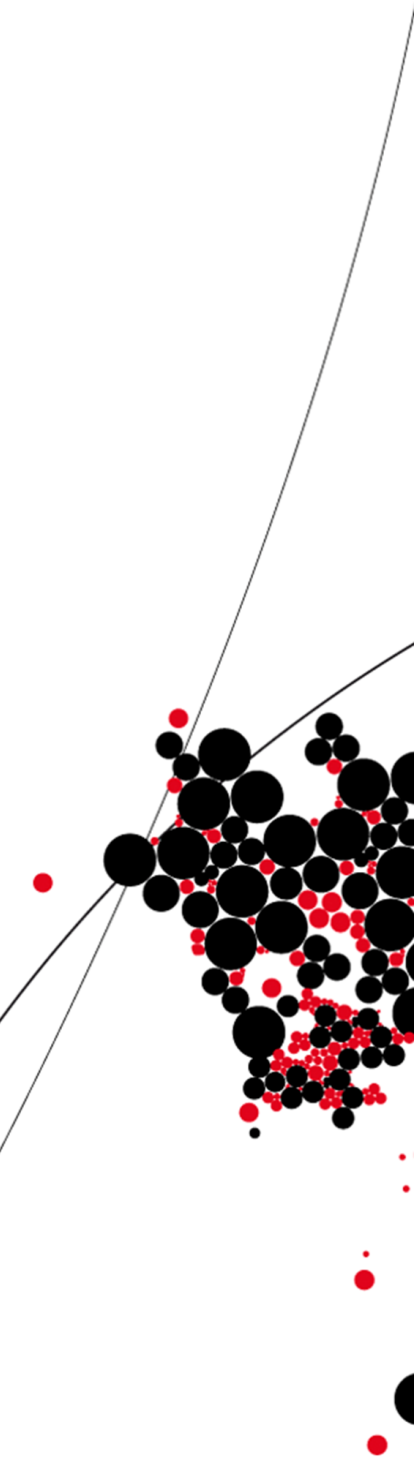
UNIVERSITY OF TWENTE.

**Mechanical Engineering**
**Faculty of Engineering Technology**

# Design Optimization Strategies for Geometrically Nonlinearly Deforming Structures using Analytical Gradients

**Tobias Posthumus**

**M.Sc. Thesis**
**July 2024**



**Exam commitee:**
prof. dr. ir. A. H. van den Boogaard
dr. ir. G. T. Havinga
dr. ir. J. J. de Jong EngD

Nonlinear Solid Mechanics
Faculty of Engineering Technology
University of Twente

**Abstract**

Optimization of nonlinearly deforming structures requires solving the system state variables while searching for the optimal design variables. For large structures with numerous state variables or a high number of design variables, using computationally efficient strategies in the optimization process becomes crucial. Efficient gradient-based optimization can be achieved by using analytical gradients. Creating an algorithm to systematically compute analytical gradients can be cumbersome and time-consuming, but significantly increases optimization speed.

In this work, two optimization strategies are investigated: nested and SAND (simultaneous analysis and design) optimization. In the nested approach, the state variables are solved in a nested function for each optimization iteration. In the SAND approach, the state variables are solved along with the design variables. The nested approach involves an implicit relationship between state and design variables. Here, the direct and adjoint methods are used.

The research shows that analytical gradient computation is several orders of magnitude faster than the numerical approach using finite differences, with increasing effect for optimization problems with more design variables.

Additionally, the work shows that using analytical gradients for both the nested and SAND approaches yields feasible and efficient optimization results for complex structures with up to hundreds of design variables and more than a thousand state variables.

Obtaining analytical gradients systematically proves to be a worthwhile investment, as it enables the optimization of far more complex systems than current gradient-based optimization using finite differences can achieve within a reasonable time.

# Nomenclature

| | |
|---|---|
| $\beta$ | Angle of an element with the $x$ axis of the global coordinate system [-]. |
| $\boldsymbol{\lambda}$ | Lagrange multipliers |
| $\boldsymbol{\lambda}_e$ | Lagrange multipliers, representing the elemental reaction forces [N]. |
| $\boldsymbol{\lambda}_n$ | Lagrange multipliers, representing the nodal reaction forces [N]. |
| $\boldsymbol{C}$ | Compliance matrix [m/N]. |
| $\boldsymbol{F}$ | External forces [N]. |
| $\boldsymbol{F}_p$ | Perturbation force [N]. |
| $\boldsymbol{g}$ | Constraint functions. |
| $\boldsymbol{g}_e$ | Element constraints [m]. |
| $\boldsymbol{g}_n$ | Nodal constraints [m]. |
| $\boldsymbol{h}$ | Equality constraints. |
| $\boldsymbol{J}$ | Jacobian of residual equations w.r.t the solution vector. |
| $\boldsymbol{L}$ | The Lagrangian used in the adjoint method. |
| $\boldsymbol{p_0}$ | Initial guess for the design variables. |
| $\boldsymbol{p}$ | Design variables. |
| $\boldsymbol{Q}$ | Prescribed displacements [m]. |
| $\boldsymbol{q}$ | Nodal displacements and element elongations [m]. |
| $\boldsymbol{q}_e$ | Element elongations [m]. |
| $\boldsymbol{q}_n$ | Nodal displacements [m]. |
| $\boldsymbol{r}$ | Residual equations. |

$\boldsymbol{u}$ Solution vector of the residual equations.

$\Delta L$ Change in length [m].

$\Delta x$ Actual $x$ difference between two nodes [m].

$\Delta x_0$ Initial $x$ difference between two nodes [m].

$\Delta y$ Actual $y$ difference between two nodes [m].

$\Delta y_0$ Initial $y$ difference between two nodes [m].

$\epsilon$ Strain [-].

$\mathcal{L}$ The Lagrangian used in the potential energy formulation [mJ].

$\Pi$ Potential energy [mJ].

$\rho$ Density [kg/m$^3$].

$\sigma$ Stress [Pa].

$\sigma_{max}$ Maximum allowable stress [Pa].

$A$ Area [m$^2$].

$E$ Young's Modulus [Pa].

$f$ objective function.

$K$ Stiffness matrix [N/m].

$k$ Element stiffness [N/m].

$L$ Actual length [m].

$L_0$ Initial length [m].

$L_x$ Length test structure [m].

$L_y$ Height test structure [m].

$M$ Mass [kg].

$n_p$ Number of design variables.

$ndiv_x$ Number of divisions of the test structure in $x$-direction.

$ndiv_y$ Number of divisions of the test structure in $y$-direction.

$nDOF$ Number of degrees of freedom.

$nEl$ Number of elements.

$nElCo$ Number of element constraint equations.

$nElEq$ Number of element equilibrium equations.

$nNo$ Number of nodes.

$nNoCo$ Number of nodal constraint equations.

$nNoEq$ Number of nodal equilibrium equations.

$nUnknown$ Number of elements in the solution vector, $\boldsymbol{u}$.

$u$ Nodal displacement in $x$ direction.

$v$ Nodal displacement in $y$ direction.

$W_{ext}$ External work [mJ].

$W_{int}$ Internal work or strain energy [mJ].

$X$ Actual $x$ coordinate.

$x$ Initial $x$ coordinate.

$Y$ Actual $y$ coordinate.

$y$ Initial $y$ coordinate.

d Nodal displacement [m].

$\mathrm{d}_{max}$ Maximum nodal displacement [m].

ElCo Set of elemental constraint equation numbers w.r.t. the residual equations.

ElEq Set of elemental equilibrium equation numbers w.r.t. the residual equations.

NoCo Set of nodal constraint equation numbers w.r.t. the residual equations.

NoEq Set of nodal equilibrium equation numbers w.r.t. the residual equations.

# Contents


**1 Introduction** **9**

1.1 Context . . . 9
1.2 Problem definition . . . 10
1.3 Research goals . . . 11
1.4 Scope . . . 11
1.4.1 Structures and elements . . . 11
1.4.2 Gradients . . . 12
1.4.3 Nonlinearities . . . 13
1.4.4 Optimization . . . 14
1.5 Outline . . . 14

**2 Flexible multibody approach** **15**

2.1 Problem introduction . . . 15
2.2 Potential energy formulation . . . 16
2.3 Newton-Rahpson . . . 19
2.4 Residual equations . . . 20
2.5 Example . . . 21
2.5.1 Truss definition . . . 21
2.5.2 Model equations . . . 22

**3 Optimization** **28**

3.1 Nested approach . . . 28
3.2 SAND approach . . . 29
3.3 fmincon . . . 29
3.4 Scaling and normalization . . . 30

**4 Theory of gradient computation** **32**

4.1 Numerical gradient computation . . . 32
4.1.1 Finite difference . . . 32
4.2 Analytical gradient computation . . . 33
4.2.1 Nested . . . 33

4.2.2 SAND . . . . . 37

**5 Implementation of analytical gradients** **39**
5.1 Theoretical framework . . . . . 39
5.2 Partial derivatives of the truss length . . . . . 40
5.3 Partial derivatives of the residual function . . . . . 42
5.4 Objective and constraint functions . . . . . 45
5.4.1 Mass objective . . . . . 45
5.4.2 Stress constraint . . . . . 46
5.4.3 Compliance objective . . . . . 47
5.4.4 Displacement constraint . . . . . 49
5.5 Concluding remarks on the analytical gradients . . . . . 50

**6 Method** **51**
6.1 Test problem formulation . . . . . 51
6.2 Gradients . . . . . 52
6.3 Optimization . . . . . 53

**7 Results** **54**
7.1 Gradients . . . . . 54
7.1.1 Accuracy of gradients . . . . . 54
7.1.2 Direct method vs finite difference . . . . . 55
7.1.3 Direct method vs Adjoint method . . . . . 57
7.2 Optimization . . . . . 59
7.2.1 Scaling . . . . . 59
7.2.2 Increasing complexity . . . . . 60
7.2.3 Convergence . . . . . 62
7.2.4 Failed runs . . . . . 64
7.2.5 Constraints . . . . . 65
7.2.6 The problem solution . . . . . 66
7.3 Bridge . . . . . 66
7.3.1 Optimization problem . . . . . 67
7.3.2 Results . . . . . 67

**8 Conclusion** **70**

**9 Recommendations** **71**

# Chapter 1

# Introduction

## 1.1 Context

Optimization allows engineers to explore a wide range of design possibilities and configurations, leading to innovative solutions that may not be obvious through traditional trial-and-error approaches. By iteratively improving designs based on specified objectives and constraints, optimization methods can significantly enhance the performance of structures, mechanisms and systems. This includes reducing costs, reducing weight, improving structural strength, and increasing overall performance.

Numerical computing and the development of optimization algorithms allows optimization problems to increase in complexity, thereby allowing models to more accurately represent reality [1]. As optimization algorithms become more efficient, the computational resources can be allocated to optimize more detailed and complex models. This increase in model detail and complexity enhances solution realism, ultimately leading to more effective implementations in practical applications.

Gradient-based optimization algorithms use the gradient of the objective and constraint functions to predict the direction of the next step in the solution space to eventually converge to an optimal solution. The gradients are commonly computed using finite differences, yet this approach proves inefficient due to the necessity of approximating each design variable's gradient individually.

Analytical gradients avoid this issue, since the gradients with respect to all design variables are computed at once. Hence analytical gradients can potentially offer increased optimization efficiency. The challenge lies in obtaining these analytical gradients, as it requires a thorough understanding of the model equations, design variables, and objective/constraint functions. In this report, the terms "model equations" and "residual equations" will be used interchangeably to refer to the set of equilibrium and constraint equations of the system. There are two types of constraint equations: the model constraint equations, and the optimization constraint equations. The context will make it

evident which set of constraint equations is being referred to. A set of analytical gradients must be systematically determined, such that all structures within the defined domain can automatically be optimized, regardless of variations in shape, size, loading conditions, objectives, and constraints. Hence, the possible constraints, objectives, and design variables must be selected in the beginning.
In this work, two main optimization approaches are used: nested and SAND (simultaneous analysis and design). The nested approach incorporates a nested function to solve the model equations. Once the system is solved for a specific set of design variables, the optimization solver evaluates the objective and constraint functions, along with their gradients if needed, and decides whether to initiate another iteration or terminate the optimization process.
The SAND approach solves the system state variables (for the model used in this work: the nodal/elemental displacements and nodal/elemental reaction forces) together with the design variables. The residual equations are integrated as equality constraints. This results in a larger optimization problem compared to the nested approach. However, the advantage lies in not needing to solve the residual equations for each optimization step, providing greater flexibility during the optimization process. Both approaches are expected to benefit significantly from using analytical gradients. The fundamental basis for gradient computation is similar for both approaches. However, the nested optimization is performed under the requirement that the model equations are satisfied, meaning that the optimization gradients are to be determined conditionally to the residual functions remaining constant. In other words, the optimization gradient must account for the change in state variables caused by a change in design variables. The two methods used in this research to achieve that goal are the direct and adjoint methods.

## 1.2 Problem definition

Optimizing structures that are able to capture nonlinear behavior, for example to allow large displacements, are computationally expensive. This is because the nonlinear model equations require iterative solution methods. When optimizing a structure a substantial portion of computational time is devoted to solving these nonlinear model equations. Determining optimization gradients using finite differences relies on evaluating the model at its current design and a set of perturbations thereof. With the number or required perturbations being equal to the number of design variables in the optimization problem, the computational cost of gradient computational increases linearly with the number of design variables.
This research investigates the potential computational efficiency gains of using analytic gradients instead of finite difference approximations for gradient-based optimization in the context of static nonlinear structural mechanics. To achieve these results, a spe-

cially designed flexible multibody model is presented. The flexible multibody approach is validated by comparing the results with those generated using the specialized software package SPACAR [2]. Two different approaches are tested: the nested approach and the SAND approach. Additionally, two solvers incorporated in MATLAB's fmincon function are evaluated: sequential quadratic programming (SQP) and interior-point (IP). These methods are tested on a series of test structures with increasing complexity. The methods can be compared in terms of efficiency, robustness, ease of implementation, and practical usability.

## 1.3 Research goals

In this research, the focus is on investigating how the computational efficiency can be increased in gradient-based structural optimization for geometric nonlinear truss problems using existing optimization algorithms. To achieve this, a model is built that accurately describes geometrically nonlinearly deforming truss behavior, and allows for practical implementation of optimization techniques. The computational efficiency of the optimization is increased by implementing analytical gradients. Two main optimization approaches are investigated: the nested and SAND approaches. A set of problems within the predefined domain can be optimized with both approaches using analytical gradients. To implement analytical gradients in the nested approach, either the direct method or adjoint method can be used. The goal of this study is to demonstrate the circumstances and conditions under which each approach is most suitable, highlighting that both approaches are significantly more efficient when analytical gradients are used instead of numerical gradients. The gain in computational efficiency enables to optimize more complex structures, allowing for more state and design variables.

## 1.4 Scope

To allow for practical results, the problems discussed are subject to certain assumptions and simplifications. In this section the most important and impactful choices made to balance relevance and manageability are outlined. This enables the testing and comparison of both the nested and the SAND approach.

### 1.4.1 Structures and elements

In this research, all structures are considered static. Consequently, the equilibrium equations are given by:

$$\boldsymbol{K}(\boldsymbol{q})\boldsymbol{q} = \boldsymbol{f}, \tag{1.1}$$

where $\boldsymbol{K}(\boldsymbol{q})$ is the total stiffness matrix of the system, $\boldsymbol{q}$ represents the displacements, and $\boldsymbol{f}$ represents the external applied loads. The stiffness matrix depends on the displacements, introducing geometric nonlinearities into the model equations. Further details on the nonlinear behavior are provided in subsection 1.4.3. This is the general equation for static structures. However, the method of obtaining the residual equations in this research is different, and explained in Chapter 2.

The primary goal of this research is to demonstrate the efficiency of analytical gradients in gradient-based optimization for structural mechanical problems. The flexible multibody approach is serving as a tool to obtain useful results. Consequently, only truss elements are used. Truss elements can only take axial forces, while beam elements can also capture bending moments. Systems consisting solely of trusses are generally insufficient for practical applications. Therefore, the high-level approach is generic, enabling the implementation of beam elements with additional stiffness components and degrees of freedom in future research.

Lastly, only 2D structures are used. In many real-life applications, transforming problems to 2D does not significantly impact the accuracy of the results or limit the insights that the model provides.

### 1.4.2 Gradients

For optimization, it is necessary to find the gradients of the objective and constraint functions with respect to the design variables. To obtain these gradients analytically, a selection of possible objective functions, constraint functions, and design variables must be made. This selection limits the optimization possibilities but can always be extended if required.

For the design variables ($\boldsymbol{p}$), six basic types are chosen. These are:

$$\boldsymbol{p} = [A^{(k)}, E^{(k)}, x_i, y_i, x_j, y_j]. \tag{1.2}$$

$A^{(k)}$ is the cross sectional area, $E^{(k)}$ is the Young's modulus. The superscript $(k)$ refers to element $k$. $x_i$ is the initial $x$ position of the $i$-th node, $y_i$ is the initial $y$ position of the $i$-th node, $x_j$ is the initial $x$ position of the $j$-th node, and $y_j$ is the initial $y$ position of the $j$-th node. Nodes are not element specific as multiple elements can share nodes.

The above variables constitute a complete set of variables in which any structure can be defined within the scope of this work. Other high-level design variables may be defined, such as the cross-section of a subset of elements, or a geometrical relation between the initial coordinates of multiple nodes. The computations of the gradients to such design variables will require the computation of the gradients of the basic variables, with the additional application of the chain rule to account for the relation between the basic variable and the design variable.

In the nested approach, the design variables $\boldsymbol{p}$ encompass all optimization variables. However, in the SAND approach, $\boldsymbol{p}$ represents a subset of the optimization variables since this approach includes the variables in the solution vector of the residual equations, $\boldsymbol{u}$, as additional optimization variables.
Converting between an objective function and a constraint function does not significantly alter the gradient computation. In this research, four key objective and constraint functions are selected: mass, stress, displacement, and compliance. An important function for compliant mechanisms that is excluded here is eigenfrequency. This exclusion is due to the additional requirement of introducing mass matrices, which is beyond the scope of this research. However, future research can incorporate eigenfrequency as an objective or constraint function, following a similar roadmap as used for the selected four objective and constraint functions.

### 1.4.3 Nonlinearities

The contrast between using analytic and numerical gradients becomes significant for nonlinear problems, where the solution of the model equations cannot be determined in a single operation but requires multiple iterations. Therefore, this research is concerned with nonlinear problems, meaning that the used system of equations accounts for large deformations.
Several nonlinear factors are relevant for truss-like structures. A distinction can be made between material nonlinearities and geometric nonlinearities. In this research only geometric nonlinearities are included. This means that the element force is assumed to be linearly dependent on the element elongations, meaning that the element stiffness is independent of its elongation. Therefore, it is assumed that there is no transverse contraction upon longitudinal loads (i.e. the element cross-section remains constant), and that the stress-strain relation can be defined in terms of engineering strain:

$$\sigma = E\epsilon, \tag{1.3}$$

were $\sigma$ denotes the stress and $\epsilon$ represents the engineering strain within an element, which is equal to $\Delta L/L_0$. The constant stiffness $k$ of a single truss can therefore be expressed as

$$k = \frac{A_0 E}{L_0}. \tag{1.4}$$

$A_0$ denotes the initial cross sectional area, and $L_0$ is the original length of the element. Both $A_0$ and $L_0$ do not depend on the deformation of the truss element [3].
The global stiffness of an element changes with the global orientation of that element, meaning that the stiffness of the structure depends on the displacement. For large displacements this effect becomes significant. This introduces nonlinearities in the equations.

These geometric nonlinearities are included in the model presented in this research.

### 1.4.4 Optimization

For optimization, MATLAB's fmincon function from the 2023a version of MATLAB is used. Two built-in solvers are tested: sequential quadratic programming (SQP) and interior-point (IP). Unless specified otherwise, the default settings from fmincon are used. The nested and SAND approaches supplied with analytical gradients are compared. The SAND approach is expected to be more efficient for large structures than the nested approach [4]. In the nested approach, analytical gradients are computed using the direct or adjoint method. The comparison between the direct and adjoint methods, as well as a comparison with finite difference is performed. This comparison measures the gradient computation time and does not involve the optimization process.

## 1.5 Outline

The report begins with an explanation of the flexible multibody approach used to solve nonlinear static structural problems. In Chapter 3, the general optimization problem for both the nested and SAND approaches are outlined, and the MATLAB function fmincon is shortly discussed. Following the definition of the optimization problem, the gradients needed for optimizing are discussed. This starts with a theoretical exploration of the methods to obtain these derivatives, including finite difference, the direct method, and the adjoint method. Following this discussion, the generic derivation of the derivatives is outlined. This includes the derivation of derivatives for four possible objective and constraint functions: mass, stress, displacement, and compliance. Chapter 6 presents sample problems used to test the impact of analytical derivatives and the efficiency of the nested and SAND approaches. Chapter 7 visualizes the findings of the research using meaningful graphs. The report concludes with a conclusion and recommendations for potential further research.

# Chapter 2

# Flexible multibody approach

The assumptions and simplifications for the types of problems investigated in this research are presented in Chapter 1. This chapter offers a general formulation of the flexible multibody approach, facilitating extension to more complex problems and allowing for greater flexibility in adjustments. In Section 2.5, a simple problem is solved using the flexible multibody approach.

## 2.1 Problem introduction

All structures are considered static. Hence, the system is in equilibrium. The structures are composed of nodes and elements. Each element is connected to two nodes, the $i$ and $j$ nodes. At these nodes, multiple elements can be connected to each other. The multibody model treats the displacements of nodes in $x$ and $y$ direction as degrees of freedom. Furthermore, the elongation of the elements, $\Delta L^{(k)}$, are defined as additional degrees of freedom. The superscript $k$ indicates the change in length of element $k$. This way a pre-strain can directly be applied to an element. There are two equilibrium equations per node and one equilibrium equation per element. The set of degrees of freedom per element changes when using beam elements, introducing a rotational degree of freedom. The equilibrium equations couple the reaction forces in the constraint nodes and elements with the nodal displacements and elemental elongations. Therefore, the unknowns are two displacements per node, elongation of each element, and the internal reaction forces in each element and in restricted nodes. The equilibrium equations can only be solved when additional boundary conditions and constraints are defined to assure the set of equations is sufficiently defined and therefore solvable. Boundary conditions or constraint equations can be applied at nodes, when nodes are physically restricted in one or multiple directions or a displacement of a node in a certain direction is forced. Furthermore, a constraint equation is defined for every element, to relate the element elongation $\Delta L^{(k)}$ with the nodal displacements of the connected degrees of freedom. The residual equations consists

of the equilibrium equations and the constraint equations.

## 2.2 Potential energy formulation

The equilibrium equations can be formulated based on the principle that, for static elastic structures, the potential energy is minimized. This implies that the derivative of the potential energy with respect to the displacements must be zero. A constraint minimization problems arises when adding the constraint equations. The potential energy equation with nodal constraints and element constraints is expressed in Lagrangian form [5]. The variables included in each term are shown in parentheses as function arguments:

$$\mathcal{L}(\boldsymbol{q}_n, \boldsymbol{q}_e, \boldsymbol{\lambda}_n, \boldsymbol{\lambda}_e) = W_{int}(\boldsymbol{q}_e) - W_{ext}(\boldsymbol{q}_n) + \boldsymbol{g}_n(\boldsymbol{q}_n)^\top \boldsymbol{\lambda}_n + \boldsymbol{g}_e(\boldsymbol{q}_n, \boldsymbol{q}_e)^\top \boldsymbol{\lambda}_e \tag{2.1}$$

where $\boldsymbol{q}_n$ are nodal displacements, $\boldsymbol{q}_e$ represent elemental displacements, $\boldsymbol{g}_n$ are nodal constraints, and $\boldsymbol{g}_e$ are element constraints. Furthermore, $\boldsymbol{\lambda}_n$ are nodal reaction forces, $\boldsymbol{\lambda}_e$ are elemental reaction forces. The displacement vector is given by

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_n & \boldsymbol{q}_e \end{bmatrix}^\top, \tag{2.2}$$

and the constraint vector:

$$\boldsymbol{g} = \begin{bmatrix} \boldsymbol{g}_n & \boldsymbol{g}_e \end{bmatrix}^\top. \tag{2.3}$$

When loads are applied to a body, the body will deform. Assuming no energy is lost in the form of heat, the external work, $W_{ext}$, done by the applied loads is converted into internal work called strain energy, $W_{int}$. Hence, the potential energy, $\Pi$ of a system can be written as

$$\Pi(\boldsymbol{q}) = W_{int}(\boldsymbol{q}_e) - W_{ext}(\boldsymbol{q}_n). \tag{2.4}$$

A general 3D formulation for the strain energy and external work is

$$W_{\text{int}} = \frac{1}{2} \int_V \sigma_{ij} \varepsilon_{ij} \, dV, \tag{2.5a}$$

$$W_{\text{ext}} = \int_V f_i u_i \, dV + \int_S t_i u_i \, dS, \tag{2.5b}$$

where $\sigma_{ij}$ and $\epsilon_{ij}$ are the stress and strain tensor respectively. $V$ is the volume, $S$ the surface, $f_i$ are body forces, for example gravity, $u_i$ is the deformation vector and $t_i$ represents surface forces. The Einstein summation convention is used.

From now on, the function arguments are omitted. Equilibrium is found when the gra-

dients in all directions are equal to zero:

$$\nabla_{\boldsymbol{q}_n}\mathcal{L} = -\nabla_{\boldsymbol{q}_n}W_{ext} + \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n + \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e = 0, \tag{2.6a}$$

$$\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top + \nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e = 0, \tag{2.6b}$$

$$\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \boldsymbol{g}_n^\top = 0, \tag{2.6c}$$

$$\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \boldsymbol{g}_e^\top = 0. \tag{2.6d}$$

The $\nabla$ operator is defined as a column vector. The first derivative terms of the Lagrangian are all column vectors. As the model does consider geometric nonlinearity, these conditions lead to a set of nonlinear model equations denoted as $\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{\lambda}) = \boldsymbol{0}$. Equation (5.1a) corresponds to the nodal equilibrium equations, and equation (5.1b) both arising from the minimization of the potential energy. Equation (5.1c) gives rise to the nodal constraint equations, and equation (5.1d) represents the element constraints. To solve the model equations with Newton-Raphson, the gradient of these equations with respect to the variable set is needed. This will be the Hessian of the Lagrangian. Given that the relation between the external work $W_{ext}$ and the nodal displacements $\boldsymbol{q}_n$ is linear, the term $\nabla_{\boldsymbol{q}_n}\nabla_{\boldsymbol{q}_n}W_{ext}$ is equal to zero. Taking the derivatives of equations (5.1a)-(5.1d) to $\boldsymbol{q}_n$ gives:

$$\nabla_{\boldsymbol{q}_n}\nabla_{\boldsymbol{q}_n}\mathcal{L} = \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top + \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{2.7a}$$

$$\nabla_{\boldsymbol{q}_n}\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{2.7b}$$

$$\nabla_{\boldsymbol{q}_n}\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\right)^\top, \tag{2.7c}$$

$$\nabla_{\boldsymbol{q}_n}\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\right)^\top. \tag{2.7d}$$

In many situation the nodal constraints $\boldsymbol{g}_n$ are a linear function of $\boldsymbol{q}_n$, if this is the case the term $\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top = \boldsymbol{0}$. The derivatives of equations (5.1a)-(5.1d) to $\boldsymbol{q}_e$ gives:

$$\nabla_{\boldsymbol{q}_e}\nabla_{\boldsymbol{q}_n}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{2.8a}$$

$$\nabla_{\boldsymbol{q}_e}\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top + \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{2.8b}$$

$$\nabla_{\boldsymbol{q}_e}\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \boldsymbol{0}, \tag{2.8c}$$

$$\nabla_{\boldsymbol{q}_e}\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\right)^\top. \tag{2.8d}$$

$$\tag{2.8e}$$

In many cases the element constraints can be written as:

$$\boldsymbol{g}_e = f(\boldsymbol{g}_n) - \boldsymbol{g}_e. \tag{2.9}$$

This results in: $\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e = -\boldsymbol{1}$, a diagonal matrix with $-1$ on its diagonal entries. Taking

the derivative of equations (5.1a)-(5.1d) to $\boldsymbol{\lambda}_n$ gives:

$$\nabla_{\boldsymbol{\lambda}_n}\nabla_{\boldsymbol{q}_n}\mathcal{L} = \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top, \tag{2.10a}$$
$$\nabla_{\boldsymbol{\lambda}_n}\nabla_{\boldsymbol{q}_e}\mathcal{L} = \mathbf{0}, \tag{2.10b}$$
$$\nabla_{\boldsymbol{\lambda}_n}\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \mathbf{0}, \tag{2.10c}$$
$$\nabla_{\boldsymbol{\lambda}_n}\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \mathbf{0}. \tag{2.10d}$$
$$\tag{2.10e}$$

Taking the derivative of Equations (5.1a)-(5.1d) to $\boldsymbol{\lambda}_e$ gives:

$$\nabla_{\boldsymbol{\lambda}_e}\nabla_{\boldsymbol{q}_n}\mathcal{L} = \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top, \tag{2.11a}$$
$$\nabla_{\boldsymbol{\lambda}_e}\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top, \tag{2.11b}$$
$$\nabla_{\boldsymbol{\lambda}_e}\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \mathbf{0}, \tag{2.11c}$$
$$\nabla_{\boldsymbol{\lambda}_e}\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \mathbf{0} \tag{2.11d}$$

This can be aggregated in matrix form as:

$$\begin{bmatrix} \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top + \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top & \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top \\ \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top & \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top + \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top & \mathbf{0} & \nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\right)^\top & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\right)^\top & \left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\right)^\top & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.12}$$

If all constraint equations are defined as linear equations in $\boldsymbol{q}_n$ and $\boldsymbol{q}_e$, and the element constraints are defined as: $\boldsymbol{g}_e = f(\boldsymbol{g}_n) - \boldsymbol{g}_e$ then:

$$\nabla_{\boldsymbol{q}_e}\left(\nabla\boldsymbol{q}_n\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top = \mathbf{0}, \tag{2.13a}$$
$$\nabla_{\boldsymbol{q}_n}\left(\nabla\boldsymbol{q}_e\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top = \mathbf{0}, \tag{2.13b}$$
$$\nabla_{\boldsymbol{q}_e}\left(\nabla\boldsymbol{q}_e\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right) = \mathbf{0}^\top, \tag{2.13c}$$
$$\nabla_{\boldsymbol{g}_e}\boldsymbol{g}_e^\top = -\mathbf{1}, \tag{2.13d}$$
$$\left(\nabla_{\boldsymbol{g}_e}\boldsymbol{g}_e^\top\right)^\top = -\mathbf{1}^\top = -\mathbf{1}. \tag{2.13e}$$
$$\tag{2.13f}$$

The matrix form then simplifies to:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \begin{bmatrix} \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top & \mathbf{0} & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top \\ \mathbf{0} & \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top & \mathbf{0} & -\mathbf{1} \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\right)^\top & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\right)^\top & -\mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.14}$$

The second derivative terms of the Lagrangian are all matrices. The size of $\boldsymbol{J}$ is ($nUnknown \times nUnknown$), were $nUnknown$ is the number of entries of the solution vector $\boldsymbol{u}$. The term $\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top$ has a size of ($nNoDOF \times nNoDOF$), were $nNoDOF$ are the number of nodal degrees of freedom. The size of $\nabla^2_{\boldsymbol{q}_e}W_{int}$ is ($nEl \times nEl$), with $nEl$ the number of elements. $\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top$ has a size of $nNoDOF \times nNoCo$, $nNoCo$ is the number of nodal constraints. Lastly, the $\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top$ has a size of $nNoDOF \times nElCo$, $nElCo$ is the number of element constraints.
Newton-Raphson uses this Jacobian to solve the residual equations. The total solution vector of the state variables is defined as

$$\boldsymbol{u} = \begin{bmatrix}\boldsymbol{g}_n & \boldsymbol{g}_e & \boldsymbol{\lambda}_n & \boldsymbol{\lambda}_e\end{bmatrix}^\top. \tag{2.15}$$

## 2.3 Newton-Rahpson

There are various methods available for solving nonlinear equations, one of which is the Newton-Raphson (NR) method. For a single nonlinear equation ($r(u) = 0$), the local linearizaton of the system equation at position $u_n$ can be use to define the next trial solution $u_{n+1}$:

$$u_{n+1} = u_n - \frac{r(u_n)}{r'(u_n)}, \tag{2.16}$$

were $r'$ means the first derivative of $r$ with respect to $u$. To solve a system of nonlinear equations, the NR method employs the Jacobian of the residual equations with respect to the solution vector to iteratively converge towards the solution [6]

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n - \boldsymbol{J}(\boldsymbol{u}_n)^{-1}\boldsymbol{r}(\boldsymbol{u}_n). \tag{2.17}$$

In this equation, $\boldsymbol{u}$ is the solution vector, denoted as $\boldsymbol{u} = [\boldsymbol{q}, \boldsymbol{\lambda}]^T$, and $\boldsymbol{r}(\boldsymbol{u}_n)$ is the residual vector evaluated at the solution of iteration $n$. The Jacobian $\boldsymbol{J}$, of the residual equations and the solution vector is given in equation (2.18). Jacobians in this work use this definition.

$$\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \begin{bmatrix} \frac{\partial r_1}{\partial u_1} & \frac{\partial r_1}{\partial u_2} & \cdots & \frac{\partial r_1}{\partial u_n} \\ \frac{\partial r_2}{\partial u_1} & \frac{\partial r_2}{\partial u_2} & \cdots & \frac{\partial r_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial u_1} & \frac{\partial r_m}{\partial u_2} & \cdots & \frac{\partial r_m}{\partial u_n} \end{bmatrix} \tag{2.18}$$

The general Jacobian of the residual equations with the state variables is described in Section 2.2 in (2.12) and (2.37).
The iterative process continues until one of the following conditions is met: either the maximum number of iterations is surpassed, the absolute change between successive solutions $|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k|$ becomes smaller than the specified solution tolerance, or the residual

of the nonlinear equations $\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{\lambda})$ falls below the equation tolerance. There are multiple ways to define the tolerances: for example when the absolute value of each component of the vector $\boldsymbol{r}(\boldsymbol{u}_n)$ falls below the equation tolerance. Alternatively, an $L_2$ norm can condense the residual vector into a single value. Consequently, it is expected that the equation tolerance must be set tighter to achieve the same level of accuracy. Determining the appropriate equation tolerance is problem-specific and depends on factors beyond the NR procedure itself.

## 2.4 Residual equations

This research only includes truss elements. Subsequently, the equations and solution vector will be further analyzed, specifically for truss elements.
As discussed in Section 2.1, the system of equations is defined with the following unknowns: nodal displacements, element deformations, nodal reaction forces and element forces. A sufficient set of equations can be found by solving for the equilibrium of the Lagrangian function, see equation 5.1. This yields a set of equations that can be categorized in four sets: nodal equilibrium equations, element equilibrium equations, nodal constraint equations and element constraint equations. When determining the system derivatives, each of these types of equations has to be treated differently. Therefore, it is of utmost importance to properly track the position of each equation type in the system of equations. To structure the notation, the following sets are defined, indicating the equation numbering for each of the four sets of equations:

$$\mathrm{NoEq} = \{1, \ldots, n\mathrm{NoEq}\}. \tag{2.19a}$$

$$\mathrm{ElEq} = \{n\mathrm{NoEq} + 1, \ldots, n\mathrm{NoEq} + n\mathrm{ElEq}\}. \tag{2.19b}$$

$$\mathrm{NoCo} = \{n\mathrm{NoEq} + n\mathrm{ElEq} + 1, \ldots, n\mathrm{NoEq} + n\mathrm{ElEq} + n\mathrm{NoCo}\}. \tag{2.19c}$$

$$\begin{aligned}\mathrm{ElCo} = \{&n\mathrm{NoEq} + n\mathrm{ElEq} + n\mathrm{NoCo} + 1, \ldots, \\ &n\mathrm{NoEq} + n\mathrm{ElEq} + n\mathrm{NoCo} + n\mathrm{ElCo}\}.\end{aligned} \tag{2.19d}$$

$nNoEq$ are the number of nodal equilibrium equations, $nElEq$ are the number of element equilibrium equations, $nNoCo$ are the number of nodal constraint equations, and $nElCo$ are the number of element constraint equations. Hence, the combination of the sets $NoEq$ and $ElEq$ include all the equilibrium equations. The combination of the sets $NoCo$ and $ElCo$ represent the constraint equations. The equations will always be ordered in the order as presented above.
External forces can only be applied at nodes, affecting only equations in set (2.19a). Forced displacements are imposed as nodal constraints, affecting only set (2.19c). Consequently, the right-hand side in the linearized solution procedure includes both external

forces and prescribed displacements.
The solution vector $\boldsymbol{u}$ consists of both degrees of freedom (DOFs) $\boldsymbol{q}$, and Lagrange multipliers $\boldsymbol{\lambda}$. However, both DOFs and Lagrange multipliers can be further categorized into two distinct types of variables. The DOFs encompass nodal displacements and element elongations, while the Lagrange multipliers include nodal reaction forces and element reaction forces. To differentiate, between displacements and reaction forces, subscripts are used. As explained in Section 2.2, $\boldsymbol{q}_n$ signifies the nodal displacements, $\boldsymbol{q}_e$ are the element elongations, $\boldsymbol{\lambda}_n$ are the nodal reaction forces, and $\lambda_e$ are the elemental reaction forces.
As some residual equations are given in terms of forces, and other in terms of displacements, it is necessary to scale the residual equations to avoid an ill-conditioned Jacobian. In this report, the scaling of equations is not explicitly indicated, as it would introduce many additional terms compromising the clarity of the fundamental equations presented. However, to achieve accurate results, it is crucial that all equations in the matrix-vector equations are of similar scale. After solving the equations, solutions can be scaled back to obtain the correct results.

## 2.5 Example

A simple example of a truss structure is included to provide insights into the model equations.

### 2.5.1 Truss definition

Trusses are the only elements that are used in this research. This section provides the truss definition that is used, [7] served as inspiration. A single free floating truss with the degrees of freedom (DOFs) indicated by arrows can be seen in Figure 2.1. The position

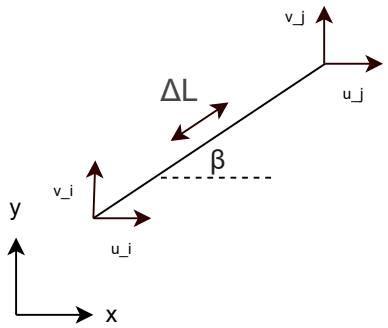



Figure 2.1: Free floating truss

of the truss can in principle be described by the coordinates of each end point. The initial coordinates are defined as $(x_i, y_i)$ and $(x_j, y_j)$. These are constant in the solution

procedure and are provided during the initialization. The final coordinates (after the model is solved) are indicated with a capital letter and consists of the initial coordinate plus the corresponding displacement.

$$X_i = x_i + u_i, \tag{2.20a}$$

$$Y_i = y_i + v_i, \tag{2.20b}$$

$$X_j = x_j + u_j, \tag{2.20c}$$

$$Y_j = y_j + v_j. \tag{2.20d}$$

Each truss is connected to five degrees of freedom: the nodal displacements $u_i$, $v_i$, $u_j$ and $v_j$, and the truss elongation $\Delta L$. While the nodal displacements can be connected to multiple bar elements, each truss elongation only belongs to a single bar element. $\Delta L$ as a separate DOF introduces an element constraint equation for every element $k$:

$$L^{(k)} - L_0^{(k)} - \Delta L_0^{(k)} = 0, \tag{2.21}$$

were $L^{(k)}$ is the current length (being a function of $X_i$, $Y_i$, $X_j$ and $Y_j$, and consequently also a function of the nodal displacements), $L_0^{(k)}$ the initial length (a constant), and $\Delta L_0^{(k)}$ the change in length of element $k$ (he element elongation degree of freedom). This expression is important because the derivatives are primarily constructed element-wise, with most terms originating from these element constraints. Boundary conditions can be added as nodal constraints.

### 2.5.2 Model equations

An example of the multibody approach is presented here. This is done in order to show the implementation of the generic equations as described.
The structure can be seen in Figure 2.2. The structure consists of 2 trusses, 3 nodes, and 8 degrees of freedom. 2 DOFs per node and 1 DOF per element. The displacement of node 2 is zero in both directions, the displacement of node 1 is zero in $x$ direction, and in $y$ direction a displacement, $c$ of 0.1m is enforced. An external force of -1e7N in $y$ direction is applied at node 3.

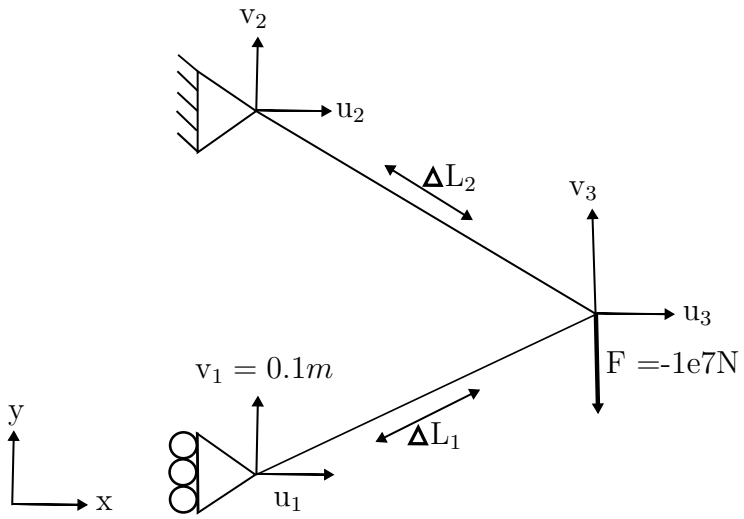



Figure 2.2: Simple two truss system.

Before starting the explanation of the example some definitions used are provided. The sine and cosine terms are abbreviated and given by:

$$\cos\beta^{(k)} = c_k = \frac{X_j^{(k)} - X_i^{(k)}}{L^{(k)}}, \tag{2.22a}$$

$$\sin\beta^{(k)} = s_k = \frac{Y_j^{(k)} - Y_i^{(k)}}{L^{(k)}}, \tag{2.22b}$$

where the superscript $k$ refers to element $k$. Furthermore, the initial and current length are given as:

$$L_0^{(k)} = \sqrt{\left(x_j^{(k)} - x_i^{(k)}\right)^2 + \left(y_j^{(k)} - y_i^{(k)}\right)^2}, \tag{2.23a}$$

$$L^{(k)} = \sqrt{\left(X_j^{(k)} - X_i^{(k)}\right)^2 + \left(Y_j^{(k)} - Y_i^{(k)}\right)^2}. \tag{2.23b}$$

The structure yields six constraint equations, $\boldsymbol{g}(\boldsymbol{q})$. The four boundary conditions or nodal constraints are

$$u_1 = 0, \tag{2.24a}$$

$$v_1 = c, \tag{2.24b}$$

$$u_2 = 0, \tag{2.24c}$$

$$v_2 = 0, \tag{2.24d}$$

and the two element constraints are

$$L_1 - L_1^0 - \Delta L_1 = 0, \tag{2.25a}$$

$$L_2 - L_2^0 - \Delta L_2 = 0. \tag{2.25b}$$

The specific strain energy and external work depend on the element type. For 2D truss elements:

$$W_{int} = \frac{1}{2}\sum_{i=1}^{\text{nEl}} k_i \Delta L_i^2, \tag{2.26}$$

and

$$W_{ext} = \sum_{l=1}^{\text{nNo}} F_{xl} u_l + F_{yl} v_l. \tag{2.27}$$

Here $F_x$ and $F_y$ are the external forces in $x$ and $y$ direction respectively. Furthermore, $u$ and $v$ are the displacements of the corresponding node in $x$ and $y$ direction respectively [8]. The loads are applied at the nodes.
The full Lagrangian becomes:

$$\begin{aligned}\mathcal{L} = \frac{1}{2}k_1\Delta L_1^2 + \frac{1}{2}k_2\Delta L_2^2 - v_3 F + \lambda_{n1}u_1 + \lambda_{n2}(v_1 - 0.01) + \lambda_{n3}u_2 + \lambda_{n4}v_2 \\ + \lambda_{e1}(L_1 - L_1^0 - \Delta L_1) + \lambda_{e2}(L_2 - L_2^0 - \Delta L_2).\end{aligned} \tag{2.28}$$

The general residual equations are described in (5.1), and repeated here for readability.

$$\nabla_{\boldsymbol{q}_n}\mathcal{L} = -\nabla_{\boldsymbol{q}_n} W_{ext} + \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top \boldsymbol{\lambda}_n + \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top \boldsymbol{\lambda}_e = 0, \tag{2.29a}$$

$$\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e} W_{int}\right)^\top + \nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top \boldsymbol{\lambda}_e = 0, \tag{2.29b}$$

$$\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \boldsymbol{g}_n = 0, \tag{2.29c}$$

$$\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \boldsymbol{g}_e = 0. \tag{2.29d}$$

For this problem, the set of nodal displacements and element elongations is

$$\boldsymbol{q} = \begin{bmatrix} q_{n1} & q_{n2} & q_{n3} & q_{n4} & q_{n5} & q_{n6} & q_{e1} & q_{e2} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & \Delta L_1 & \Delta L_2 \end{bmatrix}, \tag{2.30}$$

and the set of nodal and elemental reaction forces is

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_{n1} & \lambda_{n2} & \lambda_{n3} & \lambda_{n4} & \lambda_{e1} & \lambda_{e2} \end{bmatrix}. \tag{2.31}$$

In this example, $u$ with a subscript represents the nodal displacement in $x$ direction. These are different variables than the solution vector $\boldsymbol{u}$. Only in this example $u_i$ is used in this context to provide insights into the physical meaning of the variables.
There are 8 equilibrium equations, and 6 constraint residual equations ($\boldsymbol{r}(\boldsymbol{q}, \boldsymbol{\lambda}) = \boldsymbol{0}$). The six nodal equilibrium equations, two per node ($\nabla_{\boldsymbol{q}_n}\mathcal{L}$)

$$\frac{\partial \mathcal{L}}{\partial u_1} = \lambda_{n1} - \cos(\beta_1)\lambda_{e1} = 0, \tag{2.32a}$$

$$\frac{\partial \mathcal{L}}{\partial v_1} = \lambda_{n2} - \sin(\beta_1)\lambda_{e1} = 0, \tag{2.32b}$$

$$\frac{\partial \mathcal{L}}{\partial u_2} = \lambda_{n3} - \cos(\beta_2)\lambda_{e2} = 0, \tag{2.32c}$$

$$\frac{\partial \mathcal{L}}{\partial v_2} = \lambda_{n4} - \sin(\beta_2)\lambda_{e2} = 0, \tag{2.32d}$$

$$\frac{\partial \mathcal{L}}{\partial u_3} = \cos(\beta_1)\lambda_{e1} + \cos(\beta_2)\lambda_{e2} = 0, \tag{2.32e}$$

$$\frac{\partial \mathcal{L}}{\partial v_3} = \sin(\beta_1)\lambda_{e1} + \sin(\beta_2)\lambda_{e2} - F. \tag{2.32f}$$

The two element equilibrium equations, one per element ($\nabla_{\boldsymbol{q}_e}\mathcal{L}$)

$$\frac{\partial \mathcal{L}}{\partial \Delta L_1} = k_1 \Delta L_1 - \lambda_{e1} = 0, \tag{2.33a}$$

$$\frac{\partial \mathcal{L}}{\partial \Delta L_2} = k_2 \Delta L_2 - \lambda_{e2} = 0. \tag{2.33b}$$

Four nodal constraint equations ($\nabla_{\boldsymbol{\lambda}_n}\mathcal{L}$)

$$\frac{\partial \mathcal{L}}{\partial \lambda_{n1}} = u_1 = 0, \tag{2.34a}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{n2}} = v_1 - c = 0, \tag{2.34b}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{n3}} = u_2 = 0, \tag{2.34c}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{n4}} = v_2 = 0. \tag{2.34d}$$

Two element constraint equations ( $\nabla_{\boldsymbol{\lambda}_e}\mathcal{L}$)

$$\frac{\partial \mathcal{L}}{\partial \lambda_{e1}} = L_1 - L_1^0 - \Delta L_1 = 0, \tag{2.35a}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_{e1}} = L_2 - L_2^0 - \Delta L_2 = 0. \tag{2.35b}$$

The residual equations are analytically known for every problem. The last step before solving the problem is obtaining the Jacobian of the residual equations with respect to the solution vector. Which yields the second order gradient of the Lagrangian function. In this case the complete solution vector is

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{q} & \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & u_2 & v_2 & u_3 & v_3 & \Delta L_1 & \Delta L_2 & \lambda_{n1} & \lambda_{n2} & \lambda_{n3} & \lambda_{n4} & \lambda_{e1} & \lambda_{e2} \end{bmatrix}. \tag{2.36}$$

There are 14 residual equations and the solution vector consists of 14 elements, hence

the Jacobian is a 14x14 matrix. To keep the equations organized and clear, the Jacobian is not entirely written out, but instead sectioned into parts corresponding to different types of residual equations and variables from the solution vector. The general form of the Jacobian matrix (from Section 2.2) is repeated here for readability.

$$\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \begin{bmatrix} \nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top \boldsymbol{\lambda}_n\right)^\top & \boldsymbol{0} & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top & \nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top \\ \boldsymbol{0} & \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e} W_{int}\right)^\top & \boldsymbol{0} & \boldsymbol{-1} \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\right)^\top & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\right)^\top & \boldsymbol{-1} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}. \tag{2.37}$$

The Jacobian is the first derivative of the residual equations with respect to the state variables, and it is also the second derivative of the Lagrangian with respect to the state variables. In equation form:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = \boldsymbol{r}, \tag{2.38a}$$

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{u}^2} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \boldsymbol{J}. \tag{2.38b}$$

The four distinct terms present in this Jacobian will be given. The geometric term $\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top \boldsymbol{\lambda}_n\right)^\top =$

$$\begin{bmatrix} \frac{\lambda_{e1}}{L_1}s_1^2 & -\frac{\lambda_{e1}}{L_1}s_1c_1 & 0 & 0 & -\frac{\lambda_{e1}}{L_1}s_1^2 & \frac{\lambda_{e1}}{L_1}s_1c_1 \\ -\frac{\lambda_{e1}}{L_1}s_1c_1 & \frac{\lambda_{e1}}{L_1}c_1^2 & 0 & 0 & \frac{\lambda_{e1}}{L_1}s_1c_1 & -\frac{\lambda_{e1}}{L_1}c_1^2 \\ 0 & 0 & \frac{\lambda_{e2}}{L_2}s_2^2 & -\frac{\lambda_{e2}}{L_2}s_2c_2 & -\frac{\lambda_{e2}}{L_2}s_2^2 & \frac{\lambda_{e2}}{L_2}s_2c_2 \\ 0 & 0 & -\frac{\lambda_{e2}}{L_2}s_2c_2 & \frac{\lambda_{e2}}{L_2}c_2^2 & \frac{\lambda_{e2}}{L_2}s_2c_2 & -\frac{\lambda_{e2}}{L_2}c_2^2 \\ -\frac{\lambda_{e1}}{L_1}s_1^2 & \frac{\lambda_{e1}}{L_1}s_1c_1 & -\frac{\lambda_{e2}}{L_2}s_2^2 & \frac{\lambda_{e2}}{L_2}s_2c_2 & \frac{\lambda_{e1}}{L_1}s_1^2 + \frac{\lambda_{e2}}{L_2}s_1^2 & -\frac{\lambda_{e1}}{L_1}s_1c_1 - \frac{\lambda_{e2}}{L_2}s_2c_2 \\ \frac{\lambda_{e1}}{L_1}s_1c_1 & -\frac{\lambda_{e1}}{L_1}c_1^2 & \frac{\lambda_{e2}}{L_2}s_2c_2 & -\frac{\lambda_{e2}}{L_2}c_2^2 & -\frac{\lambda_{e1}}{L_1}s_1c_1 - \frac{\lambda_{e2}}{L_2}s_2c_2 & \frac{\lambda_{e1}}{L_1}c_1^2 + \frac{\lambda_{e2}}{L_2}c_2^2 \end{bmatrix}. \tag{2.39}$$

The stiffness related term

$$\nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e} W_{int}\right)^\top = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}. \tag{2.40}$$

The nodal constraint term is

$$\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.41}$$

The ones represent which direction in which nodes are constraints, in this case both directions of node 1 and node 2. The quantity of the prescribed displacements will appear in the right hand side of the final matrix equation. The element constraint term

is given by

$$\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_e = \begin{bmatrix} -c_1 & -s_1 & 0 & 0 & c_1 & s_1 \\ 0 & 0 & -c_2 & -s_2 & c_2 & s_2 \end{bmatrix}. \tag{2.42}$$

The following form in the linearized equations can be recognized:

$$\begin{bmatrix} \nabla_{\boldsymbol{q}_n} \left( \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top \boldsymbol{\lambda}_n \right)^\top & \mathbf{0} & \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top & \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_e^\top \\ \mathbf{0} & \nabla_{\boldsymbol{q}_e} \left( \nabla_{\boldsymbol{q}_e} W_{int} \right)^\top & \mathbf{0} & -\mathbf{1} \\ \left( \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top \right)^\top & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \left( \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_e^\top \right)^\top & -\mathbf{1} & \mathbf{0} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{q}_n \\ \boldsymbol{q}_e \\ \boldsymbol{\lambda}_n \\ \boldsymbol{\lambda}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{ext} \\ \mathbf{0} \\ \boldsymbol{Q}_{ext} \\ \mathbf{0} \end{bmatrix}, \tag{2.43}$$

were $\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n$ represents the nodal constraint equations, $\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_e$ are the element constraint equations, $\nabla_{\boldsymbol{q}_n} \left( \nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top \boldsymbol{\lambda}_n \right)^\top$ are the element orientation related terms and $\nabla_{\boldsymbol{q}_e} \left( \nabla_{\boldsymbol{q}_e} W_{int} \right)^\top$ represent the stiffness per element, that contributes to the element equilibrium equations. $\boldsymbol{F}_{ext}$ represents the external applied forces and $\boldsymbol{Q}_{ext}$ represents the prescribed displacements.

Newton-Rahpson is used to solve the equations:

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k - \boldsymbol{J}(\boldsymbol{u}_k)^{-1} \boldsymbol{r}(\boldsymbol{u}_k), \tag{2.44}$$

were $\boldsymbol{r}(\boldsymbol{u}_k)$ and $\boldsymbol{J}(\boldsymbol{u}_k)$ are the residual equations and the Jacobian evaluated at $\boldsymbol{u}_k$ respectively. Newton-Raphson iterates towards the solution until the convergence criteria are satisfied. From all the terms in the Jacobian only $\boldsymbol{G}$ and $\boldsymbol{C}_e$ depend on the state variables $\boldsymbol{u}$.

# Chapter 3

# Optimization

In this chapter, the general frameworks of the nested and sand approaches are described. The differences between these methods are explained. Furthermore, a brief note on the optimization solvers used, as well as the scaling and normalization of the involved equations, is made.

## 3.1 Nested approach

The nested approach assumes that the residual equations are already solved, meaning that $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) = 0$ is solved for $\boldsymbol{u}$ for a set of design variables $\boldsymbol{p}$ using Newton-Raphson. The general optimization problem can then be described as

$$\begin{aligned} \underset{\boldsymbol{p}}{\text{minimize}} \quad & f(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) \\ \text{subject to} \quad & \boldsymbol{g}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) \leq \boldsymbol{0}, \\ & \boldsymbol{h}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) = \boldsymbol{0}, \end{aligned}$$

where:

- $f(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p}))$ is the objective function to be minimized,
- $\boldsymbol{g}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) \leq \boldsymbol{0}$ are inequality constraints, and
- $\boldsymbol{h}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) = \boldsymbol{0}$ are equality constraints.

For every optimization iteration the residual equations need to be solved for that specific set of design variables $\boldsymbol{p}$. The computational cost of the Newton-Raphson iterations for solving the residual equations at each optimization step is the main disadvantage of the nested approach. Furthermore, it is required that the Newton-Raphson iterations converge for each evaluated design, which may become problematic if the optimizer probes

certain structure designs. These issues could potentially be alleviated by using the solution of similar designs as an initial guess for the Newton-Raphson iterations. That is however not considered in this work. Instead, each sequence of Newton-Raphson iterations starts at the same initial guess, being a vector full of zeros.

## 3.2 SAND approach

With SAND analysis and design the residual equations are solved at the same time as the optimization is performed [9]. This means that the residual equations are implemented as equality constraints. The SAND approach can outperform the nested approach as is showed in [10] in for numerical gradients.
The advantage is that the implicit relation between $\boldsymbol{p}$ and $\boldsymbol{u}$ is no longer present. This is the reason why for the analytical gradient computation the SAND approach does not need to use the direct or adjoint method. The general optimization problem is described as:

$$\begin{aligned} \text{Minimize} \quad & f(\boldsymbol{u}, \boldsymbol{p}) && (3.1)\\ \text{Subject to} \quad & \boldsymbol{g}(\boldsymbol{u}, \boldsymbol{p}) \leq \boldsymbol{0}, && (3.2)\\ & \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0} && (3.3)\\ & \boldsymbol{h}(\boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0}, && (3.4) \end{aligned}$$

where:

- $f(\boldsymbol{u}, \boldsymbol{p})$ is the objective function to be minimized,
- $\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{p}) \leq \boldsymbol{0}$ are inequality constraints,
- $\boldsymbol{r}(\boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0}$ are the residual equations, and
- $\boldsymbol{h}(\boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0}$ are other equality constraints.

The initial guess for $\boldsymbol{u}$ is found by solving the model one time for the initial $\boldsymbol{p}$, as in the nested approach. This is a relatively realistic estimate for only a modest increase in computing time.

## 3.3 fmincon

In the fmincon function in MATLAB the preferred optimization algorithm can be selected. The three most relevant algorithms for gradient-based optimization with fmincon are: sequential quadratic programming (SQP), active-set, and interior-point (IP). SQP algorithms are similar to active-set algorithms. SQP algorithms are in most situations

preferred over active-set algorithms [11]. Interior-point algorithms differ from SQP and active-set algorithms. IP algorithms incorporates barrier functions. This ensure that the iterations remain within the feasible region, causing IP algorithms to be stable to find the optimal solution. SQP algorithms solve a sequence of quadratic sub problems to approximate the original nonlinear problem. The steps taken by SQP can be in directions that momentarily violate constraints. Therefore, SQP algorithms are generally faster but can be less stable than IP algorithms [12]. The robustness and speed of each solver are highly dependent on the specific problem. These are general assumptions, therefore, in this research, both the SQP and IP algorithms will be tested across various problems. This approach is taken because one algorithm might be better suited to the nested approach than the SAND approach and vice versa. The solvers will be evaluated based on computational efficiency and robustness, specifically whether the solver can consistently converge to the correct optimum, even with changes in initial guesses or slight variations in the problem.

## 3.4 Scaling and normalization

It is important to note that in the residual equations being solved, the solution vector may consist of entities with various different units and magnitudes. This can lead to poorly scaled matrix equations, resulting in inaccurate solutions. Therefore, when implementing the algorithm, it is crucial to ensure appropriate scaling of the equations is applied to facilitate accurate solution finding.

Similarly, normalization of the design variables and objective/constraint functions is a common practice to aid optimization algorithms in finding good solutions. It should be recognized that these normalization factors affect the derivative terms and consequently also affect the optimization search direction. For example, if t $p_n$ is a normalized design variable, defined as $p_n = p * c$, with $c$ a constant. Then

$$\frac{df(p)}{dp_n} = \frac{\partial f(p)}{\partial p}\frac{dp}{dp_n} = \frac{\partial f(p)}{\partial p} \cdot \frac{1}{c}. \tag{3.5}$$

In this work, only $\frac{df}{dp}$ is mentioned. While these factors are omitted from the equations in this report for readability, it is essential to acknowledge that normalization is applied and implemented for all examples and results presented herein, for both design variables and objective/constraint functions.

For the nested approach, the residual equations are scaled to have roughly the same magnitude, and these scaled residual equations are used in the SAND approach. This means it is important to scale the residual equations back to sizes of the other objective and constraint functions, which are normalized. The scaling of the residual equations in the SAND approach can have such a large influence on the computational time that it is

investigated separately. This is explained further in Chapter 6 and Chapter 7.

# Chapter 4

# Theory of gradient computation

In this chapter the theory of gradient computation is detailed. The chapter begins with an explanation of numerical gradient computation in Section 4.1, which is used by most gradient-based optimization algorithms. The chapter then proceeds to explain analytical gradient computation for both the nested and SAND approaches in Section 4.2. To obtain analytical gradients using the nested approach, two methods are elaborated: the direct method and the adjoint method. It is essential to clarify which gradients are being sought. These are the derivatives of the objective and constraint functions with respect to the design variables. The theory applies equally to both objective and constraint functions. Therefore, throughout this explanation, $\boldsymbol{g}$ (or $g$ for a single constraint) will be used, but the exact same approach applies for an objective function, $f$ or an equality constraint, $h$. This chapter provides the theoretical explanation of the gradient computation. The implementation of the analytical gradients is discussed in Chapter 5.

## 4.1 Numerical gradient computation

### 4.1.1 Finite difference

By default, fmincon utilizes finite differences for computing the derivatives of the objective and constraint functions in MATLAB. The basic types of finite difference are central difference, backward difference, and forward difference. Backward and forward difference are $\mathcal{O}(h)$ accurate, while central difference is $\mathcal{O}(h^2)$ accurate, with $h$ the step size of the discretization. However, central difference requires two extra function evaluations for each gradient term to be determined, while forward and backward difference only require one extra function evaluation.

The truncation error decreases as the step size decreases. However, a smaller step size value may lead to larger rounding errors due to the subtraction in the finite difference equation. Even tough there exists an optimal step size $h$. This trade-off is not further discussed here and the default settings of fmincon are used. Forward difference is the

default derivative computation method in fmincon. The approximation of the derivative of a function $g$ to a design variable $p$ using forward difference is

$$\frac{\partial g}{\partial p}(p) = \frac{g(p+h) - g(p)}{h} + \mathcal{O}(h). \tag{4.1}$$

This approximation has to be performed for every component of the design variables $\boldsymbol{p}$. Higher-order derivatives can also be obtained using finite differences, provided that the function is differentiable to the required order.
Using finite difference the model is treated as a so-called black box, see Figure 4.1. This means that only the input and output of the system determine the characteristics of the problem. To compute the derivative of objective or constraint functions with respect to the vector of design variables, each design variable must be perturbed, and the model must be solved. This necessitates solving the model $N+1$ times, where $N$ is the number of design variables. Finite differences can be used for both the nested and SAND approach.

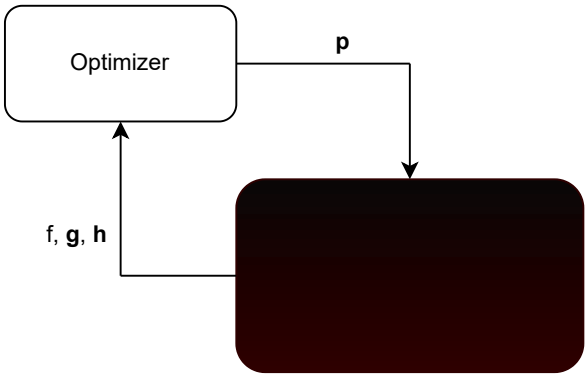



Figure 4.1: Finite difference: black box model.

## 4.2 Analytical gradient computation

To find the best design, an objective function and constraint functions are introduced. The optimization algorithm needs the derivatives of the objective function, $f$, and constraint functions, $\boldsymbol{g}$, to the design variables $\boldsymbol{p}$. To obtain the analytical gradients for the nested approach, either the direct or adjoint method can be used. In this section, both methods are derived and explained. A comparison of the direct and adjoint methods is included. Subsequently, the required gradients for both the nested and SAND approaches are elaborated upon.

### 4.2.1 Nested

Although the derivations of the direct and adjoint methods differ, the outcomes are the same. The primary distinction lies in the sequence of steps performed in each method,

and in which system of equations to solve first. Although it is not necessary for the final result, both derivations are provided. These derivations offer different perspectives, which can help in understanding the underlying processes.

### Direct method

The direct method as described in [9] is a method that rewrites the derivative formulation in such a way that the derivatives can be obtained in analytical form. The model can be solved for a certain given set of design variables. The objective and constraint functions depend on the design variables, but also on the solution of the model $\boldsymbol{u}$,

$$\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})). \tag{4.2}$$

The derivation presented here is done for the constraint functions. Notice that $\boldsymbol{u}$ implicitly depends on $\boldsymbol{p}$. Hence, using the chain rule, the total derivative is given by:

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\boldsymbol{p}}. \tag{4.3}$$

The model is solved, thus $\boldsymbol{r}(\boldsymbol{p}, \boldsymbol{u}(\boldsymbol{p})) = \boldsymbol{0}$. This means that any perturbation $d\boldsymbol{p}$ must be accompanied by a perturbation $d\boldsymbol{u}$ such that the governing equations remain satisfied, meaning that $d\boldsymbol{r}$ should be 0. Therefore, the differential of the residuals is written as

$$d\boldsymbol{r} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}} d\boldsymbol{p} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} d\boldsymbol{u} = \boldsymbol{0}. \tag{4.4}$$

This equation can be rewritten into,

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{d\boldsymbol{p}} = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}. \tag{4.5}$$

Given that the partial derivatives $\partial \boldsymbol{r} \partial \boldsymbol{u}$ and $\partial \boldsymbol{r} \partial \boldsymbol{p}$ are known from the residual function, the total derivative $d\boldsymbol{u}/d\boldsymbol{p}$ can be found by solving the above linear system:

$$\frac{d\boldsymbol{u}}{d\boldsymbol{p}} = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}. \tag{4.6}$$

Substituting expression 4.6 in equation 4.3 results in the expression for the total derivative,

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} \left( \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \right)^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}, \tag{4.7}$$

which can be solved in two steps. For the direct method first $\boldsymbol{\phi}$ is found as the solution of the linear system of equation (4.5), and then the full derivative is solved:

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\boldsymbol{\phi} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}, \tag{4.8a}$$

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\boldsymbol{\phi}. \tag{4.8b}$$

The partial derivatives of $\boldsymbol{r}$ and $\boldsymbol{g}$ to $\boldsymbol{p}$ and $\boldsymbol{u}$ may be obtained analytically, but can be obtained by finite differences as well if needed.
Note that $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}$ is equal to the Jacobian of the residual function (equation (2.37)), and that the system of equations with the Jacobian is already solved during the Newton-Raphson iterations, see equation (2.44). The matrix decomposition that is determined while solving the set of equations can be re-used, making that solving subsequent equations with the same Jacobian matrix is relatively inexpensive.

### Adjoint method

The adjoint method is similar to the direct method, the difference is in the order in which the final derivative is calculated. The derivation of the adjoint method also differs, and will be presented here [13]. The same names for the objective and constraint functions, design variables, state variables and residual function as in the direct method derivation are used. Suppose

$$\boldsymbol{L} = \boldsymbol{g} - \boldsymbol{\lambda}^T \boldsymbol{r}, \tag{4.9}$$

were $\boldsymbol{L}$ is a Lagrangian (a different Lagrangian then the one used for the potential energy, $\mathcal{L}$), and $\boldsymbol{\lambda}$ are Lagrange multipliers for the residual functions. This represents the Lagrangian of the function to be differentiated under the constraint of residual function satisfaction. Given that the residual equation $\boldsymbol{r} = 0$ should remain satisfied, it can be seen that the total derivative of the Lagrangian $\frac{d\boldsymbol{L}}{d\boldsymbol{p}}$ is equal to the total derivative of the function of interest $\frac{d\boldsymbol{g}}{d\boldsymbol{p}}$.
The total derivative of the Lagrangian is given as:

$$\frac{d\boldsymbol{L}}{d\boldsymbol{p}} = \frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}}{d\boldsymbol{p}} - \boldsymbol{\lambda}^T \left( \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\frac{d\boldsymbol{u}}{d\boldsymbol{p}} \right). \tag{4.10}$$

The equation can be rearranged by grouping the terms with $\frac{d\boldsymbol{u}}{d\boldsymbol{p}}$:

$$\frac{d\boldsymbol{L}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} + \left( \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} \right) \frac{d\boldsymbol{u}}{d\boldsymbol{p}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}} \tag{4.11}$$

Now, select the Lagrange multipliers $\boldsymbol{\lambda}$ such that

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = 0. \tag{4.12}$$

As with the direct method, the total derivative of the function of interest can be determined in two steps. First, find $\boldsymbol{\lambda}$ by solving the following set of linear equations:

$$\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda} = \left(\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}\right)^T, \tag{4.13}$$

Then find the total derivative of the function of interest by substituting the Lagrange multipliers in the following equation:

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} - \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}. \tag{4.14}$$

**Direct vs adjoint**

The direct method ((4.8)) and the adjoint method (equations (4.13) and (4.14)) give exactly the same result, with the only difference being the order of solving the set of equations.
The direct and adjoint method are visualized in Figure 4.2. The model, $\boldsymbol{r}(\boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{0}$, has to be solved only once to obtain analytical expressions for $f$, $\boldsymbol{g}$ and $\boldsymbol{h}$.

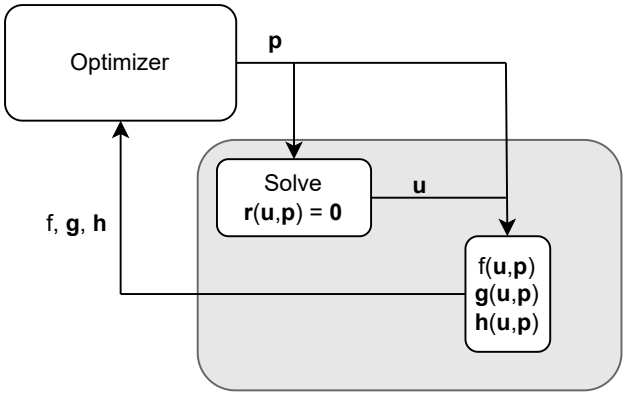



Figure 4.2: Implicit analytic differentiation using residuals.

The difference between the direct method and the adjoint method lies in the size of the linear system that has to be solved. For the direct method that system is equation 4.8a, This means the right hand side depends on the number of the design variables. In the adjoint method the linear system of equation 4.13 is solved, here the right hand side depends on the total number of objective and constraint equations. To observe a significant difference in computational efficiency between the direct and adjoint method, there should be a substantial discrepancy in the size of the right-hand sides of both

methods:

$$\text{Method} = \begin{cases} \text{Direct}, & \text{if } n_f \gg n_p \\ \text{Adjoint}, & \text{if } n_p \gg n_f \end{cases}, \tag{4.15}$$

were $n_f$ is the number of objective and constraint equations and $n_p$ is the number of design variables.

To find analytical derivatives for the nested approach, it is necessary to determine the partial derivatives of the objective and constraint functions with respect to the design variables and state variables. Additionally, the partial derivatives of the residual equations with respect tot the design variables and state variables must also be obtained. In symbols this mean the following analytical derivatives have to be found:

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}, \tag{4.16a}$$

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}, \tag{4.16b}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}}, \tag{4.16c}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}}, \tag{4.16d}$$

were $\boldsymbol{g}$ represents both objective and constraint functions. These are partial derivatives so the implicit relation between $\boldsymbol{u}$ and $\boldsymbol{p}$ does not impact these terms.

### 4.2.2 SAND

In the SAND approach, the design variables are extended to include the state variables $\boldsymbol{u}$. This eliminates the implicit relationship between $\boldsymbol{p}$ and $\boldsymbol{u}$. Consequently, the direct and adjoint methods are no longer necessary, as the full gradients can be computed directly. The primary difference now is that the full derivatives of the objective and constraint functions need to be found with respect the state variables $\boldsymbol{u}$ and design variables $\boldsymbol{p}$.

Another difference with the nested approach is that the residual equations now become constraints. Therefore, the derivatives of the residual equations with respect to $\boldsymbol{u}$ and $\boldsymbol{p}$ are also required. The final gradients to be determined in analytical form for the SAND approach are:

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \frac{d\boldsymbol{r}}{d\boldsymbol{u}}, \tag{4.17a}$$

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}} = \frac{d\boldsymbol{r}}{d\boldsymbol{p}}, \tag{4.17b}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{u}} = \frac{d\boldsymbol{g}}{d\boldsymbol{u}}, \tag{4.17c}$$

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{p}} = \frac{d\boldsymbol{g}}{d\boldsymbol{p}}, \tag{4.17d}$$

Because the implicit relation between $\boldsymbol{p}$ and $\boldsymbol{u}$ is gone, these four partial gradients are exactly equal to the four full gradients. For the nested and SAND approach the same four analytical gradients have to be obtained.

# Chapter 5

# Implementation of analytical gradients

In this chapter, the framework for obtaining the four gradient terms needed for optimization is presented. The derivative terms are written as partial derivatives following the nested approach. These are, in fact, exactly the same terms needed for the SAND approach, equation (4.17). The chapter starts with the gradients of the residual function and concludes with the gradients of the four chosen potential objective and constraints functions: mass, stress, displacement, and compliance.

## 5.1 Theoretical framework

The general residual equations are given in equation (5.1). The gradients of the residual equations to the state variables are already determined for the Newton-Raphson process, see equation (2.12). Hence, only the gradients of the residual equations $\boldsymbol{r}$ to the design variables $\boldsymbol{p}$ have to be obtained. Without specifying the type of design variable

$$\nabla_{\boldsymbol{p}}\nabla_{\boldsymbol{q}_n}\mathcal{L} = \nabla_{\boldsymbol{p}}\left(-\nabla_{\boldsymbol{q}_n}W_{ext}\right)^\top + \nabla_{\boldsymbol{p}}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top + \nabla_{\boldsymbol{p}}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{5.1a}$$

$$\nabla_{\boldsymbol{p}}\nabla_{\boldsymbol{q}_e}\mathcal{L} = \nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top + \nabla_{\boldsymbol{p}}\left(\nabla_{\boldsymbol{q}_e}\boldsymbol{g}_e^\top\boldsymbol{\lambda}_e\right)^\top, \tag{5.1b}$$

$$\nabla_{\boldsymbol{p}}\nabla_{\boldsymbol{\lambda}_n}\mathcal{L} = \nabla_{\boldsymbol{p}}\boldsymbol{g}_n^\top, \tag{5.1c}$$

$$\nabla_{\boldsymbol{p}}\nabla_{\boldsymbol{\lambda}_e}\mathcal{L} = \nabla_{\boldsymbol{p}}\boldsymbol{g}_e^\top. \tag{5.1d}$$

This gradients depend on the choice of design variables. There are many combinations of variables that can be used as design variables. The basis of these combinations consists of six different types of design variables as outlined in Chapter 1. These are

$$\boldsymbol{p} = [A^{(k)}, E^{(k)}, x_i, y_i, x_j, y_j]. \tag{5.2}$$

$A^{(k)}$ is the cross-section of element $k$, $E^{(k)}$ is the Young's modulus of element $k$. Furthermore, $x_i$, $y_i$, $x_j$ and $y_j$ are the initial nodal coordinates. It is assumed that $W_{ext}$ is linear in $\boldsymbol{q}_n$. Therefore, $\nabla_{\boldsymbol{p}} \left(-\nabla_{\boldsymbol{q}_n} W_{ext}\right)^\top = 0$. Furthermore, when it is assumed that the nodal constraint are linear in $\boldsymbol{q}_n$ then $\nabla_{\boldsymbol{p}} \left(\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top \boldsymbol{\lambda}_n\right)^\top = 0$ for any design variable. The nodal constraints only depend on the nodal displacement $\boldsymbol{q}_n$ hence $\nabla_{\boldsymbol{p}} \boldsymbol{g}_n^\top$ is also zero for all of the basis design variables.
If the design variable is $A$ or $E$ the gradient is fully determined by the strain energy term, $\nabla_{\boldsymbol{q}_e} W_{int}$. This is the only component that depends on $A$ and $E$. All other terms will be zero in this case. When the design variable is an initial nodal coordinate the gradients involve more terms. Nodal constraint equations are considered to be linear in $\boldsymbol{q}_n$, meaning that $\nabla_{\boldsymbol{p}} \left(\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top\right) \boldsymbol{\lambda}_n = 0$. The second derivative terms that are not zero, involve either $L^{(k)}$ or $L_0^{(k)}$. Therefore, the next section is devoted to systematically derive the derivative of $L^{(k)}$ with respect to the four nodal coordinates:$x_i, y_i, x_j$, and $y_j$. These derivatives are equal to the derivatives of $L^{(k)}$ to the nodal displacements. The derivatives of $L_0^{(k)}$ to the initial nodal coordinates are similar to the derivatives of $L^{(k)}$ to the nodal coordinates.

## 5.2 Partial derivatives of the truss length

In this section the general first, second and third order derivatives of $L^{(k)}$ and $L_0^{(k)}$ with respect to the initial nodal coordinates are derived.
Recall the formulation of a truss. The four basic variables that define the truss are the nodal coordinates ($x_i$, $y_i$, $x_j$ and $y_j$). These coordinates are present in the full Lagrangian, in $L^{(k)}$ and $L_0^{(k)}$:

- $L^{(k)} = L^{(k)}\left(x_i, u_i y_i, v_i, x_j, u_j, y_j, v_j\right)$,
- $L_0^{(k)} = L_0^{(k)}\left(x_i, y_i, x_j, y_j\right)$.

The partial derivatives of the current length to the state variables $u_i, v_i, u_j$ and $v_j$, and to the design variables $x_i, y_i, x_j$ and $y_j$ will be used in multiple of the required partial derivatives. These partial derivatives of the current element length will therefore be first derived. Even tough elements can share nodes, the analytical derivatives can be obtained per element. All contributions can be added to the correct entry. Before specifying which derivatives are needed, the derivatives of the common factor $L^{(k)}$ with respect to the nodal coordinates will be written out. First, some important definitions are stated:

$$\Delta x = x_j + u_j - x_i - u_i, \tag{5.3a}$$

$$\Delta y = y_j + v_j - y_i - v_i, \tag{5.3b}$$

$$\Delta x_0 = x_j - x_i, \tag{5.3c}$$

$$\Delta y_0 = y_j - y_i, \tag{5.3d}$$

$$L = \sqrt{\Delta x^2 + \Delta y^2} = \sqrt{(x_j + u_j - x_i - u_i)^2 + (y_j + v_j - y_i - v_i)^2}, \tag{5.4a}$$

$$L_0 = \sqrt{\Delta x_0^2 + \Delta y_0^2} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \tag{5.4b}$$

$$\cos(\beta) = \frac{\Delta x}{L} = \frac{x_j + u_j - x_i - u_i}{\sqrt{(x_j + u_j - x_i - u_i)^2 + (y_j + v_j - y_i - v_i)^2}} = s, \tag{5.5a}$$

$$\sin(\beta) = \frac{\Delta y}{L} = \frac{y_j + v_j - y_i - v_i}{\sqrt{(x_j + u_j - x_i - u_i)^2 + (y_j + v_j - y_i - v_i)^2}} = c, \tag{5.5b}$$

were $\beta$ is the angle between the truss in its deformed position, and the $x$-axis of the global coordinate system. From this point onward these terms will not be fully written out. It should be recognized that the derivatives of $L^{(k)}$ to the nodal displacements are equal to the derivatives of $L^{(k)}$ to the nodal coordinates. This is helpful as the nodal displacements are part of the solution vector $\boldsymbol{u}$ and these derivatives are desired. With these definitions the derivatives can be built up.

The current length $L^{(k)}$ is present in the element constraints. The elements constraints are implemented in the Lagrangian from Equation (2.1) in the following form

$$g_e^{(k)}\lambda_e^{(k)} = \lambda_e^{(k)}(L^{(k)} - L_0^{(k)} - \Delta L^{(k)}) = 0, \tag{5.6}$$

for every element $k$. Only the factor $L^{(k)}$ depends on the nodal displacements, $\boldsymbol{q}_n$. Notice that for the derivatives to the nodal coordinates an extra term is present, the derivative of $L_0^{(k)}$.

The first derivative of $\lambda_e^{(k)}L^{(k)}$ with respect to the nodal displacements is

$$\nabla_{\boldsymbol{q}_n} g_e^{(k)}\lambda_e^{(k)} = \begin{bmatrix} \frac{\partial \lambda_e L}{\partial x_i} & \frac{\partial \lambda_e L}{\partial y_i} & \frac{\partial \lambda_e L}{\partial x_j} & \frac{\partial \lambda_e L}{\partial y_j} \end{bmatrix}^\top = \lambda_e \begin{bmatrix} -\cos\beta & -\sin\beta & \cos\beta & \sin\beta \end{bmatrix}^\top. \tag{5.7}$$

For this and the following derivative terms the element indication $k$ is left out for clearness. Once more the derivatives can be taken, resulting in

$$\nabla_{\boldsymbol{q}_n} \left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top = \frac{\lambda_e}{L} \begin{bmatrix} s^2 & -sc & -s^2 & sc \\ -sc & c^2 & sc & -c^2 \\ -s^2 & sc & s^2 & -sc \\ sc & -c^2 & -sc & c^2 \end{bmatrix} \tag{5.8}$$

In the definition of the compliance the second order derivatives of the Lagrangian are present, this is explained the next section. Therefore, the third order derivatives are necessary to obtain the derivatives of the compliance. The third order derivative to the

nodal displacements is a third order tensor. All four indices are written out as matrices.

$$\nabla_{qxi}\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top = \frac{\lambda_e}{L^2}\begin{bmatrix} -3s^2c & -2c^2s - s^3 & 3s^2c & 2c^2s + s^3 \\ -2c^2s - s^3 & 2s^2c - c^3 & 2c^2s + s^3 & -2s^2c + c^3 \\ 3s^2c & 2c^2s + s^3 & -3s^2c & -2c^2s - s^3 \\ 2c^2s + s^3 & -2s^2c + c^3 & -2c^2s - s^3 & 2s^2c - c^3 \end{bmatrix}, \quad (5.9)$$

$$\nabla_{qyi}\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top = \frac{\lambda_e}{L^2}\begin{bmatrix} 2c^2s + s^3 & -2s^2c + c^3 & -2c^2s - s^3 & 2s^2c - c^3 \\ -2s^2c + c^3 & -3c^2s & 2s^2c - c^3 & 3c^2s \\ -2c^2s - s^3 & 2s^2c - c^3 & 2c^2s + s^3 & -2s^2c + c^3 \\ 2s^2c - c^3 & 3c^2s & -2s^2c + c^3 & -3c^2s \end{bmatrix}, \quad (5.10)$$

$$\nabla_{qxj}\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top = \frac{\lambda_e}{L^2}\begin{bmatrix} -3s^2c & 2c^2s - s^3 & 3s^2c & -2c^2s + s^3 \\ 2c^2s - s^3 & 2s^2c - c^3 & -2c^2s + s^3 & -2s^2c + c^3 \\ 3s^2c & -2c^2s + s^3 & -3s^2c & 2c^2s - s^3 \\ -2c^2s + s^3 & -2s^2c + c^3 & 2c^2s - s^3 & 2s^2c - c^3 \end{bmatrix}, \quad (5.11)$$

$$\nabla_{qyj}\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top = \frac{\lambda_e}{L^2}\begin{bmatrix} 2c^2s - s^3 & 2s^2c - c^3 & -2c^2s + s^3 & -2s^2c + c^3 \\ 2s^2c - c^3 & -3c^2s & -2s^2c + c^3 & 3c^2s \\ -2c^2s + s^3 & -2s^2c + c^3 & 2c^2s - s^3 & 2s^2c - c^3 \\ -2s^2c + c^3 & 3c^2s & 2s^2c - c^3 & -3c^2s \end{bmatrix}. \quad (5.12)$$

The notation $\nabla_{qxi}$ means the derivative to the $x$ coordinate of the $i$th side of the element. The notation takes the derivatives to the initial nodal coordinates. These expressions are the same for the derivatives to the nodal displacements.
These are the first, second and third order derivatives of $L^{(k)}$. Derivatives of $L_0^{(k)}$ to the nodal displacements are zero. However, derivatives of $L_0^{(k)}$ to the initial nodal coordinates are not zero. These derivatives are not explicitly written out here because the expressions are obtained in a similar way as the derivatives of $L^{(k)}$.

## 5.3 Partial derivatives of the residual function

The two derivatives of the residual equations needed for the implementation of the analytical derivatives are

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{u}^2} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}, \quad (5.13a)$$

$$\frac{\partial^2 \mathcal{L}}{\partial \boldsymbol{u} \partial \boldsymbol{p}} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}. \quad (5.13b)$$

The partial derivative $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}$ corresponds with the Jacobian $J$ that is used in the Netwon-Raphson procedure to solve the system of equations. The Jacobian matrix is given in general terms in Section 2.2.
The general linearized equations are given in equation (2.43). The geometric term $\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top\boldsymbol{\lambda}_n\right)^\top$ is obtained by assembling all the matrices $\nabla_{\boldsymbol{q}_n}\left(\nabla_{\boldsymbol{q}_n}g_e\lambda_e\right)^\top$, see Equation (5.8). The element contributions are added together in the correct entry of the matrix. The term $\nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top$ results in a diagonal matrix with on the diagonal the stiffness $k_i$ of all elements. Hence, the size of this matrix is $nEl \times nEl$. The nodal constraint term, $\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_n^\top$, is a matrix of size $nNoDOF \times nNoCo$ and consists zeros, with ones placed to constraint the desired nodal degrees of freedom. The element constraint term, $\nabla_{\boldsymbol{q}_n}\boldsymbol{g}_e^\top$ are the first derivatives of the the length $L^{(k)}$ as the other terms in the element constraint do not depend on $\boldsymbol{q}_n$. Therefore these are given by $\nabla_{\boldsymbol{q}_n}g_e^{(k)}$, without the $\lambda_e^{(k)}$ as in Equation (5.7). Every element constraint will be placed in a new column, and the four terms correspond to the four nodal degrees of freedom of each element and should be placed accordingly.
The final Jacobian found is the Jacobian one step before convergence is reached. It can be argued to directly use this Jacobian from the NR procedure. Instead of creating the updated Jacobian for the final solution vector, because during this procedure the decomposition of the Jacobian is already made. This is computationally the most expensive step in the solution algorithm. When the convergence tolerance is set 'high enough' this Jacobian can produce derivatives of the objective and constraint functions that are sufficiently accurate. This prevents one calculation of the decomposition of the Jacobian for every optimization iteration.

The derivative of the residual function to the design variables is more involved. This derivative depends on the choice of design variables. The derivative $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{p}}$ is built up element-wise. This means there are six possibilities for every column, where the row represents a residual equation, and the column the design parameter of interest, which can be of one of the six above types. The values and placement in the columns may vary, but the equations can generically be described in terms of element $k$. All other design variables can be obtained from this basis and are left for the reader to construct. For example, using the initial length of element $k$ as a design variable: $L_0^{(k)} = L_0^{(k)}(x_i, y_i, x_j, y_j)$.
The derivatives of the residual function to these six basis design variables are elaborated here. If $\boldsymbol{p}$ is $A$ or $E$ then only the strain energy term, $\nabla_{\boldsymbol{q}_e}\left(\nabla_{\boldsymbol{q}_e}W_{int}\right)^\top$ results in a nonzero term. If, $p_i = A^{(k)}$ then

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} \frac{E^{(k)}\Delta L^{(k)}}{L_0^{(k)}} & \text{if } l = nNoEq + k \\ 0 & \text{if } l \neq nNoEq + k \end{cases} \text{ for } l \in ElEq. \tag{5.14}$$

If, $p_i = E^{(k)}$ then

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} \frac{A^{(k)}\Delta L^{(k)}}{L_0^{(k)}} & \text{if } l = nNoEq + k \\ 0 & \text{if } l \neq nNoEq + k \end{cases} \text{ for } l \in ElEq. \tag{5.15}$$

When $\boldsymbol{p}$ is one of the initial nodal coordinates, three terms result in nonzero derivative terms. The geometric term $\nabla_{\boldsymbol{p}} \left(\nabla_{\boldsymbol{q}_n} \boldsymbol{g}_n^\top \boldsymbol{\lambda}_n\right)^\top$ which is now a single row of the matrix $\nabla_{\boldsymbol{q}_n} \left(\nabla_{\boldsymbol{q}_n} g_e \lambda_e\right)^\top$ from Equation (5.8), as the derivative to just one nodal coordinate is sought. If, $p_i = x_i$ then,

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{[x_{iDof}, y_{iDof}, x_{jDof}, y_{jDof}], i} = \begin{bmatrix} s^2 & -sc & -s^2 & sc \end{bmatrix} \frac{\lambda_e^{(k)}}{L^{(k)}}. \tag{5.16}$$

The second nonzero term is caused by the $L_0^{(k)}$ present in the $k_i$ in the strain energy term: $\nabla_{\boldsymbol{q}_e} \left(\nabla_{\boldsymbol{q}_e} W_{int}\right)^\top$.

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} E^{(k)} A^{(k)} \Delta L^{(k)} \frac{x_j - x_i}{\left(L_0^{(k)}\right)^3} & \text{if } l = nNoEq + k \\ 0 & \text{if } l \neq nNoEq + k \end{cases} \text{ for } l \in ElEq, \tag{5.17a}$$

The last nonzero term comes from the element constraint equations, $\nabla_{\boldsymbol{p}} \boldsymbol{g}_e^\top$.

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} -\cos\beta^{(k)} + \frac{x_j - x_i}{\left(L_0^{(k)}\right)^3} & \text{if } l = nNoEq + nElEq + nNoCo + k \\ 0 & \text{if } l \neq nNoEq + nElEq + nNoCo + k \end{cases} \text{ for } l \in ElCo. \tag{5.17b}$$

No matter which nodal coordinate is taken as design variable, the same components are nonzero. The results are similar but not the same. If, $p_i = y_i$ then

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{[x_{iDof}, y_{iDof}, x_{jDof}, y_{jDof}], i} = \begin{bmatrix} -sc & c^2 & sc & -c^2 \end{bmatrix} \frac{\lambda_e^{(k)}}{L^{(k)}}, \tag{5.18a}$$

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} E^{(k)} A^{(k)} \Delta L^{(k)} \frac{y_j - y_i}{\left(L_0^{(k)}\right)^3} & \text{if } l = nNoEq + k \\ 0 & \text{if } l \neq nNoEq + k \end{cases} \text{ for } l \in ElEq, \tag{5.18b}$$

$$\left(\frac{\partial \boldsymbol{r}}{\partial p_i}\right)_{l,i} = \begin{cases} -\sin\beta^{(k)} + \frac{y_j - y_i}{\left(L_0^{(k)}\right)^3} & \text{if } l = nNoEq + nElEq + nNoCo + k \\ 0 & \text{if } l \neq nNoEq + nElEq + nNoCo + k \end{cases} \text{ for } l \in ElCo. \tag{5.18c}$$

Lastly, it can be noticed that

$$\left(\frac{\partial \boldsymbol{r}}{\partial x_i}\right) = -\left(\frac{\partial \boldsymbol{r}}{\partial x_j}\right) \tag{5.19a}$$

and

$$\left(\frac{\partial \boldsymbol{r}}{\partial y_i}\right) = -\left(\frac{\partial \boldsymbol{r}}{\partial y_j}\right). \tag{5.19b}$$

These are the derivatives for one element. Elements share nodes, resulting in multiple contributions to the same entries of the final matrix. It is important that in that case the contributions are added together. With this method both partial derivatives of the residual function can be computed.

## 5.4 Objective and constraint functions

In this section the generic derivatives for four different objective and constraint functions are described: mass, stress, compliance and displacement. The mass and the compliance are treated as an objective function, the stress and displacement are set up as a constraint function. However, all four functions can be used as objective and constraint functions.

### 5.4.1 Mass objective

The mass of the system is defined as the sum of the mass of all elements

$$M(\boldsymbol{p}) = \sum_{k=1}^{nElements} \rho^{(k)} A^{(k)} L_0^{(k)}. \tag{5.20}$$

The mass is always independent of the solution vector $\boldsymbol{u}$

$$\frac{\partial M}{\partial \boldsymbol{u}} = \boldsymbol{0}. \tag{5.21}$$

However, derivatives with respect to $\boldsymbol{p}$ are not necessarily zero. $A^{(k)}$ can be a component of $\boldsymbol{p}$ and $L_0^{(k)} = L_0^{(k)}(x_i, x_j, y_i, y_j)$ may depend on components in $\boldsymbol{p}$.

$$\frac{\partial M}{\partial A^{(k)}} = L_0^{(k)}, \tag{5.22a}$$

$$\frac{\partial M}{\partial x_i} = -A^{(k)} \frac{x_j - x_i}{L_0^{(k)}}, \tag{5.22b}$$

$$\frac{\partial M}{\partial y_i} = -A^{(k)} \frac{y_j - y_i}{L_0^{(k)}}. \tag{5.22c}$$

Again, the derivatives to nodal coordinates of the other side of the element $(x_j, y_j)$ are simply negated:

$$\frac{\partial M}{\partial x_i} = -\frac{\partial M}{\partial x_j}, \tag{5.23a}$$

$$\frac{\partial M}{\partial y_i} = -\frac{\partial M}{\partial y_j}. \tag{5.23b}$$

Lastly, the trivial derivative to $E^{(k)}$

$$\frac{\partial M}{\partial E^{(k)}} = 0. \tag{5.24}$$

### 5.4.2 Stress constraint

The stress constraint is set up in such a way that the stress in every element should be below a certain threshold, $\sigma_{max}$. It is not known beforehand whether a certain element is in tension or compression. If the constraint would be defined in terms of the absolute value of the stress, it is not continuously differentiable, which can become problematic when using gradient-based optimization algorithms. Therefore, two constraint functions are defined, separately for both maximum allowed compressive and tensile stress. The stress constraint is given as

$$-\sigma_{max} \leq \frac{\Delta L^{(k)} E^{(k)}}{L_0^{(k)}} \leq \sigma_{max}, \tag{5.25}$$

or the way the stress constraint is implemented in the optimization algorithm

$$\frac{\Delta L^{(k)} E^{(k)}}{L_0^{(k)}} \leq \sigma_{max}, \tag{5.26a}$$

$$-\frac{\Delta L^{(k)} E^{(k)}}{L_0^{(k)}} \leq \sigma_{max}. \tag{5.26b}$$

The derivatives of these constraints to $\boldsymbol{u}$ and $\boldsymbol{p}$ need to be derived. It is evident that the derivatives of constraint equation 5.26b equal the negative of the derivatives of 5.26a. Therefore, only the derivatives of 5.26a are derived here.

$$\left(\frac{\partial \sigma^{(k)}}{\partial \boldsymbol{u}}\right)_l = \begin{cases} \frac{E^{(k)}}{L_0^{(k)}} & \text{if } l = nNoEq + k \\ 0 & \text{if } l \neq NoEq + k \end{cases} \quad \text{for } l \in ElEq \tag{5.27}$$

The term $\frac{\partial \sigma}{\partial \boldsymbol{p}}$ is a bit more involved.
If, $p_i = A^{(k)}$ then

$$\frac{\partial \sigma^{(k)}}{\partial A^{(k)}} = 0. \tag{5.28}$$

If, $p_i = E^{(k)}$ then

$$\frac{\partial \sigma^{(l)}}{\partial E^{(k)}} = \begin{cases} \frac{\Delta L^{(k)}}{L_0^{(k)}} & \text{if } l = k \\ 0 & \text{if } l \neq k \end{cases} \text{ for l = 1, 2, ..., } nEl. \tag{5.29}$$

$nEl$ are the number of elements in the structure.
If, $p_i = x_i$ then

$$\frac{\partial \sigma^{(l)}}{\partial x_i} = \begin{cases} \frac{\Delta L^{(k)} E^{(k)} \Delta x_0^{(l)}}{\left(L_0^{(k)}\right)^3} & \text{if } l = k \\ 0 & \text{if } l \neq k \end{cases} \text{ for l = 1, 2, ..., } nEl. \tag{5.30}$$

If, $p_i = x_i$ then all derivatives are the same as for $p_i = x_i$ but multiplied by minus one.
If, $p_i = y_i$ then

$$\frac{\partial \sigma^{(l)}}{\partial y_i} = \begin{cases} \frac{\Delta L^{(k)} E^{(k)} \Delta y_0^{(k)}}{\left(L_0^{(k)}\right)^3} & \text{if } l = k \\ 0 & \text{if } l \neq k \end{cases} \text{ for l = 1, 2, ..., } nEl. \tag{5.31}$$

If, $p_i = y_j$ then all equations are the same as for $p_i = y_i$ but with a minus in front of the added terms.

### 5.4.3 Compliance objective

Obtaining the compliance is more involved than the other objective and constraint functions. The derivation is presented here. Compliance is the inverse of stiffness, being the derivative of displacement with respect to force at a certain degree of freedom. The relevant degrees of freedom in the context of compliance are node displacements in $x$ and $y$ directions. Meaning the perturbation forces, $\boldsymbol{F}_p$ are added to the nodal equilibrium equations. Therefore, change in the applied perturbation force $d\boldsymbol{F}_p$ results in a corresponding reaction $d\boldsymbol{u}$ to assure the residual equations are still satisfied. The perturbation forces are added to nodes but all residual equations have to hold after the perturbation, hence the reaction $d\boldsymbol{u}$ are not only the nodal displacements.
This results in the following equation:

$$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p} d\boldsymbol{F} + \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} d\boldsymbol{u} = \boldsymbol{0}. \tag{5.32}$$

This expression follows from the chain rule after the derivative of $\boldsymbol{r}(\boldsymbol{u}(\boldsymbol{F}_p), \boldsymbol{F}_p)$ with respect to $\boldsymbol{F}_p$ is taken. The $d\boldsymbol{u}$ term can be isolated:

$$d\boldsymbol{u} = -\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p} d\boldsymbol{F}_p. \tag{5.33}$$

The compliance is the inverse of the stiffness. Using expression 5.33 for $d\boldsymbol{u}$, the compliance can be written as

$$\boldsymbol{C} = \frac{1}{\boldsymbol{K}} = \frac{d\boldsymbol{u}}{d\boldsymbol{F}_p} = -\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\right)^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}. \tag{5.34}$$

$\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}$ is an identity matrix with size $nUnknown \times nNoEq$. This means the matrix is not square. Consequently, the bottom half consists completely of zeros. This is because the force perturbations are only applied at the nodal degrees of freedom.
For optimization the derivatives of the compliance with respect to the solution vector, $\boldsymbol{u}$ and to the design variables, $\boldsymbol{p}$ have to be computed. With some matrix algebra both derivatives can be described as

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}}\left(-\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}\right) = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \cdot \frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u}^2} \cdot \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \cdot \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}, \tag{5.35a}$$

$$\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{p}} = \frac{\partial}{\partial \boldsymbol{p}}\left(-\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}\right) = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \cdot \frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u}\partial \boldsymbol{p}} \cdot \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}^{-1} \cdot \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{F}_p}. \tag{5.35b}$$

Recognize that: $\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}} = \boldsymbol{J}$, the Jacobian of the residual equations with respect to the solution vector, this term is already used in the Newton-Raphson procedure. The term, $\frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u}^2}$ is a third order tensor. Two different forms of this third order tensor can be distinguished. When the derivatives are taken to a nodal displacement $\nabla \boldsymbol{q}_n(\boldsymbol{J})$. As an example the derivatives to the nodal coordinate $x_i$ are taken, to show the general structure of the tensor, $\nabla \boldsymbol{q}_{x_i}(\boldsymbol{J})$. The derivatives to the other nodal coordinates are similar. the form is given by:

$$\nabla \boldsymbol{q}_{x_i}\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\right) = \begin{bmatrix} \nabla \boldsymbol{q}_{x_i} \nabla \boldsymbol{q}_n \left(\nabla \boldsymbol{q}_n g_e \lambda_e\right)^\top & \left(\nabla \boldsymbol{q}_{x_i}\left(\nabla \boldsymbol{q}_n g_e \lambda_e\right)^\top\right)^\top \\ \nabla \boldsymbol{q}_{x_i}\left(\nabla \boldsymbol{q}_n g_e \lambda_e\right)^\top & 0 \end{bmatrix}, \tag{5.36}$$

When the derivatives of $\boldsymbol{J}$ are taken to an element elongation $\boldsymbol{q}_e$ or a nodal reaction force $\boldsymbol{\lambda}_n$ that entire matrix corresponding to the index equals zero. For the derivative to an elemental reaction force $\boldsymbol{\lambda}_e$ an example is used, the derivative is taken to $\lambda_e^{(k)}$. A similar expression holds for the other elements.

$$\nabla \lambda_e^{(k)}\left(\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}\right) = \begin{bmatrix} \frac{1}{\lambda_e^{(k)}} \nabla \boldsymbol{q}_{x_i}\left(\nabla \boldsymbol{q}_n g_e \lambda_e\right)^\top & 0 \\ 0 & 0 \end{bmatrix}. \tag{5.37}$$

The terms used in the above equations were derived in Section 5.2.
It should be noted that these matrices are constructed element-wise, with each element contributing to the final tensor. All contributions should be assigned to the corresponding degree of freedom within that element.
The $\frac{\partial^2 \mathbf{u}}{\partial \mathbf{u}\partial \mathbf{p}}$ term is obtained in a similar fashion when the design variables are nodal coordi-

nates. The third-order derivatives of the Lagrangian with respect to the nodal coordinates are equal to the derivatives with respect to the nodal displacements, with one exception: the $\nabla \mathbf{q}_n \left(\nabla \mathbf{q}_e W_{\text{int}}\right)$ term. When the design variable is $x_i$:

$$\frac{\partial^2 \mathbf{r}}{\partial \mathbf{u} \partial \mathbf{p}} \left(\text{nNoEq} + k, \text{nNoEq} + k, i\right) = E^{(k)} A^{(k)} \frac{x_j - x_i}{\left(L_0^{(k)}\right)^3}. \tag{5.38}$$

The expression for the other nodal coordinates are similar. There are two exceptions were the internal work term is the only nonzero term. These are when the design variable is $A^{(k)}$ or $E^(k)$.

If, $p_i = A^{(k)}$

$$\frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u} \partial \boldsymbol{p}} \left(nNoEq + k, nNoEq + k, i\right) = \frac{E^{(k)}}{L_0^{(k)}}. \tag{5.39}$$

If, $p_i = E^{(k)}$

$$\frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u} \partial \boldsymbol{p}} \left(\text{nNoEq} + k, \text{nNoEq} + k, i\right) = \frac{A^{(k)}}{L_0^{(k)}}, \tag{5.40}$$

were nNoEq are the number of nodal equilibrium equations. In both cases all other entries of $\frac{\partial^2 \boldsymbol{r}}{\partial \boldsymbol{u} \partial \boldsymbol{p}}$ equal zero.

### 5.4.4 Displacement constraint

Displacements can also be used as constraints. In this case it makes most sense to pinpoint certain important node displacements to restrict. Every node can be constraint in $x$ and $y$-direction. A single constraint would look like

$$d_i \leq d_{imax}, \tag{5.41}$$

$\boldsymbol{d}_i$ contains all $u_i$ and $v_i$, $i$ is the number of nodes, were $u_i$ and $v_i$ are the nodal displacements in $x$ and $y$ direction respectively. The derivatives for a single displacement are trivial

$$\frac{\partial d}{\partial \boldsymbol{u}} = \begin{cases} 1 & \text{if } d = u_l \\ 0 & \text{if } d \neq u_l \end{cases} \text{ for l = 1, 2, \ldots, } nNoDOF. \tag{5.42}$$

$nNoDOF$ the number of nodal degrees of freedom. The relationship between $\boldsymbol{u}$ and $\boldsymbol{p}$ is implicit. Hence, the partial derivatives of $\boldsymbol{u}$ with respect to $\boldsymbol{p}$ are

$$\frac{\partial g}{\partial \boldsymbol{p}} = \mathbf{0}. \tag{5.43}$$

## 5.5 Concluding remarks on the analytical gradients

In this chapter the general analytical expressions needed for optimization were derived. The gradients described in Equation (4.16a) and Equation (4.17) can now be implemented in analytical form for optimization problems. The limitations are the chosen four objective and constraint functions. These four functions already allow to test a wide range of optimization problems, but in future research analytical gradients of other objective and constraint function can be obtained in the same way, allowing to solve even more optimization problems.

# Chapter 6

# Method

## 6.1 Test problem formulation

One of the goals of this research is to see how well the different optimization strategies perform. This has to be tested for a range of problems varying in complexity. Therefore, the truss structure that is used is designed in such a way that the complexity of the system is easily changed. The structure is defined in 2D with dimensions: total length $L_x$ and height $L_y$. These dimensions are discretized into $ndiv_x$ and $ndiv_y$ parts respectively, allowing for simple modification of the structure's complexity. The test structure for a chosen set of input variables is presented in Figure 6.1.
In the tested problems, the design variables only consist of cross-sectional areas. This approach ensures that the problems remain similar and comparable while increasing in complexity. For example, increasing the number of elements will make the structure stiffer, assuming the initial guesses for the cross sections are the same. This change will affect the computational time needed to optimize the structure, depending on how close the initial guess is to a local minimum. To address this, random guesses are implemented. However, since the lower and upper bounds are the same for all problems, the optimization algorithm might attempt a set of design variables that deforms the structure in such a way that the residual equations cannot be solved within the allowed Newton-Raphson iterations. This issue is not unique to the nested approach. The SAND approach can encounter a similar problem when the optimization algorithm gets stuck at an incorrect solution, causing the solver to stop prematurely without reaching a local minimum. In Chapter 7, a problem with other types of design variables is optimized to demonstrate that it is possible. The complexity of the system is increased in two different ways:

1. Increasing the number of design variables, $n_p$.

2. Increasing the number of elements, $nEl$.

In the first scenario the number of elements is kept constant while the number of design

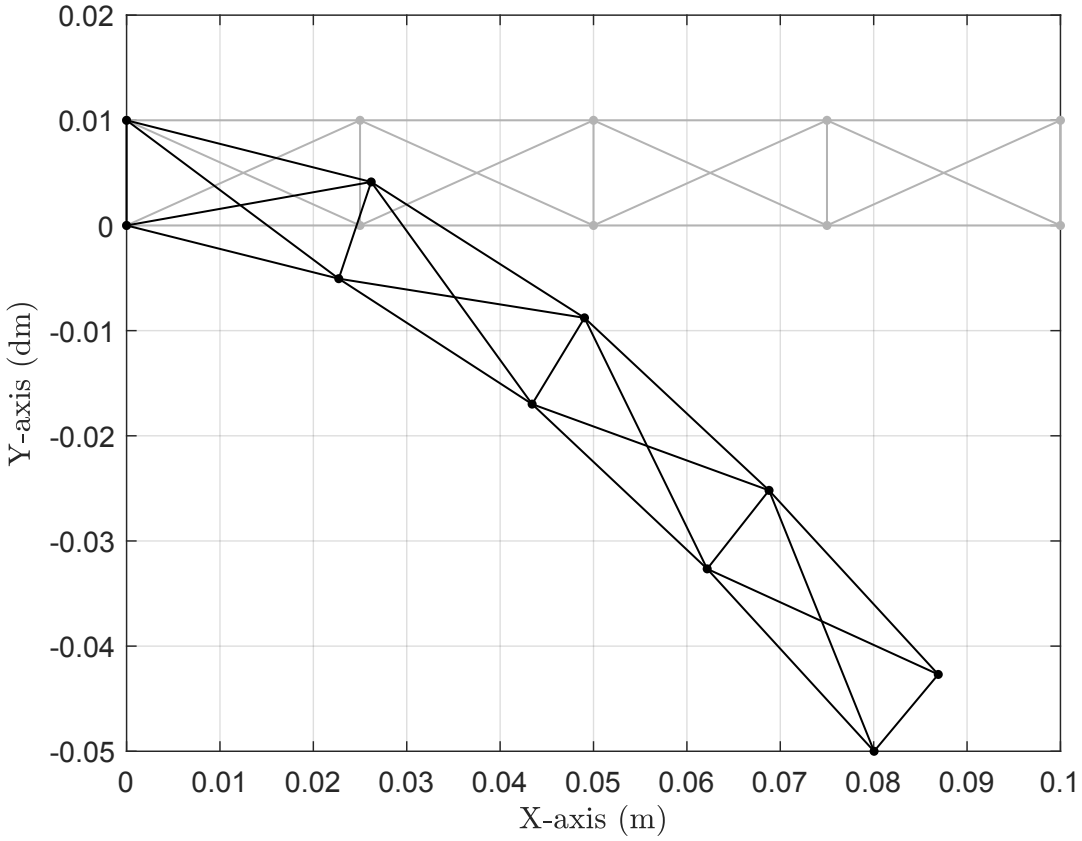



Figure 6.1: Example of the deformed truss test structure. The input variables are: $L_x = 0.1$m, $L_y = 0.001$, $ndiv_x = 4$, $numdiv_y = 1$ and a forced tip displacement of $-0.05$m.

variables is increased. The elements are grouped in sets with the same cross sectional area. Increasing the number of groups, increases the number of design variables, up to the point where every cross section is a separate design variable. In the second scenario the number of elements are increased while keeping the number of design variables constant. In this scenario the stress constraints will also increase. The elements are added by increasing $ndiv_x$ and $ndiv_y$.

## 6.2 Gradients

The accuracy of the gradients is checked by comparing the analytical gradients with numerical gradients obtained with finite differences. The computational time to compute the numerical gradients is compared with the computational time of the direct and adjoint methods. To do this, the analytical and numerical gradients of the mass, stress and displacements are computed. The compliance will be tested on accuracy but is not included in the gradient efficiency study because computing the entire compliance matrix will cost too much computer storage. In reality, only the compliance of a single or couple

of nodes would be of interest. Furthermore, a displacement of the tip of the structure in $y$ direction is prescribed. Two cases are tested: a more linear one with a tip displacement of 10% of the length of the structure, and a more nonlinear case in which the tip displacement is 50% of the total length of the structure.

## 6.3 Optimization

In optimization, comparing methods within MATLAB is more challenging due to the presence of numerous, often uncontrollable variables and settings within the optimization algorithm. The nested and SAND approach differ in the way the methods solve the problems. This causes comparison problems, considering tolerances, standard fmincon settings in MATLAB, and chosen settings for the Newton-Raphson scheme implemented in the nested approach. The choice to use either the nested or the SAND approach, with SQP vs IP, can heavily depend on the specific problem. Nevertheless general trends and behavior of the different optimization strategies can be observed.
The optimization comparison is conducted only for situation where the gradients are analytically available, excluding comparisons with finite differences. The efficiency differences between analytical and numerical gradients is already investigated with the gradient computation. Using the same two scenario's: either adding design variables or elements. The four main solution procedures compared are: Nested-SQP, Nested-IP, SAND-SQP, and SAND-IP.
In the optimization problems the mass is the objective function, the stresses are constraint functions and a constraint on the tip displacement of 40% of the length of the structure is enforced. In the optimization runs the initial guess for the design variables will be randomly selected between chosen upper and lower bounds, to test the robustness of both methods and solvers.

# Chapter 7

# Results

## 7.1 Gradients

In this section some variables have the same value for all problems. Unless mentioned otherwise these following values are used: a prescribed tip displacement of either 0.01m or 0.05m, $E = 210$GPa, $L_x = 0.1$m, and $L_y = 0.001$m.

### 7.1.1 Accuracy of gradients

The analytical gradients of the mass, stress, compliance and displacement are compared to the numerical gradients obtained with finite difference. This is done using the test structure from Chapter 6. In Figure 7.1 the accuracy of the gradients is visualized. The maximum error for each derivative is plotted. The error is the difference between the direct method and finite differences. Different values of perturbation $h$ are tried for finite differences.

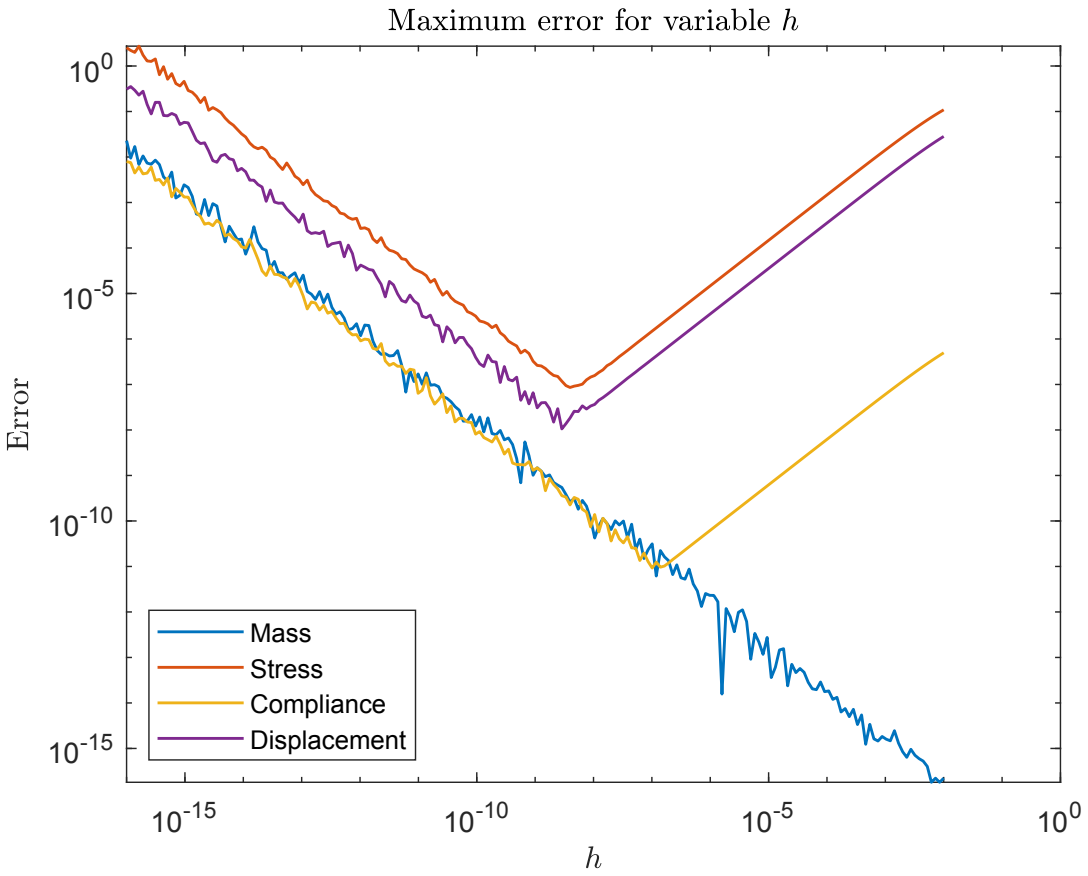



Figure 7.1: Finite difference vs direct method for different discretization values. $ndiv_x = 25$, $ndiv_y = 2$, $n_p = 2$, $nEl = 227$, and $nDOF = 616$.

From Figure 7.1, it becomes clear that the optimal value of $h$ lies around $h$ =1e-8, which is therefore the value used throughout this work.

### 7.1.2 Direct method vs finite difference

In this section, the efficiency of the direct method is compared to the conventionally used finite differences. The algorithm will compute the gradients of the mass, stress per element, and all nodal displacements with respect to the design variables, $\boldsymbol{p}$.
The computational efficiency of both methods is measured by the CPU time needed to compute the gradients. To mitigate the effects of potential noise or disturbances during the measurements, the derivatives using finite differences and the direct method are calculated 5 times in alternating order. The average time of these runs is then taken to represent the computational time needed to calculate the gradients. The two scenario's described in Chapter 6 are performed. The results of the first scenario (increasing the number of design variables) can be seen in Figure 7.2.

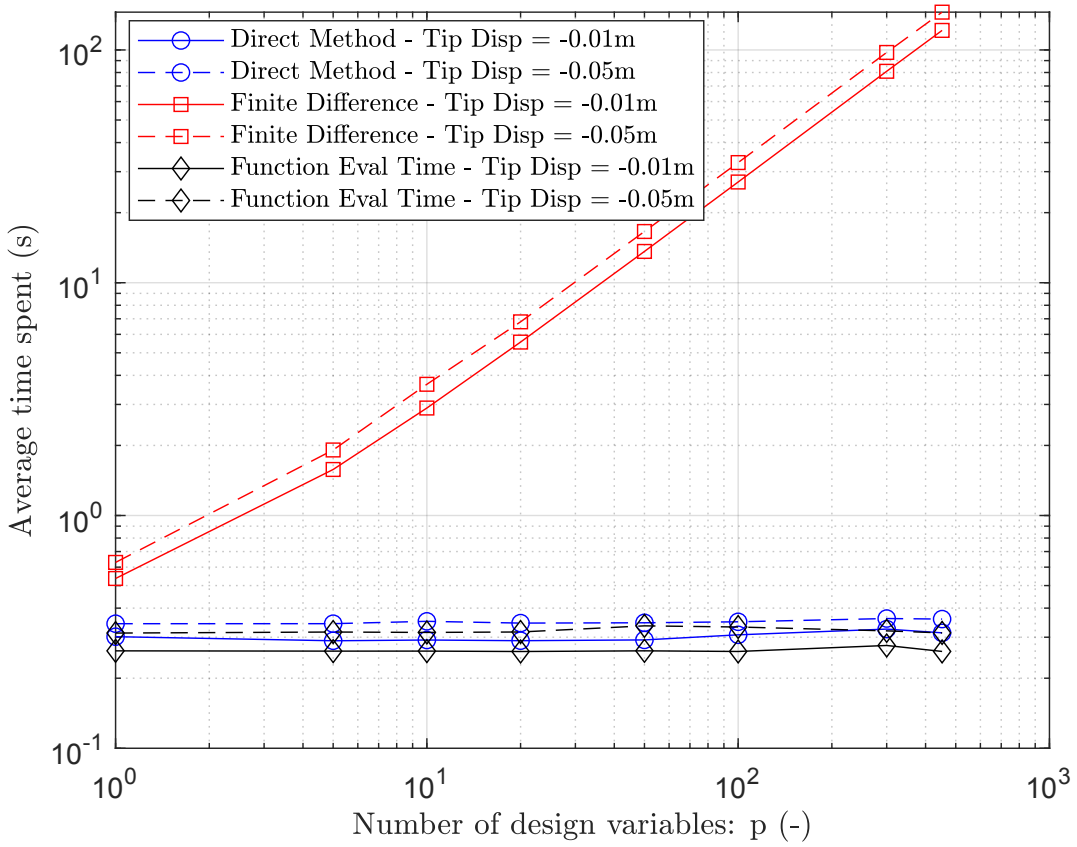



Figure 7.2: Direct method vs finite difference. Increasing the number of design variables. Mean of 5 runs. Gradient computation time and function evaluation time plotted for a tip displacement of 10% and 50% of the total length of the structure. $n_p = [1, 5, 10, 20, 50, 100, 300, 452]$, $nDOFs = 1216$ and $p_0 = 0.0025m^2$ for all design variables.

As the number of elements is constant the function evaluation time is expected to be constant. The additional time required to compute the gradients using the direct method is marginal. This allows rapid gradient computation of 452 design variables. As expected, the difference between the direct method and finite difference is approximately a factor $n_p$. For problems of this size the computational time of the matrix operations used in the direct method do not have significant influence on the computational time of the gradients. The largest portion of the computational time is used to solve the residual equations. The more linear tip displacement of -0.01m requires 4 Newton-Raphson iteration while the more nonlinear tip displacement of -0.05 requires 5 Newton-Raphson iterations to converge. This explains the extra time needed to compute the gradients for the tip displacement of -0.05m.

The results of the second scenario (increasing the number of elements) are given in Figure 7.3. Computing the gradients using the direct method requires, as in the first scenario, almost no additional time beyond the computational time of the function evaluations.

Increasing the number of elements does increase the computational time of the function evaluations. In this case the number of design variables is six. The difference between the direct method and finite differences is indeed approximately a factor six for all data points.

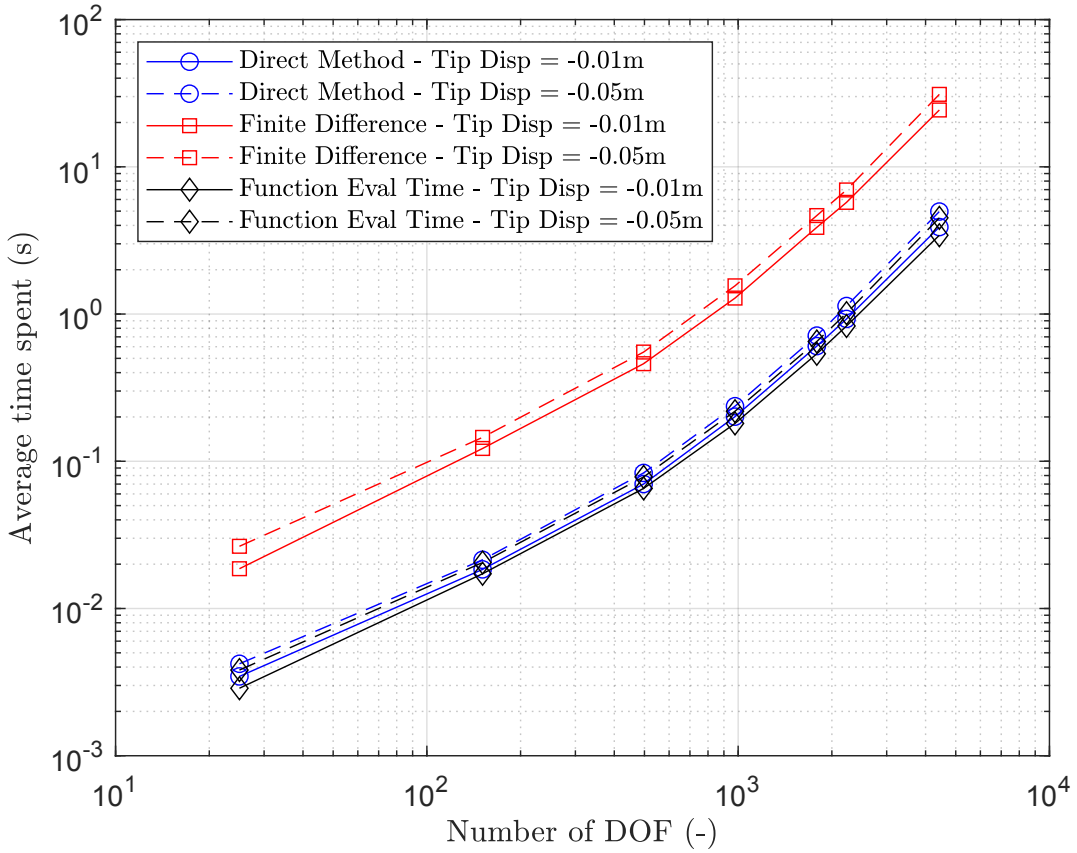



Figure 7.3: Direct method vs finite difference. Increasing the number of elements and constraints. Mean of 5 runs. Gradient computation time and function evaluation time plotted for a tip displacement of 10% and 50% of the total length of the structure. $n_p = 6$, $nDOFs = [24, 150, 496, 976, 1788, 2228, 4428]$ and $p_0 = 0.0025m^2$ for all design variables.

### 7.1.3 Direct method vs Adjoint method

In this section the direct method is compared to the adjoint method. In Figure 7.5 the results of scenario 1 are presented. The time to compute the gradients slightly increases with increasing number of design variables. However, the computational time remains very low. The direct method performs marginally better than the adjoint method, which is expected as there are more constraint functions than design variables.

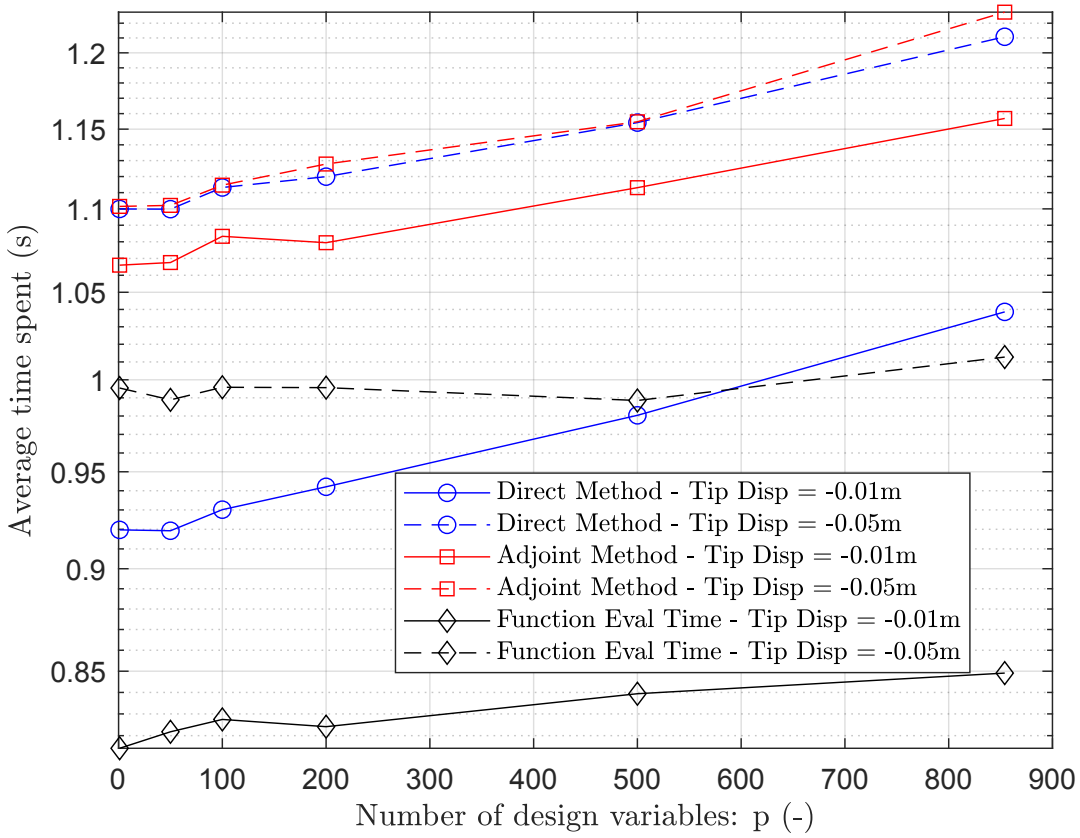



Figure 7.4: Direct method vs adjoint method. Increasing the number of design variables. Mean of 200 runs. Gradient computation time and function evaluation time plotted for a tip displacement of 10% and 50% of the total length of the structure. $n_p = [1, 50, 100, 200, 500, 854]$, $nDOFs = 2228$ and $p_0 = 0.0025m^2$ for all design variables.

The results of the second scenario are presented in Figure 7.5. Increasing the number of elements causes the computational time to increase. Both the direct and adjoint methods perform well, as the computational time is almost equal to the function evaluation time.

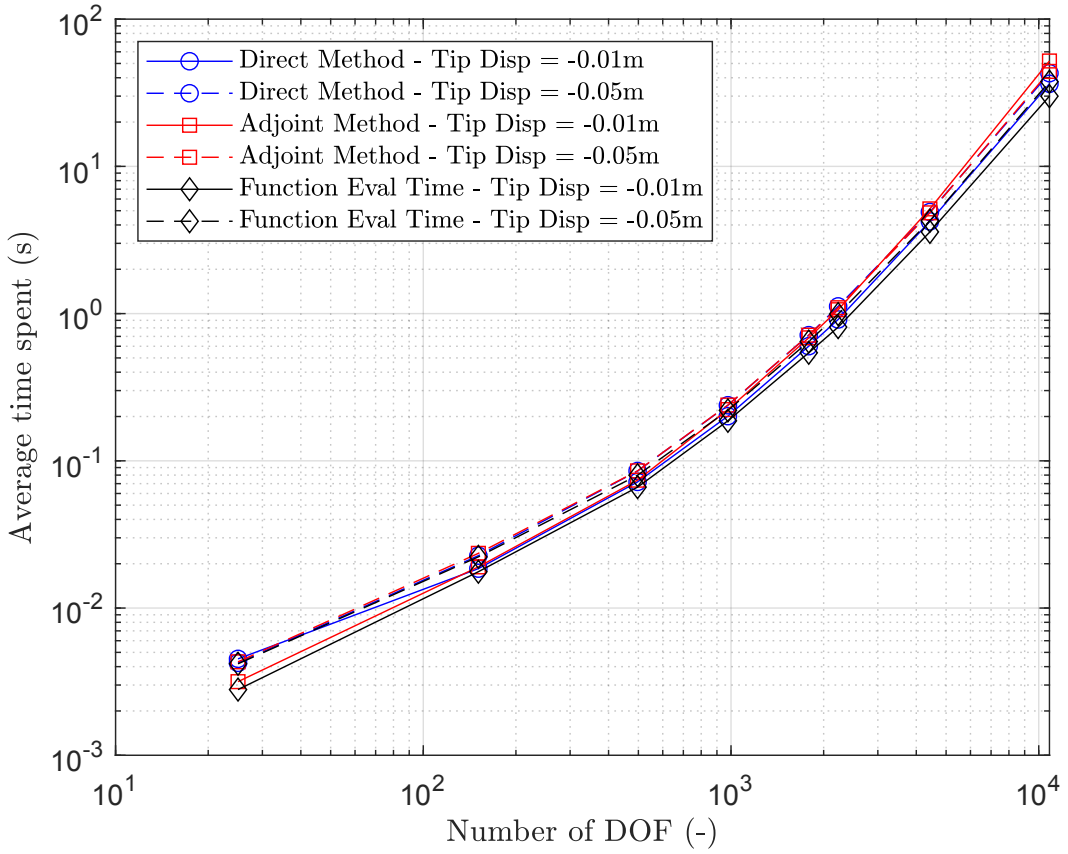



Figure 7.5: Direct method vs adjoint. Increasing the number of elements and constraints. Mean of 5 runs. Gradient computation time and function evaluation time plotted for a tip displacement of 10% and 50% of the total length of the structure. $n_p = 6$, $nDOFs = [24, 150, 496, 976, 2228, 4428, 5434]$ and $p_0 = 0.0025m^2$ for all design variables.

## 7.2 Optimization

Some variables used in the optimization problems presented here are most of the time the same. Unless mentioned otherwise the following values are used. The maximum allowed tip displacement is 0.04m. The force applied at the tip in $y$ direction equals -1e4N. Furthermore, $L_x = 0.1$m, $L_y = 0.001$m, $E = 210$GPa and $lb = 0.0001\text{m}^2$, $ub = 0.01\text{m}^2$ are the lower and upper bounds for all design variables.

### 7.2.1 Scaling

In the SAND approach the residual equations are implemented as equality constraints. These constraints should be scaled to have magnitudes comparable to the other constraint equations to increase the optimization performance. The effect of scaling the residual equations for different scaling factors is visualized in Figure 7.6. Applying no scaling to the residual equations significantly increases the computational time. Furthermore, the

amount of scaling within the range of this plot does not greatly influence the computation time. Finding the best scaling factor is extensive and beyond the scope of this work. In the following optimizations, a scaling factor equal to the inverse of the magnitude of the applied tip force is used, as this will result in the residual equations approximately having a magnitude between 0 and 1. Which is in the same order of magnitude as the other constraints.

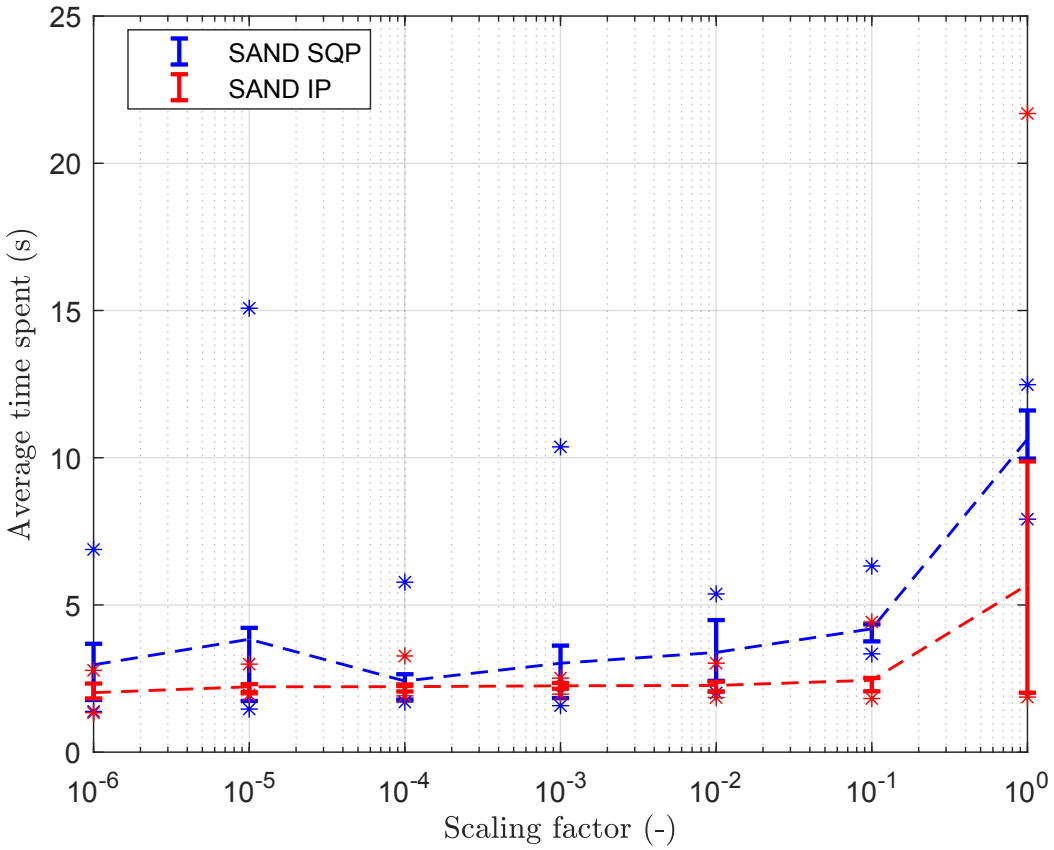
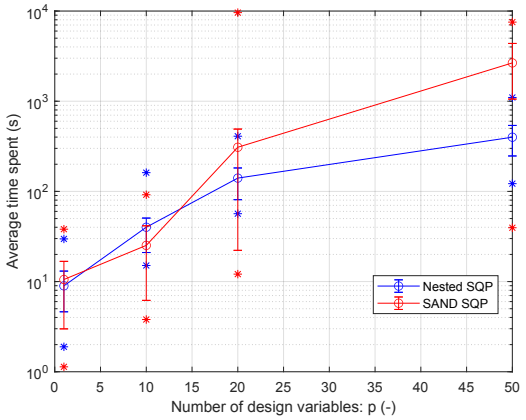



Figure 7.6: Average optimization time using SAND with SQP and IP with varying scaling factors for the residual equations. $nRuns = 20$, $nDOFS = 150$, $nEl = 51$, $n_p = 51$, $ndiv_x = 10$, $ndiv_y = 1$. The error bars plot the 25th and 75th percentiles, the minimum and maximum outliers are represented by asterisks.

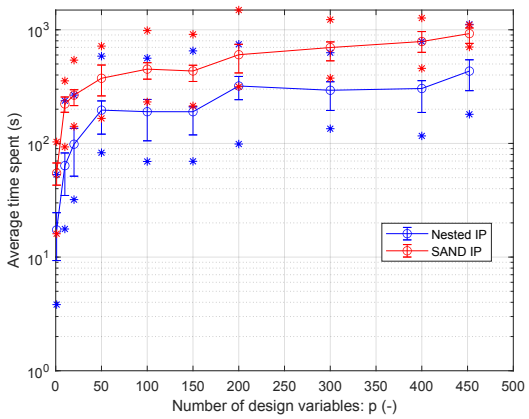
### 7.2.2 Increasing complexity

The two scenario from Chapter 6 to systematically increase the complexity are also used for the optimization results. The optimization runs using SQP are conducted with only the first four sample problems. This is because solving problems that are more complex using SQP results in computation times that are too long. The graphs in Figures 7.7 and 7.8 show that using interior-point for both the nested and SAND approach allows optimizing with hundreds of design variables and thousands of degrees of freedom within a reasonable timeframe.

The IP method performs better than SQP, possibly because IP methods prioritize maintaining feasibility, which leads to more stable progress. In contrast, SQP can suffer from constraint violations due to large steps in the solution space, requiring more frequent adjustments. Furthermore, for simple problems, SAND performs slightly better than the nested approach. However, for more complex problems, the nested approach significantly outperforms the SAND approach.

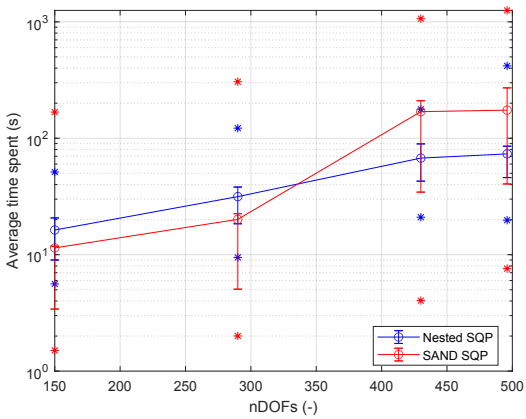



(a) SQP, $nRuns$: nested: 60, SAND: 55.

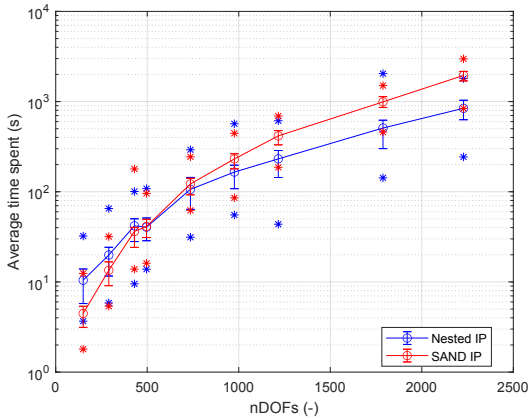



(b) IP, $nRuns$: nested: 55, SAND: 55.

Figure 7.7: Increasing the number of design variables. The error bars plot the 25th and 75th percentiles, the minimum and maximum outliers are represented by asterisks. $ndiv_x = 50$, $ndiv_y = 2$, $nDOF = 1216$, $nEl = 452$. For SQP: $n_p = [1, 10, 20, 50]$ . For IP: $n_p = [1, 10, 20, 50, 100, 150, 200, 300, 400, 452]$ .




(a) SQP, $nRuns$: nested: 90, SAND: 85.




(b) IP, $nRuns$: nested: 86, SAND: 86.

Figure 7.8: Increasing the number of elements. The error bars plot the 25th and 75th percentiles, the minimum and maximum outliers are represented by asterisks. $n_p = 50$. For SQP: $ndiv_x = [10, 20, 30, 20]$, $ndiv_y = [1, 1, 1, 2]$, $nDOF = [150, 290, 430, 496]$, $nEl = [51, 101, 151, 182]$. For IP: $ndiv_x = [10, 20, 30, 20, 30, 40, 50, 40, 50]$, $ndiv_y = [1, 1, 1, 2, 2, 2, 2, 4, 4]$, $nDOF = [150, 290, 430, 496, 736, 976, 1216, 1788, 2228]$, $nEl = [51, 101, 151, 182, 272, 362, 452, 684, 854]$.

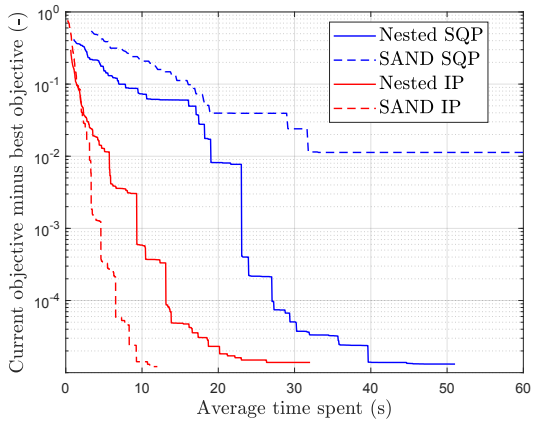
These results are problem specific, meaning it cannot be concluded that the nested approach using IP always outperforms the SAND approach for complex problems. The most important conclusion from these results is that by using analytical gradients and the right optimization strategy, problems with hundreds of design variables can be optimized in less than 15 minutes.

### 7.2.3 Convergence

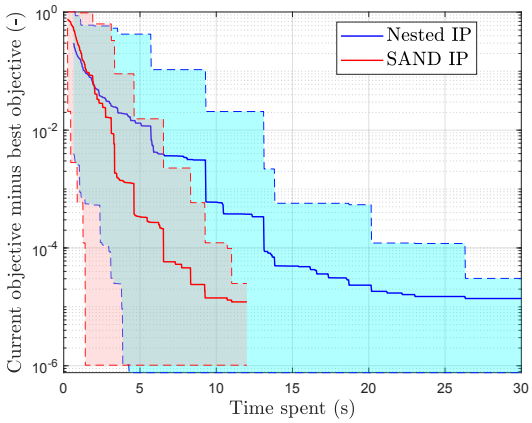
Zooming in and analyzing the results of individual problems helps to understand the performance of the different optimization strategies. In this section four problems are analyzed: a simple problem, a medium complex problem and two complex problem. The complex problems are only performed using the interior-point method, as SQP is not used for this level of complexity. The convergence of the optimization algorithm is plotted, with on the $y$ axis the objective found at that point minus the best objective found, and on the $x$ axis the time spent. The objective used in the plots is the minimum viable objective at that point in time, meaning that all the constraint equations are satisfied. Some of the optimization runs do not reach the best objective but a higher local minimum, or no minimum at all. These failed runs are deleted from the data, this is also done for the graphs in Section 7.2.2. Some remarks concerning the number of failed runs per optimization approach are presented in Section 7.2.4.

#### Simple problem

The convergence of the simple problem is presented in Figure 7.9. Interior-point outperforms SQP and the combination of SAND and IP converges the fastest. This is in line with the conclusions from Section 7.2.2 as this is a simple problem.




(a) The mean convergence of the nested and SAND approach using SQP or IP.

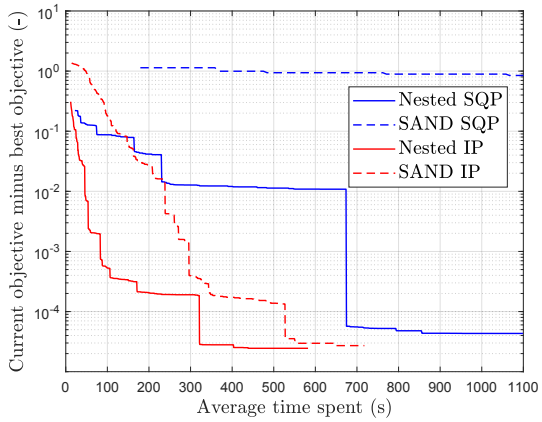



(b) The mean convergence of the nested and SAND approach using IP. The shaded area indicates the minimum and maximum convergence.

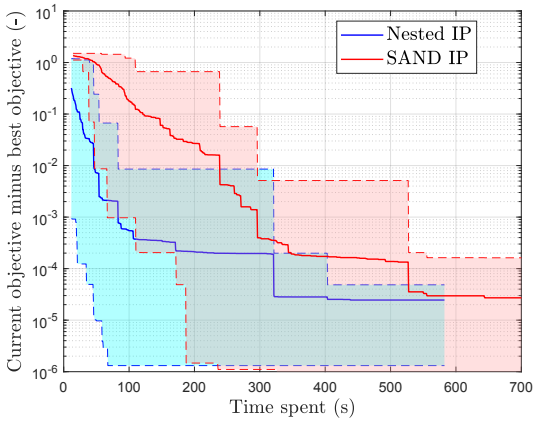
Figure 7.9: Convergence of the objective function. $ndiv_x = 10$, $ndiv_y = 1$, $nDOF = 150$, $nEl = 51$, and $n_p = 50$.

### Medium complex problem

In Figure 7.10 the results of the medium problem can be seen. Already the nested approach converges significantly faster than the SAND approach.




(a) The mean convergence of the nested and SAND approach using SQP or IP.



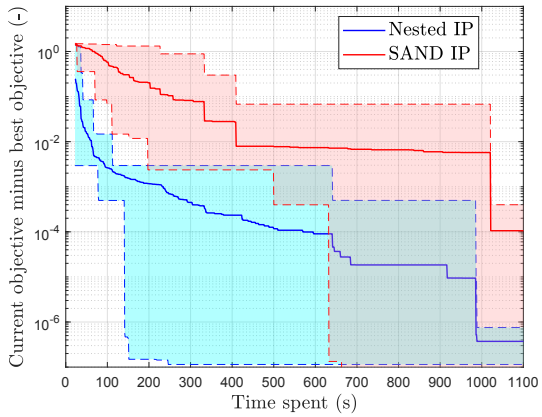

(b) The mean convergence of the nested and SAND approach using IP. The shaded area indicates the minimum and maximum convergence.

Figure 7.10: Convergence of the objective function. $ndiv_x = 50$, $ndiv_y = 2$, $nDOF = 1216$, $nEl = 854$, and $n_p = 50$.

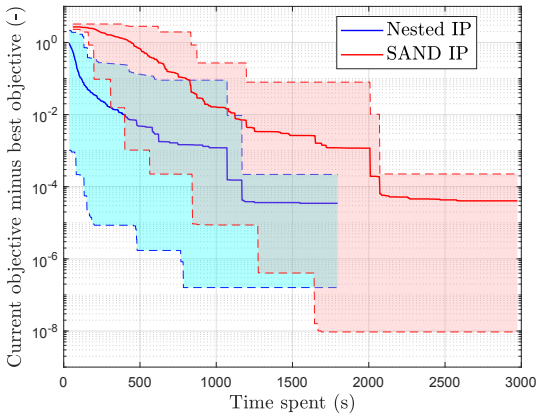
### Complex problems

In Figure 7.11 the convergence of two complex problems is visualized. Figure 7.11a shows the most complex problem obtained from scenario 1 (increasing the number of

design variables), while Figure 7.11b shows the convergence of the most complex problem obtained from scenario 2 (increasing the number of elements). The difference between the nested and SAND approach is the largest in Figure 7.11b, for that problem the nested approach converges almost three times faster than the SAND approach.




(a) The mean convergence of the nested and SAND approach using IP. The shaded area indicates the minimum and maximum convergence. $ndiv_x = 50$, $ndiv_y = 2$, $nDOF = 1216$, $nEl = 452$, and $np = 452$




(b) The mean convergence of the nested and SAND approach using IP. The shaded area indicates the minimum and maximum convergence. $ndiv_x = 50$, $ndiv_y = 4$, $nDOF = 2228$, $nEl = 854$, and $np = 50$

Figure 7.11: Convergence of two complex problems

### 7.2.4 Failed runs

The optimization runs are performed with random initial guesses. Therefore, it can happen that runs converge to a different local minimum, or not converge at all. If the objective found by a certain run is not within 1e-3 of the best objective found the run is classified as a failed run and deleted from the data. The optimization approach used should not only be time efficient but also robust, meaning the percentage of failed runs is low. The percentages of failed runs for the different optimization approaches used are presented in Table 7.1. SQP has more failed runs than IP, but the percentages of failed runs are low. The combination of SAND and IP for complex problems causes many failed runs. It becomes clear that for complex problems the SAND approach is not only less time efficient but also less robust than the nested approach.

| Category | Increasing $np$, failed runs (%) | Increasing $nEl$, failed runs (%) |
|---|---|---|
| **nested SQP** | [0, 0, 0, 13.33] | [1.11, 3.33, 7.78, 2.22] |
| **nested IP** | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] | [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] |
| **SAND SQP** | [0, 0, 0, 3.64] | [1.18, 0, 5.88, 4.71] |
| **SAND IP** | [0, 0, 0, 0, 0, 0, 0, 52.73, 43.64, 76.36] | [0, 0, 0, 0, 0, 0, 0, 0, 2.33, 4.65] |

Table 7.1: Percentage of failed runs

### 7.2.5 Constraints

The optimization approaches and solvers use different ways of searching for a minimum, which can be visualized by examining the behavior of the constraints. In Figure 7.12, the displacement constraint during the optimization process is plotted.

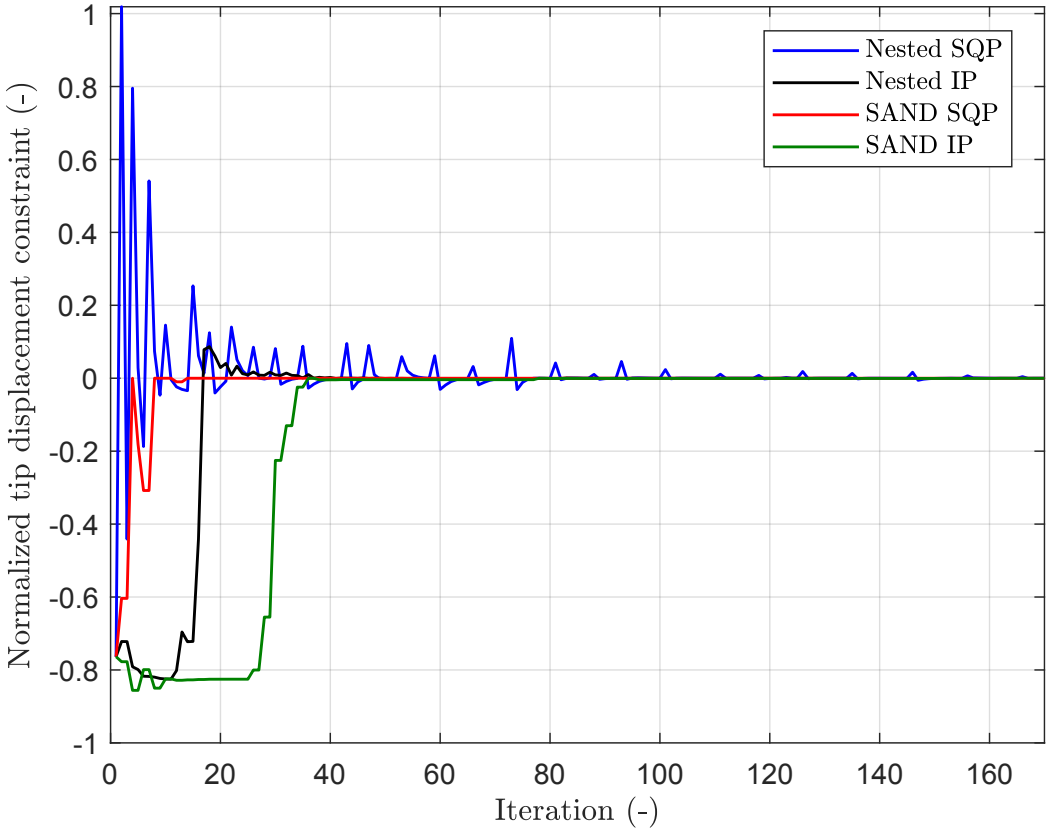


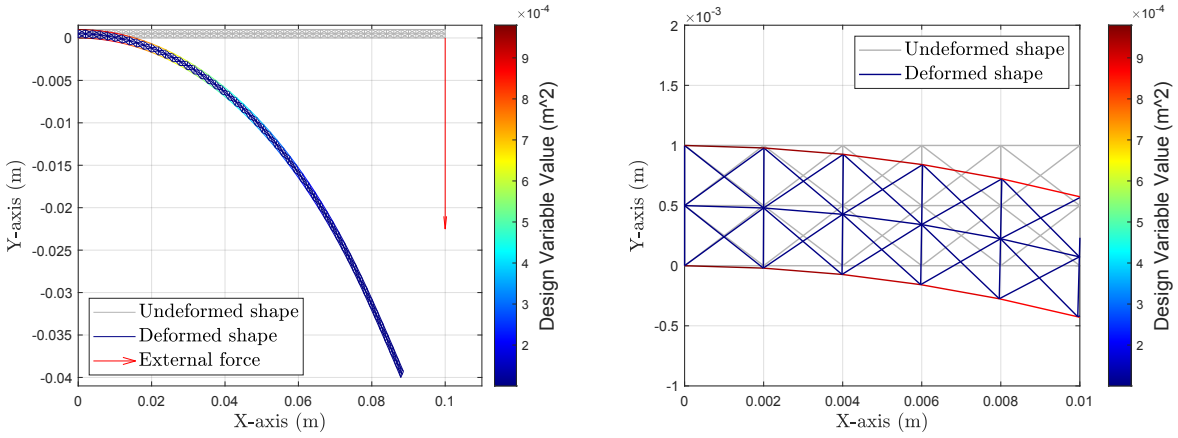

Figure 7.12: Displacement constraint of the Nested and SAND approach. $ndiv_x = 10$, $ndiv_y = 1$, $nDOF = 150$, $nEl = 51$, and $n_p = 50$.

The SAND approach does not violate the constraint. This is because in the SAND approach the residual equations are implemented as constraints, meaning the residual equations do not have to be satisfied for every iteration. This provides the SAND ap-

proach with some flexibility allowing for small changes in the state variables to stay closer to the previous found objective. For the nested approach, the state variables must be solved at every optimization iteration. This can cause larger constraint violations.
The differences between SQP and IP, as outlined in Chapter 3 are also visible in the constraint behavior. SQP allows larger constraint violations than the interior-point algorithm.

### 7.2.6 The problem solution

After analyzing the optimization results, it is interesting to look at the optimal solution found. The solution of the complex problem from Figure 7.11a can be seen in Figure 7.13. The mass is minimized, with stress constraints and a constraint on the maximum tip displacement. The optimal cross-sectional areas are plotted using a color bar. As expected, the cross-sectional areas are largest at the beginning and gradually reduce towards the end of the structure. The outside elements seem to bear most of the forces, as the other elements are close to the minimum cross-sectional area.




(a) The optimal cross sections are plotted using a color bar.

(b) Zoomed in.

Figure 7.13: Optimized truss problem. $ndiv_x = 50$, $ndiv_y = 2$, $nDOF = 1216$, $nEl = 452$, and $np = 452$.

## 7.3 Bridge

In this section, a problem with different types of design variables is optimized. Until now, only the cross-sectional areas have been used as design variables for the optimization results. In the following example, it is demonstrated that analytical gradients can also be used with other types of design variables.

### 7.3.1 Optimization problem

The optimized structure and the results can be seen in Figures 7.14 and 7.15. The optimization problem is described by
Find:

$$\boldsymbol{p} = \{A_1, A_2, ..., A_{53}, x_2, x_5, ..., x_{28}, y_3, y_5, y_7, y_9, y_{11}, y_{13}, y_{15}, y_{17}, y_{19}, y_{21}, y_{23}, y_{25}, y_{27}\}$$

Minimize:

$$M(\boldsymbol{p}) = \sum_{i=1}^{nElements} \rho_i A_i L_i^0.$$

Subject to:

$$\begin{aligned}
&0.0025 \leq (A_1, A_2, ..., A_{53}) \leq 0.02 \text{ m}^2 \\
&|u_{28}| \leq 0.5 \text{ m} \\
&-5 \leq \frac{\Delta L_i E_i}{L_i^0} \leq 5 \text{ GPa} \quad \forall i = 1, \ldots, \text{nElements}
\end{aligned}$$

Furthermore, the design variables $y_i$ are limited between 0.5m and 5m, while the design variables $x_i$ are constrained to have a maximum change from their initial positions of -0.5m and 0.5m to prevent overlapping coordinates. An external force of -1e7N is applied at node 14 in $y$ direction. The nodes are numbered from left to right with even numbers assigned to the low nodes and odd numbers assigned to the high nodes, except for node 1.

### 7.3.2 Results

The bridge consists of 53 elements, 165 degrees of freedom and is optimized for 92 design variables within 1 minute. In Figure 7.14 the initial guess of the bridge structure is plotted in gray, the undeformed optimal solution is plotted in color. The color scheme represents the optimal values of the cross-sectional areas for the elements. The cross sections of the elements in the middle are larger, indicating that these elements are subjected to the highest forces.

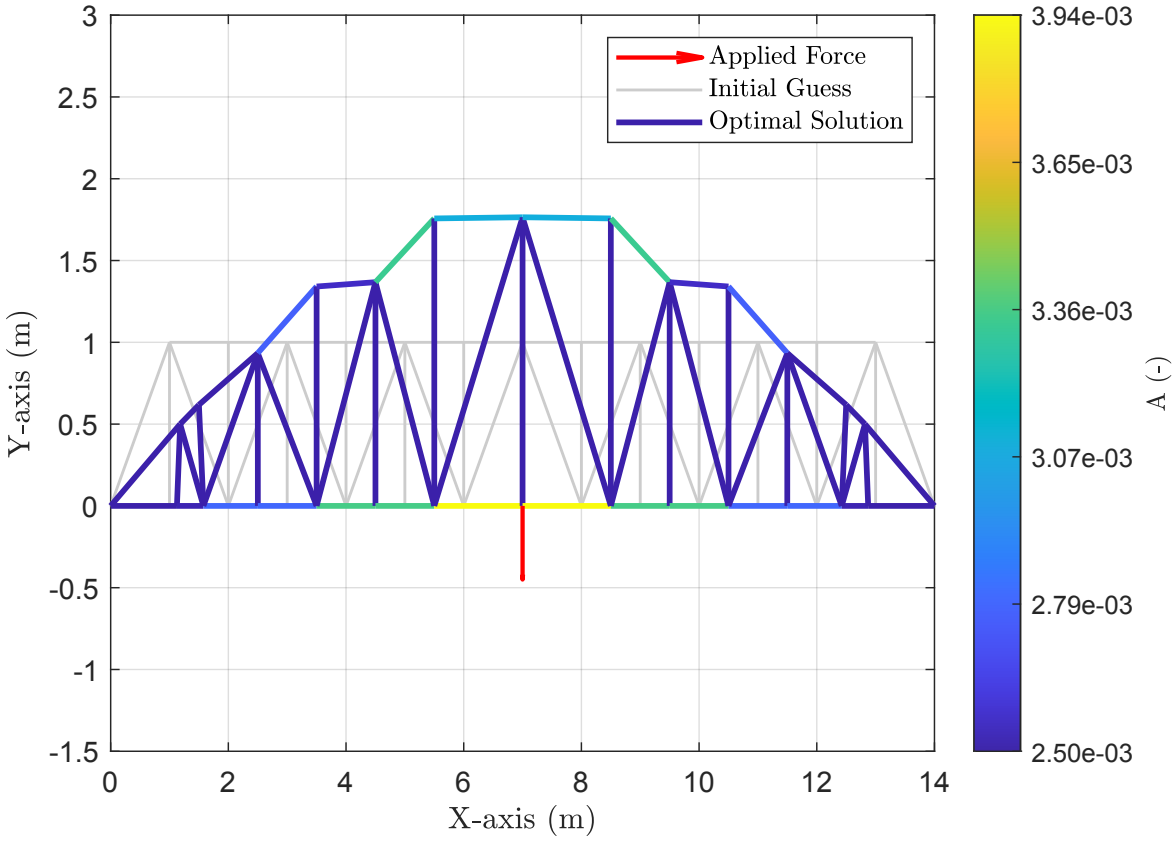



Figure 7.14: The optimal solution visualized.

In Figure 7.15 the deformed bridge is showed. The optimal undeformed structure is plotted in gray. The color scheme now indicates the stresses in the elements. These are the normalized stress constraints, meaning that the maximum value is 0 and the minimum value is -1. The stresses in the horizontal elements are the highest, these elements bear the most forces.

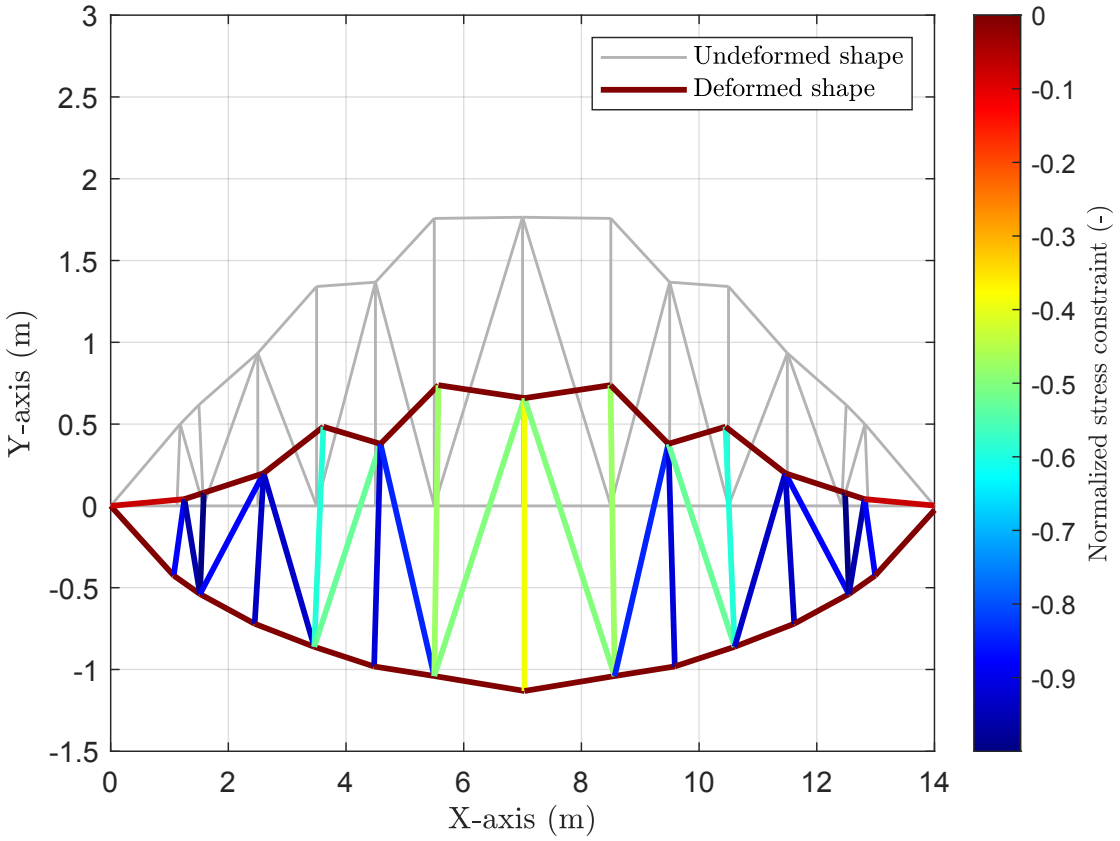



Figure 7.15: The stresses in the elements of the deformed bridge.

# Chapter 8

# Conclusion

The flexible multibody approach introduced in Chapter 2 is verified with SPACAR. The Jacobian $\boldsymbol{J} = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{u}}$, required to solve the nonlinear residual equations resulting from the flexible multibody approach, is decomposed into components based on the different types of residual equations and variables in the solution vector. The components are written as generic analytical expressions. This Jacobian is one of the four gradient terms required for optimization, as is explained in Chapter 3 and Chapter 4. Analytical expressions for the other three gradient terms are derived in Chapter 5. This chapter also details analytical expressions of the gradients with respect to the state variables and design variables of four possible objective and constraint functions: mass, stress, compliance, and displacement.

In Chapter 7, the results of the research are presented. The comparison between the computational time of numerical and analytical gradients demonstrates that when using the direct or adjoint methods, gradients of the objective and constraint functions can be computed with minimal additional computation time beyond the time required for the function evaluations. This results in a significant computational efficiency advantage of computing analytical gradients over numerical gradients. This advantage increases for an increasing number of design variables.
The reduced computational time enables optimizing problems with hundreds of design variables and thousand of degrees of freedom within a reasonable timeframe. For the types of problems tested in this research, the IP solver outperforms the SQP algorithm. IP converges faster than SQP and is able to reliably find feasible optimal solutions, even for the most complex problems. Furthermore, for complex problems, the nested approach performs better than the SAND approach. The nested approach shows faster convergence and fewer failed runs, indicating that the nested approach finds the optimal solution more reliably than the SAND approach. However, both approaches demonstrate the capability of solving complex optimization problems with hundreds of design variables.

# Chapter 9

# Recommendations

This research shows that using analytical gradients for optimizing geometrically nonlinear deforming structures can significantly reduce computational time. The nested and SAND approaches are compared, along with two solver, IP and SQP, that are available in the fmincon function in MATLAB. However, for the most efficient implementation of the optimization approaches, specialized solvers tailored to the specific approach and problem may perform better. Additionally, the scaling of the residual equations in the SAND approach specifically, and the scaling of any objective and constraint functions, has only been explored to a limited extend.
Both the nested and SAND approaches have advantages and disadvantages. The performance of an optimization approach is highly problem-specific, so the SAND approach should not be discarded for complex problems based on this work. For each type of optimization problem, it is recommended to test multiple approaches, solvers, and scaling methods.

Furthermore, it is interesting to extend the list of objective and constraint functions, to increase the range of problems that can be optimized. The eigenfrequency is commonly used in compliance mechanisms and could provide optimization of a new set of problems. To compute the eigenfrequency, stiffness and mass matrices should be established. The compliance matrix obtained in this work is not square, meaning the inverse cannot be taken to obtain the stiffness matrix. Additionally, the implementation of beam elements as an additional element type will increase the optimization possibility's. Beam elements will increase the number of degrees of freedom and reaction forces in the solution vector, influencing all gradient terms. However, by following the same systematic approach as is provided in this work for truss elements, it would be possible to include beam elements.

# Bibliography


[1] U. Kirsch, *Engineering Design Optimization*. Springer-Verlag, 1993.

[2] J. B. Jonker and J. P. Meijaards, "Spacar-computer program for dynamic analysis of flexible spatial mechanisms and manipulators," *Multibody Systems Handbook*, pp. 123–143, 1990.

[3] N. Vasios, "Nonlinear analysis of structures," *The Arc-Length method. Harvard*, 2015.

[4] R. Haftka and M. Kamat, "Simultaneous nonlinear structural analysis and design," *Computational Mechanics*, 1989.

[5] J. H. He, "A tutorial and heuristic review on lagrange multiplier for optimal problems," *Nonlinear Sci. Lett. A*, vol. 8, no. 2, pp. 121–148, 2017.

[6] S. Hartmann, "A remark on the application of the newton-raphson method in nonlinear finite element analysis," *Computational Mechanics*, vol. 36, no. 2, pp. 100–116, 2005.

[7] J. B. Jonker, R. G. K. M. Aarts, and J. P. Meijaard, *Flexible multibody dynamics for design purposes*, Lecture Notes, 2022.

[8] S. Ohkubo, Y. Watada, and F. Toshio, "Nonlinear analysis of truss by energy minimization," *Computers & Structures*, vol. 27, no. 1, pp. 129–145, 1987.

[9] J. R. R. A. Martins and A. Ning, *Engineering Design Optimization*. Cambridge University Press, 2021.

[10] O. N. Ghattas and C. E. Orozco, "A sparse approach to simultaneous analysis and design of geometrically nonlinear structures," *AIAA Journal*, 1991.

[11] J. ten Hagen, "Optimization of flexure mechanisms using gradient-based methods," M.S. thesis, University of Twente, 2023.

[12] Z. Xiao, "A comparative analysis of an interior-point method and a sequential quadratic programming method for the markowitz portfolio management problem," Honors Papers. No. 248, M.S. thesis, Honors Papers, 2016.

[13] J. Havinga, *Computational optimization, lecture 6: Gradients*, PowerPoint presentation, 2023.