist`) and returns a positive constraint value (`c`) if the deflection criterion is not met.

Main Truss Analysis Function - `analyze`

This function does not directly carry out analysis operations but manages the successive steps of the truss analysis process by calling other functions. The analysis is triggered by the objective function, which transmits the values of the optimization variables (`x`). The functions corresponding to the 5 steps of the analysis process are called successively:

1. TRUSS DEFINITION. The values of the optimization variables (`x`) are passed to the truss definition function (`define_truss`), which returns information about the truss geometry and mechanical properties: list of truss nodes (`Nt`) and associated degrees of freedom (`DOFt`) and list of truss members (`Mt`) containing information such as their length, orientation, stiffness ...

2. STIFFNESS MATRIX CALCULATION. The truss members information (`Mt`) and the list of degrees of freedom (`DOFt`) are passed to the stiffness matrix calculation function (`stiffness`), which returns the stiffness matrix of the full truss (`Kt`).

3. STIFFNESS REDUCTION. The full stiffness matrix (`Kt`) cannot be directly used to solve the problem. Some intermediate matrices, described in section 2.1.3, are needed. The lists of control degrees of freedom (`DOFc`) and fixed degrees of freedom (`DOFf`) are defined in the analysis function and passed to the matrix reduction function (`reduce`) along with the full stiffness matrix (`Kt`) and the full list of degrees of freedom (`DOFt`). The flexibility matrices (`Fc, Fu`), the force equilibrium matrix (`E`), the list of unconstrained degrees of freedom (`DOFu`) and the rearranged full list of degrees of freedom (`rDOFt`) are returned.

4. PROBLEM SOLUTION. A list of loads (`P`), corresponding to the control degrees of freedom (`DOFc`) is defined in the analysis function. It is passed to the problem solution function (`solve`) along with the intermediate matrices (`Fc, Fu, E`) and the lists of degrees of freedom (`DOFc, DOFu, DOFf`). The lists of the truss nodal displacements (`Ulist`) and support reactions (`Flist`) are returned.

5. DEFORMATION ANALYSIS. Truss members information (`Mt`) and nodal displacements (`Ulist`) are passed to the deformation analysis function (`deformations`). The truss members information (`Mt`) is returned, containing additional data about the truss members in the deformed configuration (displacements, strains ...)

Apart from the main analysis process, the weight calculation function (`weight`) is called. The weight (`w`) is calculated from the truss members information (`Mt`).

Truss Definition Function - `define_truss`

This function contains two lists of data entered by the user and defining the truss to be optimized.

1. NODES LIST. A list of nodes (`Nt`) defines the coordinates and degrees of freedom of the truss nodes (in trusses, the degrees of freedom are all nodal displacements).

2. MEMBERS LIST. A member definition list (`dMt`) defines the truss members, assigning to each member two nodes and some mechanical properties (section area, material elasticity). This list is passed to a processing function (`members`) which returns a modified version of the list (`Mt`), containing additional information to be used in the following steps of the analysis process.

The truss definition function accepts the design variables (`x`) as input, and these variables can be used in both lists defining the truss. For example, if the goal is to optimize the truss geometry, the coordinates of some nodes can be defined by optimization variables. If the material distribution is being optimized, then the cross-sectional area of the truss members can be defined by optimization variables.

Truss Members Processing Function - `members`

In the truss members definition list (`dMt`), the geometry of each truss member is simply defined by its two nodes. This list (`dMt`) is passed with the nodes list (`Nt`) to the truss members processing function (`members`). This function retrieves the nodes coordinates from the nodes list (`Nt`) to calculate the length (`L`) and orientation (θ, ϕ) of each truss member by calling a geometry function (`geometry`). This information is returned to the truss definition function (`define_truss`) as a more detailed list of truss member properties (`Mt`).

Truss Member Geometry Function - `geometry`

This function accepts the coordinates of both nodes of a truss member as arguments (`xa, ya, za, xb, yb, zb`) and returns the length (`L`) and orientation (θ, ϕ) of this member.

Truss Stiffness Matrix Calculation Function - `stiffness`

This function receives the full list of the truss degrees of freedom (`DOFt`) and the truss members list (`Mt`) as inputs. For each truss member in the list (`Mt`), the function sends the member length (`L`), its orientation (θ, ϕ), its cross-sectional area (`A`) and elasticity (`E`) to the member stiffness matrix calculation function (`member_stiffness`), which returns the member stiffness matrix (`Km`). The terms of the member stiffness matrix (`Km`) are distributed in the truss stiffness matrix (`Kt`) as described in section 2.1.3. The order of the terms is defined by the list of the truss degrees of freedom (`DOFt`). When the contribution of every truss member has been taken into account, the truss stiffness matrix (`Kt`) is returned.

Member Stiffness Matrix Calculation Function - `member_stiffness`

This function accepts the length (`L`), the orientation (θ, ϕ), the cross-sectional area (`A`) and the elasticity (`E`) of a member as inputs. The stiffness matrix of this member (`Km`) in the global coordinates is calculated as in section 2.1.2 and returned.

Stiffness Reduction Function - `reduce`

This function transforms the truss stiffness matrix (`Kt`) into the 3 intermediate matrices described in section 2.1.3 and needed to solve the problem. In addition to the truss stiffness matrix (`Km`), the function accepts the full list of degrees of freedom (`DOFt`), the list of the fixed degrees of freedom (`DOFf`) and the list of the control degrees of freedom (`DOFc`) as inputs. First, a list of unconstrained degrees of freedom (`DOFu`) is generated (degrees of freedom that are neither fixed nor controlling). Then the intermediate flexibility matrices (`Fc, Fu`) and the force equilibrium matrix (`E`) are calculated and returned, along with the list of unconstrained degrees of freedom (`DOFu`) and a rearranged full list of degrees of freedom (`rDOFt`).

Problem Solution Function - `solve`

This function multiplies the intermediate matrices (`Fu, Fc, E`) by the load vector (P) to calculate the nodal displacements and reaction forces. The lists of nodal displacements (`Ulist`) and reaction forces (`Flist`) are returned.

Deformations Analysis Function - `deformations`

This function uses the nodal displacements (`Ulist`) and the truss members list (`Mt`) to calculate quantities due to the truss deformation (strains, stresses, forces ...). This information is added to the truss members list (`Mt`), which is returned.

Weight Calculation Function - `weight`

The truss members list (`Mt`) is sent to the weight calculation function (`weight`), which returns the total weight of the truss (`w`).

Optimization Output Function - `output`

Though not required to use the MatLab® optimization toolbox, the output function (`output`) is directly called by the optimization algorithm. At the end of each iteration, the current values of the optimization variables (`x`) are sent to the output function (`output`), which can store the data and generate plots to represent the evolution of the optimization process.

2.3 Truss Geometry Optimization Examples

The computer program described in section 2.2 was used to find optimal geometries for various truss configurations. The results are presented in this section.

2.3.1 General Considerations

In the examples presented in this section, only the geometry of the truss is optimized, while the truss topology and members properties remain constant.

Constant Truss Topology

The topology of a truss is defined by the arrangement of its constituting members. Since the topology is constant, no member is added nor removed during the optimization process, and all member-to-member connections remain fastened.

Constant Member Properties

In truss analysis, the key properties of each member are its cross-sectional area and modulus of elasticity, both used to calculate the member stiffness. If buckling is considered, the cross-sectional moment of inertia becomes another important property for compression members. All member properties are considered constant in the geometry optimization problems presented in this section.

Variable Truss Geometry

The geometry of a truss is defined by the location of its nodes. As nodes are moved during the optimization process, member lengths and orientations are modified, changing the overall shape of the truss.

Since only the geometry was optimized, unit cross-sectional areas were assigned to all truss members. The magnitudes of the loads were adjusted so that all deformations remain elastic. The optimization variables used to parameterize the geometry of the trusses were also unitless. The resulting structures have therefore no technical meaning, but optimal geometries can still be obtained. In chapter 4, an actual structure with truss properties is optimized and constitute a more practical example, supported by numerical values.

2.3.2 Planar Cantilever Truss Beam

A cantilever truss beam made of 10 segments is defined and parametrized as shown on figure 2.4. A truss segment is a group of members arranged in a pattern that is repeated in the direction of the beam. In this truss, each segment is made of 4 members: top, bottom, vertical and diagonal. The optimization variables are the lengths of the vertical members, and the objective is to minimize the end deflection.

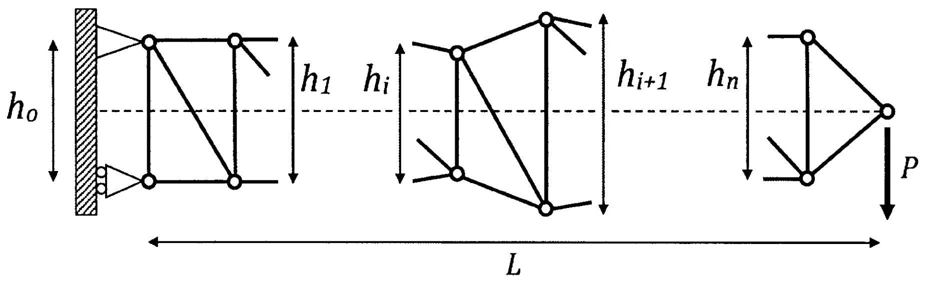


Figure 2.4: Cantilever Truss Beam Parametrizetion

The following unitless values and constraints are imposed:

$$h_0 = 10 \qquad L = 105 \qquad\qquad 1 \leq h_i \leq 50 \quad \text{for} \quad i = 1 \ldots 10$$

A lower limit of 1 is used to prevent the truss from becoming unstable. The upper limit of 50 defines a reasonable search space for the optimization algorithm. No optimization variable reaches either limiting value, meaning that the search for the optimal geometry is not restricted by the constraints. The shape evolution leading to the optimal geometry (figure 2.5) is shown on the next page.

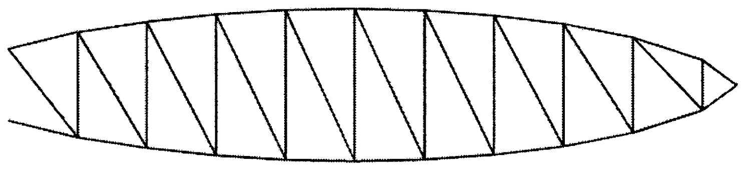
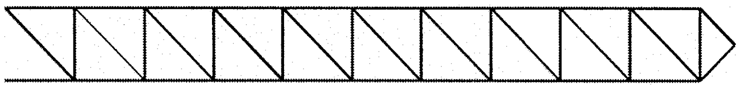


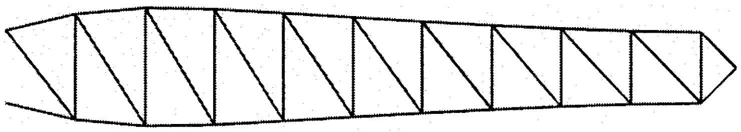
Figure 2.5: Cantilever Truss Beam Optimal Geometry

Considering the bending moment diagram, a triangular shape tapering towards the free extremity could have been expected. The oval shape obtained instead is due to the fixity. For the arbitrary combination of member sections and load magnitude considered in this example, the fixity is too short to allow for an optimized triangular shape. This phenomenon is shown on the three-dimensional cantilever truss-beam treated in section 2.3.5.

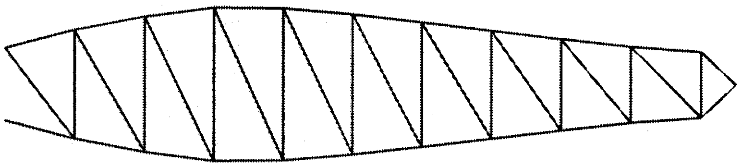
ITERATION 0. All optimization variables have an initial value of 10, so that each truss segment is square. End deflection: 14.02



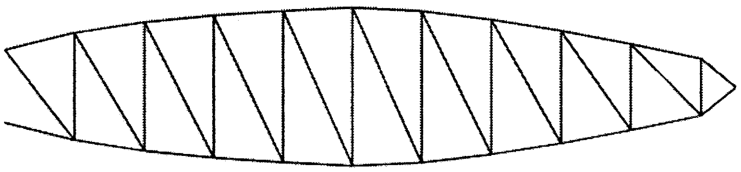
ITERATION 1. The depth of the beam starts growing at the fixity, creating a curved shape. End deflection: 11.57



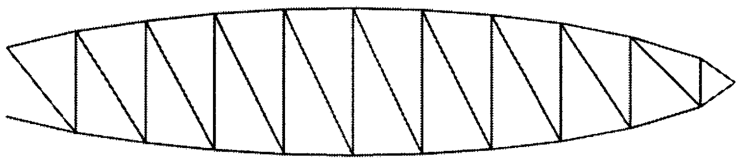
ITERATION 4. The increase in depth propagates towards the free extremity of the beam. End deflection: 9.38



ITERATION 8. Propagation stops approximately at mid-span. End deflection: 8.61



ITERATION 12. The shape is smoothed as the final adjustments of the optimization variables are made. End deflection: 8.46



2.3.3 Multi-Span Truss Bridge

Two similar truss bridges are optimized. Both bridges are made of 30 truss segments forming three spans and parametrized by the lengths of the vertical members. In the first bridge (figure 2.6), each span is an independent simply-supported truss beam. The second bridge (figure 2.7) is made of a single continuous truss beam resting on 4 supports. A uniform distributed load is applied at the bottom chord of both bridges. The objective is to minimize the aggregate deflection, defined as the sum of the maximum deflections of each span.

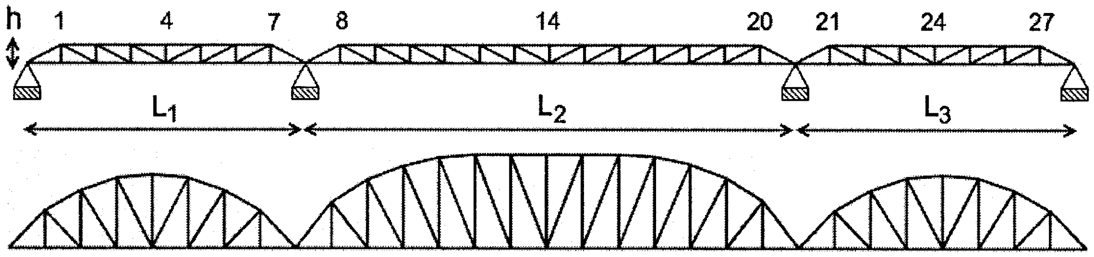


Figure 2.6: Truss Bridge 1 Parametrization and Optimal Geometry

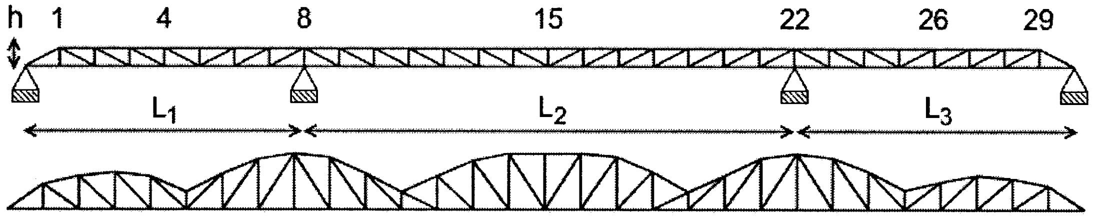


Figure 2.7: Truss Bridge 2 Parametrization and Optimal Geometry

The following unitless values and constraints are imposed to both bridges:

$$L_1 = 80 \qquad L_2 = 140 \qquad L_3 = 80 \qquad\qquad 5 \leq h_i \leq 25$$

In both cases, the optimal shape corresponds to the magnitude of the bending moment acting on the bridge. The optimal shape of the first bridge features a flat top chord at the middle of the longer span. At this location, where the bending moment is maximum, the optimization variables have reached the upper limit imposed by the constraints ($h = 25$), affecting the optimal shape expected. The shape evolution of both bridges is shown on the next page.

Shape Evolution of Truss Bridge 1

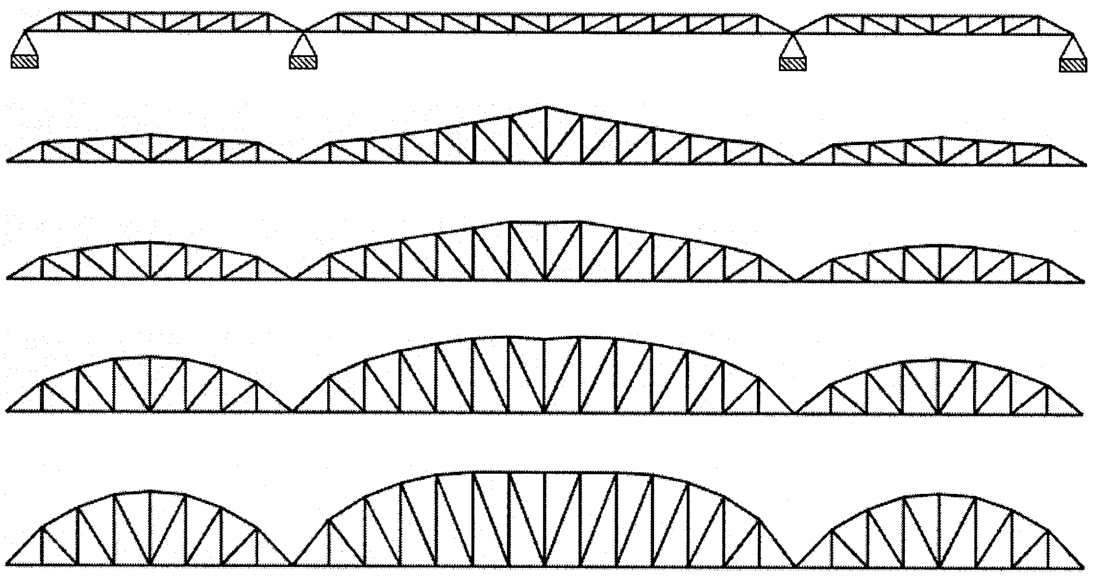


Figure 2.8: Truss Bridge 1 Shape Evolution

Shape Evolution of Truss Bridge 2

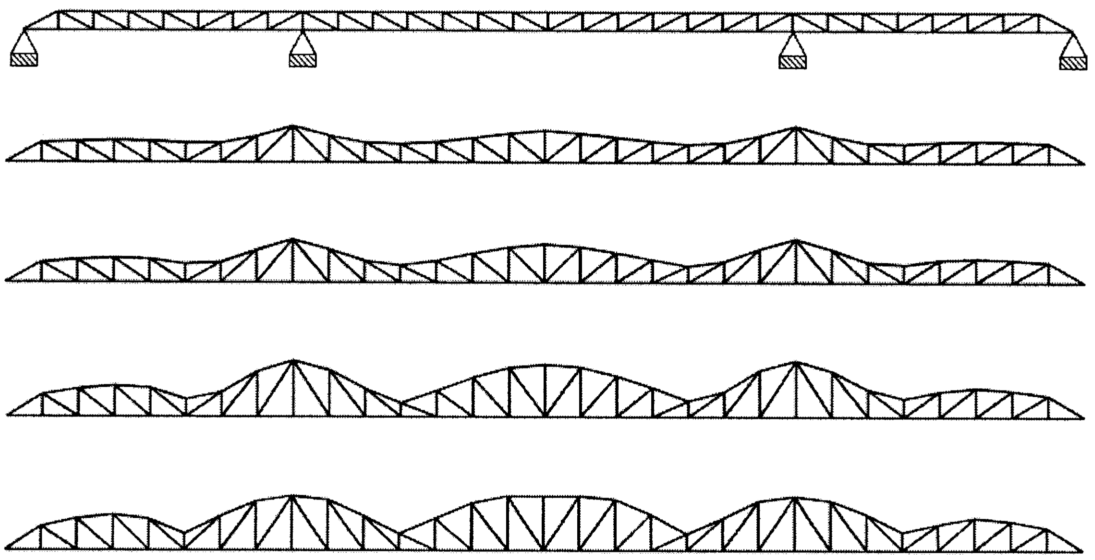


Figure 2.9: Truss Bridge 2 Shape Evolution

2.3.4 Semi-Circular Truss Arch

A semi-circular truss arch is modeled as a series of 20 truss segments, as shown on figure 2.10. The arch is pinned at both fixities and a point load is applied at mid-span.

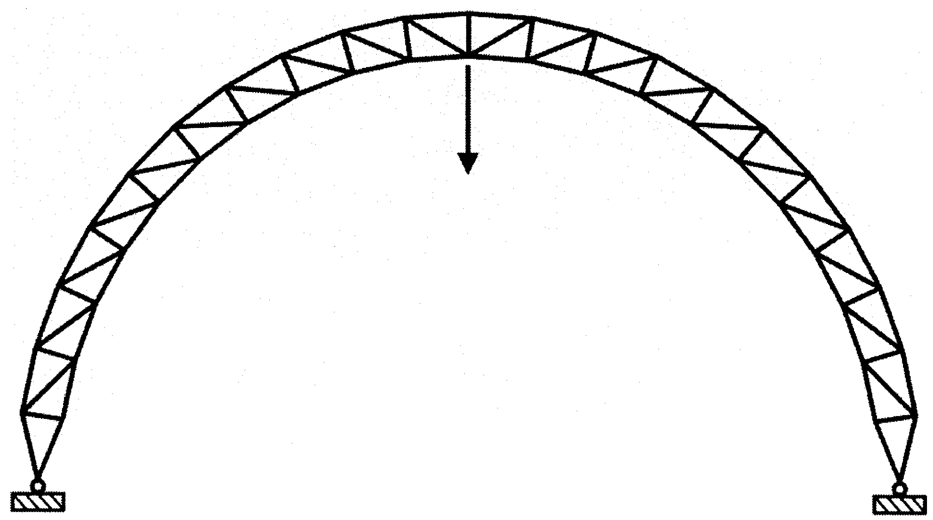


Figure 2.10: Truss Arch Topology

In this example of truss geometry optimization, the overall shape of the structure is fixed. The objective is to minimize the deflection at mid-span while keeping the semi-circular appearance of the arch. It is known that semi-circular arches are not the most efficient ones, so the design can only be partially optimized. This is a very common situation in structural engineering, as architecture does not necessarily compromise with structural efficiency.

The arch is set to remain semi-circular throughout the optimization process by appropriately parametrizing the problem (see figure 2.11 p. 39). What is being optimized is the distribution of the truss members within a semi-circular envelope. Since the problem is symmetric, optimization is carried out on one half of the truss arch only.

The half-arch considered for optimization is made of 10 truss segments, each segment being limited by two radial truss members (i.e. members oriented towards the center of the circular arch). The location of each radial member within the semi-circular envelope is parametrized by an angle θ, as shown on figure 2.11. The angles are selected as optimization variables, so that the radial members of the truss can be displaced by the optimization algorithm.

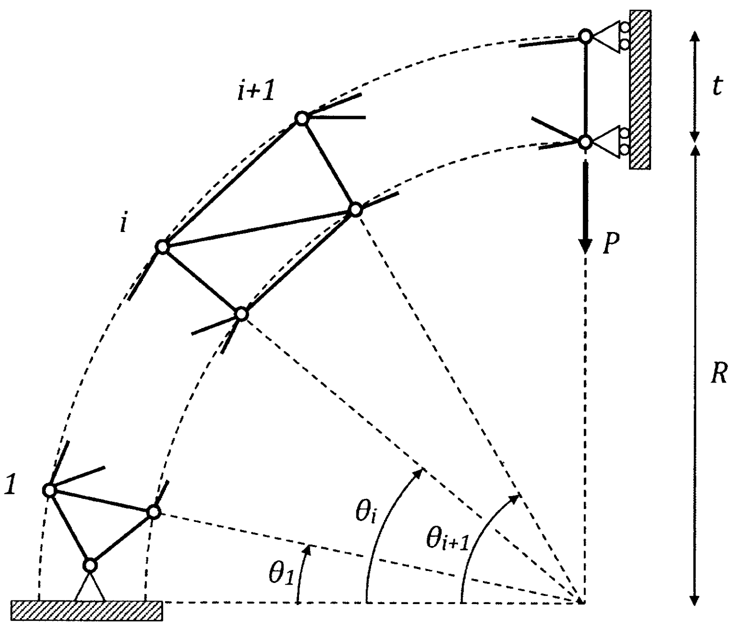


Figure 2.11: Truss Arch Parametrization

The following unitless values are fixed:

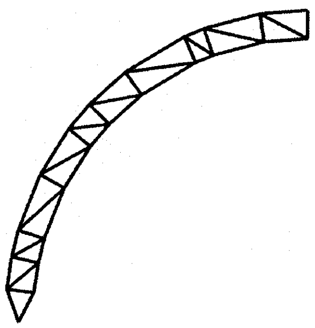
$$R = 100 \qquad t = 10$$

To prevent the truss segments from overlapping or becoming infinitely small, the following constraints are applied to the optimization variables:

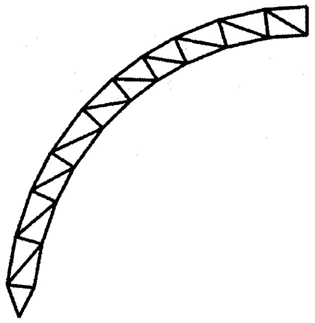
$$\begin{array}{rcccl} 1^\circ & \leq & \theta_1 & \leq & 80^\circ \\ \theta_{i-1} + 1^\circ & \leq & \theta_i & \leq & (79+i)^\circ \qquad \text{for} \qquad i = 2 \ldots 10 \end{array}$$

The optimization variables are therefore not fully independent in this problem. Starting with a random distribution of the radial members, the evolution of the truss arch is shown on the next page.

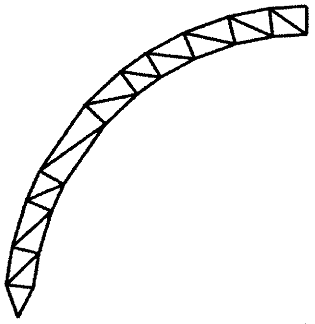
ITERATION 0
A list of angles satisfying the constraints is randomly generated and used as the initial design for the optimization process. Top deflection: 12.25



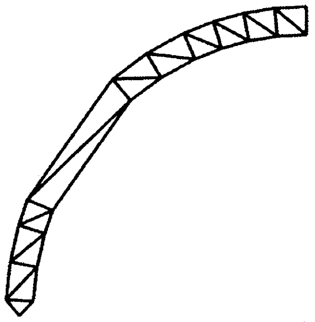
ITERATION 6
As the optimization algorithm starts running, the truss segments tend to even out. The radial truss members are almost evenly distributed over the quarter circle. Top deflection: 9.13



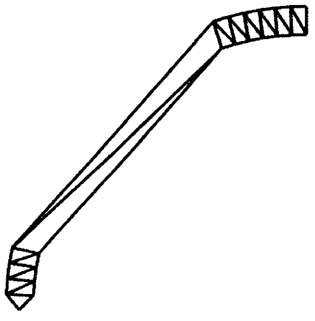
ITERATION 15.
The even design is stable during a few iterations, before a longer truss segment appears. Top deflection: 8.82



ITERATION 18.
The size of the longer segment keeps increasing, as the other segments shrink. Top deflection: 7.05



ITERATION 25.
The shrinking segments eventually collapse, leading to a final design equivalent to a single compression member. Top deflection: 2.87



The design with truss segments of even sizes was a local optimum. After a few iterations around that design, the algorithm was able to find a shape which, considering the constraints, was better. However, the final design obtained is not be acceptable since the goal was to optimize a semi-circular arch. More constraints (e.g. buckling) would be needed to make the design converge towards an more acceptable shape.

2.3.5 Triangular Cantilever Truss Beam

A triangular truss beam is modeled as a series of 9 truss segments, whose arrangement is shown on figure 2.12.

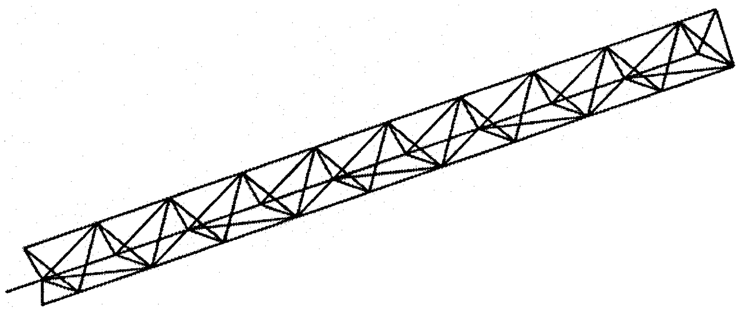


Figure 2.12: Triangular Truss Beam Topology

A cantilever problem is considered. The left extremity of the truss is fixed and two point loads are applied at the other end. The height of each truss segment (shown as h_i on figure 2.13) can be varied as an optimization variable. The objective is to minimize the deflection of the free extremity.

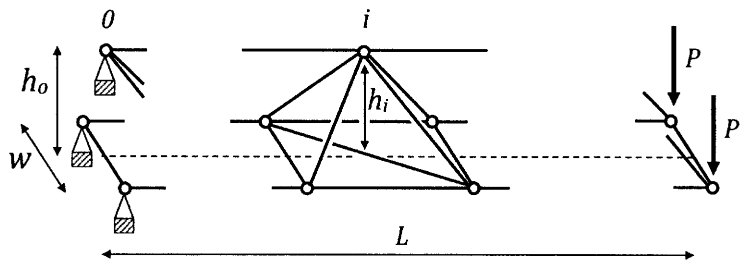


Figure 2.13: Triangular Truss Beam Parametrization

The following unitless values and constraints are imposed:

$$w = 10 \qquad L = 180 \qquad\qquad 1 \leq h_i \leq 80 \quad \text{for} \quad i = 0 \ldots 9$$

The evolution of the truss geometry leading to the optimal shape is represented on the next page.

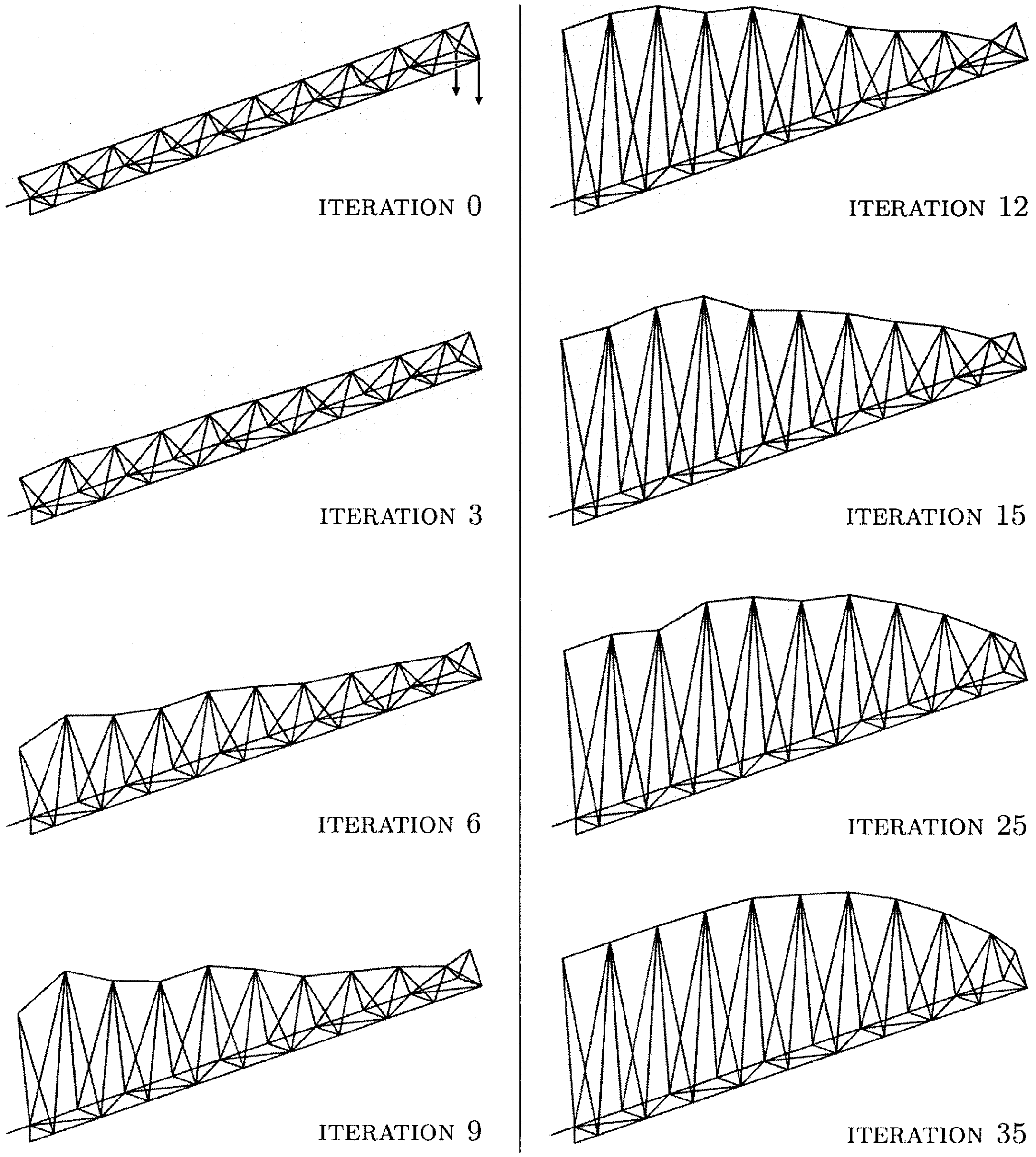


Figure 2.14: Triangular Cantilever Truss Beam Shape Evolution

In the initial design, all truss segments have a height of 10. The depth of the truss beam starts increasing at the fixity, leading to a roughly triangular but still irregular shape. The truss segments close to the fixity reach the limiting depth of 80, causing flattening of the shape. The flat portion propagates towards the free end of the beam, stopping approximately at mid-span. The remaining of the triangular shape is rounded up, leading to the final design. This result is discussed on the next page.

When optimizing a cantilever structure, a triangular shape corresponding to the bending moment is intuitively expected. In the optimal design previously obtained (reproduced on figure 2.15), several optimization variables have reached the upper limit imposed by the constraints (i.e. depth of 80), affecting the overall geometry.

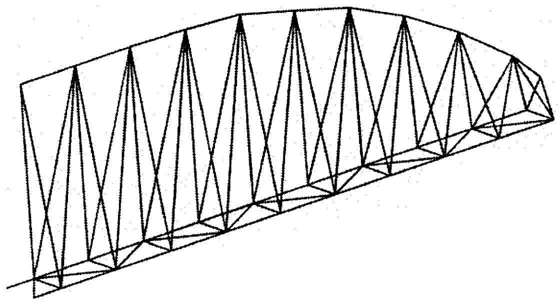


Figure 2.15: Optimal Shape Considering a Depth Constraint

The same beam was optimized with no depth constraint. The resulting optimal design (figure 2.16) is the expected triangular shape. The left part of the truss is deeper than the limit of 80 previously applied.

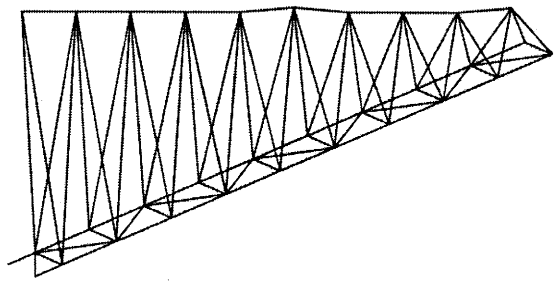


Figure 2.16: Optimal Shape with No Depth Constraint

To finish, the beam was optimized with a single constraint limiting the depth of the first segment to 20. Like in the two-dimensional example presented in section 2.3.2, an oval shape is obtained (figure 2.17).

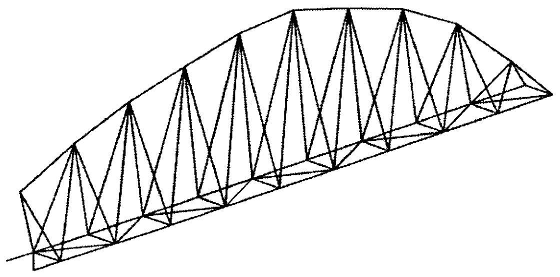


Figure 2.17: Optimal Shape Considering a Depth Constraint at Fixity only

Optimization constraints can have a significant effect on the final geometry and must therefore be carefully applied.

2.3.6 Rectangular Clamped Truss Beam

A rectangular truss beam is modeled as a series of 10 truss segments, whose arrangement is shown on figure 2.18.

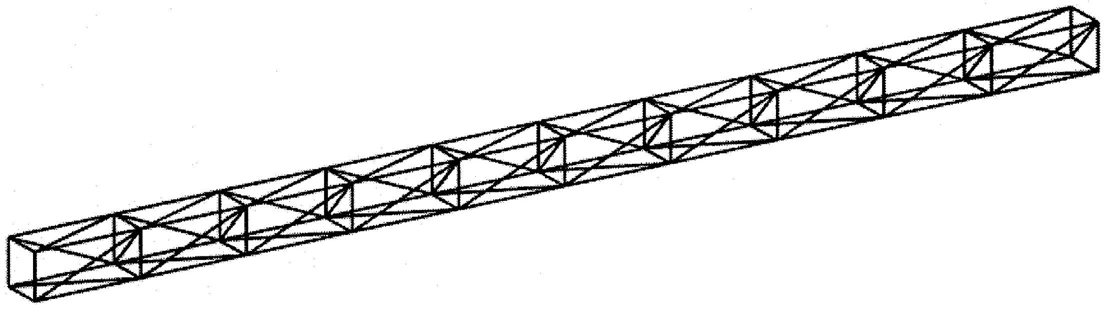


Figure 2.18: Rectangular Truss Beam Topology

A double cantilever problem is considered. Both extremities of the truss are fixed and a same point load is applied to every node on the top of the truss, resulting in a uniform distributed load. The height of each couple of vertical members (shown as h_i on figure 2.19) can be varied as an optimization variable. The objective is to minimize the deflection at mid-span.

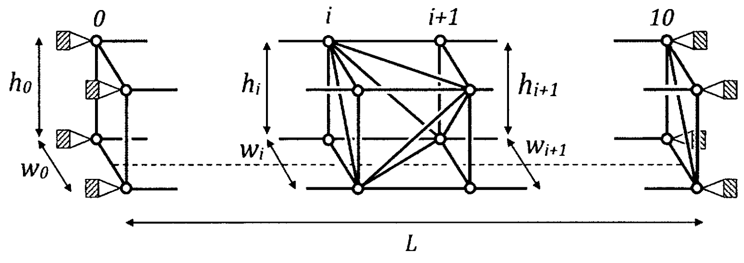


Figure 2.19: Rectangular Truss Beam Parametrization

The following unitless values are fixed:

$$L = 300 \qquad h_0 = h_{10} = 20 \qquad\qquad w_i = 10 \quad \text{for} \quad i = 0 \dots 10$$

The optimization variables are limited by the following constraints:

$$1 \leq h_i \leq 20 \qquad \text{for} \quad i = 1 \dots 9$$

The evolution of the truss geometry leading to the optimal shape is represented on the next page.

The optimization process is quick, meeting the termination criterion after 10 iterations only. The truss takes a shape that resembles the bending moment acting on a clamped beam subject to a distributed load.

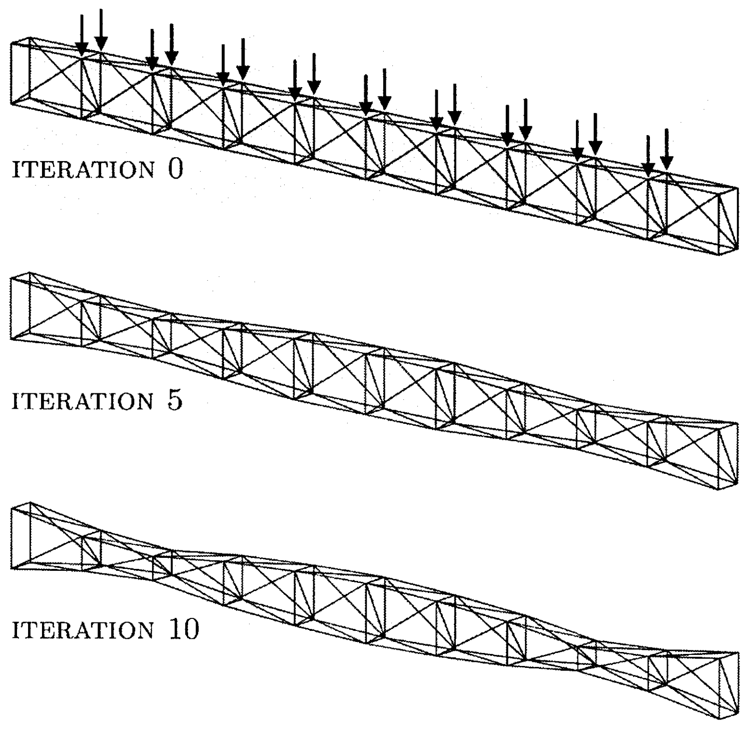


Figure 2.20: Rectangular Clamped Truss Beam Shape Evolution

2.3.7 Square Truss Shaft

The rectangular truss beam used as a double cantilever in section 2.3.6 is now modified to represent a square truss shaft. The loads and fixities are changed and a new parametrization of the geometry is used, as shown on figure 2.21.

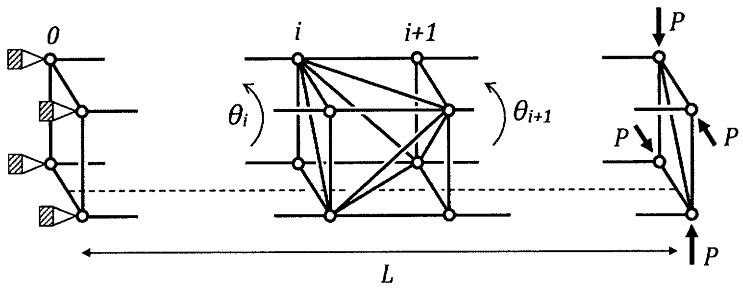


Figure 2.21: Square Truss Shaft Parametrization

The left extremity of the truss is fixed and 4 point loads are applied at the other end in order to create a torque in the axial direction of the shaft. The dimensions of all truss segments are now constant, but these segments are allowed to rotate about the axis of the shaft. The rotation θ of each segment is an optimization variable.

The following unitless values are fixed:

$$L = 300 \qquad \text{Shaft cross-section} = 20\,\text{x}\,20$$

The optimization variables are limited by the following constraints:

$$-180^\circ \leq \theta_i \leq 180^\circ \quad \text{for} \quad i = 1 \ldots 10$$

The evolution of the truss geometry leading to the optimal shape is represented on the next page.

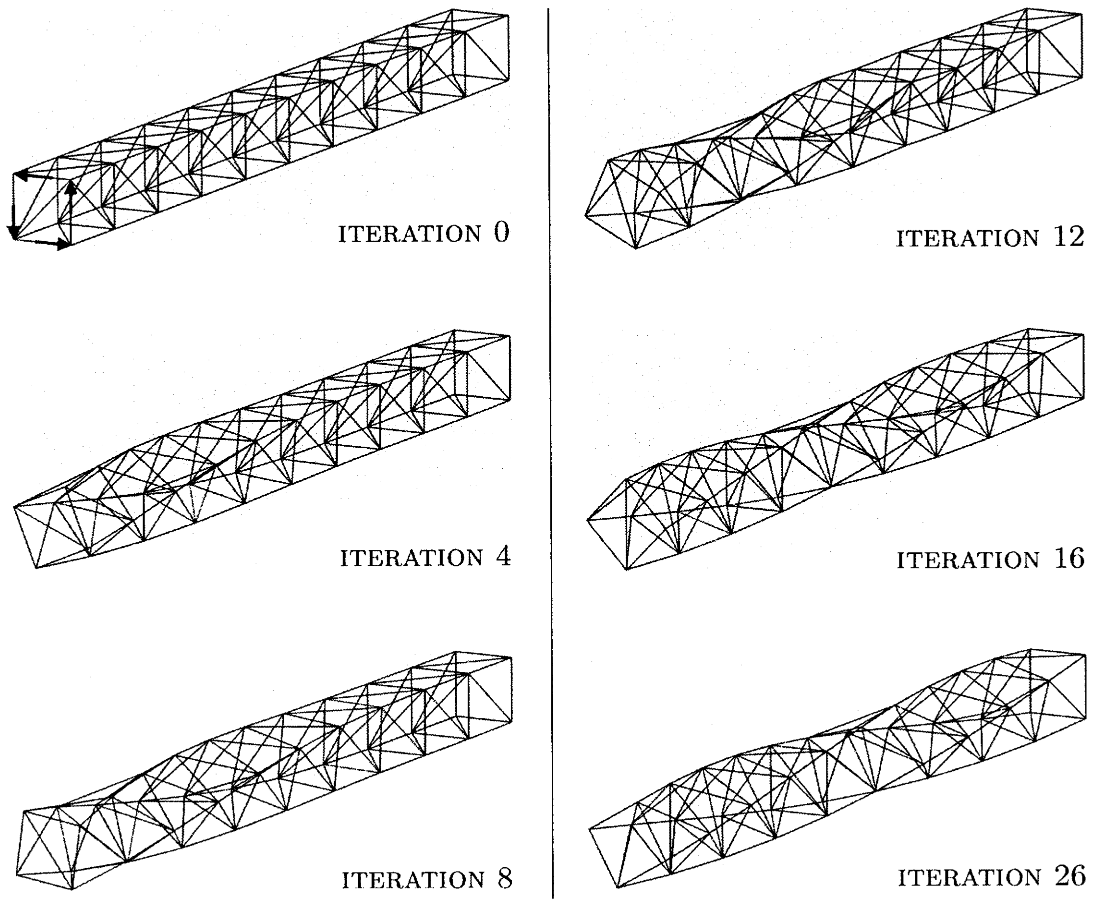


Figure 2.22: Square Truss Shaft Shape Evolution

The shaft takes a twisted shape opposite to the orientation of the applied moment. Deformation starts at the free extremity, where the torque is applied, and then propagates towards the fixity. The optimal design (figures 2.23 and 2.24) features a constant rotation rate.

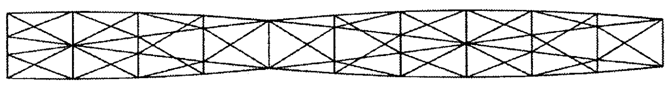


Figure 2.23: Optimal Shape - Side View

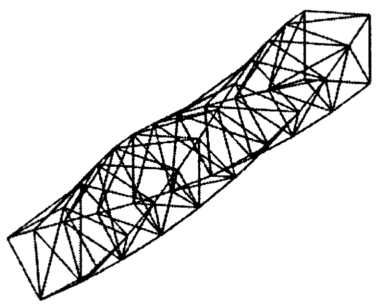


Figure 2.24: Optimal Shape - Perspective View

Chapter 3

Beam Shape Optimization

Sizing beams and columns is a necessary step in every building design project. Automating and optimizing this process is consequently of great interest to the structural engineering practice. Many methods have been proposed to optimize the performances of building frames by appropriately sizing the constitutive members. At a lower level, the design of individual structural members can also be optimized.

Beam shape optimization is to design the cross-section and distribute the material over the length of the member to improve its performance and/or reducing its cost. The performance of a beam is typically characterized by a stiffness, buckling or vibrational property, while the cost depends on the amount of material and on various fabricability aspects. Beam shape optimization has not been as widely developed as optimizing beam sizes in building frames, since non-standard beam shapes are often considered of little practical use. However, optimized shapes obtained through form-finding processes are becoming increasingly popular. Moreover some new, sustainable definitions of cost, including the concepts of *life cycle cost* and *embodied energy*, give more importance to the amount of material used in structures.

The gradient-based algorithm previously used to optimize trusses was therefore applied to beam shape optimization as well. Two beam analysis methods were considered to interact with the algorithm, and optimal shapes were obtained for various beam configurations.

3.1 Analytical Beam Optimization

The structural optimization method discussed throughout in this report involves a gradient-based algorithm in interaction with a structural analysis tool. A first application to truss geometry optimization was presented in chapter 2. The methods of this type, with an optimization routine generating successive trial designs in an effort to find the optimal one, are called *algorithmic* optimization methods. They can be opposed to the *analytical* optimization methods, whose principle is to explicitly derive optimal solutions by analyzing the structure. In practice, analytical solutions are derived by hand, whereas algorithmic solutions are obtained through computer implementation.

Algorithmic methods are necessary to optimize large-scale structures requiring computer analysis. But single structural components, such as beams or truss members, are simple enough for explicit solutions to be found. Analytical optimization can therefore be used to improve the performance of individual structural components. This section presents a strategy for analytically optimizing beams for stiffness under a weight constraint.

3.1.1 Design Problem

Lightness and stiffness are conflicting objectives in beam design. For a given beam shape, stiffer members are also heavier. (e.g. the only way to stiffen a wide flange beam of given depth is to increase the flanges thickness, resulting in a heavier member). The stiffness-to-weight ratio depends on the beam geometry. For a given beam depth, wide flange beams have higher stiffness-to-weight ratios than any other flexural member of constant cross-section. Still, this ratio can be increased by varying the cross-sectional properties. The goal of beam shape optimization is to design non-standard members with improved balance between lightness and stiffness. The problem can be approached in two different ways:

1. Finding the lightest beam meeting a given deflection criterion

2. Finding the stiffest beam of a given weight

The solutions to these problems are equivalent, as optimal beam shapes can be scaled up and down to achieve a particular weight or deflection criterion.

3.1.2 Constant Curvature Criterion

It is often said about bending members that an efficient design should resemble the bending moment diagram. Constant curvature is sometimes considered as an optimality criterion for beam design. For beam of material elasticity E and cross-sectional moment of inertia I subject to a bending moment M, the curvature is expressed as:

$$c = \frac{d\theta}{dx} = \frac{M(x)}{E\,I(x)} \tag{3.1}$$

For the curvature to be constant, the moment of inertia of the beam must be proportional to the bending moment acting on it:

$$I(x) = \alpha|M(x)| \qquad \text{where}\, \alpha = \text{constant} \tag{3.2}$$

Equation (3.2) can be used as a basis to develop an optimal beam design. The moment of inertia I needs to be replaced by its expression in terms of the chosen design parameters (e.g. beam depth, flanges thickness ...), and the parameter α is adjusted to meet a given deflection criterion or a given weight limit imposed on the full beam.

The next section presents a modification of (3.2) for beam stiffness optimization under a weight constraint.

3.1.3 Derivation of a Modified Criterion

In this section, an optimality criterion for bending member design is analytically derived. For the purpose of this analysis, the quantity whose optimal distribution is to be determined is the cross-sectional moment of inertia I. The bending rigidity distribution has less physical meaning than the material distribution, but it has the advantage of being general to all types of beam sections. Section 3.1.4 explains how the optimal bending rigidity distribution can be transformed into an optimality criterion for material distribution.

Therefore, even though this has no physical meaning, it is assumed that a total amount of available inertia is fixed. For a beam of length L:

$$I_{\text{tot}} = \int_0^{\text{L}} I(x)dx \tag{3.3}$$

The goal is to find the bending rigidity distribution $I(x)$ that minimizes the deflection at a chosen point.

The model used as illustration is a simple cantilever beam (figure 3.1), and the goal is to find the moment of inertia distribution $I(x)$ that minimizes deflection at the free extremity. This cantilever is a determinate system, for which the bending moment diagram is fully known from the applied loads. The generalization to indeterminate systems, whose bending moment diagram depends on the bending rigidity distribution, is treated is section 3.1.4.

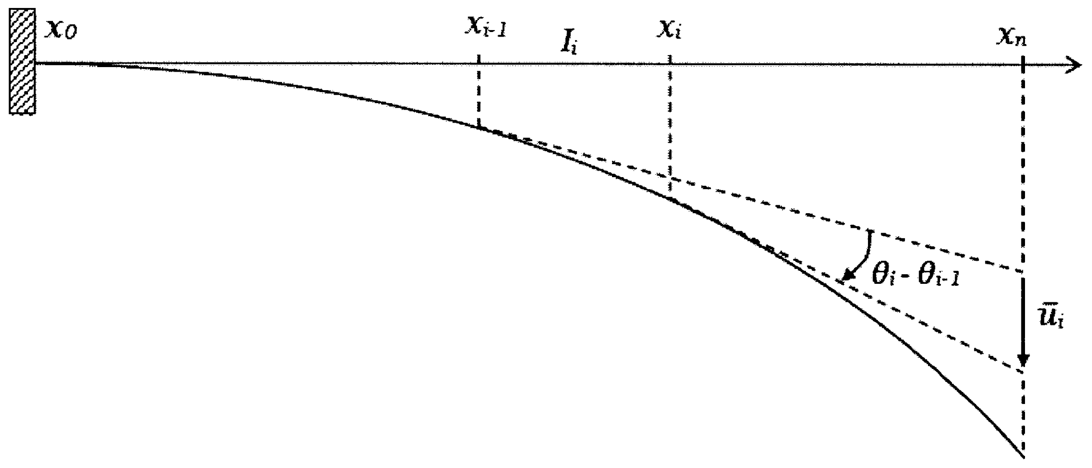


Figure 3.1: Participation of Segment Rotation in Beam Deflection

The moment of inertia distribution is derived from the discrete model shown on figure 3.1 (p.51) and then transformed into a continuous distribution. Discrete models are also considered for computer implementation.

As shown on figure 3.1 (p.51), the rotation of the i-th beam segment (segment between x_{i-1} and x_i) accounts for $\overline{u}_i$ in the total deflection. Since all rotations are considered small, $\overline{u}_i$ can be expressed as:

$$\overline{u}_i = (\theta_i - \theta_{i-1})(x_n - x_{i-1}) \tag{3.4}$$

In this discrete model, the bending moment is considered constant over each beam segment. The bending moment acting on the i-th beam segment (between x_{i-1} and x_i) is noted as M_i . When implementing optimization with numerical computer programs, the bending moment is usually known at the points x_i only. In this case, the bending moment M_i can be estimated as:

$$M_i = \frac{M(x_{i-1}) + M(x_i)}{2}$$

The moment-curvature equation (3.1) is discretized:

$$\frac{d\theta}{dx} = \frac{M(x)}{E\,I(x)} \quad \longrightarrow \quad (\theta_i - \theta_{i-1}) = \frac{M_i}{E\,I_i}(x_i - x_{i-1}) \tag{3.5}$$

Equations (3.4) and (3.5) yield:

$$\overline{u}_i = \frac{M_i}{E\,I_i}(x_i - x_{i-1})(x_n - x_{i-1}) \tag{3.6}$$

Notations are simplified:

$$\overline{u}_i = \frac{\alpha_i}{I_i} \qquad \text{where} \qquad \alpha_i = \frac{M_i}{E}(x_i - x_{i-1})(x_n - x_{i-1}) \tag{3.7}$$

Starting with a random allocation of the available inertia to the beam segments, the strategy is to transfer some inertia from segments to others in order to incrementally decrease the total deflection.

The effect of a change in I_i on the deflection participation $\overline{u}_\mathrm{i}$ is described by the derivative:

$$\frac{d\overline{u}_\mathrm{i}}{dI} = \frac{1}{dI}\left(\frac{\alpha_\mathrm{i}}{I_\mathrm{i}+dI} - \frac{\alpha_\mathrm{i}}{I_\mathrm{i}}\right) = -\frac{\alpha_\mathrm{i}}{I_\mathrm{i}^2} \tag{3.8}$$

Two beam segments a and b are now considered. Their moment of inertia are noted as I_a and I_b, and their contribution to the total deflection are $\overline{u}_\mathrm{a}$ and $\overline{u}_\mathrm{b}$ respectively. Their aggregate contribution to the total deflection is:

$$\overline{u}_\mathrm{a+b} = \overline{u}_\mathrm{a} + \overline{u}_\mathrm{b}$$

The derivatives $d\overline{u}_\mathrm{i}/dI$ can be calculated using (3.8). Assuming that:

$$\frac{d\overline{u}_\mathrm{a}}{dI} < \frac{d\overline{u}_\mathrm{b}}{dI} \tag{3.9}$$

Then a small amount of inertia ΔI is transferred from I_b to I_a. The new contributions of segments a and b to the total deflection are:

$$\overline{u}_\mathrm{a}{}' = \overline{u}_\mathrm{a} + \frac{d\overline{u}_\mathrm{a}}{dI}\Delta I \qquad \text{and} \qquad \overline{u}_\mathrm{b}{}' = \overline{u}_\mathrm{b} - \frac{d\overline{u}_\mathrm{b}}{dI}\Delta I$$

The new aggregate contribution of segments a and b to the total deflection is:

$$\begin{aligned}
\overline{u}_\mathrm{a+b}{}' &= \overline{u}_\mathrm{a}{}' + \overline{u}_\mathrm{b}{}' \\
&= \overline{u}_\mathrm{a} + \overline{u}_\mathrm{b} - \left(\frac{d\overline{u}_\mathrm{b}}{dI} - \frac{d\overline{u}_\mathrm{a}}{dI}\right)\Delta I \\
&= \overline{u}_\mathrm{a+b} - \underbrace{\left(\frac{d\overline{u}_\mathrm{b}}{dI} - \frac{d\overline{u}_\mathrm{a}}{dI}\right)\Delta I}_{>0} \\
\overline{u}_\mathrm{a+b}{}' &< \overline{u}_\mathrm{a+b}
\end{aligned}$$

The total deflection is decreased. As ΔI was transferred from I_b to I_a, segment b lost rigidity, increasing $\overline{u}_\mathrm{b}$, while segment a gained rigidity, decreasing $\overline{u}_\mathrm{a}$. But the decrease in $\overline{u}_\mathrm{a}$ was greater than the increase in $\overline{u}_\mathrm{b}$, which is what (3.9) represents.

Similar transfers of inertia can be done between any couple of beam segments whose derivatives $d\overline{u}_{\mathrm{i}}/dI$ are different, every transfer reducing the total deflection u. The smallest possible deflection is reached when no more transfer of inertia can decrease it, that is, when all the derivatives $d\overline{u}_{\mathrm{i}}/dI$ are equal.

$$\frac{d\overline{u}_{\mathrm{i}}}{dI_{\mathrm{i}}} = -\frac{\alpha_{\mathrm{i}}}{I_{\mathrm{i}}^2} = \beta_0 \qquad \longrightarrow \qquad I_{\mathrm{i}}^2 = \beta_1\,\alpha_{\mathrm{i}} \tag{3.10}$$

β_0, β_1 and all other β_{k} terms used in the following represent constants. It is convenient to use such terms to group all the constants, since the goal is to find the shape of the optimal moment of inertia distribution, which can then be scaled to achieve a given weight or deflection. The term α_{i} is replaced by its expression (3.7), giving the discrete moment of inertia distribution:

$$I_{\mathrm{i}}^2 = \beta_1\,\frac{M_{\mathrm{i}}}{E}(x_{\mathrm{i}} - x_{\mathrm{i\text{-}1}})(x_{\mathrm{n}} - x_{\mathrm{i\text{-}1}}) \tag{3.11}$$

The discrete distribution (3.11) is made continuous:

$$\left.\begin{array}{ccc} I_{\mathrm{i}} & \longrightarrow & I(x) \\ M_{\mathrm{i}} & \longrightarrow & M(x) \\ x_{\mathrm{i}} - x_{\mathrm{i\text{-}1}} & \longrightarrow & \text{constant} \\ x_{\mathrm{n}} - x_{\mathrm{i\text{-}1}} & \longrightarrow & L - x \end{array}\right\} \quad \longrightarrow \quad I(x)^2 = \beta_2\frac{M(x)}{E}(L - x) \tag{3.12}$$

The continuous distribution (3.12) can be written as:

$$I(x) = \beta\sqrt{(L - x)M(x)} \tag{3.13}$$

The term $\sqrt{(L - x)M(x)}$ is the shape of the optimal moment of inertia distribution. The scaling factor β can be adjusted in order to achieve a given weight or deflection. Further adjustments are necessary for implementation. The bending moment $M(x)$ and the distance to maximum deflection $(L - x)$ must be taken as absolute values, and a lower limit must imposed to $I(x)$ to keep some material at the inflexion points, where the bending moment is zero.

3.1.4 Implementation of the Modified Criterion

Actual Optimization Variables

An expression for the optimal bending rigidity distribution was derived in section 3.1.3. In practice, one is not directly interested in the bending rigidity distribution. Most often, the goal is to optimize for weight and stiffness, and the optimal material distribution is sought. Material distribution is described by the cross-sectional area $A(x)$. The total volume of material V and the weight of the beam W are:

$$V = \int_0^L A(x)dx \qquad \text{and} \qquad W = \rho V \qquad \text{where } \rho \text{ is the material density}$$

One can either fix the total weight and determine the stiffest beam, or fix the maximum allowable deflection and determine the lightest beam meeting that deflection criterion. For same fixity and loading conditions, the optimal material distribution differs only by a scaling factor between the two problems.

The optimal moment of inertia distribution was found by splitting the beam into small segments, expressing the contribution of each beam segment to the total deflection, and setting the derivative of that contribution with respect to the moment of inertia to be equal for all segments (see section 3.1.3)

$$\overline{u}_\mathrm{i} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{I_\mathrm{i}} \quad \longrightarrow \quad \frac{d\overline{u}_\mathrm{i}}{dI} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{I_\mathrm{i}^2} \quad \longrightarrow \quad I_{\mathrm{opt}}(x) \propto \sqrt{(L - x)M(x)}$$

The proportionality symbol ($\propto$) is used, as one is only interested in the shape of the optimal distribution, which is then scaled to achieve a given weight or deflection.

The beam property governing bending deformation is the moment of inertia I. It can be directly related to the cross-sectional area if the section has a single optimization parameter (e.g. rectangular section height, circular section radius ...). In order to find the optimal area distribution $A_{\mathrm{opt}}(x)$, one cannot simply replace the optimal inertia distribution $I_{\mathrm{opt}}(x)$ by its expression in terms of the area. The substitution must be done prior to taking the derivative, which is then taken with respect to the area. Since the inertia-area relationship depends on the cross-section geometry, the operation has to be done separately for each type of beam.

As an illustration, the optimal material distribution is derived for two types of solid cross-section.

Circular Section

For a circular cross-section of radius r, the cross-sectional area A and the moment of inertia I can be related:

$$\left.\begin{aligned} A &= \pi r^2 \\ I &= \frac{\pi}{4} r^4 \end{aligned}\right\} \quad \longrightarrow \quad I = \frac{A^2}{4\pi}$$

The segment contribution $\overline{u}_\mathrm{i}$ to the overall deflection is expressed in terms of the cross-sectional area, and the derivative is taken:

$$\overline{u}_\mathrm{i} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{I_\mathrm{i}} \quad \longrightarrow \quad \overline{u}_\mathrm{i} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{A_\mathrm{i}^2} \quad \longrightarrow \quad \frac{d\overline{u}_\mathrm{i}}{dA} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{A_\mathrm{i}^3}$$

A constant derivative corresponds to the optimal material distribution:

$$A_\mathrm{opt}(x) \propto \left[\,(L - x)M(x)\,\right]^{1/3} \tag{3.14}$$

Rectangular Section

The breadth b of a rectangular cross-section is fixed and its height h is kept as the only variable. The cross-sectional area A and the moment of inertia I can therefore be related:

$$\left.\begin{aligned} A &= bh \\ I &= \frac{bh^3}{12} \end{aligned}\right\} \quad \longrightarrow \quad I = \frac{A^3}{12b^2}$$

The segment contribution $\overline{u}_\mathrm{i}$ to the overall deflection is expressed in terms of the cross-sectional area, and the derivative is taken:

$$\overline{u}_\mathrm{i} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{I_\mathrm{i}} \quad \longrightarrow \quad \overline{u}_\mathrm{i} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{A_\mathrm{i}^3} \quad \longrightarrow \quad \frac{d\overline{u}_\mathrm{i}}{dA} \propto \frac{M_\mathrm{i}(L - x_\mathrm{i})}{A_\mathrm{i}^4}$$

A constant derivative corresponds to the optimal material distribution:

$$A_\mathrm{opt}(x) \propto \left[\,(L - x)M(x)\,\right]^{1/4} \tag{3.15}$$

Indeterminate Systems

In indeterminate systems, the bending moment acting on the beam depends on the stiffness distribution. And since the stiffness distribution is derived from the bending moment, the problem is coupled. An iterative approach is proposed to find the optimal material distribution for indeterminate beams.

If the cross-section is parametrized by a single variable (e.g. pipe diameter, tube wall thickness ...), then the cross-sectional area A and the moment of inertia I can be related. The material distribution is better represented by the cross-sectional area $A(x)$, and therefore the optimal distribution $A_{\text{opt}}(x)$ is sought.

A beam with constant cross-sectional area $A^{(1)}$ is considered as the initial design:

$$A^{(1)}(x) = \frac{\text{Beam Volume}}{\text{Beam Length}}$$

The loads P are applied and the system is solved to find the acting bending moment:

$$\left.\begin{array}{r} P \\ A^{(1)}(x) \rightarrow I^{(1)}(x) \end{array}\right\} \quad \rightarrow \quad M^{(1)}(x)$$

The material distribution is updated using a formula analytically derived and equivalent to (3.14) or (3.15):

$$A^{(2)}(x) = F(x, M^{(1)}(x))$$

The process is repeated, k representing the iteration:

$$\left.\begin{array}{r} P \\ A^{(\text{k})}(x) \rightarrow I^{(\text{k})}(x) \end{array}\right\} \quad \rightarrow \quad M^{(\text{k})}(x)$$

$$A^{(\text{k}+1)}(x) = F(x, M^{(\text{k})}(x))$$

The area distribution $A^{(\text{k})}(x)$ converges towards an the optimal distribution $A_{\text{opt}}(x)$. Convergence is quick, since the sensitivity of the bending moment to the changes in bending rigidity is limited for typical beam configurations.

3.2 Numerical Moment Integration

Following its application to truss geometry optimization (chapter 2), the gradient-based algorithm discussed in this report is proposed as a form-finding method for individual beams. Gradient-based optimization is iterative and many successive beam designs are analyzed during a beam shape optimization process. Therefore, the gradient-based algorithm needs to interact with an analysis tool that is efficient at solving beam problems. A first analysis strategy is to implement on a computer the well-known equations governing the behavior of beams: the bending moment is integrated twice to calculate the beam deformation. This method is applicable to some beam configurations and is therefore presented in this section. The other analysis technique is to model the beam with stiffness matrices and is detailed in the next section (3.3).

3.2.1 Limitations to the use of Analytical Integration

In a pure bending beam (no shear deformation), the bending moment M, the neutral axis rotation θ and the deflection u are related through:

$$M = EI\frac{d\theta}{dx} \quad \text{and} \quad \theta = \frac{du}{dx} \tag{3.16}$$

The rotation θ and deflection u can be calculated using:

$$\theta(x) = \theta_0 + \int_{x_0}^{x} \frac{M(x)}{EI(x)}\,dx \tag{3.17}$$

$$u(x) = u_0 + \int_{x_0}^{x} \theta(x)\,dx \tag{3.18}$$

However, such computations cannot always be carried out by computer programs. First, the analytical expression of the bending moment $M(x)$ is needed. Second, a computer code must be able to perform the integration. Third, even when analytical integration is technically possible, the required computations can be slow, which is problematic in the context of iterative optimization. It is easier and faster to resort to numerical integration. This allows the program to deal only with numbers and no variable.

3.2.2 Integral Approximation

Numerical integration is equivalent to approximating the integral by area calculation. In order to perform a numerical integration of the bending moment over the beam, the system must be discretized. Figure 3.2 illustrates the discretization. Instead of a continuous analytical expression $M(x)$, the bending moment is numerically known at $(n+1)$ points distributed along the beam. These points are called integration points, and their coordinates in the beam direction are noted $x_0, x_1, \cdots x_n$. The moment at the integration point x_i is noted as M_i. Integration points separate the beam into segments, and the moment of inertia of the beam is considered constant over each segment. The moment of inertia of the beam segment between integration points x_i and x_{i+1} is noted as $I_{i,i+1}$.

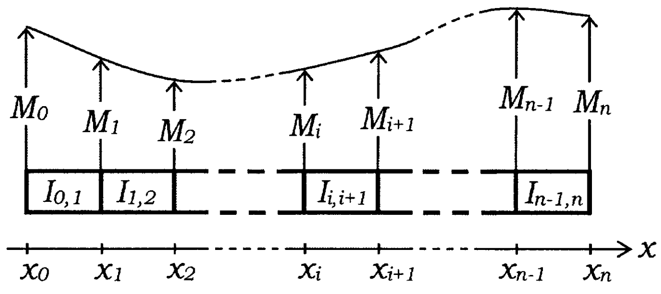


Figure 3.2: Integration Points on a Beam

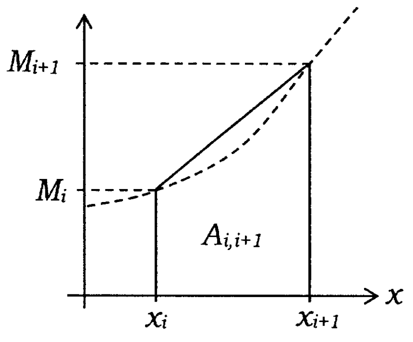


Figure 3.3: Affine Approximation

Starting with an initial rotation θ_0, the rotation θ would be analytically calculated at each integration points, using:

$$\theta_{i+1} = \theta_i + \frac{1}{E\,I_{i,i+1}} \int_{x_i}^{x_{i+1}} M(x)\,dx \tag{3.19}$$

In order to easily calculate the integral in (3.19), the moment is assumed to vary linearly between two successive integration points (figure 3.3). Then, its integral is estimated as the area $A_{i,i+1}$:

$$\int_{x_i}^{x_{i+1}} M(x)\,dx = A_{i,i+1} = \left(\frac{M_i + M_{i+1}}{2}\right)(x_{i+1} - x_i) \tag{3.20}$$

The recursive relationship used for numerical integration is therefore:

$$\theta_{i+1} = \theta_i + \left(\frac{1}{E\,I_{i,i+1}}\right)\left(\frac{M_i + M_{i+1}}{2}\right)(x_{i+1} - x_i) \tag{3.21}$$

The same process is used to calculate the deflection u by integrating the rotation θ:

$$u_{i+1} = u_i + \int_{x_i}^{x_{i+1}} \theta(x)\,dx \qquad \text{and} \qquad \int_{x_i}^{x_{i+1}} \theta(x)\,dx = \left(\frac{\theta_i + \theta_{i+1}}{2}\right)(x_{i+1} - x_i)$$

$$\rightarrow \quad u_{i+1} = u_i + \left(\frac{\theta_i + \theta_{i+1}}{2}\right)(x_{i+1} - x_i) \tag{3.22}$$

3.2.3 Boundary Conditions

Recursive relationships (3.21) and (3.22) are enough to calculate the rotation and deflection at all integration points if the initial rotation θ_0 and deflection u_0 are known. Though some beam configurations impose $u_0 = 0$, the initial rotation θ_0 is usually not known. A strategy is to first calculate a pseudo-rotation $\overline{\theta}$ and a pseudo-deflection $\overline{u}$ corresponding to zero initial rotation and deflection:

$$\begin{aligned} &\overline{\theta}_0 = 0 \quad \text{and} \quad \overline{\theta}_{i+1} = \overline{\theta}_i + \left(\frac{1}{E\,I_{i,i+1}}\right)\left(\frac{M_i + M_{i+1}}{2}\right)(x_{i+1} - x_i) \\ &\overline{u}_0 = 0 \quad \text{and} \quad \overline{u}_{i+1} = \overline{u}_i + \left(\frac{\overline{\theta}_i + \overline{\theta}_{i+1}}{2}\right)(x_{i+1} - x_i) \end{aligned} \tag{3.23}$$

The actual rotation θ and deflection u can be written as:

$$\begin{aligned} \theta_i &= \theta_0 + \overline{\theta}_i \\ u_i &= u_0 + \theta_0(x_i - x_0) + \overline{u}_i \end{aligned} \tag{3.24}$$

θ_0 and u_0 are determined by applying equations (3.24) to rotations and/or displacements known from boundary conditions.

3.2.4 Example

In this example, numerical integration is used to calculate the rotation and deflection of a simply-supported beam loaded at mid-span.

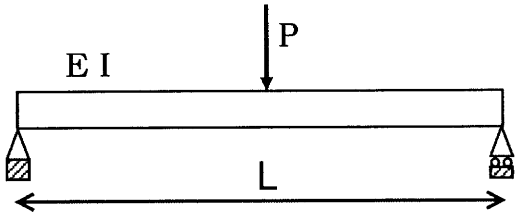


Figure 3.4: Problem

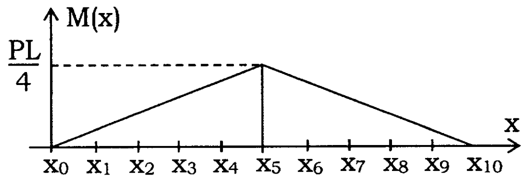


Figure 3.5: Discretization

The problem is presented on figure 3.4. The point load $P = 10\,\text{kip}$ is applied at the middle of the beam of length $L = 100\,\text{in}$ whose constant section moment of inertia is $I = 2\,\text{in}^4$. The problem is discretized at 11 integration points, evenly spaced along the beam (figure 3.5). By applying equilibrium, the bending moment is known at all integration points. Table 3.1 shows the values calculated during the numerical integration process.

i	x_i	M_i	$\overline{\theta}_i$	$\overline{u}_i$	θ_i	u_i
0	0	0	0.000	0.000	-0.108	0.000
1	10	50	0.004	0.021	-0.103	-1.056
2	20	100	0.017	0.129	-0.090	-2.026
3	30	150	0.039	0.409	-0.069	-2.823
4	40	200	0.069	0.948	-0.039	-3.362
5	50	250	0.108	1.832	0.000	-3.556
6	60	200	0.146	3.103	0.039	-3.362
7	70	150	0.177	4.720	0.069	-2.823
8	80	100	0.198	6.595	0.090	-2.026
9	90	50	0.211	8.642	0.103	-1.056
10	100	0	0.215	10.77	0.108	0.000

Table 3.1: Simply Supported Beam Numerical Integration

Once $\overline{\theta}_i$ and $\overline{u}_i$ are calculated using (3.23), boundary conditions are used with (3.24) to determine u_0 and θ_0:

$$u_0 = 0$$
$$u_{10} = 0 \quad \rightarrow \quad u_0 + \theta_0(x_{10} - x_0) + \overline{u}_{10} = 0 \quad \rightarrow \quad \theta_0 = -\frac{u_0 + \overline{u}_{10}}{x_{10} - x_0} = -0.108$$

The pseudo- ($\overline{\theta}$, $\overline{u}$) and final (θ, u) deformations are represented on figure 3.6:

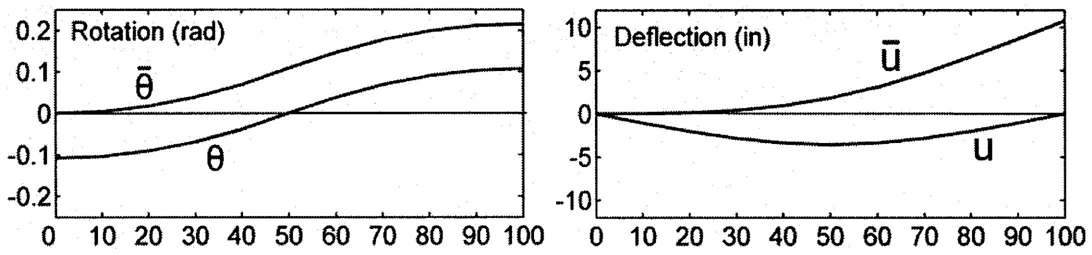


Figure 3.6: Numerical Integration Steps

The analytical solution for the rotation at $x = 0$ is:

$$\theta_{\max} = \theta_0 = \frac{PL^2}{16EI} = 0.108\,\text{rad} \tag{3.25}$$

The analytical solution for the deflection at mid-span is:

$$u_{\max} = u_5 = \frac{PL^3}{48EI} = 3.592\,\text{in} \tag{3.26}$$

In this particular case, numerical integration gives the exact rotation value. This is because the actual bending moment varies linearly between integration points, making the affine approximation (figure 3.3 p.59) equivalent to the exact integral.

Since the rotation is not linear, its numerical integral differs from the exact integral. This is why there is a 1% difference between the exact deflection and the value obtained by numerical integration. Only 11 integration points were considered, and the difference could be made even smaller by increasing the number of points.

When programming numerical integration to implement optimization, it is important to consider a sufficient number of integration points for the model to be sensitive enough to the adjustments of the optimization variables. In order to keep the total number of integration points reasonably low, more points should be allocated to the portions of the beam where the bending moment has large derivatives (e.g. around mid-span supports of multi-supported beams). Adaptive algorithms can be used to perform these more precise numerical integrations. Such algorithms detect large derivatives and increase the number of integration points where necessary. These are especially useful if many load cases are considered in the optimization process, making it necessary to numerically integrate different bending moments, whose higher derivatives are not always at the same locations.

If the applied loads are known but not the bending moment (i.e. indeterminate systems), deflection calculation by numerical integration is still possible but more complex. The shear force is calculated by summing the loads applied along the beam, and variables must be introduced to represent the reaction forces and moments. These variables are kept as unknown during the integration process. 3 successive integrations are carried out to get the deflection, and boundary conditions are applied to find the reaction forces and initial rotation and displacement. Since variables are used, much of the advantage of numerical integration is lost and the process is slower. Another analysis method is therefore needed to solve indeterminate beam configurations.

3.2.5 Numerical Integration Code

The Matlab® code below numerically integrates the bending moment on a simply-supported beam to calculate its rotation and deflection.

```
 1   function[R,U]=numerical_integration(X,I,M);
 2
 3   E=29000;
 4   n=length(X);
 5
 6   R=[0];
 7   for i=1:1:n-1;
 8       R=[R, R(i)+(M(i)+M(i+1))/(2*E*I(i))*(X(i+1)-X(i))];
 9   end;
10
11   U=[0];
12   for i=1:1:n-1;
13       U=[U, U(i)+(R(i)+R(i+1))/2*(X(i+1)-X(i))];
14   end;
15
16   u0=0;
17   r0=-Ut(n)/(X(n)-X(1));
18
19   R=R+r0;
20   U=U+u0+r0*(X-X(1));
```

`X(i)`	:	x-coordinate of i-th integration point
`I(i)`	:	moment of inertia of beam segment between i-th and (i+1)-th points
`M(i)`	:	bending moment at the i-th integration point
`R(i)`	:	rotation at the i-th integration point
`U(i)`	:	deflection at the i-th integration point

`6-9`	:	calculates rotation by integrating bending moment
`11-14`	:	calculates deflection by integrating rotation
`16-17`	:	calculates initial rotation and deflection
`19-20`	:	adds the effect of initial deformations to rotation and deflection

Lines `16` and `17` are specific to the beam support conditions. In this example, initial rotation and deflection are calculated for a beam simply-supported at both ends. Only these two lines need to be modified to deal with other beam configurations.

3.3 Beam Matrix Analysis

The numerical moment integration method presented in section 3.2 is an efficient tool to solve determinate beam problems with the speed and accuracy necessary in optimization. Indeterminate systems can still be solved by numerical moment integration, but the process is slower since integration constants must be introduced, making the method loose its advantage of handling only numbers. The matrix analysis method used in chapter 2 to optimize trusses is well-suited to handle indeterminate systems and is therefore considered for beam optimization as well. This section presents the main steps for solving beams by matrix analysis and proposes a convenient way of using this method in the context of optimization.

3.3.1 Linear Model

Beam shape optimization deals with beams of non-constant cross-sections. In matrix analysis, such beams can be modeled as series of small beam segments whose cross-sectional properties are varied independently. The behavior of an individual beam segment is governed by several force-displacement relationships summarized in a *stiffness matrix.*

Models of different complexities can be used to derive the stiffness matrix of a beam segment. In this study, a simple linear model is considered. Nonlinear terms affect the way beams deform under loading, but they do not change the solutions to the shape optimization problems considered here. Typically, a beam being optimized for stiffness converges towards the same optimal shape with linear and nonlinear models, even though the magnitudes of the deflections differ slightly. Nonlinear effects are therefore not considered, allowing for faster structural analysis and a reduced optimization time.

A two-dimensional horizontal beam model is sufficient for the beam shape optimization problems treated in this study, further simplifying the stiffness matrices.

3.3.2 Beam Segment Stiffness Matrix

Considering a linear model, the force-displacement relationship for a two-dimensional beam segment is expressed as a 4-by-4 stiffness matrix. The derivation of the stiffness matrix is detailed in appendix B, and its final expression is given below.

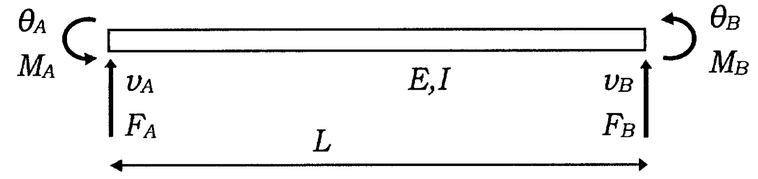


Figure 3.7: Beam Segment

The beam segment (figure 3.7) is limited by two nodes (A, B). Each node has 2 degrees of freedom [v, θ] representing a vertical displacement and a rotation respectively. The beam segment is subject to externally-applied loads and to the actions of the other beam segments connected to its nodes. The resulting loads [V, M] acting at the nodes correspond to the shear force and bending moment in the beam. The stiffness matrix relates the nodal displacements [v, θ] to the nodal forces [V, M] and depends on the properties of the beam segment.

For a beam segment of length L, cross-sectional moment of inertia I and modulus of elasticity E, the force-displacement relationship is expressed as:

$$\begin{pmatrix} F_{\mathrm{A}} \\ M_{\mathrm{A}} \\ F_{\mathrm{B}} \\ M_{\mathrm{B}} \end{pmatrix} = EI \begin{pmatrix} \frac{12}{L^3} & \frac{6}{L^2} & -\frac{12}{L^3} & \frac{6}{L^2} \\ \frac{6}{L^2} & \frac{4}{L} & -\frac{6}{L^2} & \frac{2}{L} \\ -\frac{12}{L^3} & -\frac{6}{L^2} & \frac{12}{L^3} & -\frac{6}{L^2} \\ \frac{6}{L^2} & \frac{2}{L} & -\frac{6}{L^2} & \frac{4}{L} \end{pmatrix} \begin{pmatrix} v_{\mathrm{A}} \\ \theta_{\mathrm{A}} \\ v_{\mathrm{B}} \\ \theta_{\mathrm{B}} \end{pmatrix}$$

The stiffness matrix of every segment is calculated as the first step of the matrix analysis process of a beam.

3.3.3 Full Beam Solution

Full Beam Stiffness Matrix

The stiffness matrix of every beam segment is built as described in section 3.3.2. These matrices are combined to represent the full beam as a large stiffness matrix, using the following term-by-term matrix addition:

Beam Segment 1 Beam Segment 2

$$\begin{pmatrix} \underline{F}_{\mathrm{A}} \\ \underline{F}_{\mathrm{B}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(1)}_{\mathrm{AA}} & \underline{K}^{(1)}_{\mathrm{AB}} \\ \underline{K}^{(1)}_{\mathrm{BA}} & \underline{K}^{(1)}_{\mathrm{BB}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{A}} \\ \underline{U}_{\mathrm{B}} \end{pmatrix} \qquad \begin{pmatrix} \underline{F}_{\mathrm{B}} \\ \underline{F}_{\mathrm{C}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(2)}_{\mathrm{BB}} & \underline{K}^{(2)}_{\mathrm{BC}} \\ \underline{K}^{(2)}_{\mathrm{CB}} & \underline{K}^{(2)}_{\mathrm{CC}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{B}} \\ \underline{U}_{\mathrm{C}} \end{pmatrix}$$

Beam Segments 1 and 2

$$\begin{pmatrix} \underline{F}_{\mathrm{A}} \\ \underline{F}_{\mathrm{B}} \\ \underline{F}_{\mathrm{C}} \end{pmatrix} = \begin{pmatrix} \underline{K}^{(1)}_{\mathrm{AA}} & \underline{K}^{(1)}_{\mathrm{AB}} & \underline{0} \\ \underline{K}^{(1)}_{\mathrm{BA}} & \underline{K}^{(1)}_{\mathrm{BB}} + \underline{K}^{(2)}_{\mathrm{BB}} & \underline{K}^{(2)}_{\mathrm{BC}} \\ \underline{0} & \underline{K}^{(2)}_{\mathrm{CB}} & \underline{K}^{(2)}_{\mathrm{CC}} \end{pmatrix} \begin{pmatrix} \underline{U}_{\mathrm{A}} \\ \underline{U}_{\mathrm{B}} \\ \underline{U}_{\mathrm{C}} \end{pmatrix}$$

For a beam discretized into (N-1) segments, the size of the full stiffness matrix is $2N$-by-$2N$. It relates the displacements of the nodes to the applied loads and reaction forces acting on the beam. The stiffness matrix of the full beam cannot be used is this initial form and needs to be rearranged and reduced to solve the beam analysis problem.

Rearrangement and Reduction

Three categories of degree of freedom are distinguished, as introduced by the example shown of figure 3.8.

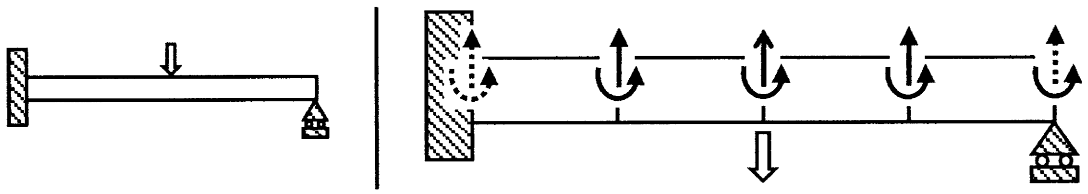


Figure 3.8: Sample Beam Configuration (left) Modeled as 4 Beam Elements (right)

Symbol	Generic Name	External Force	Displacement
⟶	Control DOF	Applied	Free
—▶	Unconstrained DOF	None	Free
····▶	Fixed DOF	Reaction	None

The stiffness matrix of the full beam is rearranged by grouping the degrees of freedom into the 3 proposed categories. The non-zero external forces and the free degrees of freedom are combined to form the column vectors $\underline{P}$, $\underline{R}$, $\underline{U}_\mathrm{C}$ and $\underline{U}_\mathrm{U}$ as follows:

Degrees of Freedom	External Forces	Displacements
Control DOFs	$\underline{P}$	$\underline{U}_\mathrm{C}$
Unconstrained DOFs	$\underline{0}$	$\underline{U}_\mathrm{U}$
Fixed DOFs	$\underline{R}$	$\underline{0}$

The external forces $\underline{P}$ are applied at the control degrees of freedom, whose displacements are noted as $\underline{U}_\mathrm{C}$. There is no force (forces $= \underline{0}$) acting on the unconstrained degrees of freedoms, whose displacements are noted as $\underline{U}_\mathrm{U}$. Reactions $\underline{R}$ occur at the fixed degrees of freedom, whose displacements are $\underline{0}$. The force-displacement relationship for the full beam is now written as:

$$\begin{pmatrix} \underline{P} \\ \underline{0} \\ \underline{R} \end{pmatrix} = \begin{pmatrix} \underline{K}_\mathrm{PC} & \underline{K}_\mathrm{PU} & \underline{K}_\mathrm{P0} \\ \underline{K}_\mathrm{0C} & \underline{K}_\mathrm{0U} & \underline{K}_\mathrm{00} \\ \underline{K}_\mathrm{RC} & \underline{K}_\mathrm{RU} & \underline{K}_\mathrm{R0} \end{pmatrix} \begin{pmatrix} \underline{U}_\mathrm{C} \\ \underline{U}_\mathrm{U} \\ \underline{0} \end{pmatrix} \tag{3.27}$$

The externally applied forces $\underline{P}$ are known, and the goal is to solve for $\underline{U}_\mathrm{C}$, $\underline{U}_\mathrm{U}$ and $\underline{R}$. By using the lines of (3.27) as 3 equations, the displacements $\underline{U}_\mathrm{C}$ and $\underline{U}_\mathrm{U}$ and the reactions $\underline{R}$ can be expressed as functions of the applied loads $\underline{P}$:

$$\underline{U}_\mathrm{C} = \underline{F}_\mathrm{C}\,\underline{P} \qquad \underline{U}_\mathrm{U} = \underline{F}_\mathrm{U}\,\underline{P} \qquad \underline{R} = \underline{E}\,\underline{P} \tag{3.28}$$

$$\begin{aligned} \text{Control DOFs Flexibility Matrix: } \underline{F}_\mathrm{C} &= (\underline{K}_\mathrm{PC} - \underline{K}_\mathrm{PU}\,\underline{K}_\mathrm{0U}^{-1}\,\underline{K}_\mathrm{0C})^{-1} \\ \text{Unconstrained DOFs Flexibility Matrix: } \underline{F}_\mathrm{U} &= -\underline{K}_\mathrm{0U}^{-1}\,\underline{K}_\mathrm{0C}\,\underline{F}_\mathrm{C} \\ \text{Force Equilibrium Matrix: } \underline{E} &= \underline{K}_\mathrm{RC}\,\underline{F}_\mathrm{C} + \underline{K}_\mathrm{RU}\,\underline{F}_\mathrm{U} \end{aligned}$$

In the following, the three matrices defined in (3.28) are referred to as *intermediate matrices*, as they are just tools to solve (3.27) but have little physical meaning. The stiffness matrix of a full beam can be very large, so the calculation of $\underline{F}_\mathrm{C}$, $\underline{F}_\mathrm{U}$ and $\underline{E}$ require many operations. When implementing optimization on computers, not all three of these matrices are always needed. For example, if a simply-supported beam loaded at mid-span is being optimized to limit the mid-span deflection, only $\underline{F}_\mathrm{C}$ is needed since one only wants to calculate the displacement of a control degree of freedom.

3.4 Beam Optimization Program

Beam optimization was implemented in Matlab®. The program created to optimize trusses (section 2.2) was modified to handle beams. The truss optimization program used a built-in optimization toolbox to run the optimization algorithm and featured a truss analysis tool based on stiffness matrices. Since the same gradient-based algorithm was used for beam optimization and since beams can also be analyzed using stiffness matrices, very few modifications of the program were necessary.

3.4.1 Schematic Diagram

Figure 3.9 shows the group of Matlab® functions used to optimize beams and the way they interact. Clarifications about the schematic diagram can be found in appendix C. The functions `objective`, `constraints` and `output` interact directly with the optimization toolbox, while all other functions form the beam analysis program. All functions are described in section 2.2.2 for the very similar truss optimization program, and the code is available in appendix D.

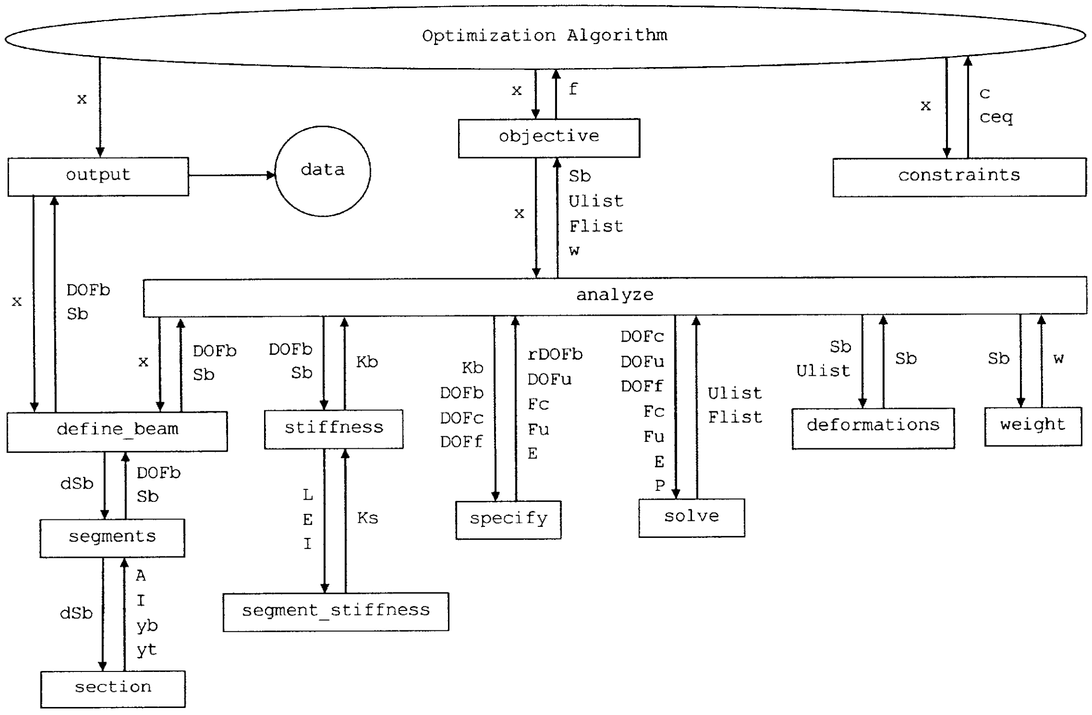


Figure 3.9: Beam Optimization Program Schematic Diagram

3.5 Beam Shape Optimization Examples

The computer program described in section 3.4 was used to find optimal beam shapes for various support conditions. The results are presented in this section.

3.5.1 General Considerations

Objective

A beam of solid rectangular cross-section (figure 3.10) is optimized by adjusting the material distribution in order to increase the overall stiffness. A solid cross-section is considered because the shape of a solid beam clearly reflects its material distribution, turning the stiffness optimization into a more visible form-finding process. In practice, however, solid cross-sections are more relevant for concrete beams.

Optimization Variables

The beam is sliced into segments whose heights are varied individually. The beam has a total length of 400 and is discretized into 40 segments of length 10. The depth (h_i) of each beam segment is an optimization variable. The number of variables can be reduced to 20 for symmetric problems. The beam has a constant width of 10. All numerical values are unitless.

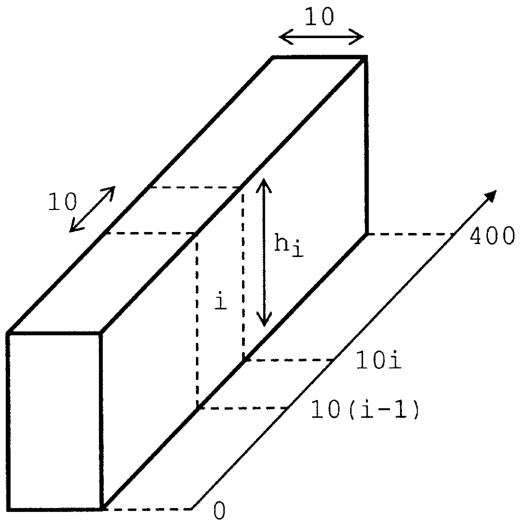


Figure 3.10: Beam Parametrization

Constraints

The beam cannot be shallower than 10 nor deeper than 30 at any point. The total volume of material is limited to 80000, corresponding to a constant depth of 20. These constraints are expressed as:

$$10 \leq h_i \leq 30 \quad \text{for} \quad i = 1 \dots 40 \qquad \sum_{i=1}^{40} h_i \leq 800$$

During the optimization process, no structural analysis is not needed to check the constraints, since they are directly expressed in terms of the optimization variables.

3.5.2 Clamped Beam

The optimal beam design process is detailed on the example of a clamped beam, with moment connections at both ends. The results for other beam configurations are presented in the next sections (3.5.3 - 3.5.7).

Problem Configuration

The beam is fully restrained (displacement and rotation) at both extremities and subject to a uniformly distributed load over its whole span.

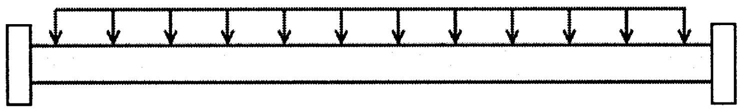


Figure 3.11: Problem Configuration

Shape Evolution

Starting with a uniform material distribution, the optimization algorithm varies the depth of the beam to minimize its mid-span deflection. The initial design given to the algorithm has an effect on the convergence speed of the process.

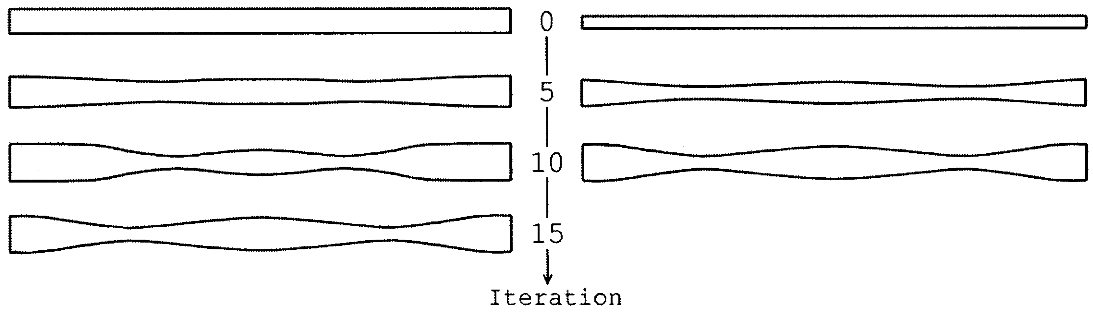


Figure 3.12: Shape Evolution Depending on Initial Depth

The left-hand side of figure 3.12 shows a few steps of the optimization process initialized with the deepest uniform beam satisfying the constraints (depth=20, limited by the amount of material available). The algorithm redistributes the material and reaches the optimal design, as defined by the stopping criterion, after 15 iterations. On the right-hand side, the process is initialized with the shallowest beam satisfying the constraints (minimum depth=10). In that case, some additional material is directly added where it is most needed, resulting in a faster convergence after only 10 iterations.

Optimal Material Distribution

The optimal material distribution (figure 3.13) generated by the algorithm features two shallow portions, corresponding to inflexion points when the load is applied.

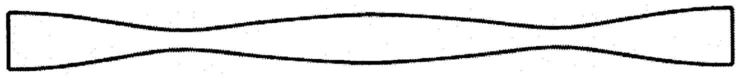


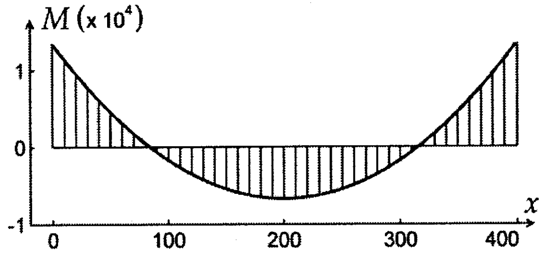
Figure 3.13: Optimal Material Distribution

The shape of the optimal material distribution resembles the bending moment diagram (figure 3.14), with the exception of the curvature at both fixities. The variation in beam thickness is limited at the fixities, while the derivative of the bending moment is maximum. The difference is due to the constraints imposed to the depth of the beam and to corrective effects presented in section 3.1.

Figure 3.14: Bending Moment Acting on Clamped Beam

$$M(x) = \frac{w}{12}(6Lx - L^2 - 6x^2)$$

Unitless graph for $L = 400$ and $w = 1$



The optimal material distribution minimizes the deflection at mid-span, as shown on figure 3.15. The deflection of the optimized beam is decreased by 40% compared with a beam made with the same amount of material uniformly distributed.

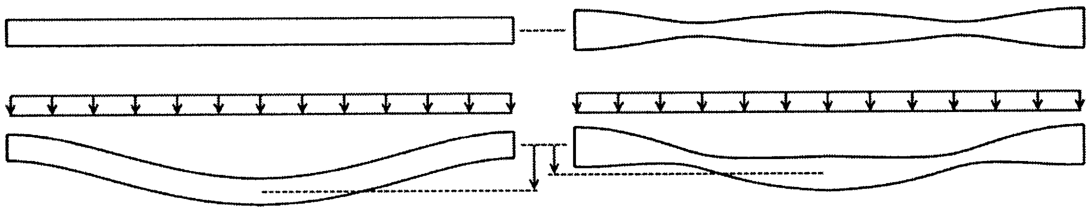


Figure 3.15: Stiffness Comparison

The application of the optimization algorithm allows for a significant increase in structural efficiency. However, the curved shape obtained would not be convenient in actual constructions. Further modifications of the material distribution are necessary to complete the design of a usable optimized beam.

Top Surface Flattening

Beams typically support horizontal surfaces, making inconvenient the use of the curved shape obtained from optimization. The material is shifted to end up with a flat top surface (figure 3.16) keeping the optimal distribution in the longitudinal direction of the beam.

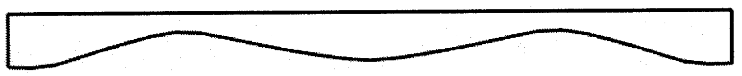


Figure 3.16: Flattened Beam

Cross-sections are not modified as the beam is flattened. Therefore, as long as the slopes on the curved bottom surface are reasonably small, the flattening of the beam does not affect its deflection (figure 3.17).

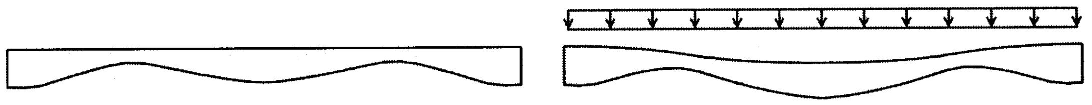


Figure 3.17: Flattened Beam Deformation

Cambering

The flattened beam can be cambered to pre-compensate for deflection. Cambering is not an additional difficulty for this type of optimized beam, since curved shapes are fabricated anyway. The final beam shape is represented on figure 3.16, with an exaggerated camber.

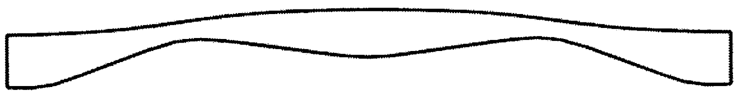


Figure 3.18: Cambered Beam

When the load is applied, the top surface of the beam becomes horizontal (figure 3.19). In its final configuration, the optimized beam can be seen as architecturally interesting.

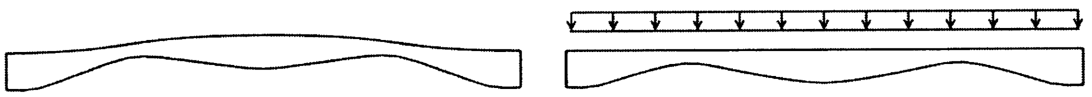


Figure 3.19: Cambered Beam Deformation

3.5.3 Two-Support Beam

The beam is supported at both extremities and subject to a uniform distributed load.

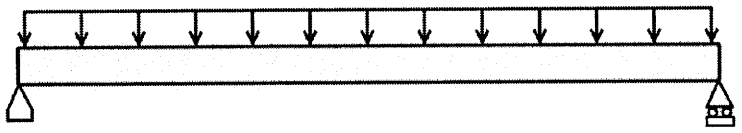


Figure 3.20: Problem Configuration

The algorithm is run to optimize the material distribution. The resulting shape (figure 3.21) is oval.

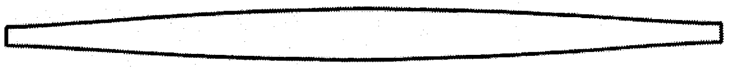


Figure 3.21: Optimal Material Distribution

The mid-span deflection of the optimized beam is decreased by 32% compared with a beam made with the same amount of material uniformly distributed.

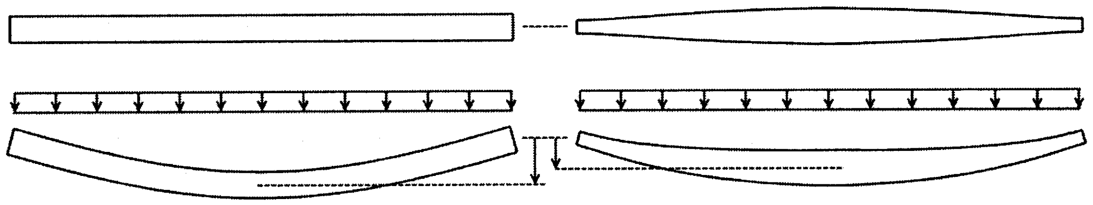


Figure 3.22: Stiffness Comparison

Based on the optimal material distribution, a practical beam (figure 3.23) is designed so that its top surface is horizontal when the load is applied.

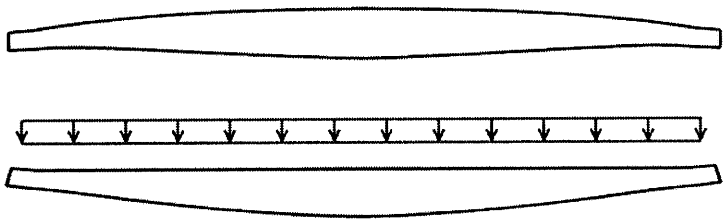


Figure 3.23: Practical Beam Design (Exaggerated Camber)

3.5.4 Three-Support Beam

The beam is supported at both extremities and at mid-span. A uniform distributed load is applied.

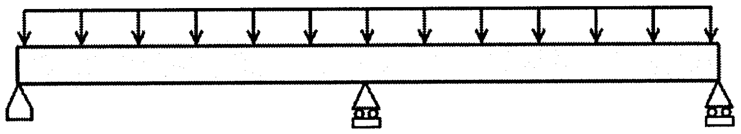


Figure 3.24: Problem Configuration

The algorithm is run to optimize the material distribution. The resulting shape (figure 3.25) features 2 shallow portions, corresponding to inflexion points when the load is applied.

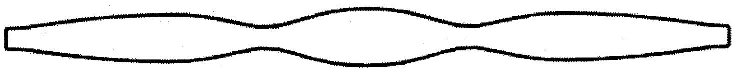


Figure 3.25: Optimal Material Distribution

The aggregate deflection of the optimized beam is decreased by 39% compared with a beam made with the same amount of material uniformly distributed.

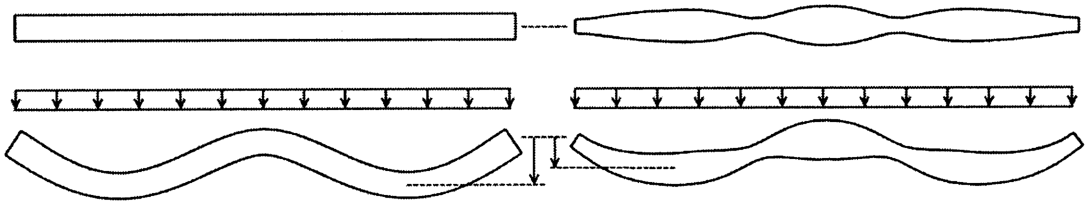


Figure 3.26: Stiffness Comparison

Based on the optimal material distribution, a practical beam (figure 3.27) is designed so that its top surface is horizontal when the load is applied.

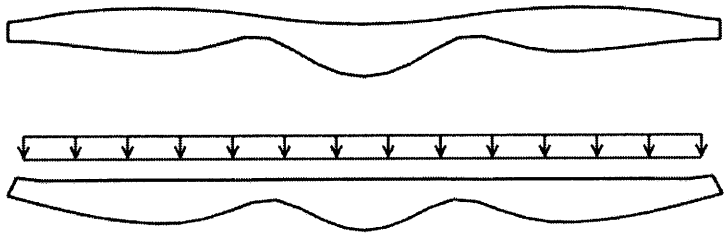


Figure 3.27: Practical Beam Design (Exaggerated Camber)

3.5.5 Four-Support Beam

The beam is supported by 4 supports evenly spaced and is subject to a uniform distributed load.

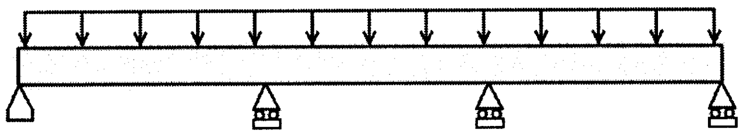


Figure 3.28: Problem Configuration

The algorithm is run to optimize the material distribution. The resulting shape (figure 3.29) features 4 shallow portions, corresponding to inflexion points when the load is applied.

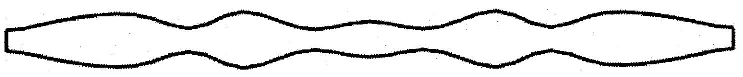


Figure 3.29: Optimal Material Distribution

The aggregate deflection of the optimized beam is decreased by 43% compared with a beam made with the same amount of material uniformly distributed.

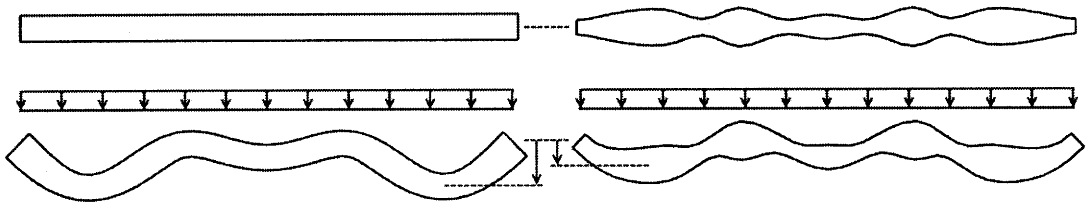


Figure 3.30: Stiffness Comparison

Based on the optimal material distribution, a practical beam (figure 3.31) is designed so that its top surface is horizontal when the load is applied.

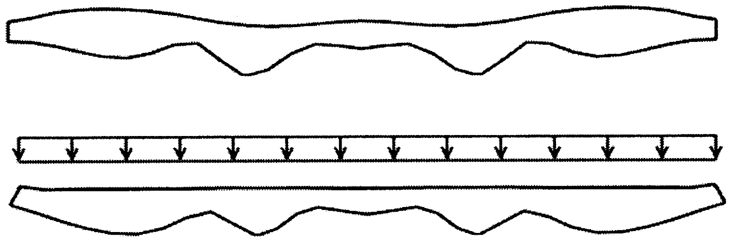


Figure 3.31: Practical Beam Design (Exaggerated Camber)

3.5.6 Simple Cantilever Beam

The simple cantilever beam is subject to a uniform distributed load.

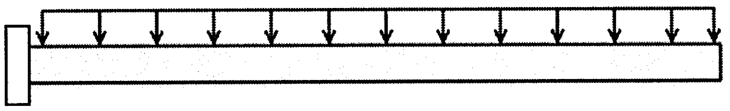


Figure 3.32: Problem Configuration

The algorithm is run to optimize the material distribution. The resulting shape (figure 3.33) tapers towards the free extremity. The shape is not exactly triangular, due to the constraints imposed to the beam depth.

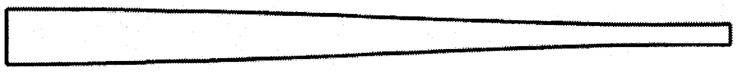


Figure 3.33: Optimal Material Distribution

The free extremity deflection of the optimized beam is decreased by 51% compared with a beam made with the same amount of material uniformly distributed.

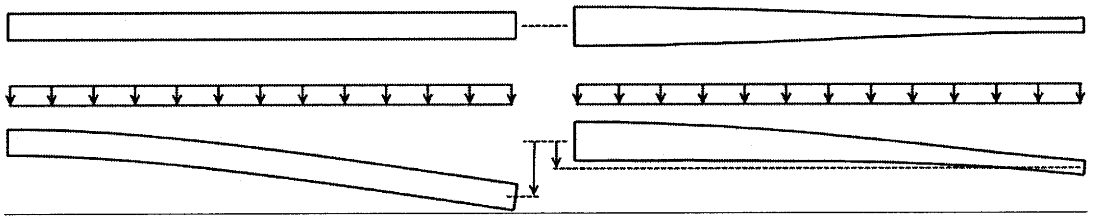


Figure 3.34: Stiffness Comparison

Based on the optimal material distribution, a practical beam (figure 3.35) is designed so that its top surface is horizontal when the load is applied.

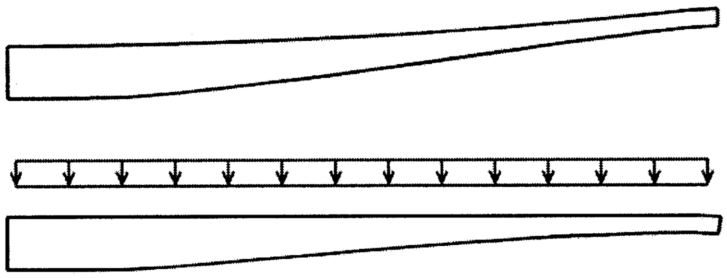


Figure 3.35: Practical Beam Design (Exaggerated Camber)

3.5.7 Hybrid Cantilever Beam

A pinned support is added at 1/3rd of a simple cantilever beam. A uniform distributed load is applied.

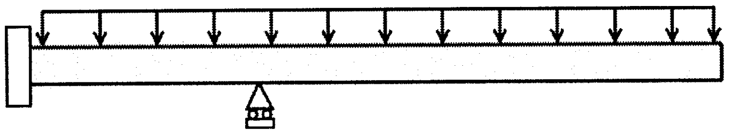


Figure 3.36: Problem Configuration

The algorithm is run to optimize the material distribution. The resulting shape (figure 3.37) features a shallow portion between the fixities, corresponding to an inflexion point when the load is applied. The beam is deeper around the pinned support, where the bending moment is maximum. The cantilever portion tapers towards the free extremity.

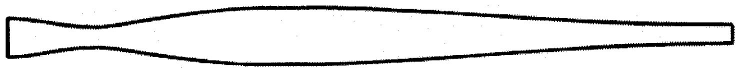


Figure 3.37: Optimal Material Distribution

The free extremity deflection of the optimized beam is decreased by 50% compared with a beam made with the same amount of material uniformly distributed.

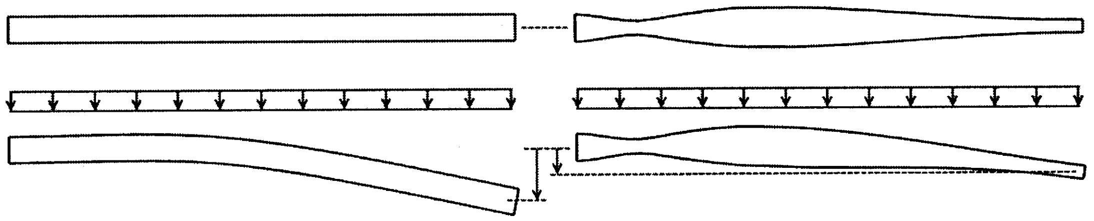


Figure 3.38: Stiffness Comparison

Based on the optimal material distribution, a practical beam (figure 3.39) is designed so that its top surface is horizontal when the load is applied.

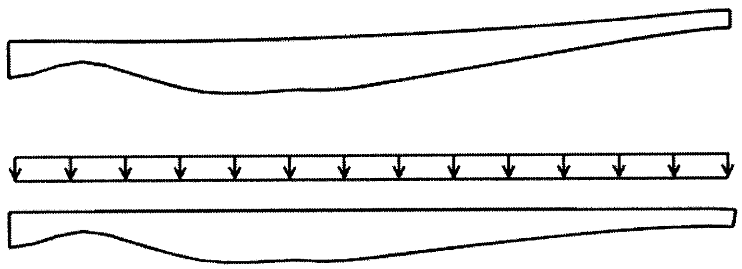


Figure 3.39: Practical Beam Design

Chapter 4

Design Optimization Example

In the previous chapters, a gradient-based optimization method was applied to truss and beam form-finding. The algorithm proved to be self-adaptive to a range of structural configurations, making implementation relatively quick and simple. In this last chapter, the robustness the gradient-based method is further evaluated by applying the same algorithm to a different type of structural design problem.

A simple girder was first optimized as part of the design process of a bridge. Compared with the form-finding problems treated before, this is a more practical application of optimization since the goal was to adjust the sizes of the constitutive members.

The girder was then stiffened with a truss system, and the resulting hybrid structure was also optimized as a whole. This last example, combining beam and truss behaviors, shows that the gradient-based method is applicable to the optimization of more complex systems.

4.1 Problem Introduction

The chapter is based on an actual design problem requiring optimization. The context, objective and constraints of the problem are presented in this section.

4.1.1 ASCE/AISC Student Steel Bridge Competition

The *Student Steel Bridge Competition* is a national contest co-sponsored by the American Society of Civil Engineers (ASCE) and the American Institute of Steel Construction (AISC). The competing teams, representing their schools, must design a 20-foot long steel bridge and fabricate its constitutive pieces. On the day of the competition, each team assembles its bridge as fast as possible. The structures are then weighted and tested for deflection. The weight, stiffness and construction speed are taken into account to determine the winner.

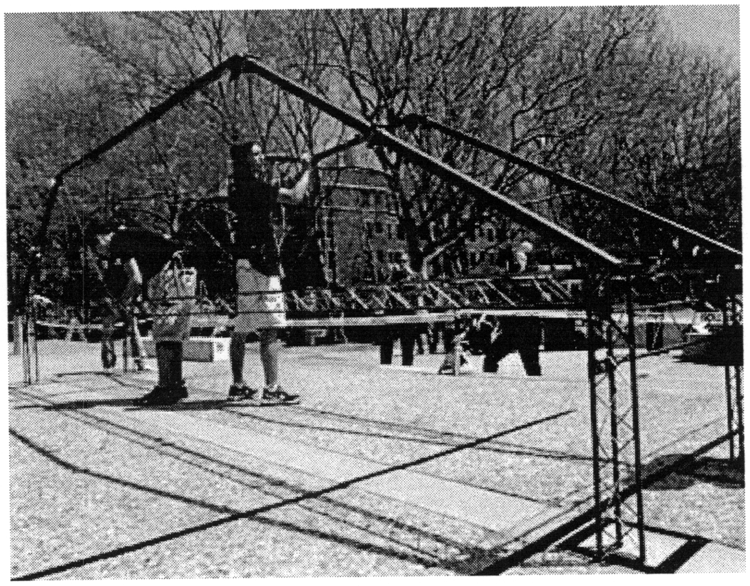


Figure 4.1: MIT Steel Bridge 2007-08

The problem posed by the *Student Steel Bridge Competition* is a good example for optimization. It is of practical use, as a structure is to be built and presented at the contest. But unlike real-world projects, the problem to be solved in the context of the competition is very well-defined through a set of rules.

4.1.2 Rules Summary

The *Student Steel Bridge Competition* is governed by new rules every year. The main rules for the 2009 edition are summarized in this section. The detailed rules can be found in the *Student Steel Bridge Competition 2009 Rules* (ASCE/AISC, 2008).

Overall Design

The bridge must fit in the bounding box represented of figure 4.2. Two deck supports must be provided over the full length of the bridge, but no actual deck is required. Pieces of deck will be installed and loaded for the deflection test. The entire structure must stand below the deck supports and rest on four legs.

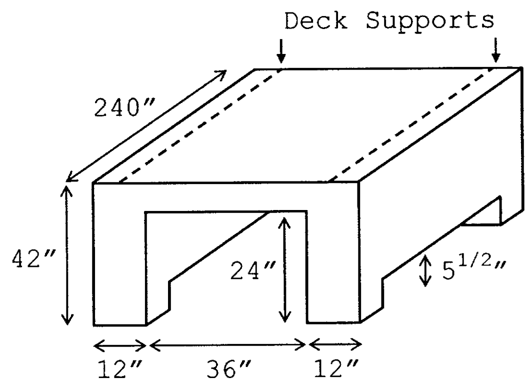


Figure 4.2: Steel Bridge Bounding Box

Fabrication

The pieces constituting the bridge are fabricated by the teams before the competition. The main constraint is that every piece must fit within its own prismatic bounding box of 42" x 6" x 6". All pieces must be made exclusively from steel and be rigid. Pieces shall be made as light as possible, as the weight of the bridge is taken into account in the evaluation.

Assembly

During the contest, the bridge is assembled from the prefabricated pieces initially placed in a staging area next to the construction site. No pre-assembly is allowed, meaning that every piece must be individually added to the bridge. Every connection must be fastened with a bolt. The assembly process is subject to many rules defining where the builders can operate and what they can do. Construction speed forms part of the evaluation of the bridge.

Load Test

Two pieces of deck loaded with 1250 lb each are successively placed on the supports. The locations of the loads, measured from the left extremity of the bridge, are determined by two dice rolls r_1 and r_2.

$$\begin{aligned} &\text{Location of First Load} &:\ & x_1 = 40 + 6\,r_1 \text{ (inches)} \\ &\text{Location of Second Load} &:\ & x_2 = 110 + 6\,r_2 \text{ (inches)} \end{aligned}$$

As a result, 36 load cases are possible. Three deflections are measured during the test to evaluate the stiffness of the bridge:

$d_{1\mathrm{a}}$	:	Deflection of one side of the bridge at x_1 due to first load
$d_{1\mathrm{b}}$	:	Deflection of other side of the bridge at x_1 due to first load
d_2	:	Deflection of either side of the bridge at x_2 due to both loads

The aggregate deflection is defined as $d = d_{1\mathrm{a}} + d_{1\mathrm{b}} + d_2$

Bridge Evaluation

The construction cost C_c is defined as:

$$\left.\begin{aligned} n &= \text{number of builders} \\ T &= \text{construction time (min)} \end{aligned}\right\} \longrightarrow \text{Construction Cost} \quad C_\mathrm{c} = nT$$

The structural cost C_s is defined as:

$$\left.\begin{aligned} w &= \text{bridge weight (lb)} \\ d &= \text{aggregate deflection (in)} \end{aligned}\right\} \longrightarrow \text{Structural Cost} \quad C_\mathrm{s} = 5w + 400d$$

The total cost C_t of the bridge is defined as $C_\mathrm{t} = C_\mathrm{c} + C_\mathrm{s}$

The bridge with the lowest total cost wins the competition. In 2008 the winning bridge (UC Berkeley) was assembled in 3 min 38 sec, weighted 142 lb and had an aggregate deflection of 0.36 in (the rules were different).

4.1.3 Optimization Problem

The design problem posed by the *Student Steel Bridge Competition* is challenging since lightness, stiffness and construction speed are often conflicting goals in structural engineering. Important questions arise in the conceptual design phase of the project. Does a light weight compensate an increased deflection? Should the structure rather be stiff but heavy? Or is construction speed the key to winning the competition? The answer certainly lies somewhere in between.

It is not easy to intuitively balance weight, stiffness and constructability when designing the bridge, and the experience of the previous editions is of limited help since the rules change every year. Regarding the balance between lightness and stiffness, the structural cost formula ($C_s = 5w + 400d$) means that an inch of aggregate deflection is worth 80 pounds of steel. This is, however, not sufficient to quickly size the bridge, as the deflection is composite and depends on the load case.

A more systematic approach is proposed. The gradient-based algorithm previously used to optimize trusses (chapter 2) and beams (chapter 3) is now applied to the steel bridge design problem. Like real-world projects, the steel bridge problem is multi-objective, whereas the gradient-based algorithm requires a single objective function. For beam and truss optimization, the strategy was to choose a single objective to be minimized and to express the other goals as constraints to be satisfied. In the case of the steel bridge, a trade-off between weight, performance and constructability is explicitly given through the definition of the cost, which can therefore be used as the unique objective function.

4.2 Girder Optimization Model

4.2.1 Girder Concept Selection

The gradient-based optimization tool considered in this study is helpful to improve structures but cannot design from scratch. A conceptual bridge design must be defined first, and only then can optimization be used to adjust the overall geometry and size the constituting elements. Girders and trusses (figure 4.3) are two structural systems that could support the loads and fit in the bounding box imposed by the rules (see section 4.1.2).

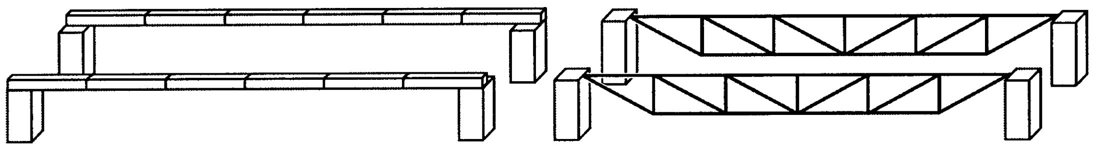


Figure 4.3: Girder and Truss Concepts

The choice of the structural system was affected by the upcoming optimization process. Weight, performance and constructability are taken into account in a single cost function to determine the winning bridge. This cost would therefore be naturally selected as the objective function to optimize the bridge. When implementing optimization, the weight and stiffness of a particular bridge design can be accurately calculated using structural analysis tools. But the construction time is more complicated to estimate, since it depends on the construction sequence and can be reduced by practicing. The assembly time of a piece could be estimated based on its weight or the number of bolts to be fasten, but such models would rely on too many assumptions to be relevant in optimization. It was therefore decided not to take construction speed into account in the optimization process. Constructability considerations will determine the choice of the bridge concept, and then will the selected structure be optimized for lightness and stiffness only.

The constitutive pieces being limited to 42" in length, 6 segments are necessary to build a girder (see figure 4.3). A truss would need many more pieces, increasing construction time dramatically. It was therefore decided to design a girder bridge.

4.2.2 Girder Overall Design

The girder bridge consists of two identical girders placed parallel and working as flexural members. Both girders are assembled from 6 pre-fabricated pieces that must fit in a 42" x 6" x 6" bounding box (figure 4.4). The girders are supported at both ends by a leg system to be designed separately. Since the contact area with the ground is limited and the legs cannot be anchored, the supports are not restrained in rotation. The girder is therefore assumed to be pin-supported.

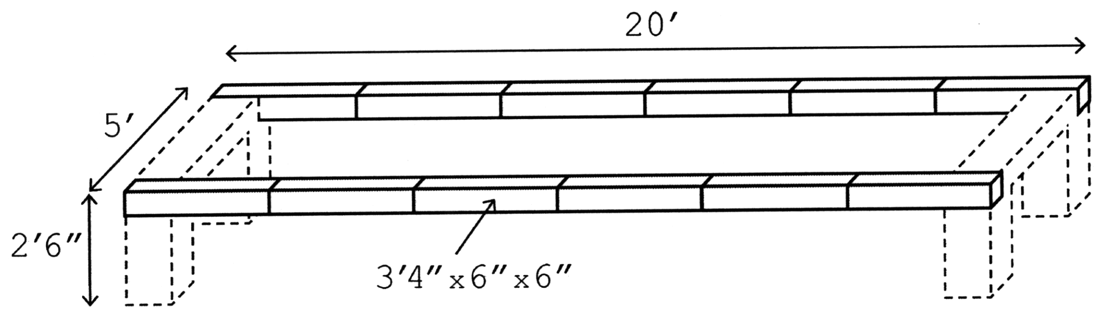


Figure 4.4: Girder Pieces Bounding Boxes

Each girder can be seen as a beam whose cross-section needs to be designed and sized. The section can vary over the length of the girder but is constant over each segment to simplify fabrication. In the following, 6 types of cross-section (figure 4.5) are considered.

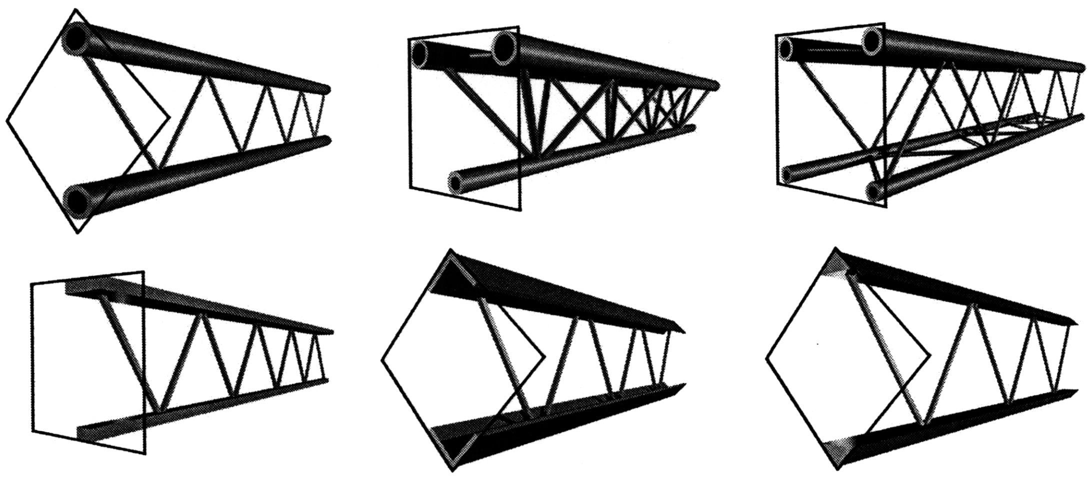


Figure 4.5: Girder Section Options, with 6"x6" Limits

4.2.3 Girder Cross-Sections Parametrization

It is proposed to optimize the girder for each potential type of section in order to select the geometry leading to the lowest structural cost. Each cross-section is parametrized, so that its properties can be varied during the optimization process.

Two-Pipe Section

In an attempt to maximize the distance between the tension and compression chords, a girder section made of two pipes placed on the diagonal of the bounding box is considered.

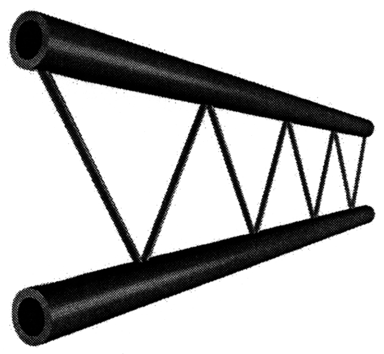


Figure 4.6: Section Rendering

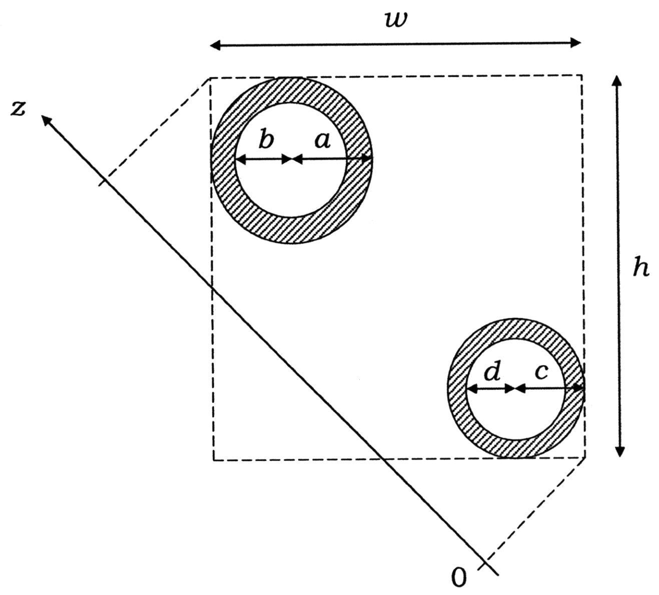


Figure 4.7: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_n and the moment of inertia I of the section are given by the following equations:

$$A = A_\mathrm{T} + A_\mathrm{B} \qquad \text{where} \quad A_\mathrm{T} = \pi(a^2 - b^2) \quad \text{and} \quad A_\mathrm{B} = \pi(c^2 - d^2)$$

$$z_\mathrm{n} = \sqrt{2}\left(\frac{A_\mathrm{T}\,(h-a) + A_\mathrm{B}\,c}{A_\mathrm{T} + A_\mathrm{B}}\right)$$

$$I = \frac{\pi}{4}(a^4 - b^4) + A_\mathrm{T}\left(\sqrt{2}(h-a) - z_\mathrm{n}\right)^2 + \frac{\pi}{4}(c^4 - d^4) + A_\mathrm{B}\left(z_\mathrm{n} - \sqrt{2}\,c\right)^2$$

This type of section has a low lateral stiffness.

Three-Pipe Section

Lateral stiffness can be increased by using a triangular shape made of three pipes. A compression chord made of two pipes is also less subjected to buckling.

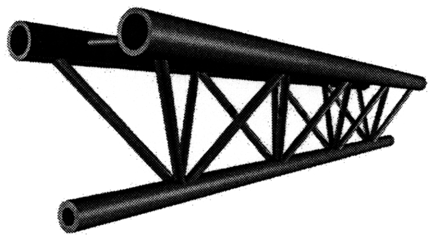


Figure 4.8: Section Rendering

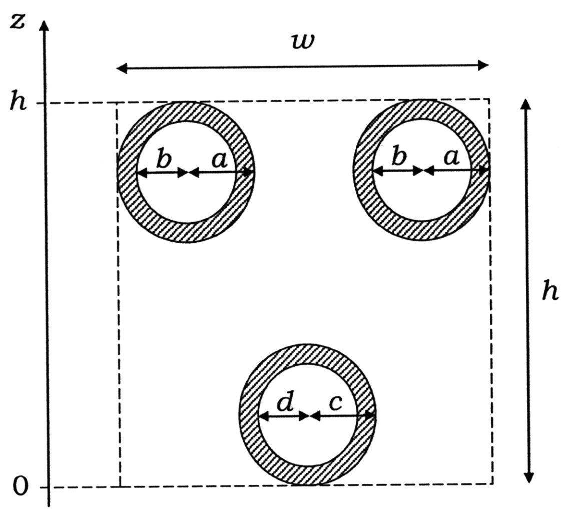


Figure 4.9: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_n and the moment of inertia I of the section are given by the following equations:

$$A = 2A_\mathrm{T} + A_\mathrm{B}$$
$$\text{where} \quad A_\mathrm{T} = \pi(a^2 - b^2) \quad \text{and} \quad A_\mathrm{B} = \pi(c^2 - d^2)$$

$$z_\mathrm{n} = \frac{2A_\mathrm{T}(h-a) + A_\mathrm{B}\,c}{2A_\mathrm{T} + A_\mathrm{B}}$$

$$I = \frac{\pi}{2}(a^4 - b^4) + 2A_\mathrm{T}(h - a - z_\mathrm{n})^2 + \frac{\pi}{4}(c^4 - d^4) + A_\mathrm{B}(z_\mathrm{n} - c)^2$$

Four-Pipe Section

A section made of four pipes has high lateral and torsional stiffness.

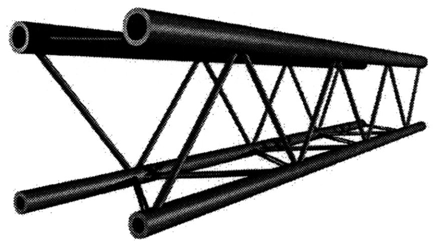


Figure 4.10: Section Rendering

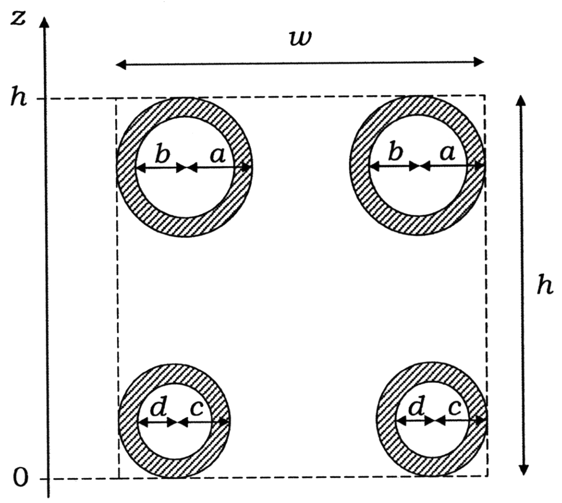


Figure 4.11: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_n and the moment of inertia I of the section are given by the following equations:

$$A = 2A_\mathrm{T} + 2A_\mathrm{B}$$
$$\text{where} \quad A_\mathrm{T} = \pi(a^2 - b^2) \quad \text{and} \quad A_\mathrm{B} = \pi(c^2 - d^2)$$

$$z_\mathrm{n} = \frac{2A_\mathrm{T}(h-a) + 2A_\mathrm{B}\,c}{2A_\mathrm{T} + 2A_\mathrm{B}}$$

$$I = \frac{\pi}{2}(a^4 - b^4) + 2A_\mathrm{T}(h - a - z_\mathrm{n})^2 + \frac{\pi}{2}(c^4 - d^4) + 2A_\mathrm{B}(z_\mathrm{n} - c)^2$$

For all the sections made from pipes, the moment of inertia is maximized by setting the values of all inner radiuses to zero, changing the pipes into rods. However, the inner radiuses are kept as potential optimization variables so that pipes can still be used if local buckling of the girder chords turns out to be a critical issue.

I-Shape Section

A girder section similar to an I-beam shape can be built, using steel plates as flanges.

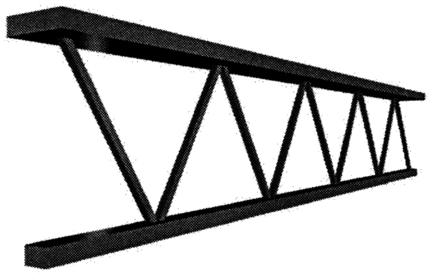


Figure 4.12: Section Rendering

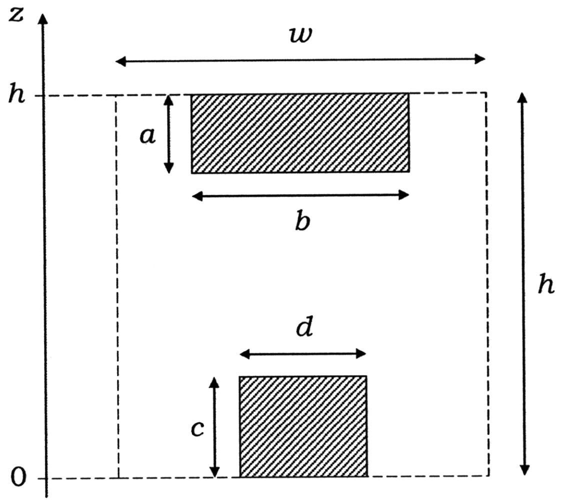


Figure 4.13: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_{n} and the moment of inertia I of the section are given by the following equations:

$$A = ab + cd$$

$$z_{\mathrm{n}} = \frac{ab\left(h - \frac{a}{2}\right) + cd\left(\frac{c}{2}\right)}{ab + cd}$$

$$I = \frac{ba^3}{12} + ab\left(h - z_{\mathrm{n}} - \frac{a}{2}\right)^2 + \frac{dc^3}{12} + cd\left(z_{\mathrm{n}} - \frac{c}{2}\right)^2$$

Using all the available width by setting $b = d = w$ would maximize the moment of inertia for a given amount of steel but could lead to very thin flanges, subject to local buckling in compression.

Two-L-Shape Section

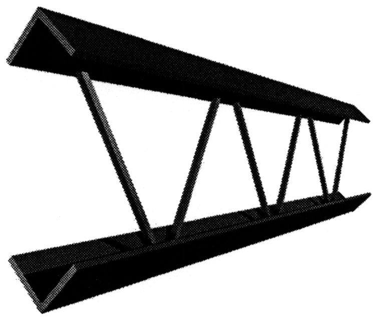


Figure 4.14: Section Rendering

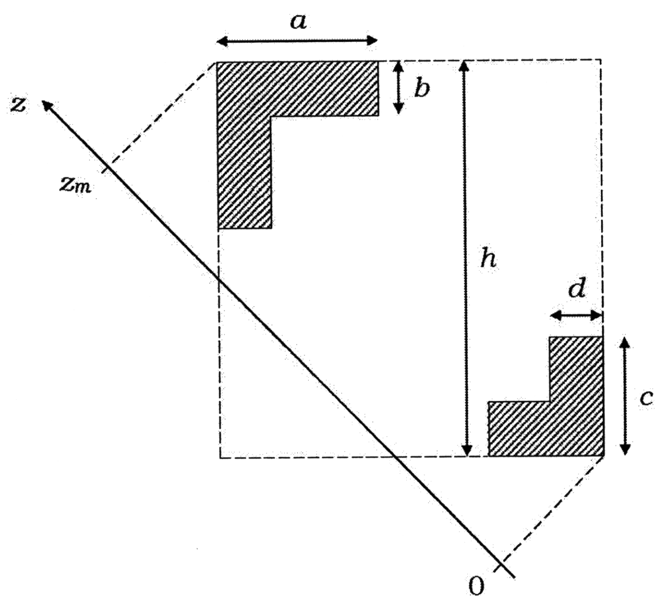


Figure 4.15: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_{n} and the moment of inertia I of the section are given by the following equations:

$$s_1 = a+b \qquad s_2 = a-b \qquad s_3 = s_4 = b \qquad s_5 = c+d \qquad s_6 = c-d \qquad s_7 = s_8 = d$$

$$b_{\mathrm{i}} = s_{\mathrm{i}}\sqrt{2} \qquad h_{\mathrm{i}} = \frac{s_{\mathrm{i}}}{\sqrt{2}} \qquad A_{\mathrm{i}} = \frac{b_{\mathrm{i}}h_{\mathrm{i}}}{2} \qquad I_{\mathrm{i}} = \frac{b_{\mathrm{i}}h_{\mathrm{i}}^3}{36} \qquad z_{\mathrm{i}} = \begin{cases} z_{\mathrm{m}} - h_1 + \dfrac{h_{\mathrm{i}}}{3} & \mathrm{i} = 1\ldots 4 \\ h_4 - \dfrac{h_{\mathrm{i}}}{3} & \mathrm{i} = 5\ldots 8 \end{cases}$$

$$A = A_1 - \sum_{\mathrm{i}=2}^{4} A_{\mathrm{i}} + A_5 - \sum_{\mathrm{i}=6}^{8} A_{\mathrm{i}}$$

$$z_{\mathrm{n}} = \frac{z_1 A_1 - \sum_{\mathrm{i}=2}^{4}(z_{\mathrm{i}}A_{\mathrm{i}}) + z_5 A_5 - \sum_{\mathrm{i}=6}^{8}(z_{\mathrm{i}}A_{\mathrm{i}})}{A}$$

$$\begin{aligned} I &= I_1 + A_1(z_1 - z_{\mathrm{n}})^2 - \sum_{\mathrm{i}=2}^{4}\left(I_{\mathrm{i}} + A_{\mathrm{i}}(z_{\mathrm{i}} - z_{\mathrm{n}})^2\right) \\ &+ I_5 + A_5(z_{\mathrm{n}} - z_5)^2 - \sum_{\mathrm{i}=6}^{8}\left(I_{\mathrm{i}} + A_{\mathrm{i}}(z_{\mathrm{n}} - z_{\mathrm{i}})^2\right) \qquad (4.1) \end{aligned}$$

Two-Triangle Section

Considering the bounding box, a girder section made of two solid triangular rods has the highest possible stiffness-to-weight ratio. The material is pushed as far as possible from the neutral axis of the section.

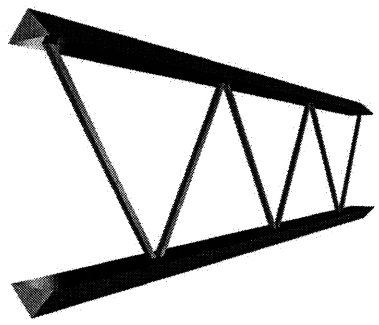


Figure 4.16: Section Rendering

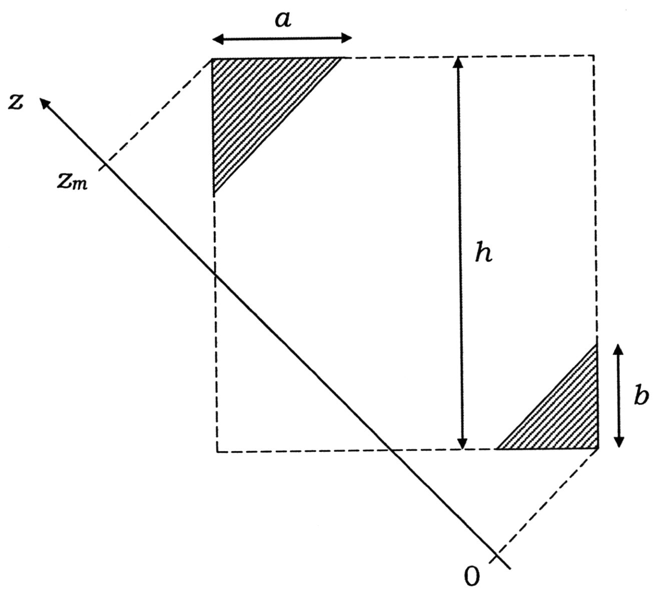


Figure 4.17: Section Parametrization

The cross-sectional area A, the z-coordinate of the neutral axis z_{n} and the moment of inertia I of the section are given by the following equations:

$$A_{\mathrm{T}} = \frac{a^2}{2} \qquad z_{\mathrm{T}} = z_{\mathrm{m}} - \frac{\sqrt{2}}{3}a \qquad A_{\mathrm{B}} = \frac{b^2}{2} \qquad z_{\mathrm{B}} = \frac{\sqrt{2}}{3}b$$

$$A = A_{\mathrm{T}} + A_{\mathrm{B}}$$

$$z_{\mathrm{n}} = \frac{A_{\mathrm{T}}\, z_{\mathrm{T}} + A_{\mathrm{B}}\, z_{\mathrm{B}}}{A_{\mathrm{T}} + A_{\mathrm{B}}}$$

$$I = \frac{a^4}{72} + A_{\mathrm{T}}(z_{\mathrm{T}} - z_{\mathrm{n}})^2 + \frac{b^4}{72} + A_{\mathrm{B}}(z_{\mathrm{n}} - z_{\mathrm{B}})^2$$

4.3 Girder Optimization Program

The girder optimization was implemented in Matlab®. A built-in optimization toolbox was used to run the gradient-based algorithm, and a program was developed to define and analyze girders in interaction with the optimization tool. The strategy for implementing structural optimization on computers was introduced in section 1.2.1 and more details on the Matlab® optimization tool can be found in the *Optimization Toolbox™ 3 User's Guide* (The MathWorks™, 2007b).

4.3.1 Program Overview

The shallow girder considered in this study behaves as a beam. Flexural members can be analyzed by numerical moment integration or using matrix analysis, as presented in sections 3.2 and 3.3 respectively. The robust matrix analysis method was used in chapter 3 as beams of different support conditions were considered, including hyperstatic configurations. The girder to be optimized here is a determinate system, for which numerical moment integration is straightforward (see example section 3.2.4). Both analysis methods are therefore applicable.

A total of 36 load cases can occur on the day of the competition. At each iteration of the design, the girder needs to be solved for all possible load scenarios in order to determine the worst case deflection and to check the stresses. The speed of the analysis tool is therefore critical. Since numerical moment integration is faster for simple beams, a girder analysis program based on this method is used to interact with the gradient-based optimization algorithm. Since the bending moment acting on the girder depends on the load case but not on the girder design, the moments corresponding to every load cases are calculated once at the initialization of the process and stored in a database.

4.3.2 Schematic Diagram

Figure 4.18 (p.93) shows the group of Matlab® functions used to optimize the girder and the way they interact. Each rectangle represents a function, that is, a piece of code contained in a separate file. The arrows represent arguments being passed between functions. The layout it top-down, meaning that a function called within a function is represented below the function that calls it. More clarifications about these schematic diagrams can be found in appendix C.

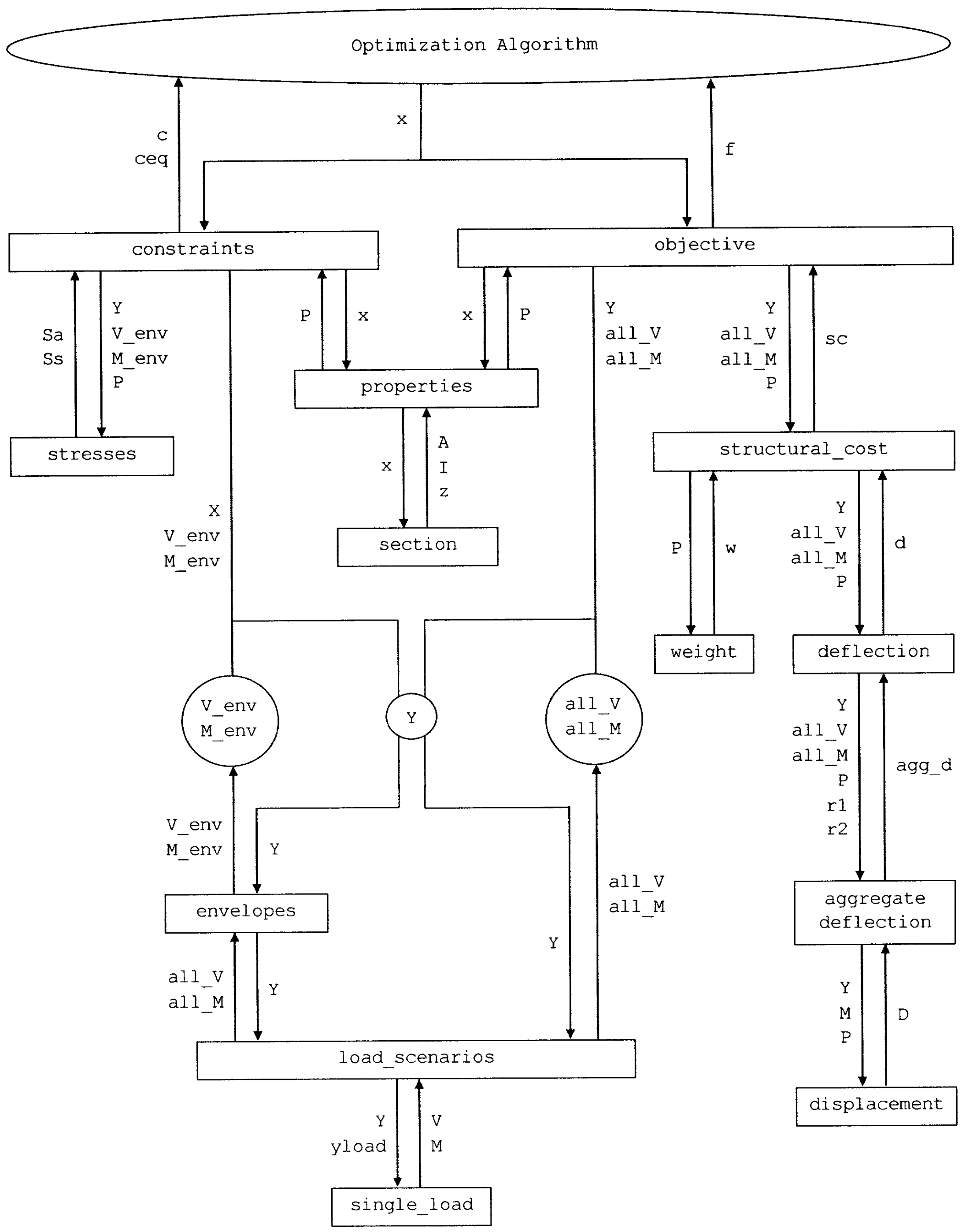


Figure 4.18: Girder Optimization Program Schematic Diagram

4.3.3 Functions Description

The operations carried out by the functions represented on the schematic diagram (figure 4.18 p.93) are described in this section. The code is available in appendix E.

Optimization Objective Function - `objective`

The objective function (`objective`) is required to use the Matlab® optimization toolbox (see figure 1.2 p.13). It is called by the optimization algorithm whenever it needs to evaluate the quality of a design scenario. The algorithm sends the current values of the set of optimization variables (`x`) to the objective function. The objective function calls to the girder properties function (`properties`) to transform these optimization variables (`x`) into a list of girder segments (`P`) desribing the properties of each segment. The current design of the girder is fully described by this list (`P`) throughout the upcoming analysis process. Then, the objective function loads from the database the list of integration points (`Y`) and the corresponding values of the shear force and bending moment for all possible load cases (`all_V`, `all_M`). All the data, along with the girder properties data (`P`), are transmitted to the analysis function (`structural_cost`), which returns the structural cost (`sc`) of the current girder design. This value (`sc`) is then returned to the optimization algorithm as the design value (`f`).

Girder Properties Function - `properties`

The role of this function is to interpret the optimization variables (`x`) and to rewrite them in a format (`P`) describing the current design of the girder and that can be conveniently used in the analysis process. This function is edited by the user to define the girder to be optimized and to assign the optimization variables. The translation of the optimization variables into girder properties is made by calling functions (`section`) that calculate the girder section properties.

Section Properties Calculation Function - `section`

Each function of the set (`section1`, `section2` ...) calculates the cross-sectional properties of a given type of girder section. The cross-sectional area (`A`), the moment of inertia (`I`) and the maximum distance to the neutral axis (`z`) are returned.

Structural Cost Calculation Function - `structural_cost`

In addition to the general data (`Y`, `all_V`, `all_M`), this function accepts the girder properties data (`P`) as argument, and its role is to calculate the structural cost (`sc`) of that particular girder, as defined by the rules. First, the weight (`w`) of the girder is obtained by sending the girder properties (`P`) to the weight calculation function (`weight`). The girders properties (`P`) are then sent to the deflection calculation function (`deflection`) along with the general data (`Y`, `all_V`, `all_M`), which returns the deflection (`d`). The weight (`w`) and the deflection (`d`) are combined to calculate the structural cost (`sc`), which is returned.

Weight Calculation Function - `weight`

This function uses the cross-sectional areas contained the girder properties data (`P`) to calculate the weight of the girder (`w`).

Worst Case Deflection Calculation Function - `deflection`

This function does not directly calculate the girder deflection but is used to find the worst-case aggregate deflection, whose value is to be used to calculate the structural cost. For each possible load combination, defined by the dice roll values (`r1`, `r2`), the general data (`Y`, `all_V`, `all_M`) and the girder properties (`P`) are transmitted to the actual lower-level deflection calculation function (`aggregate_deflection`), which returns the aggregate deflection (`agg_d`) for that load combination. The maximum aggregate deflection value (`p`) is returned to the structural cost calculation function (`structural_cost`).

Aggregate Deflection Calculation Function - `aggregate_deflection`

This function simulates the two steps of the loading process for a given load combination, specified by the dice roll values (`r1`, `r2`) received as arguments, along with the general data (`Y`, `all_V`, `all_M`) and the girder properties (`P`). The numerical integration function (`displacement`) is called twice, the bending moment to be integrated (`M`) being the one due to the first load and then the one due to both loads. These bending moments are retrieved from the bending moment data (`all_M`) using the dice roll values (`r1`, `r2`). Both times, the displacements values of the girder (`D`) corresponding to the integration points (`Y`) are returned, and the deflections at the points of interests are saved. Using the deflection values from both steps of the loading process, the aggregate deflection (`agg_d`) is then calculated and returned.

Numerical Integration Function - `displacement`

This function carries out the numerical integration of the bending moment over the girder to calculate its deformation. Computer implementation of the numerical integration is presented in section 3.2. The bending moment (`M`) and the girder properties (`P`) are used to assign a bending moment and a bending rigidity to each integration point of the list (`Y`). Two successive integrations are then carried out, and the resulting deformation values at each integration point (`D`) are returned.

Optimization Constraints Function - `constraints`

The constraints function (`constraints`) is required to use the Matlab® optimization toolbox (see figure 1.2 p.13). It is called by the optimization algorithm whenever it needs to check whether a design scenario is acceptable or not. The function receives the values of the optimization variables (`x`) from the optimization algorithm and returns two series of numbers (`c, ceq`) calculated from the optimization variables. The design scenario is acceptable if all numbers in the first list (`c`) are negative and all numbers in the second list (`ceq`) are zero. This function is edited by the user to set the boundaries of the optimization problem, that is, the minimum and maximum values of the optimization variables. The girder is optimized for stiffness and only checked for strength, the maximum allowable strength becoming a constraint to the optimization problem. The girder properties function is called to get the girder properties (`P`), which are passed to the stress calculation function (`stresses`), along with the integration points (`Y`) and the shear and moment envelopes (`V_env`, `M_env`) loaded from the database. The axial (`Sa`) and shear (`Ss`) stresses distributions are returned. If one of the stresses exceeds the maximum allowable value, a positive term is added to the constraints list (`c`), making the design is not acceptable.

Stress Calculation Function - `stresses`

A cross-sectional area, a moment of inertia and a distance to neutral axis are assigned to each integration point (`Y`) using the girder properties (`P`). Then, the axial stress (`Sa`) due to the bending moment envelope (`M_env`) and the shear stress (`Ss`) due to the shear force envelope (`S_env`), and returned.

Three additional programs (`single_load`, `load_scenarios`, `envelopes`) are used only once at the beginning of the optimization process to generate and store data to be used at every iteration.

4.4 Girder Optimization Results

The computer program described in section 4.3 was used to optimize the 6 different girder designs considered in this study. The optimization process is presented on the example of the girder made of two pipes, and results for the other 5 girder sections are given.

4.4.1 Optimized Two-Pipe Section Girder

The girder is assembled from 6 pre-fabricated segments, as shown on figure 4.19. Each segment is made of two pipes placed parallel about 8" from each other and linked by a series of diagonal webbing elements. The objective of the optimization process is to determine the size of every pipe in order to minimize the structural cost of the girder (balance between weight and stiffness, as defined by the rules of the competition). The webbing elements are not considered as design parameters in the optimization process and are to be sized separately. Working as a bending beam, the girder is assumed to have identical top and bottom chords. Both pipes of a segment are therefore of the same size. The girder is designed symmetrically, since no particular load direction is assumed (loading is asymmetric, but the side on which the first load is applied is not known). Considering all of the previous assumptions, only 3 pipe sizes need to be determined.

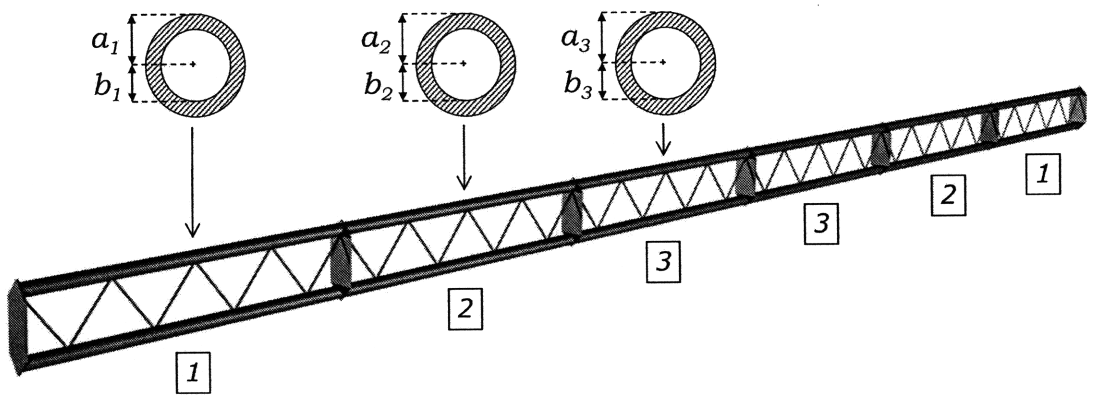


Figure 4.19: Two-Pipe Section Girder Parametrization

The girder segments are labeled with numbers, and segments with the same label are identical. The optimization variables a_i and b_i represent the outer and inner radiuses of the pipes constituting the segments of type i.

The 6 variables $(a_1, b_1, a_2, b_2, a_3, b_3)$ parametrizing the girder design are given to the gradient-based algorithm. Initial values of 1/2" and 1/4" are respectively assigned to the outer and inner radiuses of all pipes.

The algorithm is then run. At each iteration, the gradient-based algorithm updates the values of the inner and outer radiuses. The aggregate deflection of each new girder design is evaluated for the different load cases. The worst-case deflection is combined with the weight of the bridge to calculate the structural cost. The next iterations take into account the results from the previous ones to adjust the optimization variables. The algorithm stops when a pre-defined optimality criterion is met.

The convergence diagrams of the optimization process are shown on figure 4.20. The upper graph represents the evolution of the optimization variables, and the lower diagram shows the decrease of the objective function (structural cost S_c).

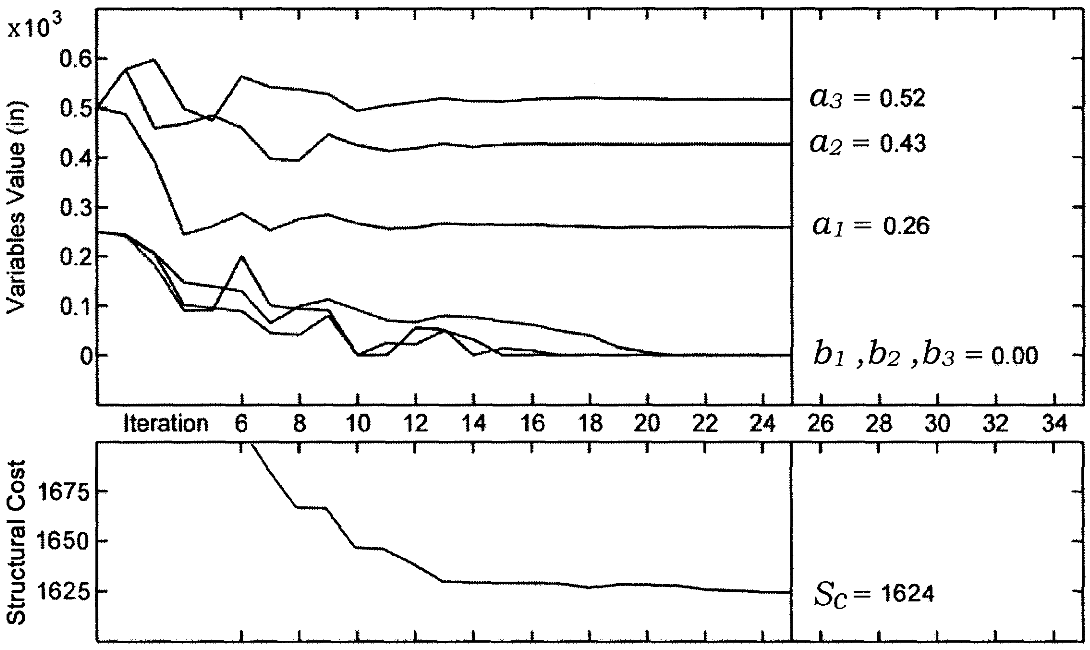


Figure 4.20: Algorithm Convergence Diagrams

The outer radiuses quickly reach the range of their respective final values. The inner radiuses evolve more steadily and slowly, since a change in inner radius has less effect on the pipe cross-sectional area than a change in outer radius.

Hollow pipes are changed into solid rods since all inner radiuses converge towards zero, meaning that buckling does not govern the design of girder chords.

The cross-sections of the three types of girder segments (labeled as 1, 2 and 3 on figure 4.19) are represented to scale on figure 4.21. Rods diameters are indicated in inches. As expected from the bending moment diagram, the optimized girder is heavier at mid-span.

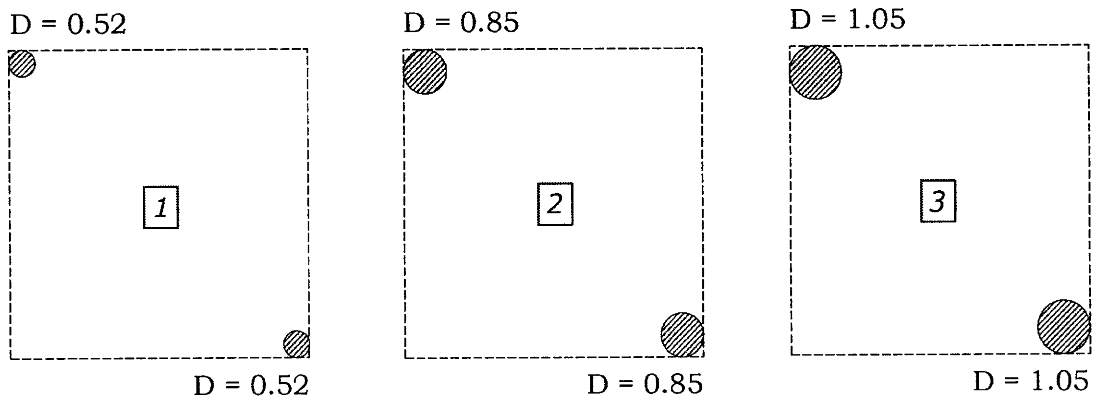


Figure 4.21: Optimized Two-Pipe Sections

If this optimized girder was selected for the actual bridge, further adjustment would be necessary. Since steel rods and are available in standard sizes, the girder chords diameters would have to be slightly modified. Another option would be to use pipes instead of rods, selecting standard sizes for the inner and outer radiuses to find members whose cross-sectional areas match the areas of the rods in the optimal design. Rods are optimal since they maximize the moment of inertia of the girder section by placing the material as far as possible in the corners of the square bounding box. However, using small pipes instead of rods would have a negligible effect on the bending capacity of the girder.

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	147 lbs
aggregate deflection	1.27 in
structural cost	\$ 1,624,000

4.4.2 Optimized Three-Pipe Section Girder

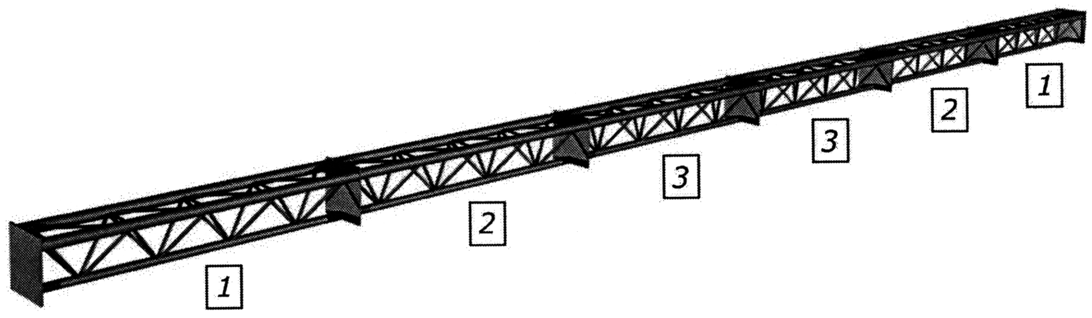


Figure 4.22: Girder made of Three-Pipe Sections

The cross-sections of the three types of girder segment, labeled as [1, 2, 3] on figure 4.22, are represented to scale on figure 4.23. Diameters are indicated in inches.

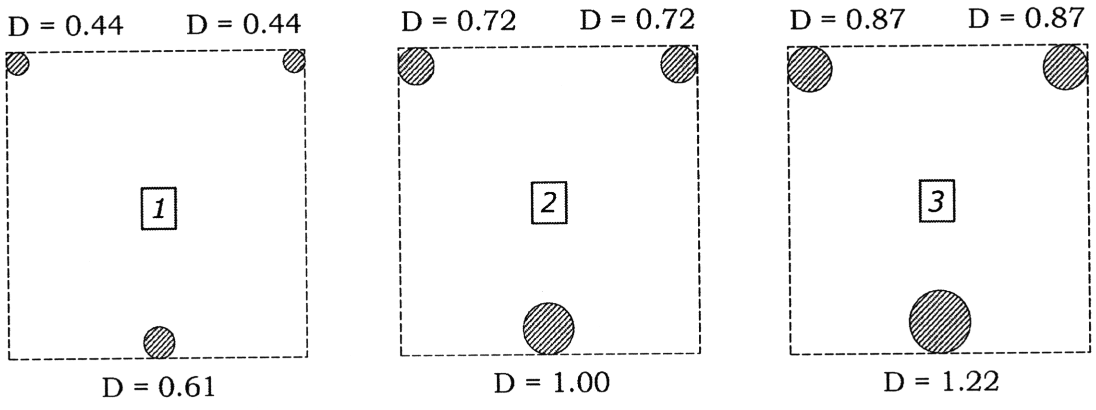


Figure 4.23: Optimized Three-Pipe Sections

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	208 lbs
aggregate deflection	1.80 in
structural cost	$ 2,300,000

The structural cost of this section is 40% higher than the structural cost of the two-pipe section presented in 4.4.1. The triangular cross-section is only 6"-deep and is therefore much less efficient than the two-pipe section, whose depth is 8"1/2.

4.4.3 Optimized Four-Pipe Section Girder

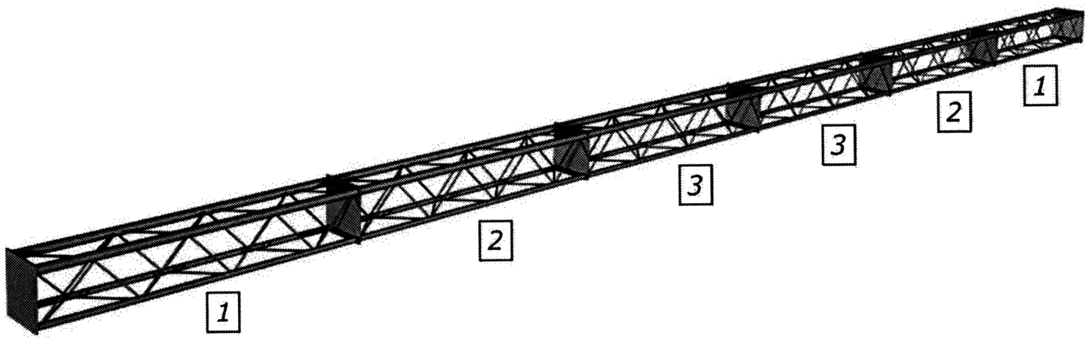


Figure 4.24: Girder made of Four-Pipe Sections

The cross-sections of the three types of girder segment, labeled as [1, 2, 3] on figure 4.24, are represented to scale on figure 4.25. Diameters are indicated in inches.

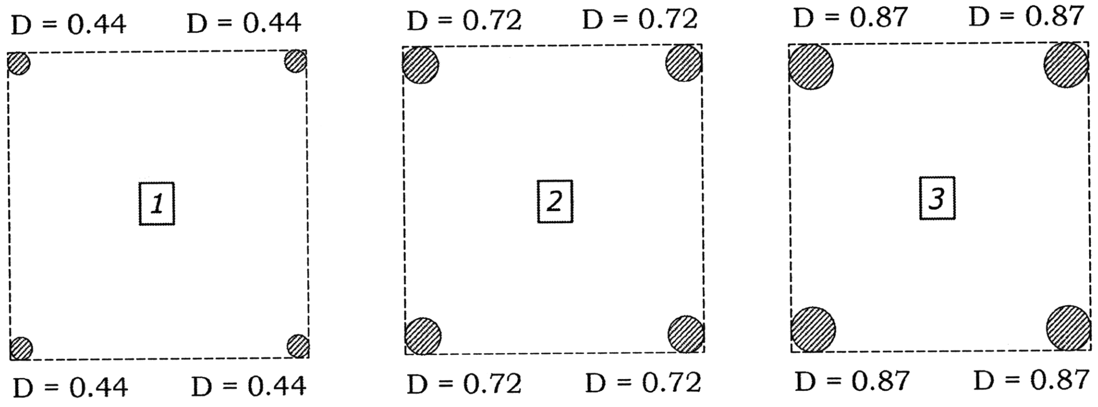


Figure 4.25: Optimized Four-Pipe Sections

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	206 lbs
aggregate deflection	1.71 in
structural cost	$ 2,227,000

4.4.4 Optimized I-Shape Section Girder

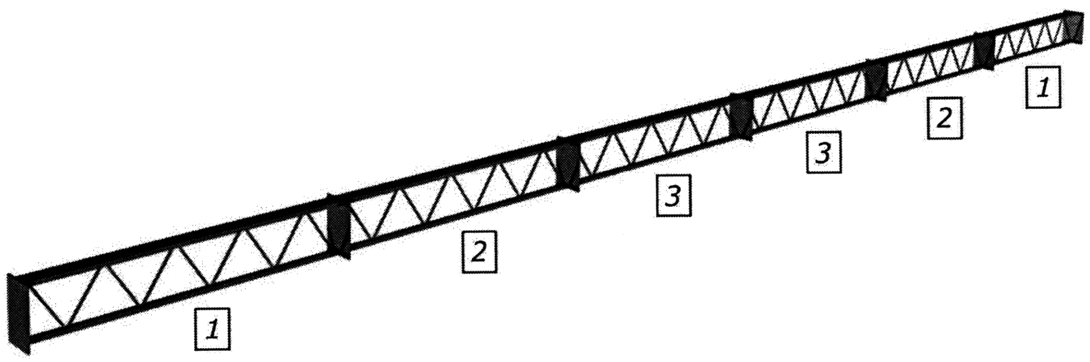


Figure 4.26: Girder Made of I-Shape Sections

The cross-sections of the three types of girder segment, labeled as [1, 2, 3] on figure 4.26, are represented to scale on figure 4.27. Diameters are indicated in inches.

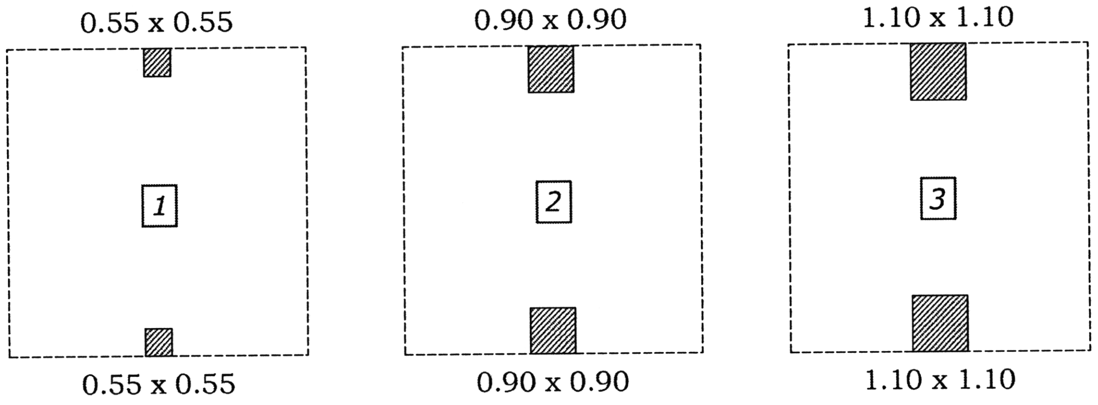


Figure 4.27: Optimized I-Shape Sections

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	210 lbs
aggregate deflection	1.79 in
structural cost	$ 2,303,000

4.4.5 Optimized Two-L-Shape Section Girder

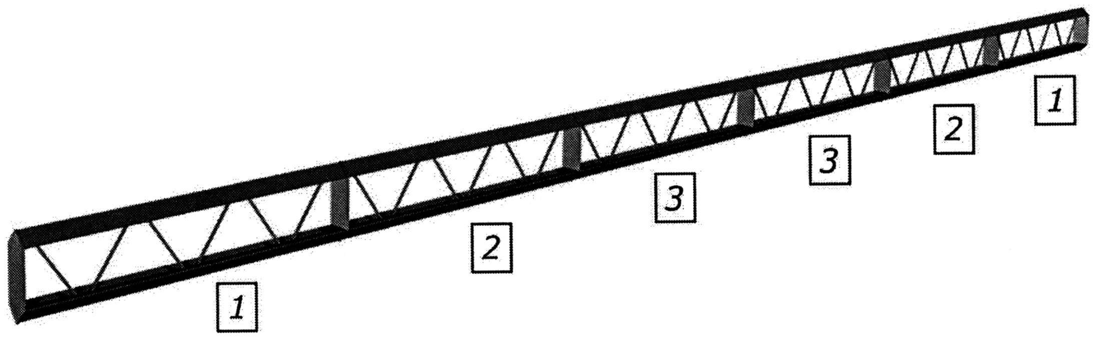


Figure 4.28: Girder made of Two-L-Shape Sections

The cross-sections of the three types of girder segment, labeled as [1, 2, 3] on figure 4.28, are represented to scale on figure 4.29. Diameters are indicated in inches.

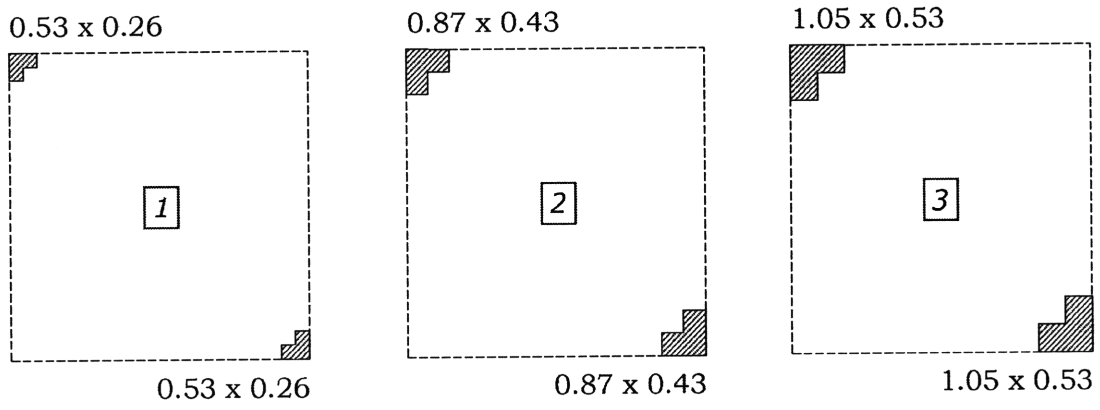


Figure 4.29: Optimized Two-L-Shape Sections

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	146 lbs
aggregate deflection	1.22 in
structural cost	$ 1,584,000

4.4.6 Optimized Two-Triangle Section Girder

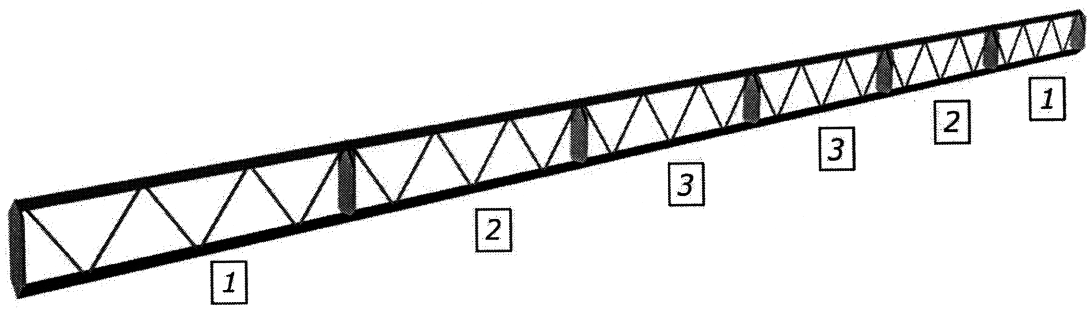


Figure 4.30: Girder made of Two-Triangle Sections

The cross-sections of the three types of girder segment, labeled as [1, 2, 3] on figure 4.30, are represented to scale on figure 4.31. Diameters are indicated in inches.

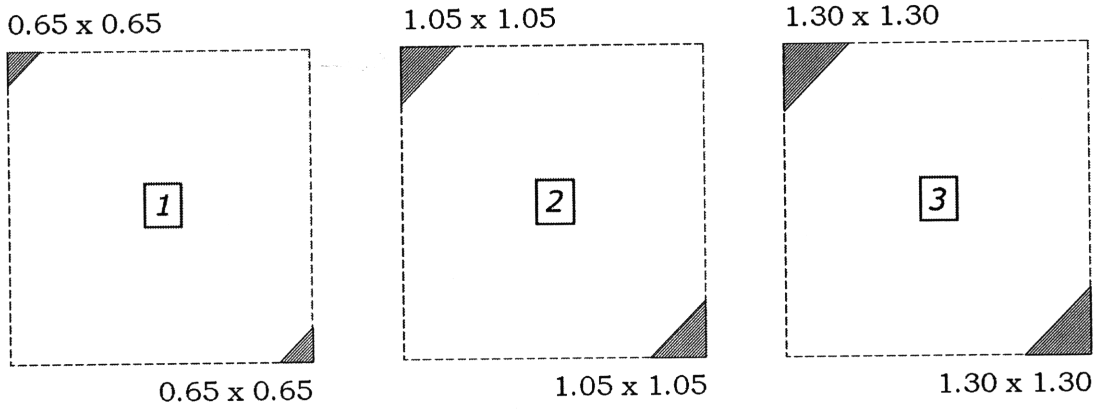


Figure 4.31: Optimized Two-Triangle Sections

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

weight	146 lbs
aggregate deflection	1.21 in
structural cost	$ 1,577,000

4.5 Convergence Analysis

In the following, the *convergence speed* refers to the number of iterations needed by the algorithm to reach the optimal design, whereas the *convergence time* is the actual duration of the optimization process.

Quick convergence towards the optimal solution is a critical issue when dealing with complex systems. Adjustments within the optimization algorithm and a relevant definition of the objective function can make the process faster. But the selection of the optimization variables and the initial value assigned to them also greatly influences the convergence speed. This section shows how selecting relevant optimization variables and assigning appropriate initial values can reduce the convergence time of a gradient-based algorithm. The steel girder presented and optimized in the previous sections is used as an example.

The convergence of the optimization variables towards their value in the optimal design scenario are represented on convergence diagrams. An example is shown on figure 4.32. The important aspect to look at is how a gradient-based algorithm handles a group of variables. What each of these variables represents does not really matter. Therefore, for simplicity and clarity of the diagrams, all variables are represented by black lines. The vertical line on each diagram represents the end of the iteration process. The process ends when a pre-defined convergence criterion is met. This convergence criteria is the same for all optimization processes discussed in this section.

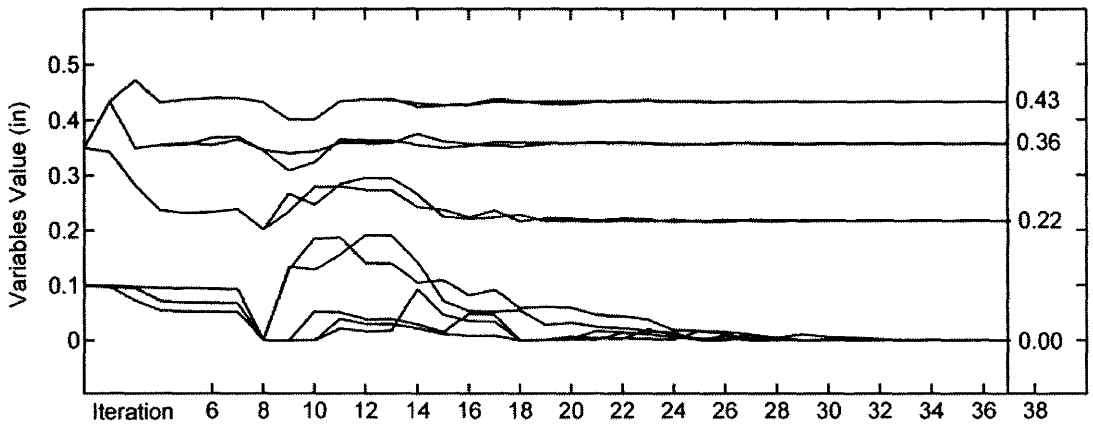


Figure 4.32: Example of Optimization Variables Convergence Diagram

4.5.1 Effect of the Number of Variables

The effect of the number of optimization variables on convergence speed is shown using the optimization of the girder made of two-pipe sections. The girder is made of 6 segments but is symmetrical with respect to the mid-span, so that only 3 segments are optimized. Each segment is made of two pipes, and each pipe is parametrized by its outer and inner radiuses. The resulting 12 optimization variables to be considered are represented on figure 4.33.

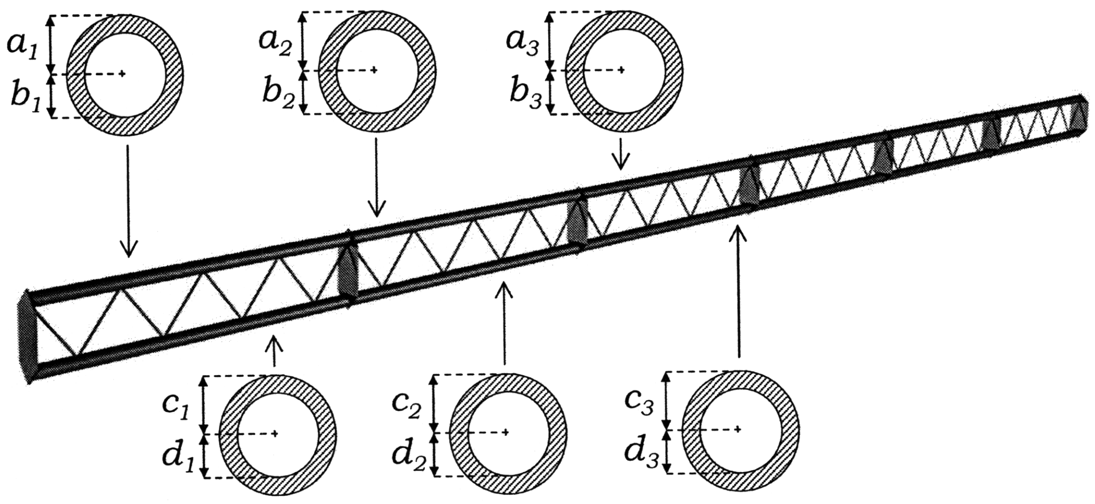


Figure 4.33: Two-Pipe Section Girder Optimization Variables

The optimal design values are (see section 4.4.1):

$$\begin{aligned} a_1 &= c_1 = 0.52\,\text{in} & b_1 &= d_1 = 0.00\,\text{in} \\ a_2 &= c_2 = 0.43\,\text{in} & b_2 &= d_2 = 0.00\,\text{in} \\ a_3 &= c_3 = 0.26\,\text{in} & b_3 &= d_3 = 0.00\,\text{in} \end{aligned}$$

Though the system was parametrized with 12 optimization variables, only 3 values of interest are to be found. Using simple engineering considerations, one can reduce the number of variables from 12 to 3. First, all pipe inner radiuses shall be zero, since the highest moment of inertia is obtained by pushing the material as far as possible from the centerline of the section and quick analysis shows that buckling is not an issue. Second, for each of the three girder segments, the top and bottom pipes (rods) have same dimensions, since this type of cross-section is most efficient when symmetrical with respect to the centerline. The following diagrams show how the duration of the optimization process is reduced by that kind of pre-analysis. In this girder optimization example, the duration of the optimization process is of the order of a few seconds in the worst case. But reducing the number of variables would be very useful when applied to more complex optimization problems, for which the gradient-based algorithm would have to run much longer.

Since the goal is to show the influence of the number of optimization variables only, the same initial values are assigned to these variables in the different cases considered. All inner radiuses start at 0.25 in and all outer radiuses start at 0.5 in.

All Optimization Variables

As no assumption is made, all of the 12 variables represented on figure 4.33 are given to the optimization algorithm. With the algorithm termination criteria considered in this study, convergence is achieved after 35 iterations.

Variables: a_1 b_1 c_1 d_1 a_2 b_2 c_2 d_2 a_3 b_3 c_3 d_3

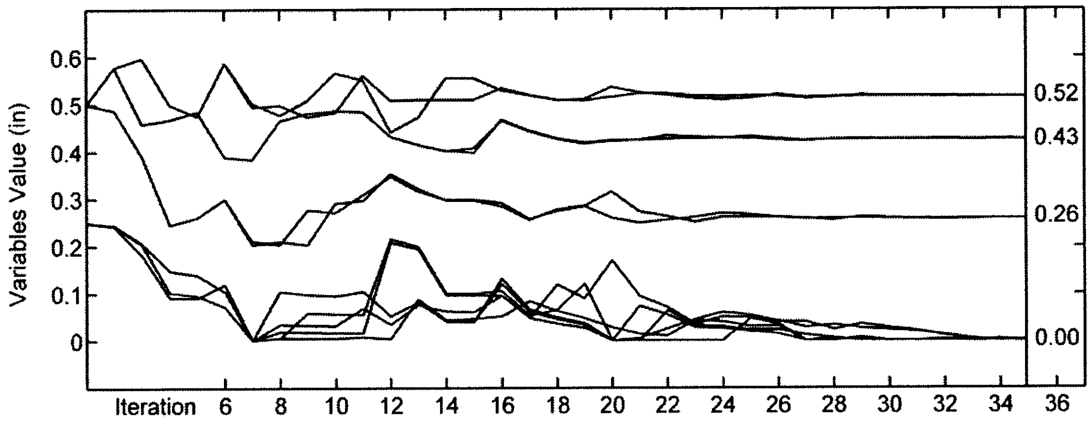


Figure 4.34: Convergence of 12 Optimization Variables

Symmetrical Sections

Each of the 3 girder segments is assumed to be made of a symmetrical section, that is, with identical top and bottom pipes. The number of optimization variables is down to 6. Convergence is achieved after 25 iterations.

Variables: a_1 b_1 a_2 b_2 a_3 b_3

Assumptions: $a_1 = c_1$ $a_2 = c_2$ $a_3 = c_3$ $b_1 = d_1$ $b_2 = d_2$ $b_3 = d_3$

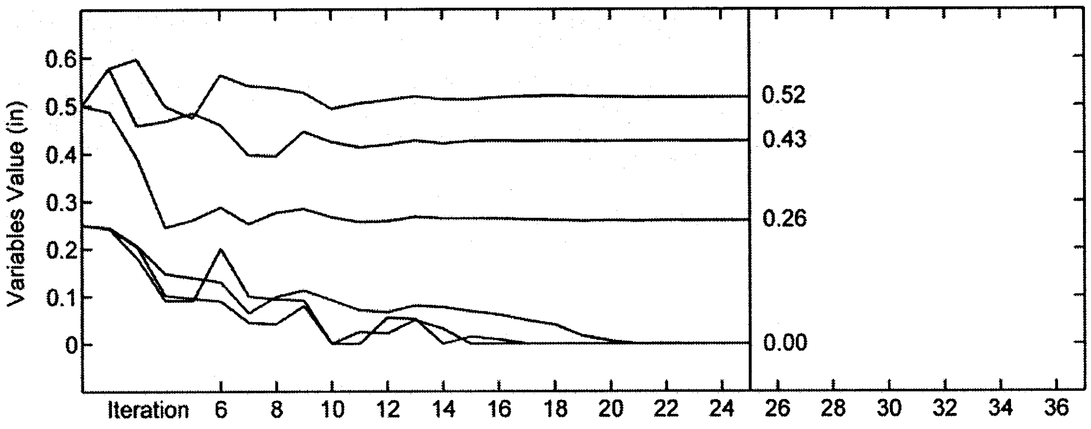


Figure 4.35: Convergence of 6 Optimization Variables

The group of 6 variables converge 40% faster than the 12 variables considered initially. The convergence time is even more reduced, as each iteration is faster when fewer variables are considered.

Zero Inner Radiuses

All pipes inner radiuses are set to zero, changing the pipes into rods. The remaining optimization variables are the 6 outer radiuses. Convergence is achieved after 15 iterations.

Variables: $a_1\ c_1\ a_2\ c_2\ a_3\ c_3$
Assumptions: $b_1 = d_1 = b_2 = d_2 = b_3 = d_3 = 0$

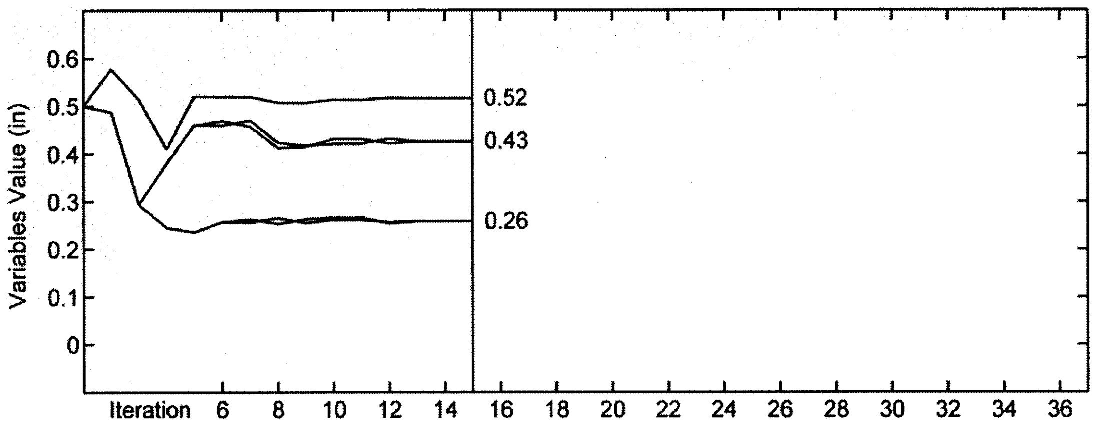


Figure 4.36: Convergence of 6 Optimization Variables

The same number of optimization variables was considered in the previous case. However, convergence is much quicker in the current case (15 vs. 25 iterations), showing that the number of variables is not the only factor affecting the convergence speed of the optimization process. The effect of other factors are presented in the following sections (4.5.2 and 4.5.3).

Symmetrical Sections and Zero Inner Radiuses

The two assumptions made separately in the previous convergence analyses are now made simultaneously. Only 3 optimization variables remain. Convergence is achieved after 13 iterations.

Variables: a_1 a_2 a_3
Relationships: $a_1 = c_1 \quad a_2 = c_2 \quad a_3 = c_3 \quad b_1 = d_1 = b_2 = d_2 = b_3 = d_3 = 0$

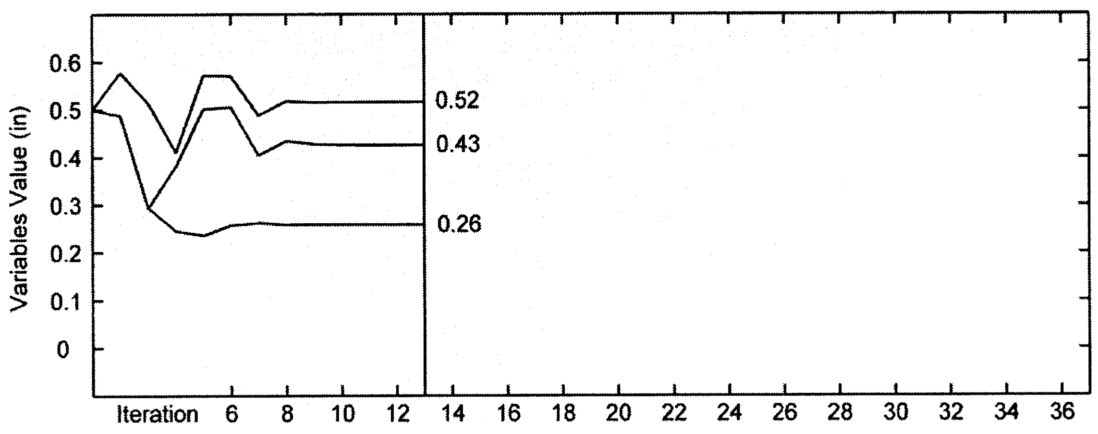


Figure 4.37: Convergence of 3 Optimization Variables

Summary

The number of variables parametrizing the system greatly affects the convergence speed of the gradient-base optimization algorithm. With less optimization variables, fewer iterations of the algorithm are necessary to reach the optimal design. The actual optimization time is even more reduced, as a smaller number of variables makes each iteration faster, due to the nature of the gradient-based algorithm. However, the number of variables is not the only parameter to significantly affect the convergence speed of the optimization process. Figure 4.35 shows a convergence 40% faster than figure 4.36, though the same number of variables was considered.

4.5.2 Effect of the Absolute Initial Values

Convergence speed is affected by the initial values assigned to the optimization variables. This effect is shown using the optimization of a girder made of L-shape sections. The girder is made of 6 segments but is symmetrical with respect to the mid-span, so that only 3 segments are to be optimized. Each segments is made of two L-shapes, each L-shape being parametrized by its side length and thickness. The resulting 12 optimization variables are represented on figure 4.38.

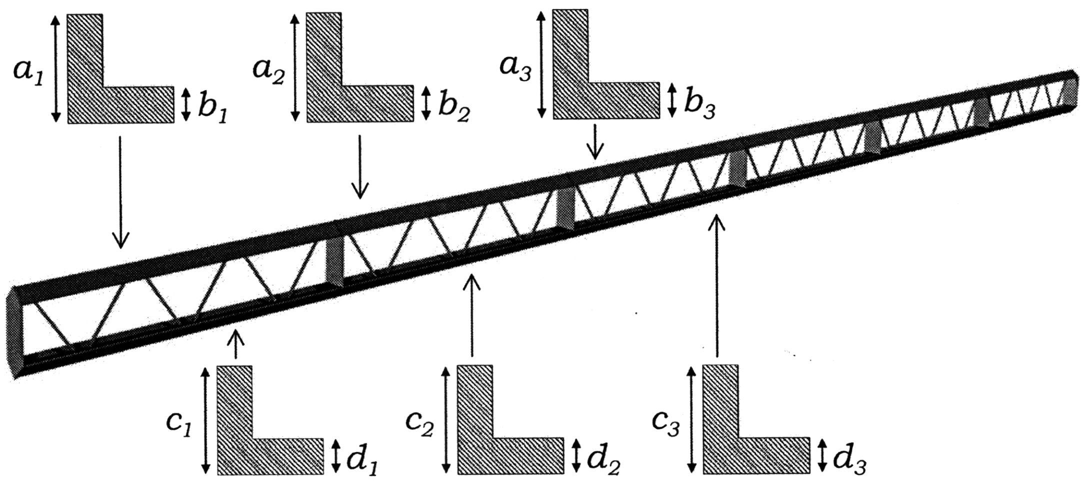


Figure 4.38: L-Shape Section Girder Optimization Variables

The optimal design values are:

$$
\begin{aligned}
a_1 = c_1 &= 0.53\,\text{in} & b_1 = d_1 &= 0.26\,\text{in}\\
a_2 = c_2 &= 0.87\,\text{in} & b_2 = d_2 &= 0.43\,\text{in}\\
a_3 = c_3 &= 1.05\,\text{in} & b_3 = d_3 &= 0.53\,\text{in}
\end{aligned}
$$

Knowing the final value reached by each variable in the optimal design, the optimization process was run with the initial values successively assigned with a 1%, 10%, 25%, 50% and 100% difference from final values. The number of iterations required before convergence are compared.

1% Difference

Side lengths start 1% longer than their final values, while thicknesses start 1% thinner. Convergence is achieved after 9 iterations.

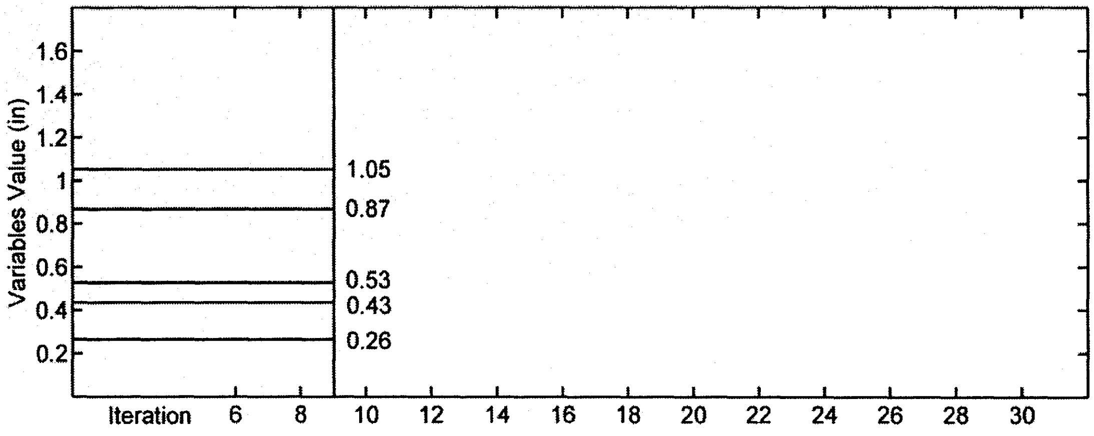


Figure 4.39: Initial Values 1% Different from Final Values

10% Difference

Side lengths start 10% longer than their final values, while thicknesses start 10% thinner. Convergence is achieved after 16 iterations.

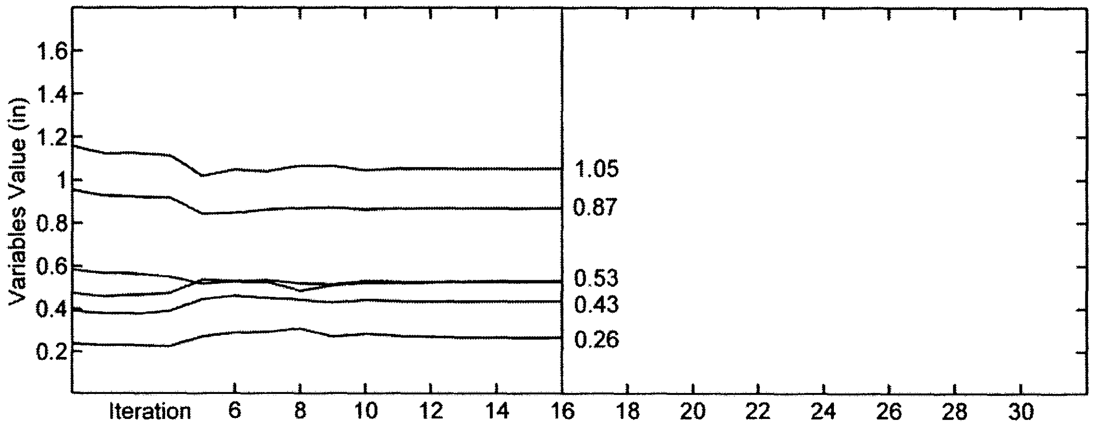


Figure 4.40: Initial Values 10% Different from Final Values

25% Difference

Side lengths start 25% longer than their final values, while thicknesses start 25% thinner. Convergence is achieved after 20 iterations.

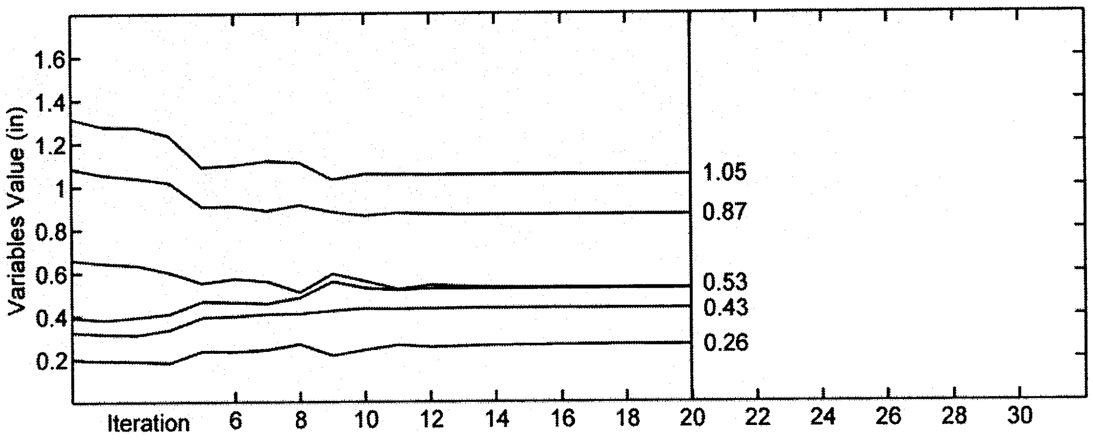


Figure 4.41: Initial Values 25% Different from Final Values

50% Difference

Side lengths start 50% longer than their final values, while thicknesses start 50% thinner. Convergence is achieved after 25 iterations.

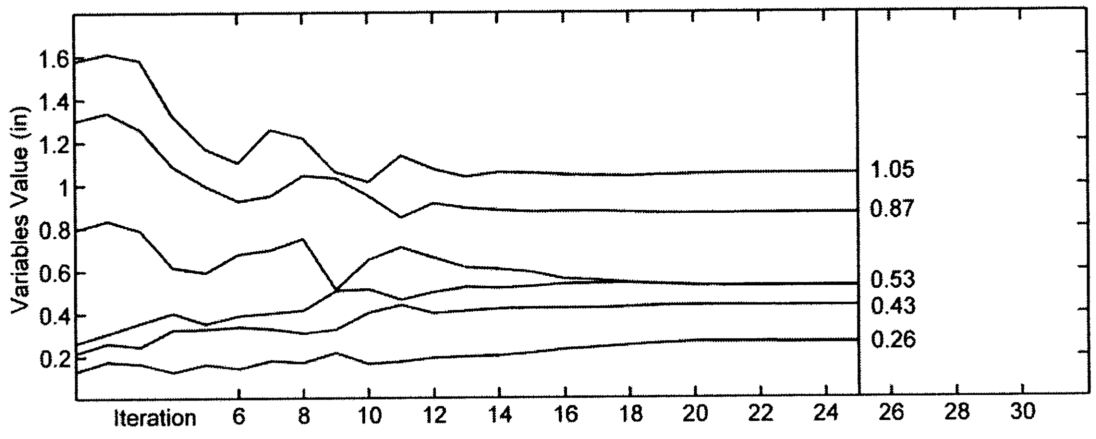


Figure 4.42: Initial Values 50% Different from Final Values

100% Difference

Side lengths start 100% longer than their final values, while thicknesses start 100% thinner. Convergence is achieved after 30 iterations.

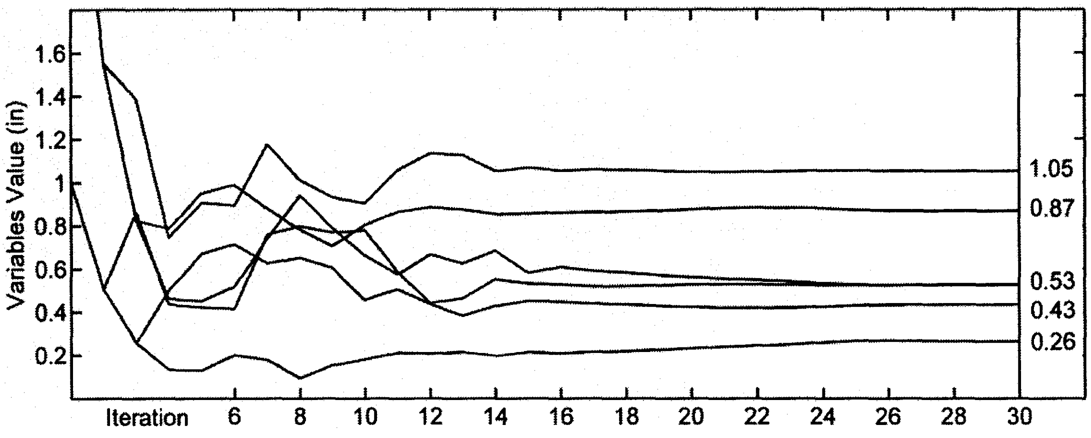


Figure 4.43: Initial Values 100% Different from Final Values

Summary

The gradient-based algorithm needs several iterations to find the appropriate range of each optimization variable before more finely adjusting them and reach the optimal solution. A good estimate of the optimal solution is therefore a better starting point for the optimization process than a random design. If the system to optimize is too complex to estimate the optimal design, the optimization variables shall be, at least, initialized with reasonable values for the design parameters they represent.

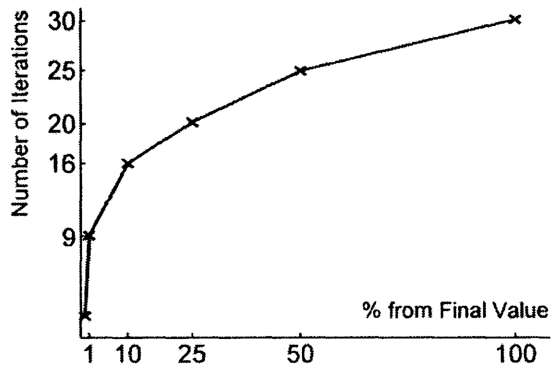


Figure 4.44: Effect of Initial Values on Convergence Speed

4.5.3 Effect of the Relative Initial Values

Section 4.5.2 shows that the absolute initial value given to the optimization variables affects the convergence speed of the optimization process. The relative initial value of each variable with respect to the others also affects the convergence speed. This effect is shown using the optimization of a girder made of 4-pipe sections. The girder is made of 6 segments but is symmetrical with respect to the mid-span, so that only 3 segments are optimized. Each segment is made of 4 pipes, with both top pipes assumed to be identical and both bottom pipes also identical (but potentially different from top pipes). Each pipe is parametrized by its outer and inner radiuses. The resulting 12 optimization variables are represented on Figure 4.45.

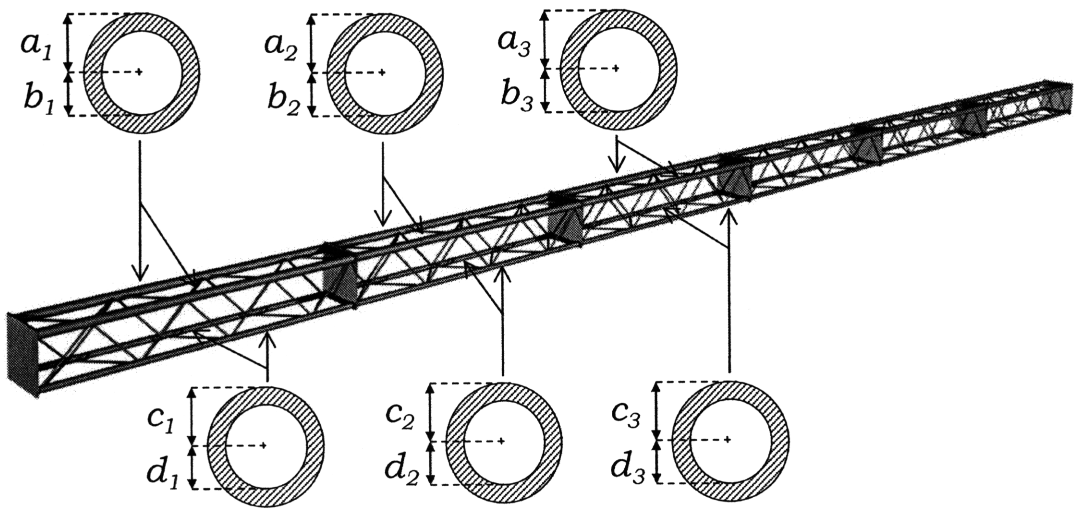


Figure 4.45: Four-Pipe Section Girder Optimization Variables

The optimal design values are:

$$
\begin{aligned}
a_1 = c_1 &= 0.44\,\text{in} & b_1 = d_1 &= 0.00\,\text{in} \\
a_2 = c_2 &= 0.72\,\text{in} & b_2 = d_2 &= 0.00\,\text{in} \\
a_3 = c_3 &= 0.87\,\text{in} & b_3 = d_3 &= 0.00\,\text{in}
\end{aligned}
$$

Knowing how the final values of the optimization variables are ordered, initial values are ordered in different ways and the number of iterations required for convergence is compared.

Finely Ordered Initial Values

The initial values of the pipes inner radiuses are increased where the bending moment is larger, and the top and bottom pipes of a same segment are given similar initial values. Convergence is achieved after 32 iterations. Optimization variables remain in their initial order, as they do not cross each other.

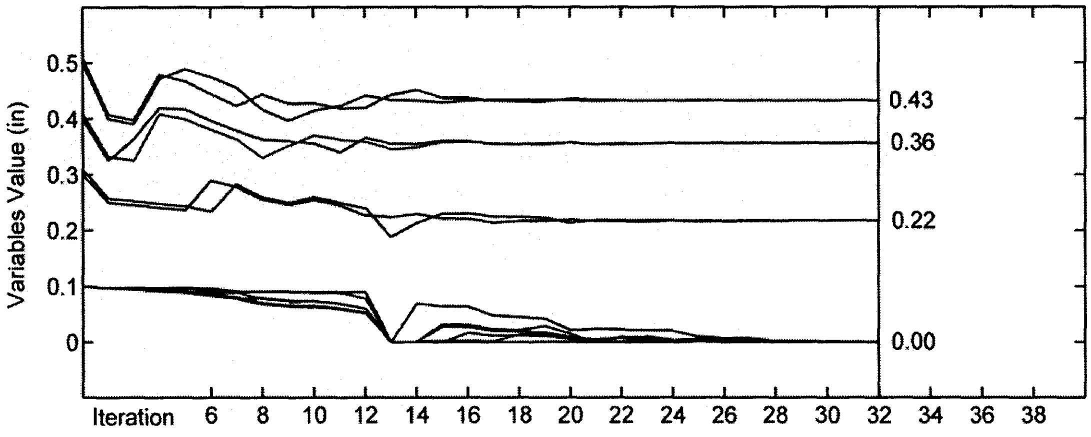


Figure 4.46: Finely Ordered Initial Values

Roughly Ordered Initial Values

The initial values of the pipes outer radiuses are still increased where the bending moment is larger, but the top and bottom pipes of a given girder segment do not have close initial values. This has little effect on convergence speed, as the process needs only one additional iteration to converge (33 vs. 32 iterations).

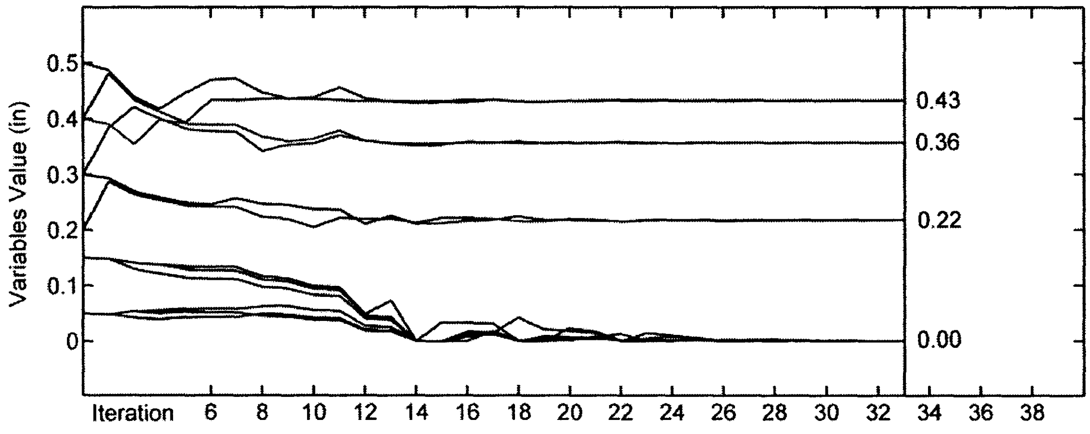


Figure 4.47: Roughly Ordered Initial Values

Equal Initial Values

All outer radiuses are given the same initial value, and all inner radiuses also start with a common initial value. The algorithm adjusts the variables in their final relative order quite quickly at the beginning of the process, but the convergence time is still increased to 37 iterations.

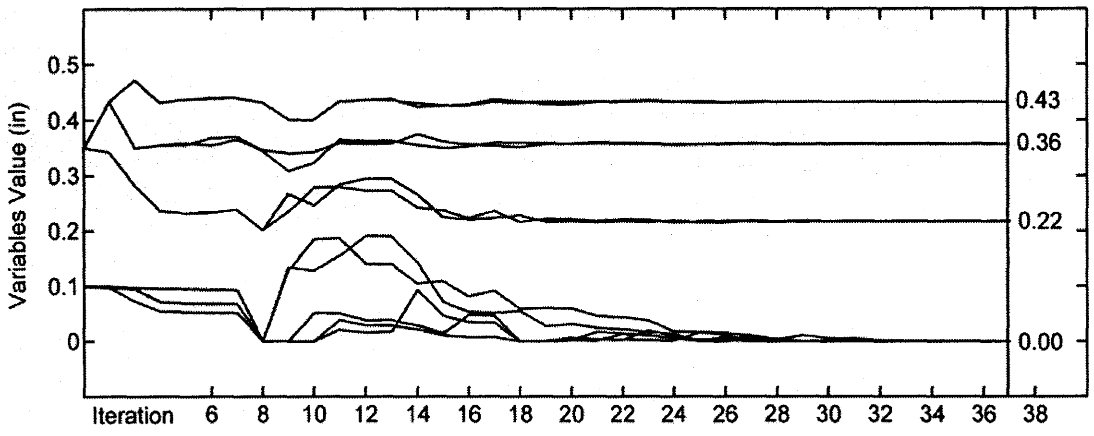


Figure 4.48: Same Initial Values

Non-Ordered Initial Values

Optimization variables are organized into 4 groups: top pipes outer radiuses (a_1 a_2 a_3), bottom pipes outer radiuses (c_1 c_2 c_3), top pipes inner radiuses (b_1 b_2 b_3), bottom pipes inner radiuses (d_1 d_2 d_3). All variables within a group are given the same initial value. Convergence is achieved after 37 iterations.

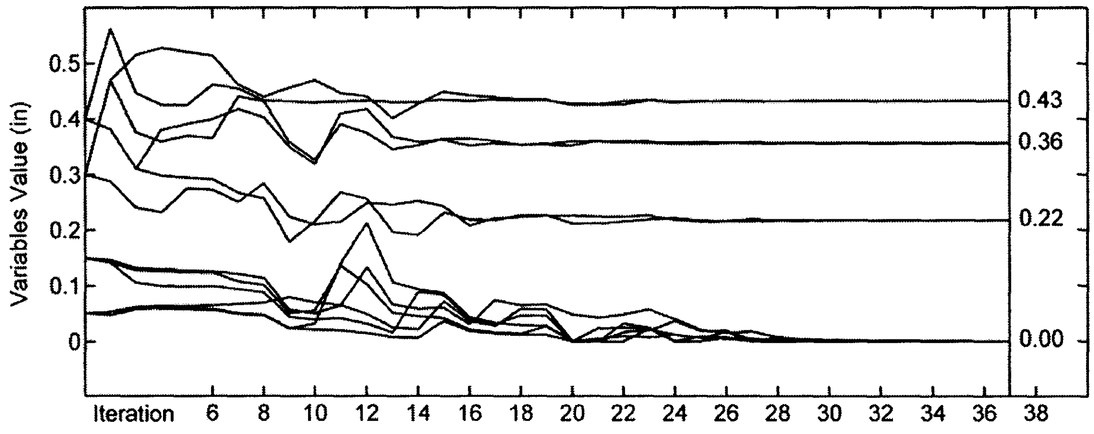


Figure 4.49: Non-Ordered Initial Values

4.5.4 Conclusions

General Algorithm Behavior

All convergence diagrams show a common general behavior. The convergence occurs in three successive phases, as illustrated on Figure 4.50.

- PHASE A: Optimization variables experience large variations from their initial values to reach the range in which their final value will eventually be. At the end of this first phase, the variables are in their final order. If starting with the same value, variables whose final value are close may experience the same evolution during this phase.
- PHASE B: Variations are smaller, as variables are adjusted to reach their final value.
- PHASE C: Optimization variables remain almost constant, as the algorithm checks that the current design is actually the global optimum.

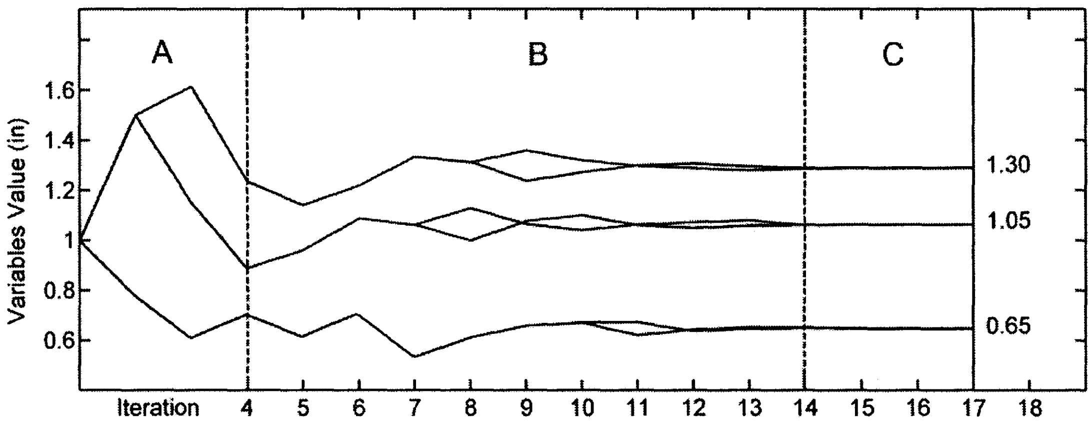


Figure 4.50: Main Phases of the Convergence Process

The absolute and relative durations of each phase depends on the problem considered and on parameters adjusting the behavior of the gradient-based algorithm. In particular, the duration of the third phase is governed by the termination criterion.

Variables Selection

The behavior of the gradient-based algorithm is greatly affected by the set of optimization variables parametrizing the design problem. The selection of the variables is an important preliminary step in algorithmic optimization. The number of variables and the absolute and relative initial values assigned to them all affect the convergence speed of the algorithm, as shown in sections 4.5.1, 4.5.2 and 4.5.3 respectively.

To reduce the number of variables is more efficient at increasing convergence speed than to finely adjust the initial values assigned to a greater number of variables. In section 4.5.3, adjusting the initial values of 12 variables allowed for a slightly faster convergence (32 vs 37 iterations). Figure 4.51 (below) shows the same problem being solved in only 19 iterations after removal of 6 irrelevant variables.

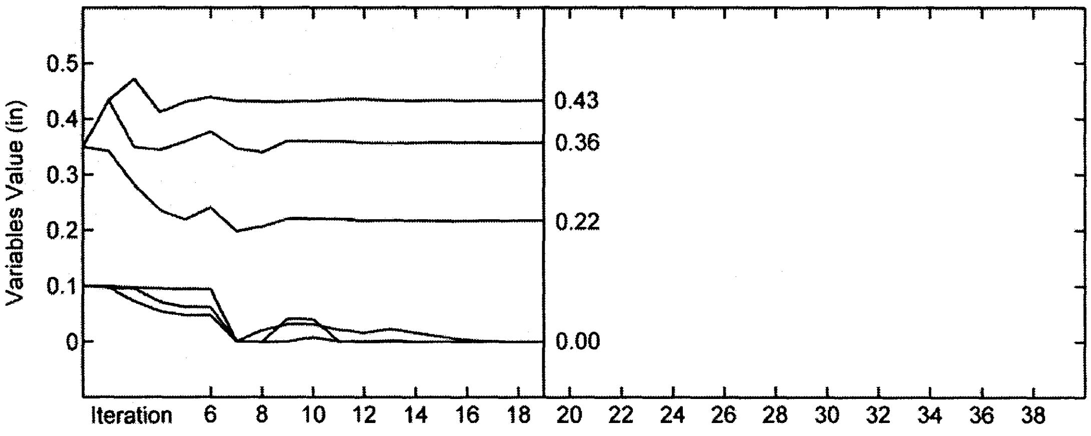


Figure 4.51: Convergence of 6 optimization variables (c)

The effect is even more important on the convergence time, since each iteration is faster with less variables.

4.6 Stiffened Girder Optimization Model

In the context of the *Student Steel Bridge Competition*, the girder option was preferred to the truss based on constructability issues. The girder concept was then optimized to minimize the structural cost and, considering its quick assembly, might be a good bridge design overall. However, it is not clear whether a more elaborate structure, longer to build but stiffer, can be a better option. Full trusses seem too long to assemble for their increased stiffness to compensate the loss in construction speed. But a hybrid scheme, consisting in a quickly assembled girder equipped with a few additional truss members, could be a good balance. It is proposed to use the gradient-based optimization method to evaluate the potential of a stiffened girder.

4.6.1 Girder Concept Modification

The components of a stiffened girder as shown on figure 4.52. The girder is a flexural member, behaving as a beam. The stiffener is made of pinned members and is also pinned to the girder, therefore behaving as a truss. Several truss configurations are considered (see section 4.6.2) but all trusses feature vertical members. These members are also called *king posts* and work in compression (refer to the hybrid schemes analytical solutions derived in appendix F). They are balanced by *tension rods*, installed horizontally and diagonally.

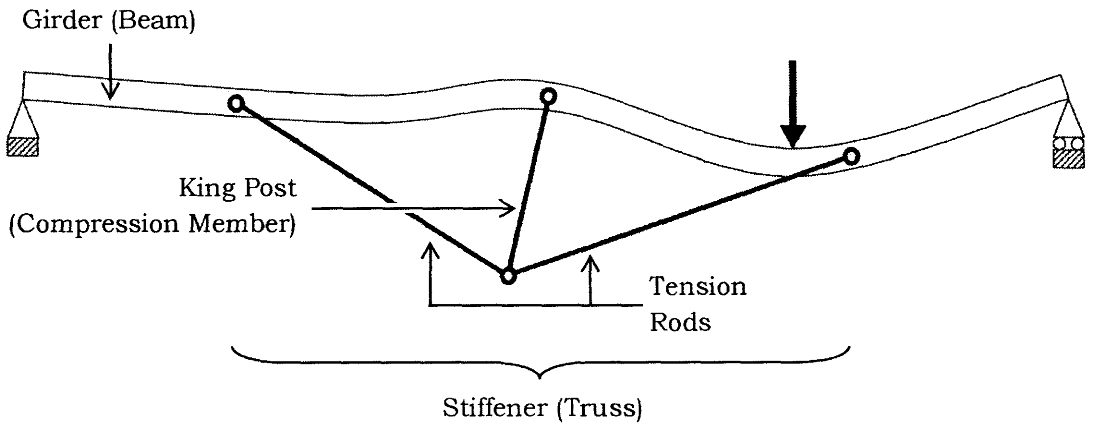


Figure 4.52: Hybrid System Description

Tension and compression in the stiffener are caused by the differential displacements of the truss nodes connected to the girder. As a result, the stiffener tends to generate an opposed bending moment on the stiffener, thus reducing its deflection.

4.6.2 Stiffener Schemes

The girder considered in this study has a triangular cross-section made of three longitudinal pipes. It is equipped with three different stiffeners. In the following, the terms *simple*, *double* and *triple stiffener* refer to the number of king posts in the stiffening truss. On the following renderings, the king posts are represented as elaborate members since they are subject to compression and would be engineered to prevent buckling. Tension members are represented as simple rods.

A simple stiffener (figure 4.53) is made of 5 members. Each diagonal tension rod requires 2 segments due to the member size limit of 42".

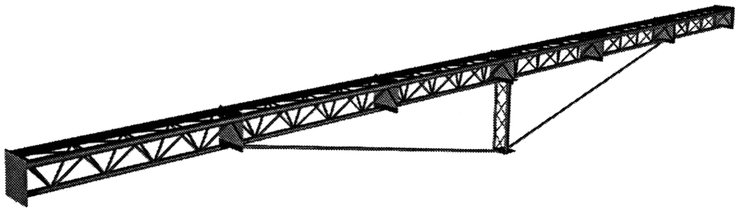


Figure 4.53: Simply-Stiffened Girder

A double stiffener (figure 4.54) is made of 6 members. The horizontal tension rod requires 2 segments due to the member size limit of 42".

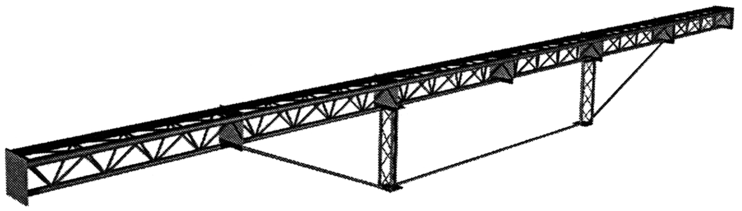


Figure 4.54: Doubly-Stiffened Girder

A triple stiffener (figure 4.55) is made of 7 pieces.

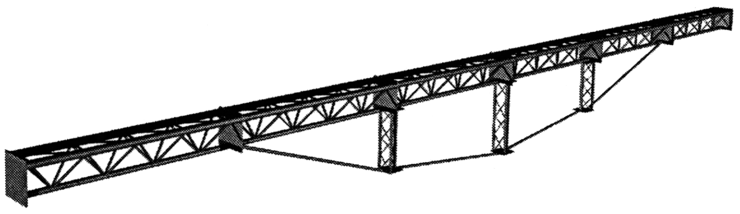


Figure 4.55: Triply-Stiffened Girder

Truss topology optimization is not considered in this study, and each hybrid system is therefore optimized separately. The design parameters are the girder and truss members sizes as well as the truss geometry, varied by adjusting the lengths of the king posts.

4.7 Hybrid System Analysis Methods

In order to optimize the stiffened girders, the gradient-based algorithm needs to interact with a structural analysis tool that is efficient at solving this type of hybrid structures. Unstiffened girders were analyzed using numerical moment integration as a fast method easily applicable to determinate structures. Trusses were analyzed using stiffness matrices as a robust method to solve more complex structures. The stiffened girders considered here are hybrid structures featuring girder and truss properties. Both analysis methods are applicable to this type of structures. Though relatively slow, matrix analysis is easily implemented by combining beam and truss stiffness matrices. The numerical moment integration method is faster but requires the derivation of an analytical solution for the beam-truss interaction. Both methods are developed in the following.

4.7.1 Hybrid System Matrix Analysis

The stiffened girder is modeled as an assembly of beam and truss elements. The beam is discretize into 30 elements, allowing for precise load placement and accurate results while keeping the analysis reasonably fast.

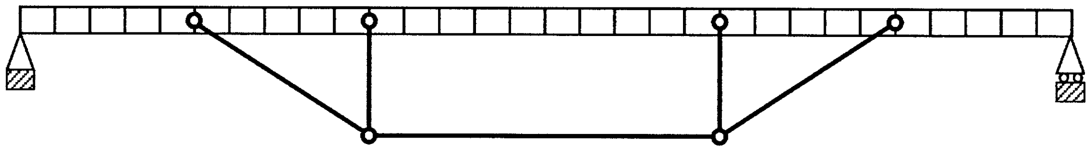


Figure 4.56: Elements Assembly for the Girder with Double Stiffener

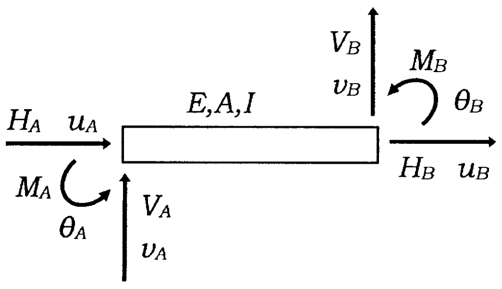


Figure 4.57: Beam Element

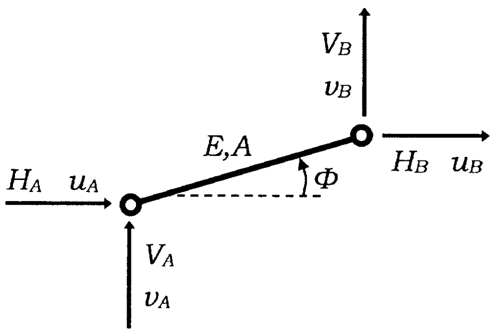


Figure 4.58: Truss Element

Beam Element Stiffness Matrix

The two-dimensional beam element, represented on figure 4.57 (p.123), is of length L and has a cross-sectional area A, a moment of inertia I and a modulus of elasticity E. The derivation of its stiffness matrix is detailed in appendix B, and the resulting force-displacement relationship is expressed as:

$$\begin{pmatrix} H_A \\ V_A \\ M_A \\ H_B \\ V_B \\ M_B \end{pmatrix} = \begin{pmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{pmatrix} \begin{pmatrix} u_A \\ v_A \\ \theta_A \\ u_B \\ v_B \\ \theta_B \end{pmatrix}$$

Truss Element Stiffness Matrix

The two-dimensional truss element, represented on figure 4.58 (p.123), is of length L, has a cross-sectional area A and a modulus of elasticity E. Its stiffness matrix is obtained by simplifying the result derived in appendix A for a the three-dimensional truss member. The resulting force-displacement relationship is expressed as:

$$\begin{pmatrix} H_A \\ V_A \\ H_B \\ V_B \end{pmatrix} = \frac{EA}{L} \begin{pmatrix} \cos^2\Phi & \sin\Phi\cos\Phi & -\cos^2\Phi & -\sin\Phi\cos\Phi \\ \cos\Phi\sin\Phi & \sin^2\Phi & -\cos\Phi\sin\Phi & -\sin^2\Phi \\ -\cos^2\Phi & -\sin\Phi\cos\Phi & \cos^2\Phi & \sin\Phi\cos\Phi \\ -\cos\Phi\sin\Phi & -\sin^2\Phi & \cos\Phi\sin\Phi & \sin^2\Phi \end{pmatrix} \begin{pmatrix} u_A \\ v_A \\ u_B \\ v_B \end{pmatrix}$$

Beam and truss stiffness matrices are combined by adding the terms corresponding to the same degrees of freedom, as described in sections 2.1.3 and 3.3.3.

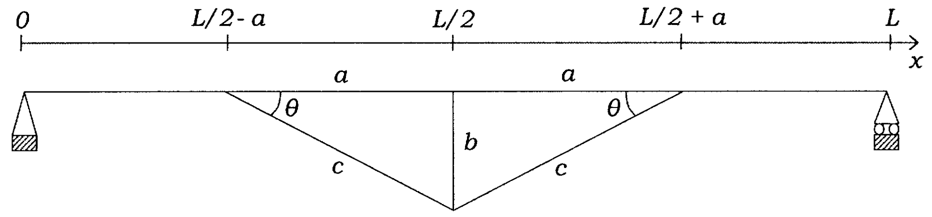
4.7.2 Hybrid System Numerical Moment Integration

Hybrid systems are more sensitive than simple girders to the different load cases considered for the competition. For example, the behavior of a stiffened girder is significantly modified as a load is moved from the top of a king post to a location between two king posts. It is risky to consider only a few significant load cases in the optimization process, as an unconsidered load scenario might become critical when the design is modified. The structure is therefore solved for all of the 36 load cases at each iteration, making the need for a quick analysis tool even more important. Numerical moment integration is faster than matrix analysis and shall therefore be considered.

The stiffened girders are indeterminate systems, making the solution by moment integration non-trivial. The computer program can carry out the moment integrations, but an analytical solution for the girder-truss interaction is still needed. The derivation of this solution is detailed in appendix F for the simple and double stiffener schemes. The results are presented in the following in the form of practical methods to solve both hybrid systems.

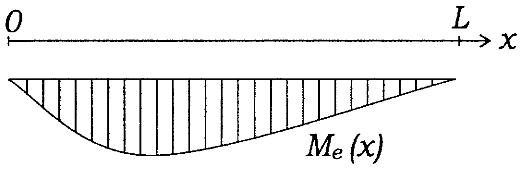
Simple Stiffener Analytical Solution

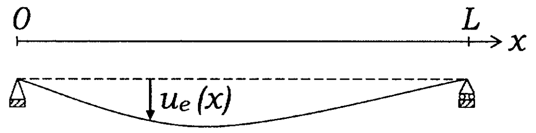
The following symmetrical scheme is considered:



The lengths L, a and b define the geometry of the problem. The moment of inertia $I(x)$ of the girder is known, as well as the cross-sectional areas A_b and A_c of the truss members.

The first step is to apply the external loads on the unstiffened girder, creating the external bending moment M_e. This moment is integrated twice to obtain the deflection u_e due to the external loads. By convention, $u_\mathrm{e} < 0$.

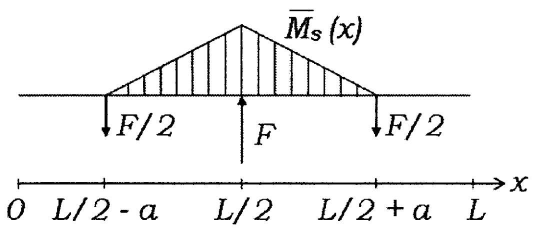


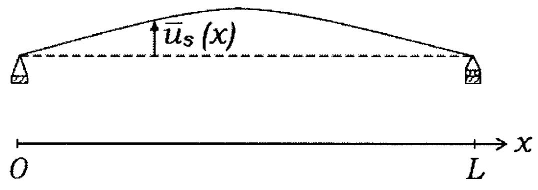


The deflection u_e is calculated at 3 particular points:

$$u_{\mathrm{e}1} = u_\mathrm{e}\left(\frac{L}{2} - a\right) \qquad u_{\mathrm{e}2} = u_\mathrm{e}\left(\frac{L}{2}\right) \qquad u_{\mathrm{e}3} = u_\mathrm{e}\left(\frac{L}{2} + a\right)$$

The reaction of the stiffener is parametrized by the force in the vertical member, noted F. Its value is not yet known. Considering a unit value for F, the stiffener generates a pseudo-bending moment $\overline{M}_\mathrm{s}$ acting on the girder. This bending moment is integrated twice to obtain the pseudo-deflection $\overline{u}_\mathrm{s}$. By convention, $\overline{u}_\mathrm{s} > 0$.





The pseudo-deflection $\overline{u}_\mathrm{s}$ is calculated at 3 particular points:

$$\overline{u}_{\mathrm{s}1} = \overline{u}_\mathrm{s}\left(\frac{L}{2} - a\right) \qquad \overline{u}_{\mathrm{s}2} = \overline{u}_\mathrm{s}\left(\frac{L}{2}\right) \qquad \overline{u}_{\mathrm{s}3} = \overline{u}_\mathrm{s}\left(\frac{L}{2} + a\right)$$

The actual value of the force F in the girder is calculated using:

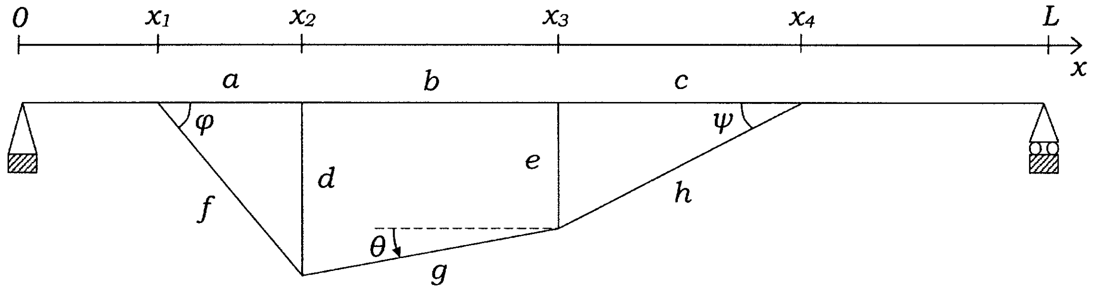
$$F\left(\overline{u_{\mathrm{s}2}} - \frac{\overline{u_{\mathrm{s}1}} + \overline{u_{\mathrm{s}3}}}{2} + \frac{c}{2\sin^2\theta\, EA_\mathrm{c}} + \frac{b}{EA_\mathrm{b}}\right) = u_{\mathrm{e}2} - \frac{u_{\mathrm{e}1} + u_{\mathrm{e}3}}{2}$$

The actual girder deflection is then given by:

$$u = u_\mathrm{e} + F\overline{u}_\mathrm{s} \qquad (> 0 \quad \text{upwards by convention})$$

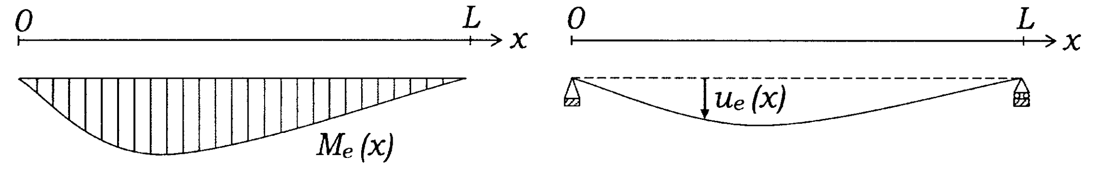
Double Stiffener Analytical Solution

The following scheme is considered:



The lengths L, d and e and the coordinates x_1, x_2, x_3 and x_4 of the truss nodes connected to the girder define the geometry of the problem. The moment of inertia $I(x)$ of the girder is known, as well as the cross-sectional areas A_d, A_e, A_f, A_g and A_h of the truss members.

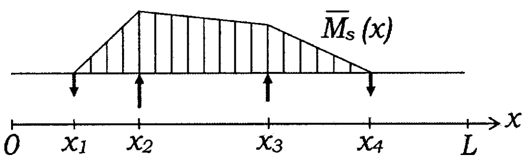
The first step is to apply the external loads on the unstiffened girder, creating the external bending moment M_e. This moment is integrated twice to obtain the deflection u_e due to the external loads. By convention, $u_\mathrm{e} < 0$.

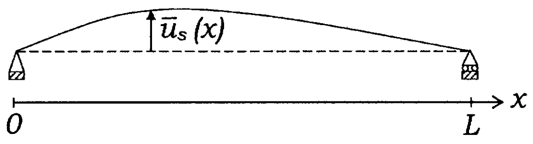


The deflection u_e is calculated at 4 particular points:

$$u_{\mathrm{e}1} = u_\mathrm{e}(x_1) \qquad u_{\mathrm{e}2} = u_\mathrm{e}(x_2) \qquad u_{\mathrm{e}3} = u_\mathrm{e}(x_3) \qquad u_{\mathrm{e}4} = u_\mathrm{e}(x_4)$$

The reaction of the stiffener is parametrized by the force in the bottom truss member, noted F. Its value is not yet known. Since the truss is determinate, the force in every member can be expressed in terms of F. Considering a unit value for F, the stiffener generates a pseudo-bending moment $\overline{M}_{\mathrm{s}}$ acting on the girder. This bending moment is integrated twice to obtain the pseudo-deflection $\overline{u}_{\mathrm{s}}$. By convention, $\overline{u}_{\mathrm{s}} > 0$.





The pseudo-deflection $\overline{u}_{\mathrm{s}}$ is calculated at 3 particular points:

$$\overline{u}_{\mathrm{s1}} = \overline{u}_{\mathrm{s}}(x_1) \qquad \overline{u}_{\mathrm{s2}} = \overline{u}_{\mathrm{s}}(x_2) \qquad \overline{u}_{\mathrm{s3}} = \overline{u}_{\mathrm{s}}(x_3) \qquad \overline{u}_{\mathrm{s4}} = \overline{u}_{\mathrm{s}}(x_4)$$

The actual force F in the girder is calculated using:

$$F\left(\begin{array}{llll} & (\overline{u_{\mathrm{s2}}} - \overline{u_{\mathrm{s3}}})\tan\theta & + \dfrac{g}{EA_{\mathrm{g}}}\dfrac{1}{\cos\theta} & \\ + & (\overline{u_{\mathrm{s2}}} - \overline{u_{\mathrm{s1}}})\tan\varphi & + \dfrac{f}{EA_{\mathrm{f}}}\dfrac{\cos\theta}{\cos^2\varphi} & + \dfrac{d}{EA_{\mathrm{d}}}\dfrac{\sin(\varphi+\theta)}{\cos\varphi}(\tan\varphi + \tan\theta) \\ + & (\overline{u_{\mathrm{s3}}} - \overline{u_{\mathrm{s4}}})\tan\psi & + \dfrac{h}{EA_{\mathrm{h}}}\dfrac{\cos\theta}{\cos^2\psi} & + \dfrac{e}{EA_{\mathrm{e}}}\dfrac{\sin(\psi-\theta)}{\cos\psi}(\tan\psi - \tan\theta) \end{array}\right)$$

$$= \quad (u_{\mathrm{e2}} - u_{\mathrm{e3}})\sin\theta \quad + \quad (u_{\mathrm{e2}} - u_{\mathrm{e1}})\cos\theta\tan\varphi \quad + \quad (u_{\mathrm{e3}} - u_{\mathrm{e4}})\cos\theta\tan\psi \tag{4.2}$$

The actual girder deflection is then given by:

$$u = u_{\mathrm{e}} + F\overline{u}_{\mathrm{s}} \qquad (> 0 \quad \text{upwards by convention})$$

4.8 Stiffened Girder Optimization Results

The three hybrid systems proposed in section 4.6.2 were optimized as more complex application examples for the gradient-based algorithm. For each hybrid system, the girder and the truss stiffener are both taken into account in a single optimization process. Reusing an optimal girder previously obtained and simply adding a stiffener would not lead to an optimal hybrid system since the girder shall be specifically designed to work best with the truss.

The simple and double stiffener schemes were analyzed by numerical moment integration, using the analytical solutions derived in section 4.7.2. The triple stiffener scheme was solved by hybrid matrix analysis, as presented in section 4.7.1. In all cases, a three-pipe section was used for the girder.

4.8.1 Simple Stiffener

The simply-stiffened girder is assembled from 11 pre-fabricated members, as shown on figure 4.59. The both the girder and the truss stiffener are symmetric, and the members labeled with the same number are identical.

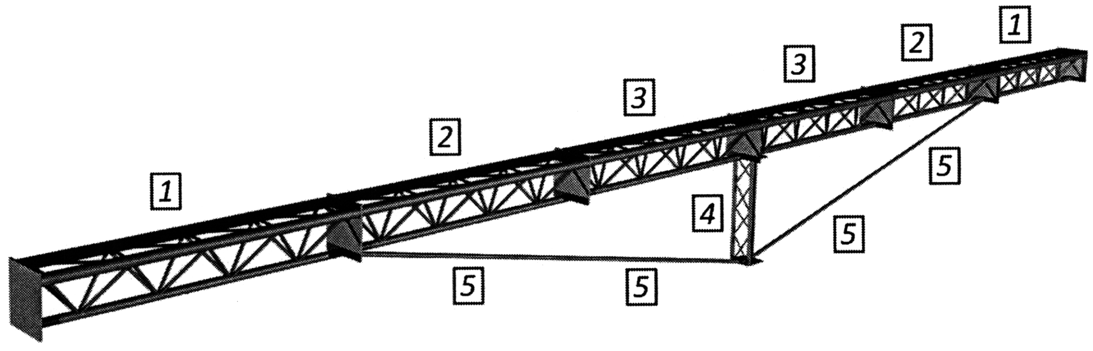


Figure 4.59: Simply-Stiffened Girder Parametrization

Optimization variables are used to parametrize the girder, the king post and the tension rods. These parameters are all considered simultaneously in the optimization process.

Table 4.1 below is a list of the variables considered in the optimization process of the simply-stiffened girder. The pipes are described by their cross-sectional areas instead of the typical inner and outer radiuses. Resistance of the pipes to buckling is not imposed as an optimization constraint, allowing for a faster optimization process. Rods can therefore be used instead of pipes, as the section area is the only relevant design parameter. The inner and outer diameters of these members are selected after the optimization process to match the optimal areas, with negligible effect on the girder capacity as long as the selected pipes remain slender.

Element	Description	Notation	Optimum Value (in, in^2)
1	Top Pipes Area	A_{T1}	0.152
	Bottom Pipe Area	A_{B1}	0.299
2	Top Pipes Area	A_{T2}	0.235
	Bottom Pipe Area	A_{B2}	0.461
3	Top Pipes Area	A_{T3}	0.107
	Bottom Pipe Area	A_{B3}	0.211
4	Length	L_4	21
	Total Area	A_4	0.192
5	Area	A_5	0.300

Table 4.1: Simply-Stiffened Girder Optimization Variables

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

Weight	121 lbs
Aggregate Deflection	0.93 in
Cost	\$ 1260

Figures 4.60 and 4.61 on the next page illustrate the behavior of the simply-stiffened girder during the loading process. The bending moment acting on the girder is represented and the values of the forces in the truss are given, as well as the deflections at the points of interest.

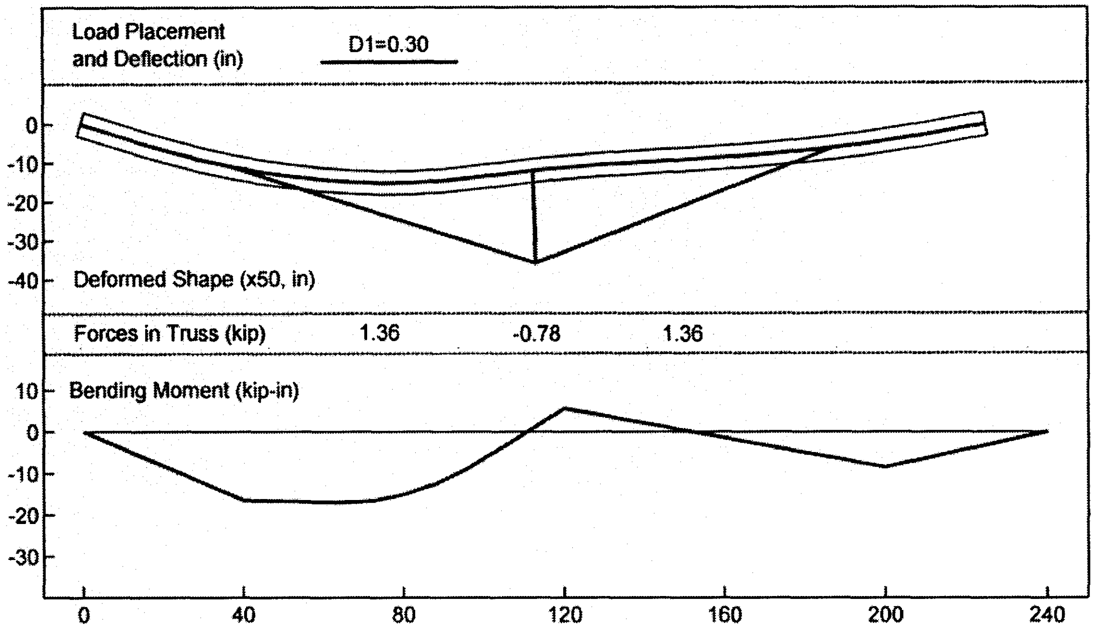


Figure 4.60: Deformation of Simply-Stiffened Girder under First Load

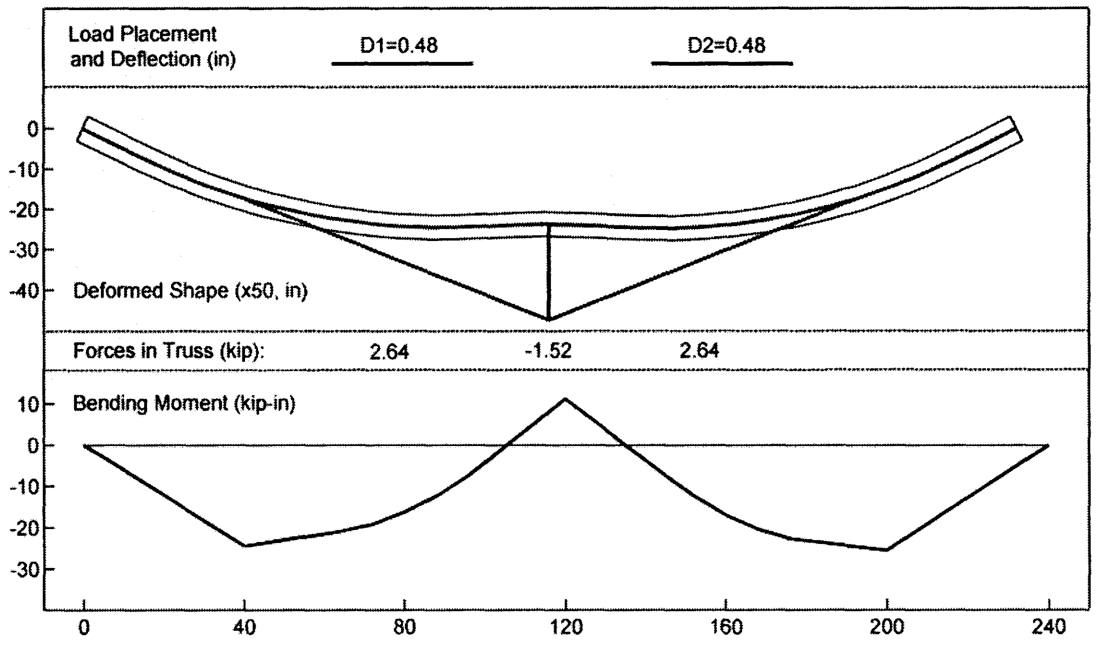


Figure 4.61: Deformation of Simply-Stiffened Girder under Both Loads

4.8.2 Double Stiffener

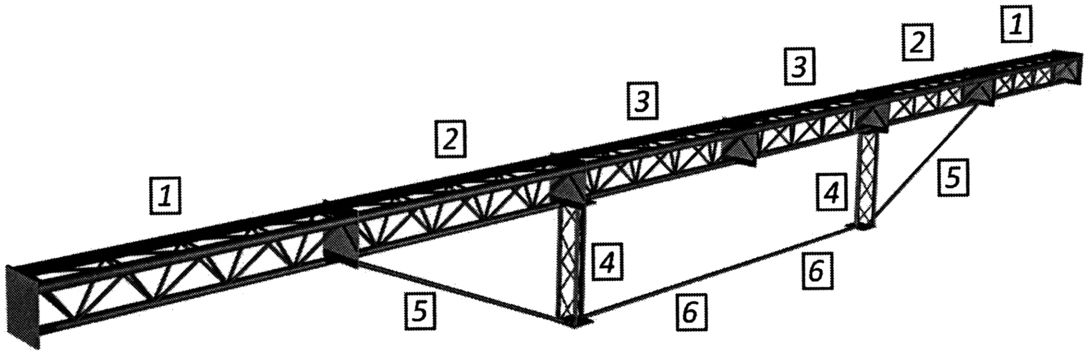


Figure 4.62: Doubly-Stiffened Girder Parametrization

Element	Description	Notation	Optimum Value (in, in^2)
1	Top Pipes Area	A_{T1}	0.153
	Bottom Pipe Area	A_{B1}	0.301
2	Top Pipes Area	A_{T2}	0.173
	Bottom Pipe Area	A_{B2}	0.340
3	Top Pipes Area	A_{T3}	0.057
	Bottom Pipe Area	A_{B3}	0.113
4	Length	L_4	21
	Total Area	A_4	0.163
5	Area	A_5	0.265
6	Area	A_6	0.228

Table 4.2: Doubly-Stiffened Girder Optimization Variables

The worst-case aggregate deflection and the total weight of the bridge give the structural cost of the optimal design.

Weight	98 lbs
Aggregate Deflection	0.75 in
Cost	\$ 1010

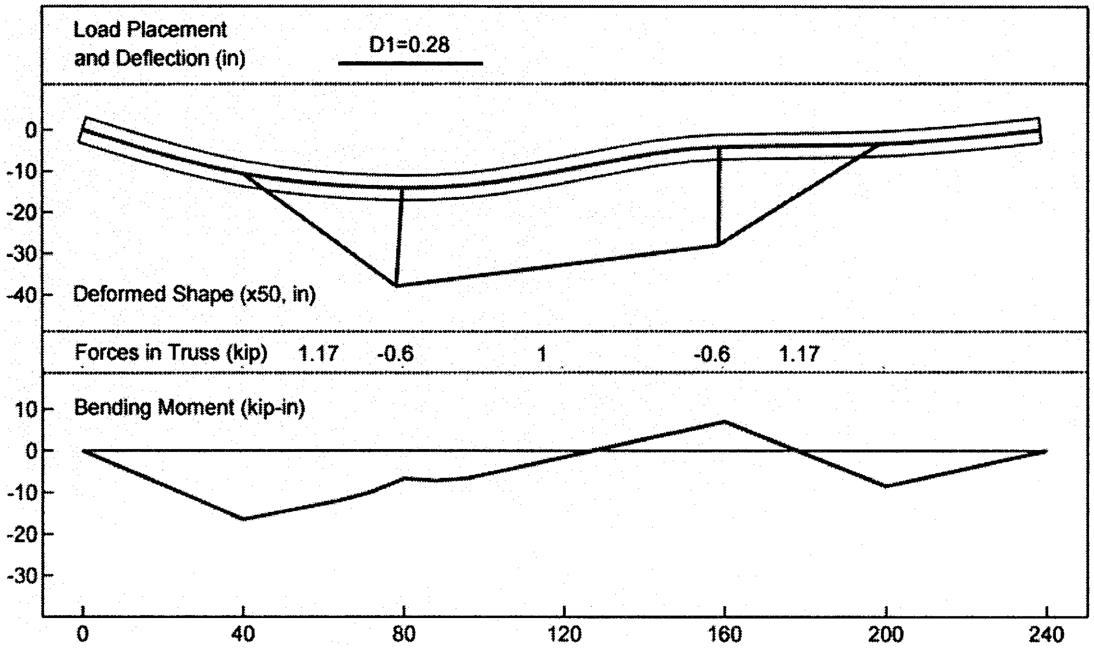


Figure 4.63: Deformation of Doubly-Stiffened Girder under First Load

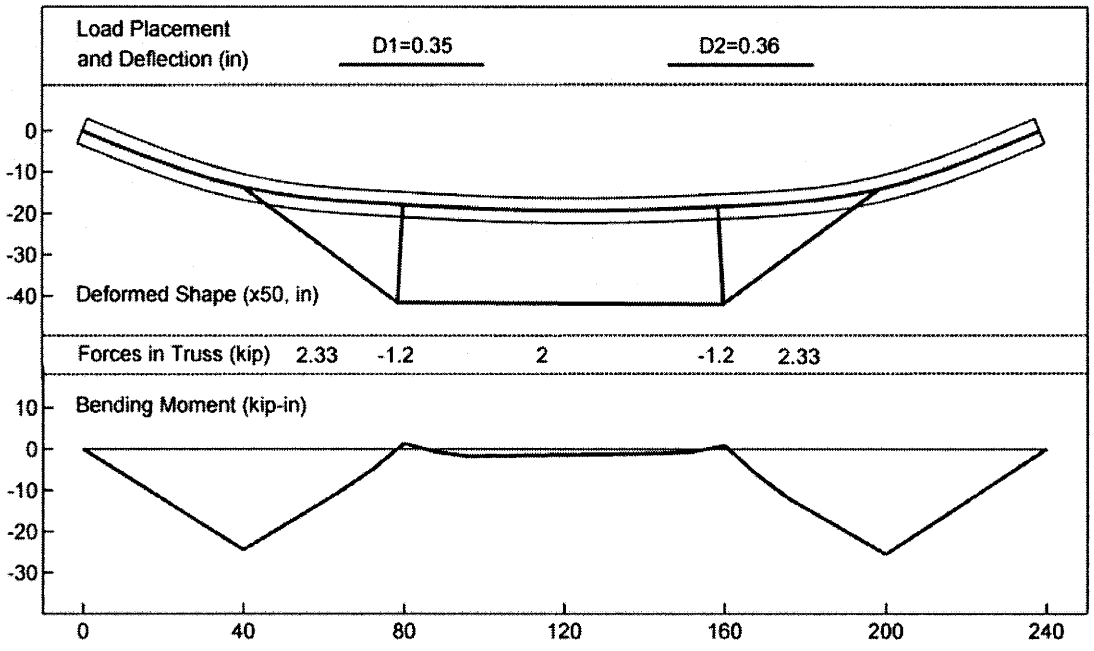


Figure 4.64: Deformation of Doubly-Stiffened Girder under Both Loads

4.8.3 Triple Stiffener

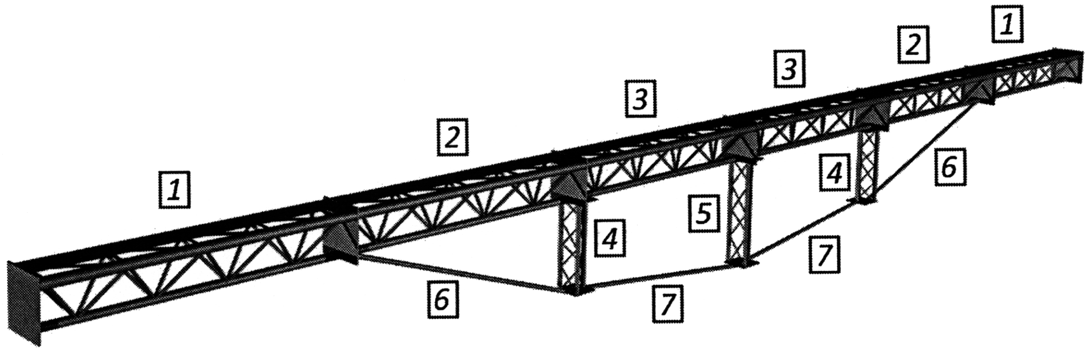


Figure 4.65: Triply-Stiffened Girder Parametrization

Element	Description	Notation	Optimum Value (in, in^2)
1	Top Pipes Area	A_{T1}	0.153
	Bottom Pipe Area	A_{B1}	0.301
2	Top Pipes Area	A_{T2}	0.173
	Bottom Pipe Area	A_{B2}	0.340
3	Top Pipes Area	A_{T3}	0.057
	Bottom Pipe Area	A_{B3}	0.113
4	Length	L_4	21
	Total Area	A_4	0.163
5	Length	L_5	21
	Total Area	A_5	0.016
6	Area	A_6	0.266
7	Area	A_6	0.227

Table 4.3: Triply-Stiffened Girder Optimization Variables

Weight	98 lbs
Aggregate Deflection	0.75 in
Cost	\$ 1010

The triply-stiffened girder behaves exactly as the doubly-stiffened girder, as the middle king post cannot take any load in the optimal geometry obtained. The double stiffener scheme is therefore selected.

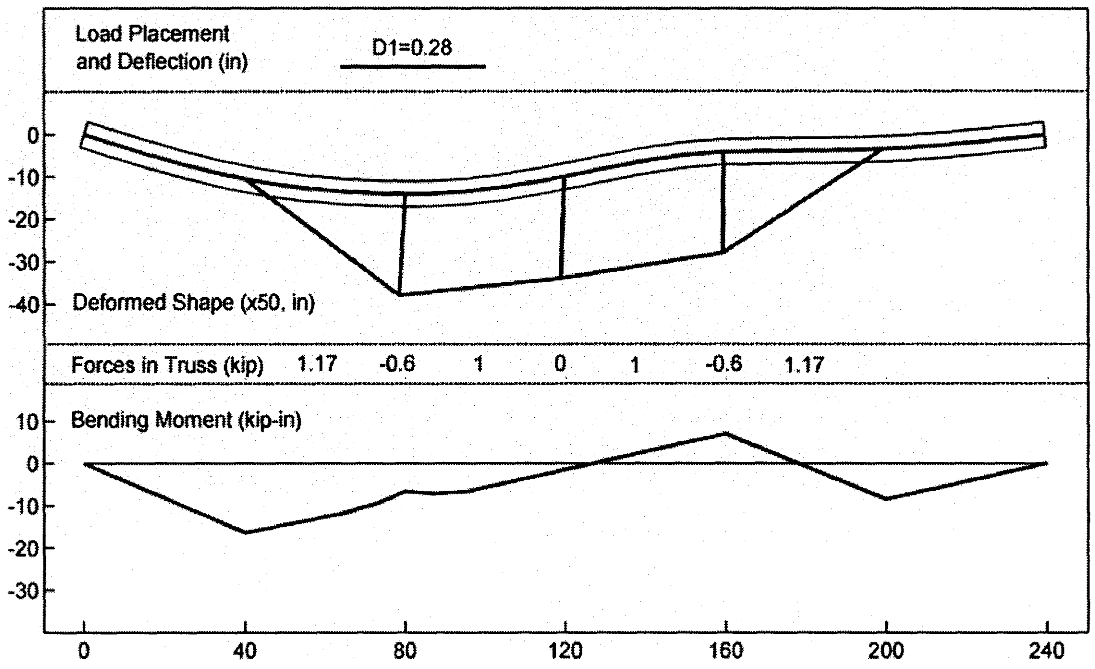


Figure 4.66: Deformation of Triply-Stiffened Girder under First Load

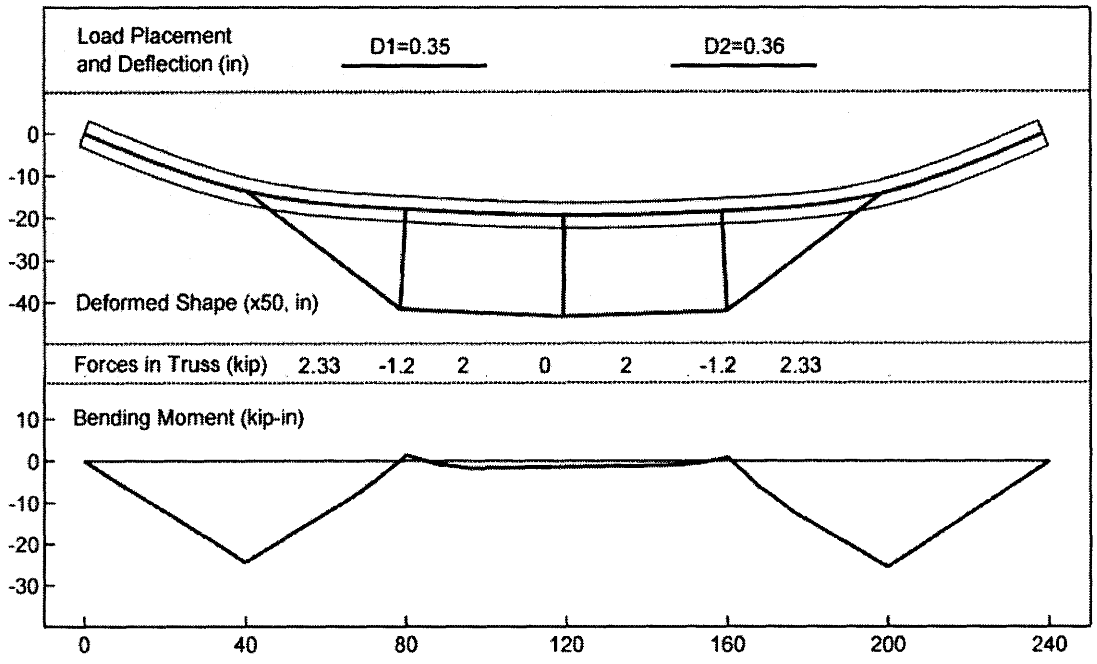


Figure 4.67: Deformation of Triply-Stiffened Girder under Both Loads

APPENDICES

Appendix A

Truss Member Stiffness Matrix

A three-dimensional truss member is shown on figure A.1. The geometry is defined by the length L of the member and the oriented angles θ and φ. The truss member is limited by two nodes (A, B). Each node has 3 degrees of freedom corresponding to the orthogonal directions $[x,\ y,\ z]$. The nodal displacements in these directions are noted as $[u,\ v,\ w]$ respectively.

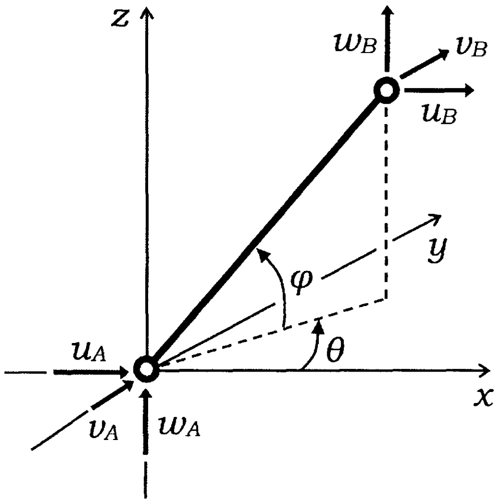


Figure A.1: Nodal Displacements

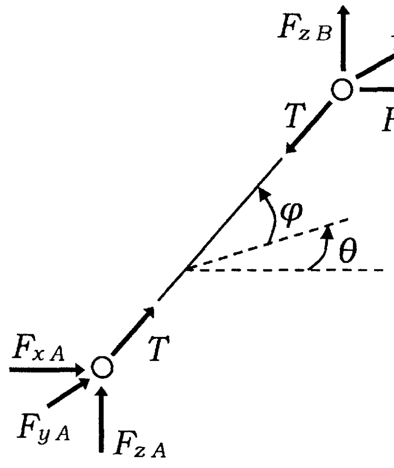


Figure A.2: Nodal Equilibrium

External and internal actions are shown on figure A.2. The external forces $[F_x,\ F_y,\ F_z]$ are applied at both nodes of the truss member in the directions of the degrees of freedom. The internal force T is due to the deformation of the truss member and acts in its the axial direction.

The elongation of the truss member is a function of the nodal displacements and depends on the geometry:

$$\Delta L = (u_\mathrm{B} - u_\mathrm{A})\cos\theta\cos\varphi + (v_\mathrm{B} - v_\mathrm{A})\sin\theta\cos\varphi + (w_\mathrm{B} - w_\mathrm{A})\sin\varphi \tag{A.1}$$

The elongation creates a force in the member. Using an elastic model, where A is the cross-sectional area of the member, E its modulus of elasticity and L its length:

$$T = AE\frac{\Delta L}{L} \tag{A.2}$$

If $T > 0$, the truss member is in tension.
If $T < 0$, the truss member is in compression.

Force equilibrium is applied at both nodes by projecting the force T in the directions of the degrees of freedom:

$$\text{Node A}\left\{\begin{array}{lcl} F_\mathrm{xA} &=& -T\cos\theta\cos\varphi \\ F_\mathrm{yA} &=& -T\sin\theta\cos\varphi \\ F_\mathrm{zA} &=& -T\sin\varphi \end{array}\right. \qquad \text{Node B}\left\{\begin{array}{lcl} F_\mathrm{xB} &=& T\cos\theta\cos\varphi \\ F_\mathrm{yB} &=& T\sin\theta\cos\varphi \\ F_\mathrm{zB} &=& T\sin\varphi \end{array}\right. \tag{A.3}$$

(A.1), (A.2) and (A.3) are combined:

$$\begin{pmatrix} F_\mathrm{xA} \\ F_\mathrm{yA} \\ F_\mathrm{zA} \\ F_\mathrm{xB} \\ F_\mathrm{yB} \\ F_\mathrm{zB} \end{pmatrix} = \underbrace{\frac{AE}{L}\begin{pmatrix} & (u_\mathrm{B} - u_\mathrm{A}) & \cos\theta\cos\varphi \\ + & (v_\mathrm{B} - v_\mathrm{A}) & \sin\theta\cos\varphi \\ - & (w_\mathrm{B} - w_\mathrm{A}) & \sin\varphi \end{pmatrix}}_{T} \begin{pmatrix} -\cos\theta\cos\varphi \\ -\sin\theta\cos\varphi \\ -\sin\varphi \\ \cos\theta\cos\varphi \\ \sin\theta\cos\varphi \\ \sin\varphi \end{pmatrix} \tag{A.4}$$

(A.4) is rearranged to express the nodal forces as functions of the displacements. The force-displacement relationship is summarized as:

$$\underline{F} = \underline{K}\,\underline{U} \tag{A.5}$$

where $\underline{F} = \begin{pmatrix} F_{\text{xA}} \\ F_{\text{yA}} \\ F_{\text{zA}} \\ F_{\text{xB}} \\ F_{\text{yB}} \\ F_{\text{zB}} \end{pmatrix}$ $\quad \underline{U} = \begin{pmatrix} u_{\text{A}} \\ v_{\text{A}} \\ w_{\text{A}} \\ u_{\text{B}} \\ v_{\text{B}} \\ w_{\text{B}} \end{pmatrix}$ $\quad \left\{ \begin{array}{lcl} c_\theta & = & \cos\theta \\ s_\theta & = & \sin\theta \\ c_\varphi & = & \cos\varphi \\ s_\varphi & = & \sin\varphi \end{array} \right.$

$\underline{K}$ is the stiffness matrix of the truss member:

$$\underline{K} = \frac{EA}{L} \begin{pmatrix} c_\theta^2\, c_\varphi^2 & c_\theta s_\theta\, c_\varphi^2 & c_\theta\, c_\varphi\, s_\varphi & -c_\theta^2\, c_\varphi^2 & -c_\theta\, s_\theta\, c_\varphi^2 & -c_\theta\, c_\varphi\, s_\varphi \\ c_\theta\, s_\theta\, c_\varphi^2 & s_\theta^2\, c_\varphi^2 & s_\theta\, c_\varphi\, s_\varphi & -c_\theta\, s_\theta\, c_\varphi^2 & -s_\theta^2\, c_\varphi^2 & -s_\theta\, c_\varphi\, s_\varphi \\ c_\theta\, c_\varphi\, s_\varphi & s_\theta\, c_\varphi\, s_\varphi & s_\varphi^2 & -c_\theta\, c_\varphi\, s_\varphi & -s_\theta\, c_\varphi\, s_\varphi & -s_\varphi^2 \\ -c_\theta^2\, c_\varphi^2 & -c_\theta\, s_\theta\, c_\varphi^2 & -c_\theta\, c_\varphi\, s_\varphi & c_\theta^2\, c_\varphi^2 & c_\theta\, s_\theta\, c_\varphi^2 & c_\theta\, c_\varphi\, s_\varphi \\ -c_\theta\, s_\theta\, c_\varphi^2 & -s_\theta^2\, c_\varphi^2 & -s_\theta\, c_\varphi\, s_\varphi & c_\theta\, s_\theta\, c_\varphi^2 & s_\theta^2\, c_\varphi^2 & s_\theta\, c_\varphi\, s_\varphi \\ -c_\theta\, c_\varphi\, s_\varphi & -s_\theta\, c_\varphi\, s_\varphi & -s_\varphi^2 & c_\theta\, c_\varphi\, s_\varphi & s_\theta\, c_\varphi\, s_\varphi & s_\varphi^2 \end{pmatrix}$$

or $\underline{K} = \dfrac{EA}{L}\,\underline{G}\,\underline{G}^{\text{T}}$ using the geometry vector $\underline{G} = \begin{pmatrix} c_\theta\, c_\varphi \\ s_\theta\, c_\varphi \\ s_\varphi \\ -c_\theta\, c_\varphi \\ -s_\theta\, c_\varphi \\ -s_\varphi \end{pmatrix}$

Appendix B

Beam Segment Stiffness Matrix

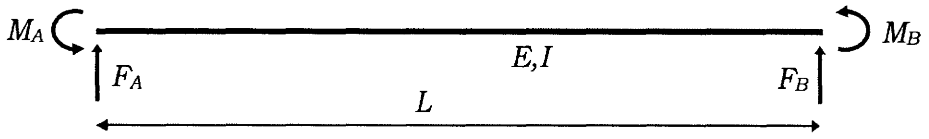


Figure B.1: Forces Acting on Beam Segment

A beam segment and the forces acting on it is represented on figure B.1. Force and moment equilibrium are applied:

$$F_A + F_B = 0 \qquad M_A + M_B + F_B L \tag{B.1}$$

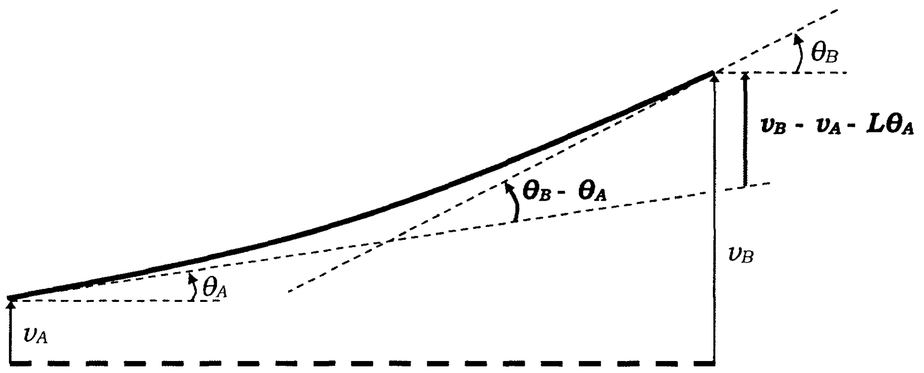


Figure B.2: Beam Segment Displacements

The deformation of the beam segment is represented by two vertical displacements (v_A, v_B) and two rotations (θ_A, θ_B), as shown on figure B.2.

An equivalent cantilever beam segment (figure B.3) is used to derive the force-displacement relationships.

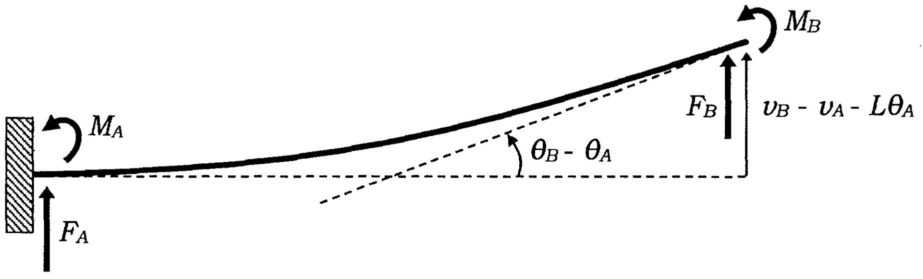


Figure B.3: Equivalent Cantilever Beam Segment

The deformation of a cantilever beam subjected to the end force F_B and the end moment M_B is known:

$$\left\{\begin{aligned} v_\mathrm{B} - v_\mathrm{A} - L\theta_\mathrm{A} &= \frac{F_\mathrm{B}L^3}{3EI} + \frac{M_\mathrm{B}L^2}{2EI} \\ \theta_\mathrm{B} - \theta_\mathrm{A} &= \frac{F_\mathrm{B}L^2}{2EI} + \frac{M_\mathrm{B}L}{EI} \end{aligned}\right. \tag{B.2}$$

The system of equations (B.2) is solved for F_B and M_B:

$$\left\{\begin{aligned} F_\mathrm{B} &= EI\left(-\frac{12}{L^3}v_\mathrm{A} - \frac{6}{L^2}\theta_\mathrm{A} + \frac{12}{L^3}v_\mathrm{B} - \frac{6}{L^2}\theta_\mathrm{B}\right) \\ M_\mathrm{B} &= EI\left(\frac{6}{L^2}v_\mathrm{A} + \frac{2}{L}\theta_\mathrm{A} - \frac{6}{L^2}v_\mathrm{B} + \frac{4}{L}\theta_\mathrm{B}\right) \end{aligned}\right. \tag{B.3}$$

Using the equilibrium equations (B.1), the load-displacement relationship for the beam segment is written:

$$\begin{pmatrix} F_\mathrm{A} \\ M_\mathrm{A} \\ F_\mathrm{B} \\ M_\mathrm{B} \end{pmatrix} = EI \begin{pmatrix} \frac{12}{L^3} & \frac{6}{L^2} & -\frac{12}{L^3} & \frac{6}{L^2} \\ \frac{6}{L^2} & \frac{4}{L} & -\frac{6}{L^2} & \frac{2}{L} \\ -\frac{12}{L^3} & -\frac{6}{L^2} & \frac{12}{L^3} & -\frac{6}{L^2} \\ \frac{6}{L^2} & \frac{2}{L} & -\frac{6}{L^2} & \frac{4}{L} \end{pmatrix} \begin{pmatrix} v_\mathrm{A} \\ \theta_\mathrm{A} \\ v_\mathrm{B} \\ \theta_\mathrm{B} \end{pmatrix}$$

Appendix C

Program Schematic Diagrams

This appendix intends to clarify the schematic diagrams used is section 3.4.1, 2.2.1 and 4.3.2 to represent optimization programs.

Functions

Rectangular boxes represent functions. A function accepts inputs, carries out various operations and returns outputs. In practice, each function consists of a series of instructions written in a file (.m files for Matlab® code used in this study).

The name of each the function is indicated in the rectangular box. Arrows arriving to the top of the rectangle represent function inputs. Arrows departing from the top of the rectangle represent function outputs. The name of input and output variables are indicated along the corresponding arrows.

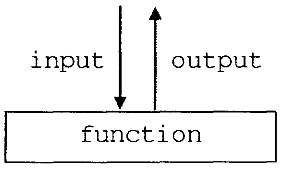


Figure C.1: Representation of a Function

Sub-Functions

A function can call other functions, then called sub-functions. Arrows departing from the bottom of the rectangle represent data passed as input to the sub-function called within the execution of the function. Arrows arriving to the bottom of the rectangle represent outputs of the sub-function returned to the function. In figure C.2, `function A` calls `sub-function B`.

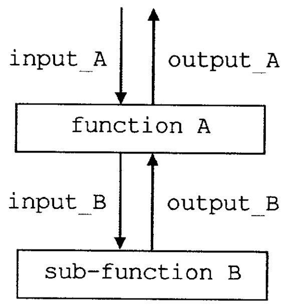


Figure C.2: Function calling a Sub-Function

Databases

Ovals represent data stored in a database. Data is stored and retrieved by functions. Arrows arriving to the oval represent data being stored in the database. Arrows departing from the oval represent data being retrieved from the database.

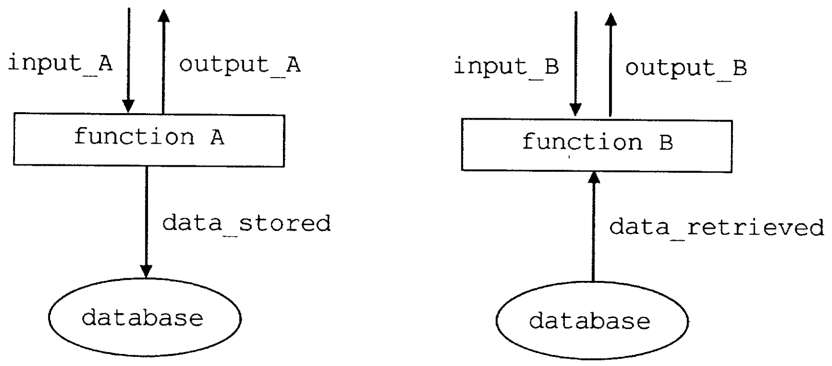


Figure C.3: Functions Interacting with a Database

Order of Operations

The diagrams are laid out to represent the order in which the main operations are carried out within a function. In the example shown in figure C.4, `function A` starts by calling `sub-function B`, then loads data from `database` and finally calls `sub-function C` before returning `output_A`.

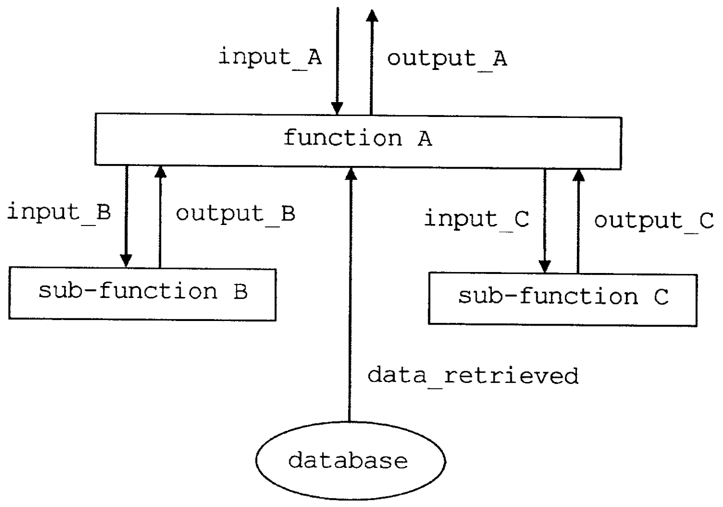


Figure C.4: Complex Function

Appendix D

Truss Optimization Program Code

Optimization Algorithm Objective Function

```
function[f]=objective(x);

[Mt,Ulist,Flist,w]=analyze(x);

f=abs(Ulist(find(Ulist(:,1)==dof),2));
```

Truss Analysis Process

```
function[Mt,Ulist,Flist,w]=analyze(x);

[Nt,DOFt,Mt]=define_truss(x);

cd('..\..\code');

[Kt]=stiffness(DOFt,Mt);

DOFf=transpose([list of fixed degrees of freedom]);
DOFc=transpose([list of control degrees of freedom]);

[rDOFt,DOFu,Fc,Fu,E]=specify(Kt,DOFt,DOFf,DOFc);
P=transpose([list of loads applied to control degrees of freedom]);
[Uc,Uu,Ulist,R,Flist]=solve(DOFc,DOFu,DOFf,Fc,Fu,E,P);
[Mt]=deformations(Mt,Ulist);
w=weight(Mt);

cd('..\problems\1');
```

Truss Definition

```
function[Nt,DOFt,Mt]=define_truss(x);

DOFt=transpose([list of all truss degrees of freedom]);

Nt=[tag x y z dofx dofy dofz];

dMt=[tag nodeA nodeB E A];

cd('..\..\code');

Mt=members(Nt,dMt);

cd('..\problems\p1');
```

Truss Members Properties

```
function[Mt]=members(Nt,dMt);

Mt=[];

for i=1:1:length(dMt(:,1));
    a=find(Nt(:,1)==dMt(i,2));
    b=find(Nt(:,1)==dMt(i,3));
    [L,T,P]=geometry(Nt(a,2),Nt(a,3),Nt(a,4),Nt(b,2),
    Nt(b,3),Nt(b,4));
    Mt=[Mt;dMt(i,1) Nt(a,:)  Nt(b,:)  L T P dMt(i,4) dMt(i,5)];
end;
```

Truss Members Geometry

```
function[L,T,P]=geometry(xa,ya,za,xb,yb,zb);

L=((xb-xa)^2+(yb-ya)^2+(zb-za)^2)^0.5;

Lh=((xb-xa)^2+(yb-ya)^2)^0.5;
```

```

if Lh>0;
    if yb>=ya;
        T=acos((xb-xa)/Lh)/pi*180;
    elseif yb<ya;
        T=-acos((xb-xa)/Lh)/pi*180;
    end;
elseif Lh==0;
    T=0;
end;

P=asin((zb-za)/L)/pi*180;
```

Truss Stiffness

```
function[Kt]=stiffness(DOFt,Mt);

Kt=zeros(length(DOFt));

for k=1:1:length(Mt(:,1));

    M=[Mt(k,:)];

    Km=member_stiffness(M(16), M(17), M(18), M(19), M(20));

    DOFm=[M(6), M(7), M(8), M(13), M(14), M(15)];

    for i=1:1:length(DOFm);

        a=find(DOFt==DOFm(i));

        for j=1:1:length(DOFm);

            b=find(DOFt==DOFm(j));

            Kt(a,b)=Kt(a,b)+Km(i,j);

        end;
    end;
end;
```

Member Stiffness

```
function[Km]=member_stiffness(L,T,P,E,A);

st=sin(T/180*pi);
ct=cos(T/180*pi);
sp=sin(P/180*pi);
cp=cos(P/180*pi);

line=[-ct*cp -st*cp -sp ct*cp st*cp sp];

Km=(E*A/L)*transpose(line)*line;
```

Matrix Reduction

```
function[rDOFt,DOFu,Fc,Fu,E]=reduce(Kt,DOFt,DOFf,DOFc);

DOFu=[];
for k=1:1:length(DOFt);
    if and(length(find(DOFf==DOFt(k)))==0,
            length(find(DOFc==DOFt(k)))==0);
        DOFu=[DOFu;DOFt(k)];
    end;
end;

rDOFt=[DOFc;DOFu;DOFf];

rKt=zeros(length(rDOFt));
for i=1:1:length(DOFt);
    a=find(rDOFt==DOFt(i));
    for j=1:1:length(DOFt);
        b=find(rDOFt==DOFt(j));
        rKt(a,b)=Kt(i,j);
    end;
end;

nc=length(DOFc);
nu=length(DOFu);
nf=length(DOFf);
nt=length(rDOFt);
```

```
if nu>=1;
    Ka=rKt(1:nc,1:nc);
    Kb=rKt(1:nc,(nc+1):(nc+nu));
    Kd=rKt((nc+1):(nc+nu),1:nc);
    Ke=rKt((nc+1):(nc+nu),(nc+1):(nc+nu));
    Kg=rKt((nc+nu+1):(nc+nu+nf),1:nc);
    Kh=rKt((nc+nu+1):(nc+nu+nf),(nc+1):(nc+nu));
    C=-inv(Ke)*Kd;
    Kc=Ka+Kb*C;
    Fc=inv(Kc);
    Fu=C*Fc;
    E=Kg*Fc+Kh*Fu;
elseif nu==0;
    Ka=rKt(1:nc,1:nc);
    Kg=rKt((nc+nu+1):(nc+nu+nf),1:nc);
    Fc=inv(Ka);
    Fu=[];
    E=Kg*Fc;
end;
```

Problem Solution

```
function[Uc,Uu,Ulist,R,Flist]=solve(DOFc,DOFu,DOFf,Fc,Fu,E,P);

Uc=Fc*P;
Uu=Fu*P;
R=E*P;

Ulist=[[DOFc,Uc];[DOFu,Uu];[DOFf,zeros(length(DOFf),1)]];
Flist=[[DOFc,P];[DOFu,zeros(length(DOFu),1)];[DOFf,R]];
```

Deformed Truss Analysis

```
function[Mt]=deformations(Mt,Ulist);

Mt=[Mt(:,1:20),zeros(length(Mt(:,1)),46-length(Mt(1,:)))];

for i=1:1:length(Mt(:,1));

    ua=Ulist(find(Ulist(:,1)==Mt(i,6)),2);
    va=Ulist(find(Ulist(:,1)==Mt(i,7)),2);
    wa=Ulist(find(Ulist(:,1)==Mt(i,8)),2);
    ub=Ulist(find(Ulist(:,1)==Mt(i,13)),2);
    vb=Ulist(find(Ulist(:,1)==Mt(i,14)),2);
    wb=Ulist(find(Ulist(:,1)==Mt(i,15)),2);
    xpa=Mt(i,3)+ua;
    ypa=Mt(i,4)+va;
    zpa=Mt(i,5)+wa;
    xpb=Mt(i,10)+ub;
    ypb=Mt(i,11)+vb;
    zpb=Mt(i,12)+wb;
    L=Mt(i,16);
    T=Mt(i,17);
    P=Mt(i,18);
    E=Mt(i,19);
    A=Mt(i,20);

    st=sin(T/180*pi);
    ct=cos(T/180*pi);
    sp=sin(P/180*pi);
    cp=cos(P/180*pi);

    dL=(ub-ua)*ct*cp+(vb-va)*st*cp+(wb-wa)*sp;
    strain=dL/L;
    stress=E*strain;
    F=A*stress;

    Mt(i,31:46)=[ua va wa ub vb wb xpa ypa zpa xpb ypb zpb dL
                strain stress F];

end;
```

Truss Weight

```
function[w]=weight(Mt);

w=0;

for i=1:1:length(Mt(:,1));
    w=w+Mt(i,16)*Mt(i,20)/12/12/12*490;
end;
```

Appendix E

Girder Optimization Program Code

Structural Cost

```
function[sc]=structural_cost(Y,all_V,all_M,P);

w=weight(P);
d=deflection(Y,all_V,all_M,P);
sc=700*d+5*w;
```

Weight

```
function[w]=weight(P);

w=0;

for i=1:1:length(P(:,1));
    w=w+2*(P(i,2)-P(i,1))*P(i,3)/(12^3)*490;
end;
```

Deflection

```
function[d]=deflection(Y,all_V,all_M,P);

Agg_D=[];
for r1=1:1:6;
    for r2=1:1:6;
        agg_d=aggregate_deflection(Y,all_V,all_M,r1,r2,P);
        Agg_D=[Agg_D;agg_d];
    end;
end;
d=max(Agg_D);
```

Aggregate Deflection

```
function[agg_d]=aggregate_deflection(Y,all_V,all_M,r1,r2,P);

M1=all_M(:,r1);
D1=displacement(Y,M1,P);
y1=40+6*r1;
d1=D1(find(Y==y1));

M2=all_M(:,6+r2);
M=M1+M2;
D2=displacement(Y,M,P);
y2=110+6*r2;
d2=D2(find(Y==y2));

agg_d=-(2*d1+d2);
```

Displacement

```
function[D]=displacement(Y,M,P);

E=29000;          % Steel Modulus of Elasticity (ksi)

nb_points=length(M(:,1));
A=[Y,zeros(nb_points,1),M,zeros(nb_points,2)];

% Find cross-section for each segment
for i=2:1:nb_points;
    A(i,2)=P(find(and(A(i-1,1)>=P(:,1),A(i,1)<=P(:,2))),4);
end;

% Calculate angle
A(1,4)=0;
for i=2:1:nb_points;
    A(i,4)=A(i-1,4)+(A(i,1)-A(i-1,1))*((A(i-1,3)+A(i,3))/2)/(E*A(i,2));
end;

% Calculate deflection
A(1,5)=0;
for i=2:1:nb_points;
    A(i,5)=A(i-1,5)+(A(i,1)-A(i-1,1))*((A(i-1,4)+A(i,4))/2);
end;

% Calculate initial angle
initial_angle=-A(nb_points,5)/A(nb_points,1);

% Re-calculate angle
A(1,4)=initial_angle;
for i=2:1:nb_points;
    A(i,4)=A(i-1,4)+(A(i,1)-A(i-1,1))*((A(i-1,3)+A(i,3))/2)/(E*A(i,2));
end;

% Re-calculate deflection
A(1,5)=0;
for i=2:1:nb_points;
    A(i,5)=A(i-1,5)+(A(i,1)-A(i-1,1))*((A(i-1,4)+A(i,4))/2);
end;

D=[A(:,5)];
```

Stresses

```
function[Sa,Ss]=stresses(Y,V_env,M_env,P);

nb_points=length(Y);
A=P(1,3:5);

% Find cross-section properties for each segment
for i=2:1:nb_points;
    A=[A;P(find(and(Y(i-1,1)>=P(:,1),Y(i,1)<=P(:,2))),3:5)];
end;

Sa=[];
Ss=[];

for i=1:1:length(Y);

    sa=M_env(i)*A(i,3)/A(i,2);
    Sa=[Sa;sa];

    ss=[V_env(i)/A(i,1)];
    Ss=[Ss;ss];

end;
```

Single Load

```
function[V,M]=single_load(Y,yload);

L=240;
d=36;
w=0.01737;

Ro=-w*d*(1-(2*yload+d)/(2*L));
Rl=-w*d*(2*yload+d)/(2*L);

V=[];
M=[];

```

```
for i=1:1:length(Y);
    y=Y(i);
    if y<=yload;
        v=-Ro;
        m=-Ro*y;
    elseif and(y>yload,y<=yload+d);
        v=-Ro+(Ro+Rl)*(y-yload)/d;
        m=-Ro*y+(Rl+Ro)/(2*d)*(y-yload)^2;
    elseif y>yload+d;
        v=Rl;
        m=-(Ro+Rl)*(yload+d/2)+Rl*y;
    end;
    V=[V;v];
    M=[M;m];
end;
```

Load Scenarios

```
function[all_V,all_M]=load_scenarios(Y);

all_V=[];
all_M=[];

for r1=1:1:6;
    y1=40+6*r1;
    [V,M]=single_load(Y,y1);
    all_V=[all_V,V];
    all_M=[all_M,M];
end;

for r2=1:1:6;
    y2=110+6*r2;
    [V,M]=single_load(Y,y2);
    all_V=[all_V,V];
    all_M=[all_M,M];
end;
```

Envelopes

```
function[V_env,M_env]=envelopes(Y);

[all_V,all_M]=load_scenarios(Y);

V_comb=[];
M_comb=[];
for r1=1:1:6;
    for r2=1:1:6;
        V_comb=[V_comb,[all_V(:,r1)+all_V(:,6+r2)]];
        M_comb=[M_comb,[all_M(:,r1)+all_M(:,6+r2)]];
    end;
end;

V_env=[];
M_env=[];

for i=1:1:length(V_comb(:,1));
    V_env=[V_env;max(abs(V_comb(i,:)))];
    M_env=[M_env;max(abs(M_comb(i,:)))];
end;
```

Appendix F

Hybrid Systems Analytical Solutions

Simple Stiffener

A symmetrical stiffener scheme with a single king post is considered first.

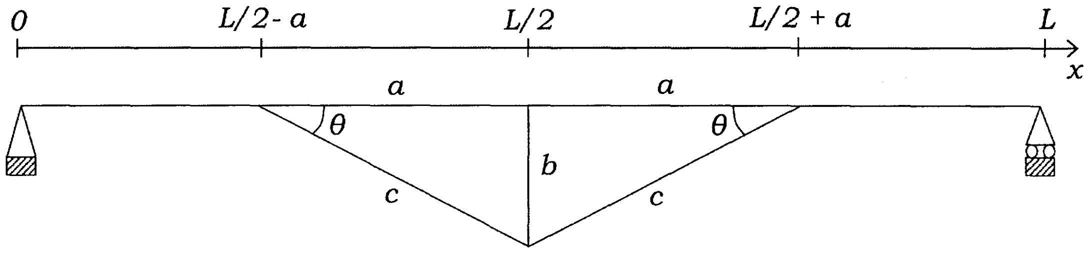


Figure F.1: Problem Geometry

The lengths a and b define the geometry. Then:

$$c = \sqrt{a^2 + b^2} \qquad \cos\theta = \frac{a}{c} \qquad \sin\theta = \frac{b}{c} \tag{F.1}$$

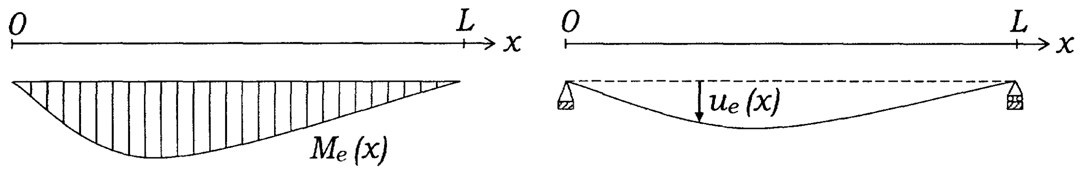


Figure F.2: External Load Action

The downward deflection u_e due to externally applied loads is calculated from the bending moment M_e created by these loads:

$$u_e = \int\int \frac{M_e}{EI} \quad \longrightarrow \quad u_e(x) \tag{F.2}$$

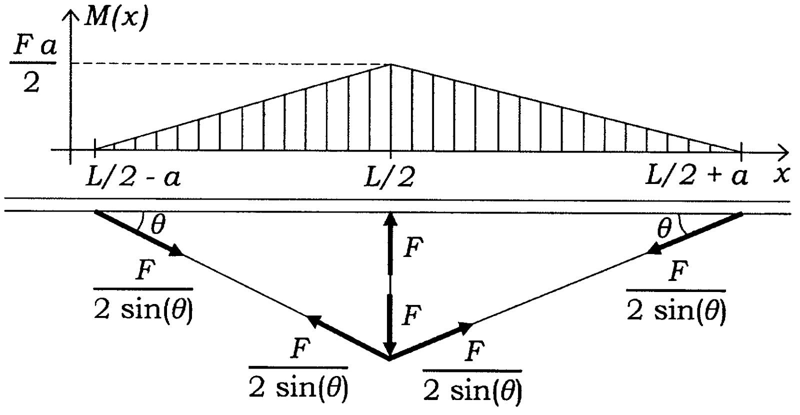

M(x)
$\frac{Fa}{2}$
L/2 - a
L/2
L/2 + a
x
θ
θ
F
F
$\frac{F}{2\sin(\theta)}$
$\frac{F}{2\sin(\theta)}$
$\frac{F}{2\sin(\theta)}$
$\frac{F}{2\sin(\theta)}$

Figure F.3: Forces in Stiffener and Moment Created on Beam

The force in the stiffener, F, is not known yet

The bending moment M_s created by the stiffener is:

$$\begin{cases} M_s(x) = 0 & \text{for} \quad 0 \leq x \leq \frac{L}{2} - a \\ M_s(x) = \frac{F}{2}\left(x - \frac{L}{2} + a\right) & \text{for} \quad \frac{L}{2} - a \leq x \leq \frac{L}{2} \\ M_s(x) = \frac{F}{2}\left(\frac{L}{2} + a - x\right) & \text{for} \quad \frac{L}{2} \leq x \leq \frac{L}{2} + a \\ M_s(x) = 0 & \text{for} \quad \frac{L}{2} + a \leq x \leq L \end{cases} \tag{F.3}$$

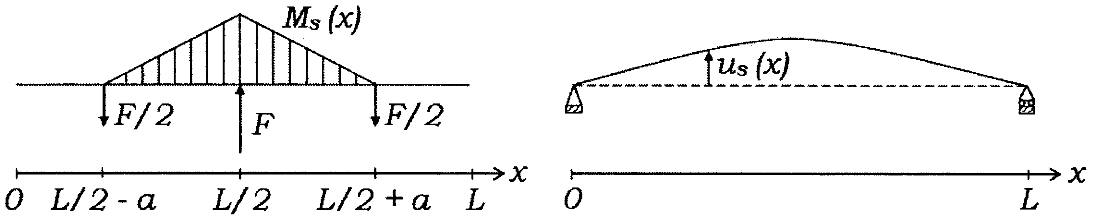


Figure F.4: Stiffener Action

The upward deflection u_s due to stiffener action is calculated from the bending moment M_s:

$$u_\mathrm{s} = \int\int \frac{M_\mathrm{s}}{EI} \quad \longrightarrow \quad u_\mathrm{s}(x) = F\,\overline{u_\mathrm{s}}(x) \tag{F.4}$$

F is the still unknown force in the stiffener.
$\overline{u_\mathrm{s}}$ is calculated and represents the deflection caused by a unit force in the stiffener. The actual deflection u, due to the combined action of external loads and stiffener reaction, is:

$$u = u_\mathrm{e} + u_\mathrm{s} \quad \longrightarrow \quad u(x) = u_\mathrm{e}(x) - F\,\overline{u_\mathrm{s}}(x) \tag{F.5}$$

The following notation is used:

$$\begin{aligned} u\left(\frac{L}{2} - a\right) &= u_1 = u_{\mathrm{e}1} - F\,\overline{u_{\mathrm{s}1}} \\ u\left(\frac{L}{2}\right) &= u_2 = u_{\mathrm{e}2} - F\,\overline{u_{\mathrm{s}2}} \\ u\left(\frac{L}{3} - a\right) &= u_1 = u_{\mathrm{e}3} - F\,\overline{u_{\mathrm{s}3}} \end{aligned} \tag{F.6}$$

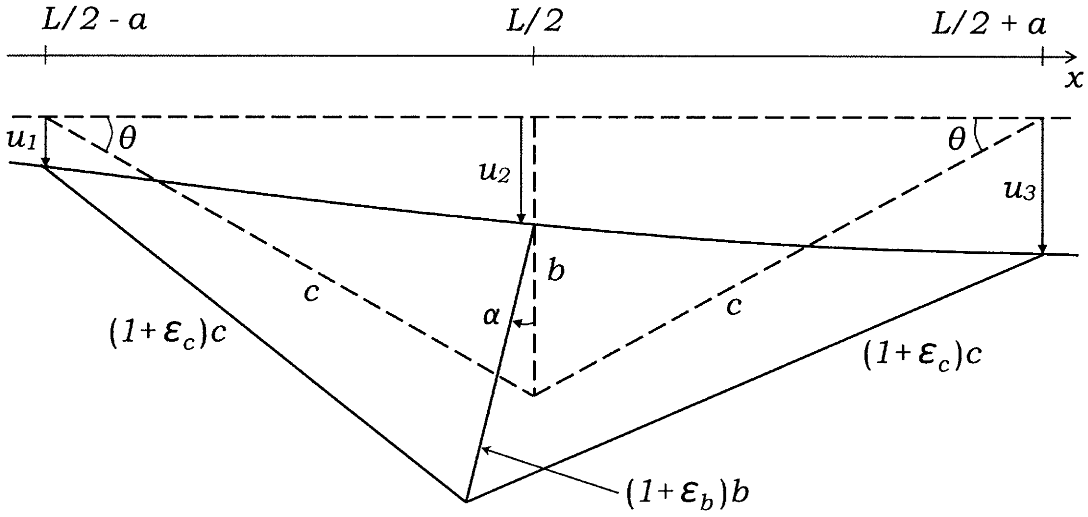


Figure F.5: System Deformation

Truss members deformations are related to beam displacements, considering the angle α small:

$$\epsilon_c\, c = (u_2 - u_1 + \epsilon_b\, b)\sin\theta - \alpha\, b\cos\theta \tag{F.7}$$

$$\epsilon_c\, c = (u_2 - u_3 + \epsilon_b\, b)\sin\theta + \alpha\, b\cos\theta \tag{F.8}$$

(F.7) and (F.8) are added to cancel the angle α:

$$\epsilon_c\, c = \left(u_2 - \frac{u_1 + u_3}{2} + \epsilon_b\, b\right)\sin\theta \tag{F.9}$$

Each truss member strain (see Fig. F.5) is related to the member force (see Fig. F.3) through the member stiffness, function of the member cross-sectional area:

$$\begin{aligned} \frac{F}{2\sin\theta} &= EA_c\,\epsilon_c & &\longrightarrow & \epsilon_c &= \frac{F}{2EA_c\sin\theta} \\ F &= -EA_b\,\epsilon_b & &\longrightarrow & \epsilon_b &= -\frac{F}{EA_b} \end{aligned} \tag{F.10}$$

In (F.9), displacements are decomposed using (F.6) and strains are expressed in terms of the member forces using (F.10).

$$\frac{Fc}{2EA_\mathrm{c}\sin\theta} = \left(u_\mathrm{e2} - F\,\overline{u_\mathrm{s2}} - \frac{u_\mathrm{e1} - F\,\overline{u_\mathrm{s1}} + u_\mathrm{e3} - F\,\overline{u_\mathrm{s3}}}{2} - \frac{Fb}{EA_\mathrm{b}}\right)\sin\theta \tag{F.11}$$

(F.11) is rearranged to be solved for F:

$$F\left(\overline{u_\mathrm{s2}} - \frac{\overline{u_\mathrm{s1}} + \overline{u_\mathrm{s3}}}{2} + \frac{c}{2\sin^2 EA_\mathrm{c}} + \frac{b}{EA_\mathrm{b}}\right) = u_\mathrm{e2} - \frac{u_\mathrm{e1} + u_\mathrm{e3}}{2} \tag{F.12}$$

Double Stiffener

A stiffener scheme with two king posts is considered.

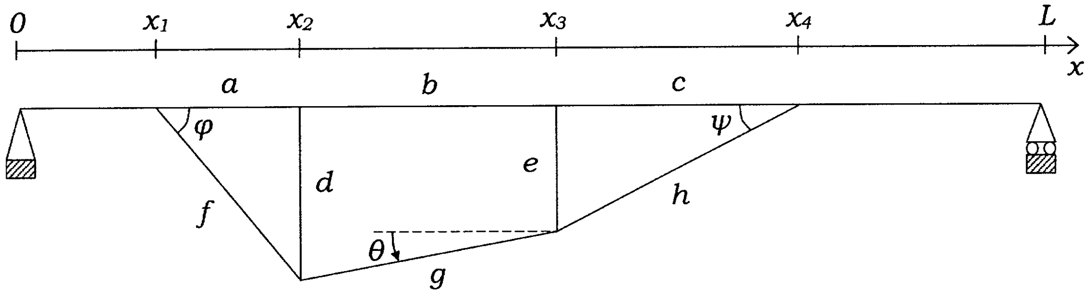


Figure F.6: Problem Geometry

To define the geometry, x_1, x_2, x_3, x_4, d and e can be chosen independently. Then:

$$a = x_2 - x_1 \qquad b = x_3 - x_2 \qquad a = x_4 - x_3$$
$$f = \sqrt{a^2 + d^2} \qquad g = \sqrt{b^2 + (e-d)^2} \qquad h = \sqrt{c^2 + e^2} \tag{F.13}$$
$$\cos\varphi = \frac{a}{f} \quad \sin\varphi = \frac{d}{f} \quad \cos\psi = \frac{c}{h} \quad \sin\psi = \frac{e}{h} \quad \cos\theta = \frac{b}{g} \quad \sin\theta = \frac{d-e}{g}$$

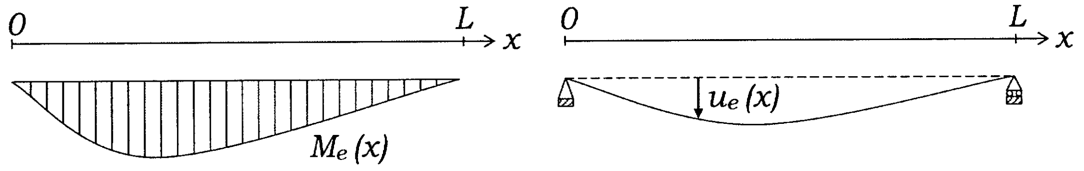


Figure F.7: External Load Action

The downward deflection u_e due to externally applied loads is calculated from the bending moment M_e created by these loads:

$$u_e = \int\int \frac{M_e}{EI} \tag{F.14}$$

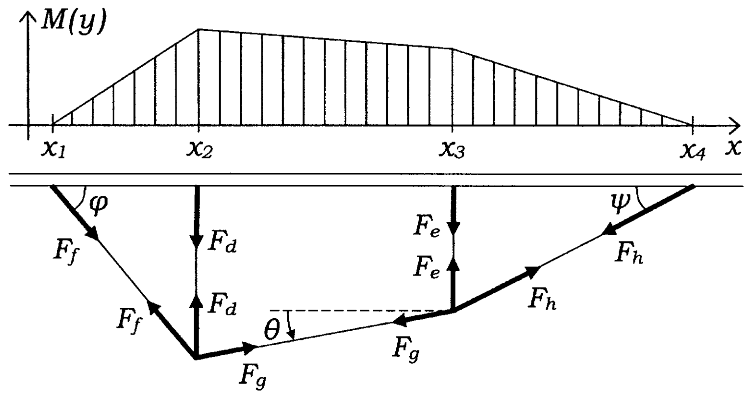


Figure F.8: Forces in Stiffener and Moment Created on Beam

The force F_g, acting in the bottom member, is chosen as the reference force F. By applying equilibrium at the nodes, all forces in the truss are expressed in terms of the reference force F:

$$F_d = -F\frac{\sin(\varphi+\theta)}{\cos\varphi} \qquad F_e = -F\frac{\sin(\psi-\theta)}{\cos\psi}$$
$$F_f = F\frac{\cos\theta}{\cos\varphi} \qquad F_g = F \qquad F_h = F\frac{\cos\theta}{\cos\psi} \tag{F.15}$$

The bending moment M_s due to stiffener reaction is:

$$\begin{cases} M_s(x) = 0 & \text{for} \quad 0 \le x \le x_1 \\ M_s(x) = F\cos\theta\tan\varphi\,(x-x_1) & \text{for} \quad x_1 \le x \le x_2 \\ M_s(x) = F\cos\theta\left(a\tan\varphi\left(1-\frac{x-x_2}{b}\right)+c\tan\psi\left(\frac{x-x_2}{b}\right)\right) & \text{for} \quad x_2 \le x \le x_3 \\ M_s(x) = F\cos\theta\tan\psi\,(c-x+x_3) & \text{for} \quad x_3 \le x \le x_4 \\ M_s(x) = 0 & \text{for} \quad x_4 \le x \le L \end{cases} \tag{F.16}$$

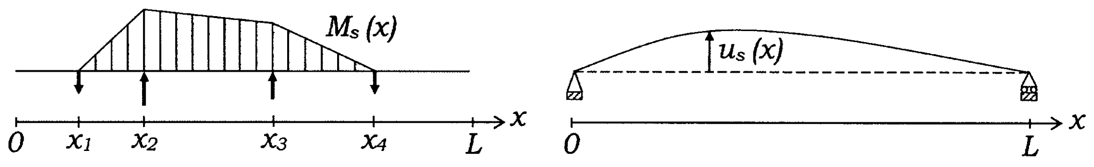


Figure F.9: Stiffener Action

The upward deflection u_s due to stiffener action is calculated from the bending moment M_s:

$$u_\mathrm{s} = \int\int \frac{M_\mathrm{s}}{EI} \quad \longrightarrow \quad u_\mathrm{s}(x) = F\,\overline{u_\mathrm{s}}(x) \tag{F.17}$$

F is the still unknown reference force in the stiffener.
$\overline{u_\mathrm{s}}$ is calculated and represents the deflection caused by a unit reference force F in the stiffener.

The actual deflection u, due to the combined action of external loads and stiffener reaction, is:

$$u = u_\mathrm{e} - u_\mathrm{s} \quad \longrightarrow \quad u(x) = u_\mathrm{e}(x) - F\,\overline{u_\mathrm{s}}(x) \tag{F.18}$$

The following notation is used:

$$\begin{aligned} u(x_1) &= u_1 = u_\mathrm{e1} - F\,\overline{u_\mathrm{s1}} \qquad & u(x_2) &= u_2 = u_\mathrm{e2} - F\,\overline{u_\mathrm{s2}} \\ u(x_3) &= u_3 = u_\mathrm{e3} - F\,\overline{u_\mathrm{s3}} \qquad & u(x_4) &= u_4 = u_\mathrm{e4} - F\,\overline{u_\mathrm{s4}} \end{aligned} \tag{F.19}$$

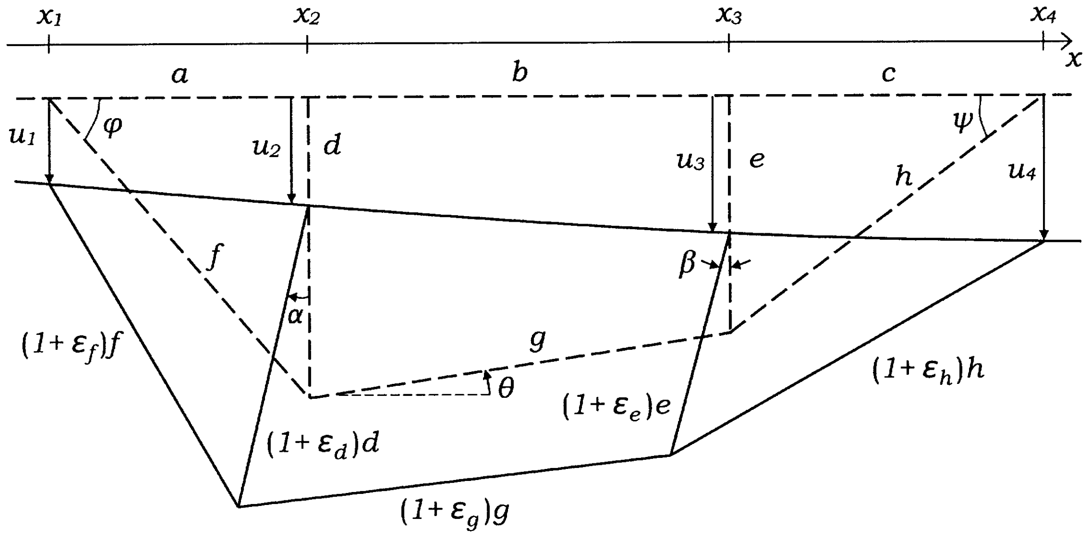


Figure F.10: System Deformation

Truss members deformations are related to beam displacements, considering angles α and β small:

$$\epsilon_f f = (u_2 - u_1 + \epsilon_d d) \sin\varphi - \alpha d \cos\varphi \tag{F.20}$$

$$\epsilon_h h = (u_3 - u_4 + \epsilon_e e) \sin\psi + \beta e \cos\psi \tag{F.21}$$

$$\epsilon_g g = (u_2 + \epsilon_d d - u_3 - \epsilon_e e) \sin\theta + (\alpha d - \beta e) \cos\theta \tag{F.22}$$

(F.20), (F.21) and (F.22) are combined to cancel angles α and β:

$$\begin{aligned} \epsilon_g g &= (u_2 + \epsilon_d d - u_3 - \epsilon_e e) \sin\theta \\ &+ \left((u_2 - u_1 + \epsilon_d d) \tan\varphi - \frac{\epsilon_f f}{\cos\varphi} \right) \cos\theta \\ &+ \left((u_3 - u_4 + \epsilon_e e) \tan\psi - \frac{\epsilon_h h}{\cos\psi} \right) \cos\theta \end{aligned} \tag{F.23}$$

Each member strain ϵ_i is related to the member force F_i through the member stiffness, function of the member cross-sectional area A_i:

$$\epsilon_d = \frac{F_d}{EA_d} \qquad \epsilon_e = \frac{F_e}{EA_e} \qquad \epsilon_f = \frac{F_d}{EA_f} \qquad \epsilon_g = \frac{F_g}{EA_g} \qquad \epsilon_h = \frac{F_h}{EA_h} \tag{F.24}$$

In (F.23), displacements are decomposed using (F.19) and strains are expressed in terms of the member forces using (F.24). These member forces are then expressed in terms of the reference force F using (F.15). The resulting equation can be solved for F:

$$F\left(\begin{array}{lllll} & (\overline{u_{\mathrm{s}2}}-\overline{u_{\mathrm{s}3}})\tan\theta & + & \dfrac{g}{EA_{\mathrm{g}}}\dfrac{1}{\cos\theta} & \\ + & (\overline{u_{\mathrm{s}2}}-\overline{u_{\mathrm{s}1}})\tan\varphi & + & \dfrac{f}{EA_{\mathrm{f}}}\dfrac{\cos\theta}{\cos^2\varphi} & + \quad \dfrac{d}{EA_{\mathrm{d}}}\dfrac{\sin(\varphi+\theta)}{\cos\varphi}(\tan\varphi+\tan\theta) \\ + & (\overline{u_{\mathrm{s}3}}-\overline{u_{\mathrm{s}4}})\tan\psi & + & \dfrac{h}{EA_{\mathrm{h}}}\dfrac{\cos\theta}{\cos^2\psi} & + \quad \dfrac{e}{EA_{\mathrm{e}}}\dfrac{\sin(\psi-\theta)}{\cos\psi}(\tan\psi-\tan\theta) \end{array}\right)$$

$$= \quad (u_{\mathrm{e}2}-u_{\mathrm{e}3})\sin\theta \quad + \quad (u_{\mathrm{e}2}-u_{\mathrm{e}1})\cos\theta\tan\varphi \quad + \quad (u_{\mathrm{e}3}-u_{\mathrm{e}4})\cos\theta\tan\psi \qquad (\mathrm{F}.25)$$

List of Figures

List of Tables

Bibliography

Alimoradi, A., S. Pezeshk, and C. M. Foley (2007). Probabilistic Performance-based Optimal Design of Steel Moment-resisting Frames. II: Applications. *Journal of Structural Engineering 133*(6), 767–776.

American Society of Civil Engineers / American Institute of Steel Construction (2008). Student Steel Bridge Competition 2009 Rules.

Arciszewski, T. and W. Ziarko (1991). Structural Optimization: Case-based Approach. *Journal of Computing in Civil Engineering 5*(2), 159–174.

Arora, J. S. (2000). Methods for Discrete Variable Structural Optimization. In M. Elgaaly (Ed.), *Structures 2000 - Part of Advanced Technology in Structural Engineering*, Proceedings of the 2000 Structures Congress.

Baldock, R., K. Shea, and D. Eley (2005). Evolving Optimized Braced Steel Frameworks for Tall Buildings using Modified Pattern Search. In L. Soibelman and F. Pea-Mora (Eds.), *Computing in Civil Engineering*, Proceedings of the 2005 ASCE International Conference on Computing in Civil Engineering.

Balling, R. J. (1991). Optimal Steel Frame Design by Simulated Annealing. *Journal of Structural Engineering 117*(6), 1780–1795.

Balling, R. J. and X. Yao (1997). Optimization of Reinforced Concrete Frames. *Journal of Structural Engineering 123*(2), 193–202.

Camp, C. V. (2007). Design of Space Trusses using Big Bang-Big Crunch Optimization. *Journal of Structural Engineering 133*(7), 999–1008.

Camp, C. V., B. J. Bichon, and S. P. Stovall (2004). Design of Low-weight Steel Frames using Ant Colony Optimization. In G. E. Blandford (Ed.), *Structures 2004*

- *Building On The Past: Securing The Future*, Proceedings of the 2004 Structures Congress.

Camp, C. V., B. J. Bichon, and S. P. Stovall (2005). Design of Steel Frames using Ant Colony Optimization. *Journal of Structural Engineering 131*(3), 369–379.

Chan, C.-M. and Q. Wang (2006). Nonlinear Stiffness Design Optimization of Tall Reinforced Concrete Buildings under Service Loads. *Journal of Structural Engineering 132*(6), 978–990.

Cheng, G., X. Guo, and N. Olhoff (2000). New Formulation for Truss Topology Optimization Problems under Buckling Constraints. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua.* Kluwer Academic Publishers, Netherlands.

Cohn, M. Z. and A. S. Dinovitzer (2004). Application of Structural Optimization. *Journal of Structural Engineering 120*(2), 617–650.

Cohn, M. Z. and Z. Lounis (1994). Optimal Design of Structural Concrete Bridge Systems. *Journal of Structural Engineering 120*(9), 2653–2674.

Foley, C. M., S. Pezeshk, and A. Alimoradi (2007). Probabilistic Performance-based Optimal Design of Steel Moment-resisting Frames. I: Formulation. *Journal of Structural Engineering 133*(6), 757–766.

Foley, C. M. and D. Schinler (2001). Optimized Design of Partially and Fully-restrained Steel Frames using Distributed Plasticity. In P. C. Chang (Ed.), *Structures 2001 - A Structural Engineering Odyssey*, Proceedings of the 2001 Structural Congress and Exposition.

Hajela, P. and S. Vittal (2000). Evolutionary Computing and Structural Topology Optimization - A State of the Art Assessment. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua.* Kluwer Academic Publishers, Netherlands.

Kargahi, M. and J. C. Anderson (2006a). Structural Weight Optimization of Frames using Tabu Search. II: Evaluation and Seismic Performance. *Journal of Structural Engineering 132*(12), 1869–1879.

Kargahi, M. and J. C. Anderson (2006b). Structural Weight Optimization with Tabu Search. In R. B. Malla, W. K. Binienda, and A. K. Maji (Eds.), *Earth and Space 2006: Engineering, Construction, and Operations in Challenging Environment*, Proceedings of 10th Biennial International Conference on Engineering, Construction, and Operations in Challenging Environments.

Kargahi, M., J. C. Anderson, and M. M. Dessouky (2006). Structural Weight Optimization of Frames using Tabu Search. I: Optimization Procedure. *Journal of Structural Engineering 132*(12), 1858–1868.

Kirsch, U. and P. Papalambros (2000). Structural Reanalysis for Topological Modifications - A Unified Approach. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua.* Kluwer Academic Publishers, Netherlands.

Kost, B. (2003). Evolutionary Shape Optimization with Self-adapting Mutation Distribution based on the Cholesky Decomposition. In C. A. Brebbia, M. E. M. El-Sayed, and S. Hernandez (Eds.), *Computer Aided Optimum Design of Structures VIII.* WIT Press, UK.

Kost, B. and B. Baumann (2001). Building up Structures by means of Stochastic Topology Optimization. In S. Hernandez and C. A. Brebbia (Eds.), *Computer Aided Optimum Design of Structures VII.* WIT Press, UK.

Koumousis, V. K. and P. G. Georgiou (1994). A Genetic Algorithm for Structural Optimization of Steel Truss Roofs. *Journal of Computing in Civil Engineering 8*(3), 309–325.

Kripka, M. and H. M. C. C. Antunes (2001). Optimum Support Positions in Building Grillages. In S. Hernandez and C. A. Brebbia (Eds.), *Computer Aided Optimum Design of Structures VII.* WIT Press, UK.

Lee, S.-L. and P. K. Basu (1992). Bracing Requirements of Plane Frames. *Journal of Structural Engineering 118*(6), 1527–1546.

Liang, Q. Q., Y. M. Xie, and G. P. Steven (2000). Optimal Topology Design of Bracing Systems for Multistory Steel Frames. *Journal of Structural Engineering 126*(7), 823–829.

Liu, H., B. W. Schafer, and T. Igusa (2004). Cold-formed Steel Member Cross-section Shape Optimization by Knowledge-based Global Optimization Method. In G. E. Blandford (Ed.), *Structures 2004 - Building on the Past: Securing the Future*, Proceedings of the 2004 Structures Congress.

Merello, R. (2006). Design of a Building Structural Skin Using Multi-objective Optimization Techniques. Master Thesis, Department of Civil and Environmental Engineering, Massachusetts Institute of Technology.

Mijar, A. R., C. C. Swan, J. S. Arora, and I. Kosaka (1998). Continuum Topology Optimization for Concept Design of Frame Bracing Systems. *Journal of Structural Engineering 124*(5), 541–550.

Mróz, Z. and D. Bojczuk (2000). Topological Derivative and its Application in Optimal Design of Truss and Beam Structures for Displacement, Stress and Buckling Constraints. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua*. Kluwer Academic Publishers, Netherlands.

Park, H. S., Y. H. Kwon, J. H. Seo, and B.-H. Woo (2006). Distributed Hybrid Genetic Algorithms for Structural Optimization on a PC Cluster. *Journal of Structural Engineering 132*(12), 1890–1897.

Pezeshk, S., C. V. Camp, and D. Chen (2000). Design of Nonlinear Framed Structures using Genetic Optimization. *Journal of Structural Engineering 126*(3), 382–388.

Place, W., O. Ferm, T. Howard, and M. Williard (2001). Computer Optimization of Innovative Steel Arena Structure Illuminated with Natural Light. In S. Hernandez and C. A. Brebbia (Eds.), *Computer Aided Optimum Design of Structures VII*. WIT Press, UK.

Rahmatalla, S. and C. C. Swan (2003). Form Finding of Sparse Structures with Continuum Topology Optimization. *Journal of Structural Engineering 129*(12), 1707–1716.

Rajeev, S. and C. S. Krishnamoorthy (1992). Discrete Optimization of Structures using Genetic Algorithms. *Journal of Structural Engineering 118*(5), 1233–1250.

Rozvany, G. I. (2000). Problem Classes, Solution Strategies and Unified Terminology of FE-based Topology Optimization. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua.* Kluwer Academic Publishers, Netherlands.

Saito, T., T. Tamaki, and E. Kita (2003). Continuum Structural Design using Local Rule. In C. A. Brebbia, M. E. M. El-Sayed, and S. Hernandez (Eds.), *Computer Aided Optimum Design of Structures VIII.* WIT Press, UK.

Sigmund, O., T. Buhl, and C. B. W. Pedersen (2000). On the Influence of Geometrical Non-linearities in Topology Optimization. In G. I. N. Rozvany and N. Olhoff (Eds.), *Topology Optimization of Structures and Composite Continua.* Kluwer Academic Publishers, Netherlands.

Sues, R. H., M. A. Cesare, S. S. Pageau, and J. Y. Wu (2001). Reliability-based Optimization considering Manufacturing and Operational Uncertainties. *Journal of Aerospace Engineering 14*(4), 166–174.

Swan, C. C. and S. Rahmatalla (2001). Does Continuum Topology Optimization Have a Place in Design of Large-scale Structures? In P. C. Chang (Ed.), *Structures 2001 - A Structural Engineering Odyssey,* Proceedings of the 2001 Structural Congress and Exposition.

Takada, T., Y. Kohama, and A. Miyamura (2001). Optimization of Shear Wall Allocation in 3D Frames by Branch-and-bound Method. In S. Hernandez and C. A. Brebbia (Eds.), *Computer Aided Optimum Design of Structures VII.* WIT Press, UK.

Teughels, A., J. Maeck, and G. D. Roeck (2001). A Finite Element Model Updating Method using Experimental Modal Parameters on a Railway Bridge. In S. Hernandez and C. A. Brebbia (Eds.), *Computer Aided Optimum Design of Structures VII.* WIT Press, UK.

The MathWorks$^{\text{TM}}$ (2007a). Genetic Algorithm and Direct Search Toolbox$^{\text{TM}}$ 2 User's Guide.

The MathWorks$^{\text{TM}}$ (2007b). Optimization Toolbox$^{\text{TM}}$ 3 User's Guide.

van de Lindt, J. W. and T. N. Dao (2007). Evolutionary Algorithm for Performance-based Shear Wall Placement in Buildings subjected to Multiple Load Types. *Journal of Structural Engineering 133*(8), 1156–1167.

van Steirteghem, J., C. Buyl, W. P. de Wilde, and P. Samyn (2003). Morphological Optimisation of 2D Stayed Columns. In C. A. Brebbia, M. E. M. El-Sayed, and S. Hernandez (Eds.), *Computer Aided Optimum Design of Structures VIII.* WIT Press, UK.

Šilih, S. and S. Kravanja (2003). Topology, Shape and Standard Dimension Optimization of Trusses. In C. A. Brebbia, M. E. M. El-Sayed, and S. Hernandez (Eds.), *Computer Aided Optimum Design of Structures VIII.* WIT Press, UK.

Wang, Q. and J. S. Arora (2006). Alternative Formulations for Structural Optimization: An Evaluation using Frames. *Journal of Structural Engineering 132*(12), 1880–1889.

Xu, L., Y. Gong, and D. E. Grierson (2006). Seismic Design Optimization of Steel Building Frameworks. *Journal of Structural Engineering 132*(2), 277–286.

Xu, L. and D. E. Grierson (1993). Computer-automated Design of Semirigid Steel Frameworks. *Journal of Structural Engineering 119*(6), 1740–1760.

Zou, X. K., C. M. Chan, G. Li, and Q. Wang (2007). Multiobjective Optimization for Performance-based Design of Reinforced Concrete Frames. *Journal of Structural Engineering 133*(10), 1462–1474.