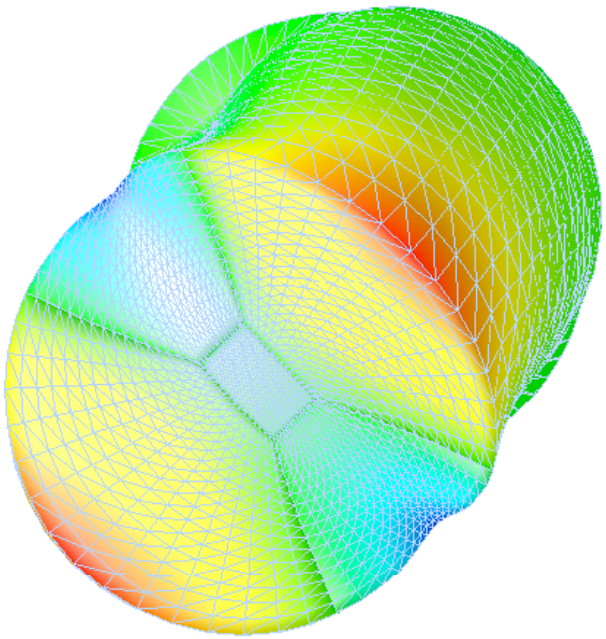
FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN-NÜRNBERG
INSTITUT FÜR INFORMATIK (MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG)

Lehrstuhl für Informatik 10 (Systemsimulation)

Calculations in Structural Mechanics in Laser Simulation

Kristina Pickl



Bachelor Thesis

Calculations in Structural Mechanics in Laser Simulation

Kristina Pickl

Bachelor Thesis

Aufgabensteller:	Prof. Dr. C. Pflaum
Betreuer:	Prof. Dr. C. Pflaum
Bearbeitungszeitraum:	15.02.2007 – 10.05.2007

Declaration:

I confirm that this Bachelor Thesis is done solely by myself and no other sources and facilities were used except those that are mentioned in this thesis. This Bachelor Thesis was never submitted in total or in part or in modified form to any other legal department or authorities or to any examination commission. All quotations taken from other sources are referenced accordingly.

Erklärung:

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, den 10.05.2007 .....

Contents

List of Figures

List of Tables

List of Algorithms

Units in Structural Mechanics

α_T	thermal coefficient of expansion $[1/K]$
λ	first Lame parameter $[N/mm^2]$
μ	second Lame parameter $[N/mm^2]$
ν	Poisson's ratio (dimensionless)
σ_{ij}	$i, j = x, y, z$ stress $[N/mm^2]$
ε_{ij}	$i, j = x, y, z$ strain (dimensionless)
E	modulus of elasticity $[N/mm^2]$
T	actual temperature $[K]$
T_0	starting temperature $[K]$
u_i	$i = x, y, z$ deformation $[mm]$

Chapter 1

Introduction

The original definition of the term "LASER", Light Amplification by Stimulated Emission of Radiation, has almost completely vanished from the common language usage. This is mainly due to the fact that lasers are used nearly everywhere in daily life. They can be found e.g. in the scanner of a supermarket check-out, in DVD players, in printers or even in modern surgery.
Optical technologies are divided into five fields: LED's (light emitting diodes), displays (e.g. plasma displays), industrial image processing (e.g. optical sensors of production robots), "productronics" (i.e. the manufacturing of microchips) and lasers. They are the basis of the photonic industry [Kro07].
The photonic industry offers a variety of job opportunities and relies on the expertise of a number of scientists, such as computer specialists, mechanical engineers, electrical engineers and physicists. According to Kröher [Kro07], around 110,000 employees are currently working in the German photonic industry. Carl Zeiss, Jenoptik and Schott are some of the most renowned enterprises. The German photonic industry possesses both the attributes and the resources it takes to become a major global player. Experts consider it a highly dynamic field offering excellent business prospects. Within the next decade they expect an annual worldwide turnover between 500 and 800 billion euro. This development could lead to the creation of hundred thousands of qualified positions in this industry. Those small and medium-sized German businesses which have invested 10-20 % of their turnover in scientific research profit the most as figures show [Kro07].
Simulation and validating the results in experiments is considered the state of the art by researchers as it allows to save both time and money. The software system LASCAD (**LA**ser **C**avity **A**nalysis and **D**esign) is a simulation tool for developing and improving solid-state lasers. It provides a thermal and structural finite element analysis and a numerical eigenmode analysis. Furthermore it offers ABCD Gaussian beam propagation code, wave optics beam propagation code and handles beam propagation outside cavity. For further information consider [Las07].
The aim of the overall project is to re-design the present LASCAD code. In the following thesis the thermal and structural finite element analysis implementation is dealt with in detail. Based on the old code the idea was to verify the equations of structural mechanics. First of all in chapter 2 the problem of the thermal effects is considered in detail to show why it is important to investigate the thermal and structural finite element analysis. Moreover the used geometries for laser crystals are shown and the structural mechanics formulation of linear elasticity is fully described with boundary conditions. After that in chapter 3 the finite element discretisation of the structural mechanics problem is derived in detail. Chapter 4 gives a short overview on the implemented algorithms with regard to the used libraries, followed by chapter 5, which presents the achieved results. Finally, chapter 6 concludes the results.

Chapter 2

Problem Formulation

In this chapter the thermal effects in solid state lasers are described briefly to show in which way the temperature-caused stresses can influence the crystal. Furthermore the used geometries are introduced and the structural mechanics calculation is explained in detail because it forms the basis of the thermal and structural finite element simulation.

2.1 Thermal Effects

In the case of solid-state lasers the heating of the laser crystal is caused by the optical pumping process (cf. [Iff90, pp. 90-110] or [Koe99, pp. 406-468]). This results in a temperature difference between the core and the surface of the laser crystal. The temperature gradient causes the thermal lensing effect which is shown exaggeratedly in figure 2.1.

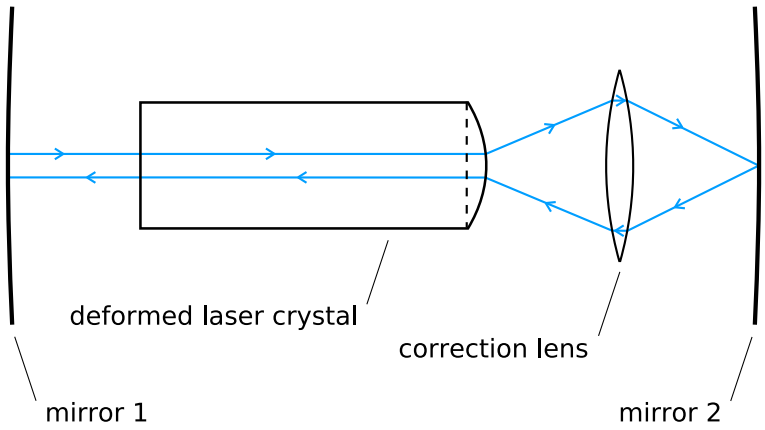


Figure 2.1: Thermal lensing effect

This effect consists of the bulging of the end faces of the geometry and therefore a change in the refractive index of the laser crystal. The arising mechanical stresses even lead to a quadratic change of the radius and the refractive index. This is called the photoelastic effect. Both effects have to be compensated by correction lenses to guarantee the correct amplification of the light beam in the resonator. At the worst the high stresses cause the crystal to break. To avoid this and to prevent a large change in the refractive index, the crystal has to be cooled

in the corresponding areas. Another disadvantage resulting from the temperature gradient is that the efficiency of the laser beam decreases with increasing temperature gradient.

2.2 Geometries

In the simulation of solid state lasers two main geometries are used: hexahedrons and cylinders. Figure 2.2 illustrates the hexahedron geometry consisting of the parameters lx, ly and lz.

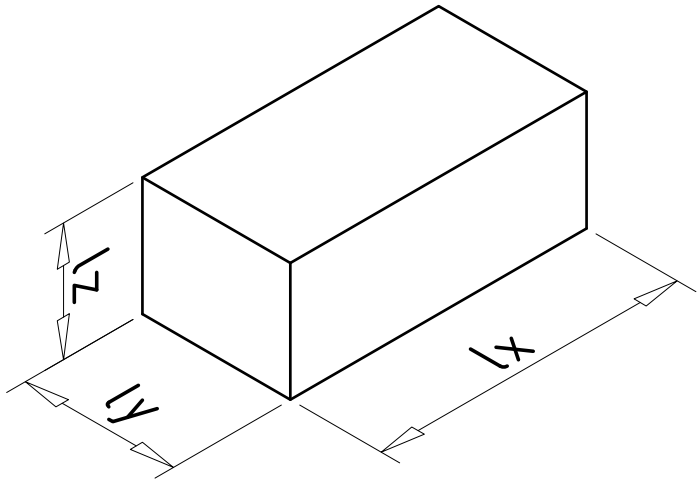


Figure 2.2: Hexahedron geometry

Figure 2.3 shows the cylinder geometry characterised by the radius R, the length l and the size of the inner block $r \times r$.

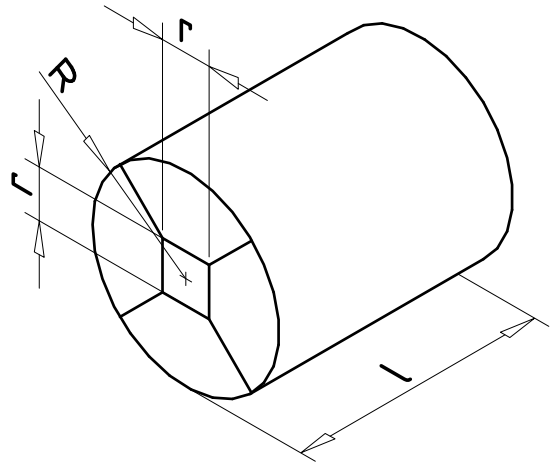


Figure 2.3: Cylinder geometry

2.3 Structural Mechanics

In general the infinitesimal deformation of a body is defined by a vector $u = (u_x, u_y, u_z)^T$ which describes how the body will be deformed in all three coordinates.
In the following the basic equations in elastostatics are considered. The descriptions in this chapter are mainly based on [GHSW04, pp. 64-108].
The displacement gradient is the first basic equation. It consists of the (infinitesimal) strain tensor and the (infinitesimal) rotation tensor. The latter can be ignored because rotations do not cause any stresses in the deformed body. Therefore the displacement gradient only consists of the strain tensor which is defined by

$$\varepsilon_{ij} = \frac{1}{2}(u_{ij} + u_{ji}) \qquad \text{for} \quad i,j = x,y,z. \tag{2.1}$$

The strain tensor is symmetric (i. e. $\varepsilon_{ij} = \varepsilon_{ji}$) and can also be written in matrix notation

$$\varepsilon = [\varepsilon_{ij}] = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{bmatrix} = \begin{bmatrix} \varepsilon_{x} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{xy} & \varepsilon_{y} & \varepsilon_{yz} \\ \varepsilon_{xz} & \varepsilon_{yz} & \varepsilon_{z} \end{bmatrix}. \tag{2.2}$$

If $i \neq j$ the components are called shear strains (in literature sometimes also credited as angular deformations $\frac{1}{2}\gamma_{ij}$). If $i = j$ they are called direct strains.
These are defined by

$$\begin{aligned} \varepsilon_{xx} = \varepsilon_x &= \frac{\partial u_x}{\partial x}, & \varepsilon_{xy} &= \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right), \\ \varepsilon_{yy} = \varepsilon_y &= \frac{\partial u_y}{\partial y}, & \varepsilon_{xz} &= \frac{1}{2}\left(\frac{\partial u_x}{\partial z} + \frac{\partial u_z}{\partial x}\right), \\ \varepsilon_{zz} = \varepsilon_z &= \frac{\partial u_z}{\partial z}, & \varepsilon_{yz} &= \frac{1}{2}\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right). \end{aligned} \tag{2.3}$$

The strains lead to stresses. These are described by the symmetric (i. e. $\sigma_{ij} = \sigma_{ji}$) Cauchy stress tensor which is defined by

$$\sigma = [\sigma_{ij}] = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix} = \begin{bmatrix} \sigma_{x} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{y} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{z} \end{bmatrix}. \tag{2.4}$$

If $i \neq j$ the components are tangential and are called shearing stresses (in literature also credited as τ_{ij}). If $i = j$ the components are normal and are called normal stresses.
The next basic equation in elastostatics is the connection between the strains and the stresses. It is defined by Hooke's Law in 3D

$$\sigma_{ij} = \lambda\, \varepsilon_{kk}\, \delta_{ij} + 2\mu\, \varepsilon_{ij} \qquad \text{for} \quad i,j = x,y,z. \tag{2.5}$$

This relation only holds for isotropic materials. λ and μ are called Lame parameters. They are defined as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}$$

and moreover

$$\varepsilon_{kk} = \varepsilon_x + \varepsilon_y + \varepsilon_z.$$

Furthermore ν is called Poisson's ratio, E modulus of elasticity and δ_{ij} Kronecker delta which is defined as

$$\delta_{ij} = \begin{cases} 1 & \text{for} \quad i = j \\ 0 & \text{for} \quad i \neq j \end{cases} .$$

The stress-strain relationship is often written in a shorter vector-matrix formulation because of its symmetry only six components are relevant:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} = \underbrace{\begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 2\mu \end{bmatrix}}_{C} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} \tag{2.6}$$

(2.7)

or short

$$\sigma = C\,\varepsilon. \tag{2.8}$$

The last basic equation is the equilibrium condition:

$$div\,(\sigma_{ij}) + f_j = 0 \qquad \text{for} \quad i,j = x,y,z. \tag{2.9}$$

Note: The divergence in structural mechanics is applied on a 3×3 matrix in the following way:

$$\begin{aligned} div \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} &= \nabla \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} := \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial x}(a_{11}) + \frac{\partial}{\partial y}(a_{21}) + \frac{\partial}{\partial z}(a_{31}) \\ \frac{\partial}{\partial x}(a_{12}) + \frac{\partial}{\partial y}(a_{22}) + \frac{\partial}{\partial z}(a_{32}) \\ \frac{\partial}{\partial x}(a_{13}) + \frac{\partial}{\partial y}(a_{23}) + \frac{\partial}{\partial z}(a_{33}) \end{pmatrix}^T \end{aligned} \tag{2.10}$$

The equilibrium condition states that the sum of the external forces must be zero. In the general case the body is loaded with the body forces f_j but in the case of a crystal in laser simulation there are no body forces. Thus the equilibrium condition (2.9) results to:

$$div\,(\sigma_{ij}) = 0 \qquad \text{for} \quad i,j = x,y,z. \tag{2.11}$$

Inserting the Lame parameters in (2.5) and solving it after ε_{ij} leads to:

$$\varepsilon_{ij} = \frac{1+\nu}{E}\,\sigma_{ij} - \frac{\nu}{E}\,\sigma_{kk}\,\delta ij, \tag{2.12}$$

where

$$\sigma_{kk} = \sigma_x + \sigma_y + \sigma_z.$$

As a result (2.6) can be written as

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} = \underbrace{\frac{1}{E} \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1+\nu \end{bmatrix}}_{C^{-1}} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz}. \end{bmatrix} \tag{2.13}$$

Additionally the temperature influence has to be included. An increase in temperature

$$\Delta T = (T - T_0)$$

with T equalling the actual temperature and T_0 equalling the starting temperature leads to linear widespread equal strains in the case of an isotropic material. They are proportional to the increase in temperature. Thus (2.12) results to

$$\varepsilon_{ij} = \frac{1+\nu}{E}\,\sigma_{ij} - \frac{\nu}{E}\,\sigma_{kk}\,\delta ij + \alpha_T\,\Delta T\,\delta_{ij}. \tag{2.14}$$

α_T is called thermal coefficient of expansion. Moreover (2.13) results to

$$\begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \varepsilon_{xy} \\ \varepsilon_{xz} \\ \varepsilon_{yz} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1+\nu \end{bmatrix} \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{xz} \\ \sigma_{yz} \end{bmatrix} + \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} (T - T_0) \tag{2.15}$$

or short

$$\varepsilon = C^{-1}\,\sigma + \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T. \tag{2.16}$$

Remark: The results presented in [Bra01] differ by a factor 2 in the calculations of the shear strains in (2.3) and consequently lead to wrong results for the deformation of the body.

2.4 Boundary Conditions

The governing equations of structural mechanics have to be completed by boundary conditions to fully specify the problem formulation. The most frequently used boundary conditions are Dirichlet and Neumann boundary conditions (cf. e.g. [Bra01, pp. 35-52]). In the case of Dirichlet boundary conditions the unknown u is given as a function

$$u = g(x, y, z) \qquad \text{on} \qquad \partial\Omega \tag{2.17}$$

or simply as a constant on the boundary $\partial\Omega$. Neumann boundary conditions state that the normal derivative of the unknown u is a function

$$\frac{\partial u}{\partial n} = g(x, y, z) \qquad \text{on} \qquad \partial\Omega \tag{2.18}$$

or a constant on the boundary $\partial\Omega$. These are automatically fulfilled in the case of finite elements.
In general there will be no unique solution if the boundary is exclusively described by pure Neumann boundary conditions. The reason for this problem is that the integration of the derivatives on the boundary results in a integration constant that cannot be defined exactly. At least one boundary has to be defined as a Dirichlet boundary and all other boundaries Neumann boundary. Then the integration constant can be determined respectively to the Dirichlet boundary. For the hexahedron geometry of the given problem this is shown in figure 2.4 where the red surface is Dirichlet boundary and the rest of the surfaces are Neumann boundary.

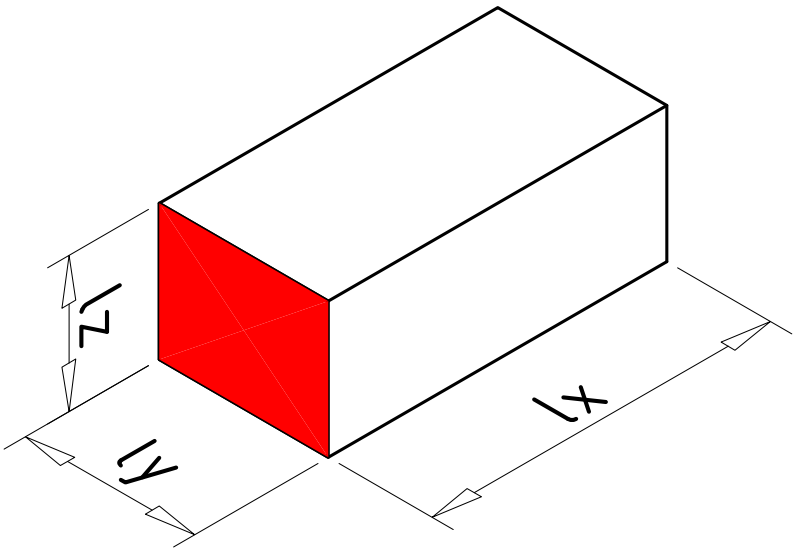


Figure 2.4: Boundary conditions for the hexahedron

This can be compared to an elastic beam that is attached to a wall. The surface touching the wall will not change its shape, however, the rest of the body will in accordance with the impact.
There is a method to handle pure Neumann boundary conditions (cf. [Bra01, p. 44-51]) but this shall for the sake of simplicity not be treated here.

Chapter 3

Finite Element Discretisation

In this chapter the finite element discretisation for the linear elasticity problem in structural mechanics will be derived in detail.
The common idea of the finite element method (FEM) is to transfer the *strong* or *classic* formulation of the problem into the *weak* or *variational* formulation. This has the advantage that the solution no longer has to fulfill the partial differential equation in every point. A detailed description of the FEM can be found in [DR06], [Bra01] or [SK04].
Note: It is also very important that in the short vector-matrix formulations the vectors with six components have to be re-transformed to their initial symmetric 3×3 matrices before applying the divergence operator as shown in (2.10).
Equation (2.15) or rather (2.16) is the strong formulation of the problem in the domain Ω. Reordering leads to

$$C\,(\varepsilon - \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T) = \sigma. \tag{3.1}$$

First the divergence has to be applied on both sides. This results in

$$div\,[C\,(\varepsilon - \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T)] = div\,(\sigma). \tag{3.2}$$

Here $div\,(\sigma) = 0$ holds because of the equilibrium condition (2.11) and therefore results:

$$div\,(C\,\varepsilon) = div\,(C \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T). \tag{3.3}$$

Now the usual steps of a finite element discretisation can be applied. (3.3) has to be multiplied by an arbitrary testfunction $v = (v_x, v_y, v_z)^T$ and integrated over Ω:

$$\int_\Omega div\,(C\,\varepsilon)\,v\,d(x,y,z) = \int_\Omega div\,(C \begin{bmatrix} \alpha_T \\ \alpha_T \\ \alpha_T \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta T)\,v\,d(x,y,z). \tag{3.4}$$

After that all known variables are inserted to calculate the local stiffness matrices. From now on (3.4) is regarded separately with left and right-hand side.
First the left-hand side is calculated:

$$\int_\Omega div\,(C\,\varepsilon)\,v\,d(x,y,z) = \int_\Omega \nabla \cdot (F\,W)\,v\,d(x,y,z) \tag{3.5}$$

with

$$W = \begin{bmatrix} (1-\nu)\varepsilon_x + \nu\varepsilon_y + \nu\varepsilon_z & (1-2\nu)\varepsilon_{xy} & (1-2\nu)\varepsilon_{xz} \\ (1-2\nu)\varepsilon_{yx} & \nu\varepsilon_x + (1-\nu)\varepsilon_y + \nu\varepsilon_z & (1-2\nu)\varepsilon_{yz} \\ (1-2\nu)\varepsilon_{zx} & (1-2\nu)\varepsilon_{zy} & \nu\varepsilon_x + \nu\varepsilon_y + (1-\nu)\varepsilon_z \end{bmatrix}$$

and

$$F = \frac{E}{(1+\nu)(1-2\nu)}.$$

The divergence is applied and the result is multiplied by the testfunction v:

$$\begin{aligned} \int_\Omega F\Big[& \frac{\partial}{\partial x}\,(1-\nu)\,\varepsilon_x\,v_x + \frac{\partial}{\partial x}\,\nu\,\varepsilon_y\,v_x + \frac{\partial}{\partial x}\,\nu\,\varepsilon_z\,v_x + \frac{\partial}{\partial y}\,(1-2\nu)\,\varepsilon_{yx}\,v_x + \\ & \frac{\partial}{\partial z}\,(1-2\nu)\,\varepsilon_{zx}\,v_x + \frac{\partial}{\partial x}\,(1-2\nu)\,\varepsilon_{xy}\,v_y + \frac{\partial}{\partial y}\,\nu\,\varepsilon_x\,v_y + \frac{\partial}{\partial y}\,(1-\nu)\,\varepsilon_y\,v_y + \\ & \frac{\partial}{\partial y}\,\nu\,\varepsilon_z\,v_y + \frac{\partial}{\partial z}\,(1-2\nu)\,\varepsilon_{zy}\,v_y + \frac{\partial}{\partial x}\,(1-2\nu)\,\varepsilon_{xz}\,v_z + \frac{\partial}{\partial y}\,(1-2\nu)\,\varepsilon_{yz}\,v_z + \\ & \frac{\partial}{\partial z}\,\nu\,\varepsilon_x\,v_z + \frac{\partial}{\partial z}\,\nu\,\varepsilon_y\,v_z + \frac{\partial}{\partial z}\,(1-\nu)\,\varepsilon_z\,v_z\Big]\,d(x,y,z). \end{aligned} \tag{3.6}$$

After using the integral theorem of Gauss the partial derivatives can be applied to the v_i (with $i = x, y, z$). The resulting boundary integrals are eliminated through given conditions, e.g. for Dirichlet boundary conditions holds $v = 0$ on $\partial\Omega$. Additionally the terms for ε_{ij} are inserted.

Therefore the left-hand side finally results to

$$
\begin{aligned}
-\int_\Omega F\Big[&(1-\nu)\,\frac{\partial u_x}{\partial x}\,\frac{\partial v_x}{\partial x}+\nu\,\frac{\partial u_y}{\partial y}\,\frac{\partial v_x}{\partial x}+\nu\,\frac{\partial u_z}{\partial z}\,\frac{\partial v_x}{\partial x}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_y}{\partial x}\,\frac{\partial v_x}{\partial y}+\\
&\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_x}{\partial y}\,\frac{\partial v_x}{\partial y}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_z}{\partial x}\,\frac{\partial v_x}{\partial z}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_x}{\partial z}\,\frac{\partial v_x}{\partial z}+\\
&\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_x}{\partial y}\,\frac{\partial v_y}{\partial x}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_y}{\partial x}\,\frac{\partial v_y}{\partial x}+\nu\,\frac{\partial u_x}{\partial x}\,\frac{\partial v_y}{\partial y}+\\
&(1-\nu)\,\frac{\partial u_y}{\partial y}\,\frac{\partial v_y}{\partial y}+\nu\,\frac{\partial u_z}{\partial z}\,\frac{\partial v_y}{\partial y}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_z}{\partial y}\,\frac{\partial v_y}{\partial z}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_y}{\partial z}\,\frac{\partial v_y}{\partial z}+\\
&\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_x}{\partial z}\,\frac{\partial v_z}{\partial x}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_z}{\partial x}\,\frac{\partial v_z}{\partial x}+\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_y}{\partial z}\,\frac{\partial v_z}{\partial y}+\\
&\frac{1}{2}\,(1-2\nu)\,\frac{\partial u_z}{\partial y}\,\frac{\partial v_z}{\partial y}+\nu\,\frac{\partial u_x}{\partial x}\,\frac{\partial v_z}{\partial z}+\nu\,\frac{\partial u_y}{\partial y}\,\frac{\partial v_z}{\partial z}+(1-\nu)\,\frac{\partial u_z}{\partial z}\,\frac{\partial v_z}{\partial z}\Big]\,d(x,y,z).
\end{aligned}
\tag{3.7}
$$

Now the right-hand side is calculated.

$$
\int_\Omega div\left(C\begin{bmatrix}\alpha_T\\ \alpha_T\\ \alpha_T\\ 0\\ 0\\ 0\end{bmatrix}\Delta T\right)v\,d(x,y,z)=\int_\Omega \nabla\cdot\left(F\,H\right)v\,d(x,y,z) \tag{3.8}
$$

with

$$
H=\begin{bmatrix}(1+\nu)\,\alpha_T\,\Delta T & 0 & 0\\ 0 & (1+\nu)\,\alpha_T\,\Delta T & 0\\ 0 & 0 & (1+\nu)\,\alpha_T\,\Delta T\end{bmatrix}.
$$

The divergence is also applied as shown in (2.10) and the result is multiplied by the testfunction v:

$$
\int_\Omega F\,\frac{\partial}{\partial x}\,(1+\nu)\,\alpha_T\,\Delta T\,v_x+\frac{\partial}{\partial y}\,(1+\nu)\,\alpha_T\,\Delta T\,v_y+\frac{\partial}{\partial z}\,(1+\nu)\,\alpha_T\,\Delta T\,v_z\,d(x,y,z). \tag{3.9}
$$

Again after using the integral theorem of Gauss the partial derivatives can be applied to the v_i (with $i=x,y,z$). The resulting boundary integrals vanish again. So the right-hand side finally results to:

$$
-\int_\Omega F\left[(1+\nu)\,\alpha_T\,\Delta T\,\frac{\partial v_x}{\partial x}+(1+\nu)\,\alpha_T\,\Delta T\,\frac{\partial v_y}{\partial y}+(1+\nu)\,\alpha_T\,\Delta T\,\frac{\partial v_z}{\partial z}\right]d(x,y,z). \tag{3.10}
$$

The minus signs in (3.7) and (3.10) cancel. So the stiffness matrices result to:

- left-hand side:

$$
\begin{bmatrix}M_{11} & M_{12} & M_{13}\\ M_{21} & M_{22} & M_{23}\\ M_{31} & M_{32} & M_{33}\end{bmatrix} \tag{3.11}
$$

  The matrix is also symmetric based on the symmetry of the matrix C in the stress-strain relationship (2.6). Therefore it is sufficient to calculate only the six submatrices M_{11}, M_{12}, M_{13}, M_{22}, M_{23} and M_{33} for the further implementation of the program.

- right-hand side:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}. \tag{3.12}$$

Chapter 4

Numerical Implementation

In this thesis several numerical solvers were examined to achieve both, best performance and a high precision of the solution.
The main task of this chapter is to show with code examples how the different methods are implemented with the used library for unstructured block grids [Uns07] in which the Colsamm library [Col07] is integrated.
With the Colsamm library the local stiffness matrix calculations can be implemented in one simple `Calculate` call as shown in algorithm 4.1 for the local stiffness matrix M_{11}.

```
double nu;
double F;
double E;

F = E / ((1.0 + nu) * (1.0 - 2.0 * nu));

Local_stiffness_matrix<double> M11(grid);

M11.Calculate(((1.0 - nu) * d_dx(v_()) * d_dx(w_()) + (d_dy(v_()) * d_dy(w_()) +
                d_dz(v_()) * d_dz(w_())) * (1.0 - 2.0 * nu) * 0.5) * F);
```

Algorithm 4.1: Local stiffness matrix calculation

Thus the Colsamm library reduces the errors in the implementation of the mathematical formulas which is the part of the program that is very prone to errors to a minimum.

4.1 Conjugate Gradient Method

For detailed analysis and description of the conjugate gradient (CG) method see [Bra01, pp. 192-199]. Basically the CG algorithm has been implemented to test whether any of the following optimised implementations will reduce the number of iterations to produce a similar solution or not.
The reduction of computational time compared to a "standard" implementation without any optimisation techniques results from the calculations by the Colsamm library. Operator overloading and fast expression templates allow the user to write the conjugate gradient algorithm almost like it is defined in Braess [Bra01, p. 194].
Algorithm 4.2 shows the computation of the residuals r_i which are equal to the first gradients g_{0_i} (with $i = x, y, z$) of the CG method in Braess. Moreover the first directions d_i (with $i = x, y, z$) (in Braess defined as d_{0_i}) are calculated and the abort criterion `delta` is calculated (in Braess defined as the numerator of α_k in the case of the first step $k = 0$). The expression `|calculation_grid` means that the operation will be evaluated for every gridpoint of the previously defined calculation grid. Furthermore r_i, u_i, f_i and d_i (with $i = x, y, z$) have to be

variables from type `Variable<double>` and the local stiffness matrices M_{lj} (with $l, j = 1, 2, 3$) have to be variables from type `Local_stiffness_matrix<double>`.

```
double delta;

r_x = M11(u_x) + M12(u_y) + M13(u_z) - f_x   |calculation_grid;
r_y = M12(u_x) + M22(u_y) + M23(u_z) - f_y   |calculation_grid;
r_z = M13(u_x) + M23(u_y) + M33(u_z) - f_z   |calculation_grid;

d_x = -1.0*r_x     |calculation_grid;
d_y = -1.0*r_y     |calculation_grid;
d_z = -1.0*r_z     |calculation_grid;

delta = product(r_x, r_x, calculation_grid) +
        product(r_y, r_y, calculation_grid) +
        product(r_z, r_z, calculation_grid);
```

Algorithm 4.2: Computation of the first gradient

4.2 Multigrid Method

For a detailed description of multigrid (MG) methods see [TOS01]. In this thesis three different MG algorithms have been implemented. All use a V-cycle as cycling strategy. They differ only in their error smoothing procedures. Gauss-Seidel iteration, Jacobi iteration with damping parameter and conjugate gradient method have been implemented as smoothers. Without damping Jacobi the iteration would not converge for 3D finite elements.
In algorithm 4.3 it is shown how easy a Gauss-Seidel iteration can be implemented with the unstructured block grids library. Here u_i and f_i (with $i = x, y, z$) have to be variables from type `Variable<double>` and the local stiffness matrices M_{lj} (with $l, j = 1, 2, 3$) have to be variables from type `Local_stiffness_matrix<double>`. The operator `Mkk.diag()` (with $k = 1, 2, 3$) represents the diagonal of the differential operator `Mkk`. Furthermore the corresponding code example for a damped Jacobi would be implemented as shown in algorithm 4.4.

```
for(int i = 0; i < iter; i++) {

  u_x = u_x - (M11(u_x) + M12(u_y) + M13(u_z) - f_x) / M11.diag()    |calculation_grid;
  u_y = u_y - (M12(u_x) + M22(u_y) + M23(u_z) - f_y) / M22.diag()    |calculation_grid;
  u_z = u_z - (M13(u_x) + M23(u_y) + M33(u_z) - f_z) / M33.diag()    |calculation_grid;

}
```

Algorithm 4.3: Gauss-Seidel iteration

```
double omega = 0.5;

for(int i = 0; i < iter; i++) {

  r_x =  (M11(u_x) + M12(u_y) + M13(u_z) - f_x) / M11.diag()   |calculation_grid;
  r_y =  (M12(u_x) + M22(u_y) + M23(u_z) - f_y) / M22.diag()   |calculation_grid;
  r_z =  (M13(u_x) + M23(u_y) + M33(u_z) - f_z) / M33.diag()   |calculation_grid;

  u_x = u_x - r_x * omega     |calculation_grid;
  u_y = u_y - r_y * omega     |calculation_grid;
  u_z = u_z - r_z * omega     |calculation_grid;

}
```

Algorithm 4.4: Damped Jacobi iteration

The restriction and the prolongation parts of the MG code are very prone to errors. With the MG operators of the unstructured block grids library (for further implementation details

consider the handbook of [Uns07]) the code for a restriction can be reduced to algorithm 4.5 and the code for a prolongation can be reduced to algorithm 4.6. In both algorithms the right hand side f_i, the unknowns u_i and the residual r_i (with $i = x, y, z$) have to be variables of type `MG_array<Variable<double>>`. Additionally the MG operator `MGop` has to be initialised as shown in algorithm 4.7 where `n` is the depth of the grid.

```
int level;

f_x[level-1] = MGop[level-1].Restrict(r_x[level]);
f_y[level-1] = MGop[level-1].Restrict(r_y[level]);
f_z[level-1] = MGop[level-1].Restrict(r_z[level]);

level = level-1;
```

Algorithm 4.5: Restriction in a multigrid algorithm

```
int level;

level = level+1;

r_x[level] = MGop[level-1].Prolongate(u_x[level-1]);
r_y[level] = MGop[level-1].Prolongate(u_y[level-1]);
r_z[level] = MGop[level-1].Prolongate(u_z[level-1]);
```

Algorithm 4.6: Prolongation in a multigrid algorithm

```
int n;

MG_array<Multigrid_operator> MGop(n-1);

for(int i = 0; i < n-1; i++){

      MGop(i) = new Multigrid_operator(blockgrids[i], blockgrids[i+1]);

}
```

Algorithm 4.7: Initialisation of a multigrid operator

4.3 Preconditioned Conjugate Gradient Method

For detailed information how the preconditioned conjugate gradient (PreCG) method can be implemented and an analysis which preconditioners are suited for the CG method refer to [Bra01, pp. 201-211].
In this thesis the PreCG method has been implemented with the MG method as a preconditioner. The MG methods again vary only in their error smoothing procedures. This time only Gauss-Seidel and damped Jacobi iteration were used as smoothers. Trottenberg et al. discuss such an implementation [TOS01, pp.278-287] in detail. They came to the conclusion that for unstructured grids as implemented in this thesis MG as a preconditioner is well suited.

4.4 Visual Validation

Besides the numerical analysis the results of the algorithms have been validated visually by testing them with different temperature functions. The results have been visualised by OpenDX. The implemented print-function can be seen in algorithm 4.8. Here the file `u_x.dx` contains the deformation of the body in x-direction with respect to the deformations in all three coordinates `u_x`, `u_y` and `u_z`.

```
ofstream FILE;

FILE.open("u_x.dx", std::ios :: out);
u_x.Print_Dx(u_x, u_y, u_z, &FILE);
FILE.close();
```

Algorithm 4.8: Visualisation by OpenDX

Chapter 5

Numerical Results

5.1 Formulas

5.1.1 Algebraic Error

The algebraic error is defined by

$$\|u - \widehat{u}\|_2. \tag{5.1}$$

u is the exact solution and $\widehat{u}$ is the approximated solution of the algorithm. In the case of the structural mechanics equations the exact solutions are not as easy to calculate as e.g. for a Poisson problem. Therefore the implemented algorithms approximate the exact solution by calculating with higher precision than necessary. Thus formula (5.1) has to be written as

$$\|u_e - \widehat{u}\|_2, \tag{5.2}$$

where u_e is the approximated exact solution (cf. [Pfl06, p. 41-42]).

5.1.2 Convergence Rate

In general the convergence rate is estimated by

$$\frac{\|u_{i+2} - u_{i+1}\|_2}{\|u_{i+1} - x_i\|_2} \leq q \quad \text{with } 0 \leq q \leq 1. \tag{5.3}$$

The convergence rate can also be approximated by the behaviour of the residual

$$r_i = A\, u_i - b. \tag{5.4}$$

The approximated convergence rate $\widehat{q}$ results to

$$\frac{\|r_{i+1}\|_2}{\|r_i\|_2} \leq \widehat{q} \quad \text{with } 0 \leq \widehat{q} \leq 1 \tag{5.5}$$

or rather

$$\left(\frac{\|r_{i+s+1}\|_2}{\|r_i\|_2}\right)^{\frac{1}{s}} \leq \widehat{q} \quad \text{with } 0 \leq \widehat{q} \leq 1 \tag{5.6}$$

for large i and $s \approx 1 - 20$ (cf. [Pfl06, p. 45].

5.2 Test Conditions

Test Case 1: Cylinder with constant temperature function

Geometry: $R = 4.0\,mm$, $r = 1.0\,mm$, $l = 9.0\,mm$,

$\Delta T = 1\,500\,K$,

$E = 300\,000\ \ N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6}\ \ 1/K$,

$\nu = 0.25$.

Test Case 2: Cylinder with non-constant temperature function

Geometry: $R = 4.0\,mm$, $r = 1.0\,mm$, $l = 9.0\,mm$,

$\Delta T = x \cdot y \cdot 1\,500\,K$,

$E = 300\,000\ \ N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6}\ \ 1/K$,

$\nu = 0.25$.

Test Case 3: Hexahedron with constant temperature function

Geometry: $lx = 9.0\,mm$, $ly = 4.0\,mm$, $lz = 4.0\,mm$,

$\Delta T = 1\,500\,K$,

$E = 300\,000\ \ N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6}\ \ 1/K$,

$\nu = 0.25$.

Test Case 4: Hexahedron with non-constant temperature function

Geometry: $lx = 9.0\,mm$, $ly = 4.0\,mm$, $lz = 4.0\,mm$,

$\Delta T = (2.0 - y) \cdot (2.0 - z) \cdot 1\,500\,K$,

$E = 300\,000\ \ N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6}\ \ 1/K$,

$\nu = 0.25$.

Test Case 5: Cylinder with small inner block $r \times r$ and constant temperature function

Geometry: $R = 4.0\,mm$, $r = 0.1\,mm$, $l = 9.0\,mm$,

$\Delta T = 1\,500\,K$,

$E = 300\,000\ \ N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6}\ \ 1/K$,

$\nu = 0.25$.

Test Case 6: Cylinder with small inner block $r \times r$ and non-constant temperature function

Geometry: $R = 4.0\,mm$, $r = 0.1\,mm$, $l = 9.0\,mm$,

$\Delta T = x \cdot y \cdot 1\,500\,K$,

$E = 300\,000 \;\; N/mm^2$,

$\alpha_T = 6.9 \cdot 10^{-6} \;\; 1/K$,

$\nu = 0.25$.

5.3 Conjugate Gradient

number of gridpoints in each direction	L2-norm of the residual			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	210 ($1.21 \cdot 10^{-6}$)	175 ($1.26 \cdot 10^{-6}$)	102 ($2.45 \cdot 10^{-6}$)	92 ($1.90 \cdot 10^{-6}$)
20	436 ($4.60 \cdot 10^{-7}$)	365 ($4.74 \cdot 10^{-7}$)	186 ($9.72 \cdot 10^{-7}$)	173 ($8.88 \cdot 10^{-7}$)
30	670 ($2.68 \cdot 10^{-7}$)	543 ($2.56 \cdot 10^{-7}$)	272 ($5.08 \cdot 10^{-7}$)	245 ($5.87 \cdot 10^{-7}$)

Table 5.1: Number of iterations for the conjugate gradient method with corresponding L2-norm of the residual in brackets

number of gridpoints in each direction	algebraic error			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	$3.15 \cdot 10^{-11}$	$2.11 \cdot 10^{-11}$	$3.59 \cdot 10^{-11}$	$2.74 \cdot 10^{-11}$
20	$3.48 \cdot 10^{-11}$	$2.55 \cdot 10^{-11}$	$6.64 \cdot 10^{-11}$	$3.06 \cdot 10^{-11}$
30	$5.23 \cdot 10^{-11}$	$3.89 \cdot 10^{-11}$	$6.02 \cdot 10^{-11}$	$7.13 \cdot 10^{-11}$

Table 5.2: Algebraic error for the conjugate gradient method with u_e after 1000 iterations

number of gridpoints in each direction	convergence rate			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	0.68	0.74	0.96	0.96
20	0.79	0.87	0.92	0.87
30	1.05 (0.95 after 20th)	0.87	0.90	0.94

Table 5.3: Convergence rate of the conjugate gradient method measured after 40 iterations

5.4 Preconditioned Conjugate Gradient

5.4.1 Preconditioner: Multigrid with Gauss-Seidel Smoother

The iterations for this algorithm are fixed to 250 iterations because otherwise the algorithm does not converge fast enough.

number of gridpoints in each direction	algebraic error			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	$1.47 \cdot 10^{-6}$	$1.14 \cdot 10^{-6}$	$3.79 \cdot 10^{-6}$	$2.55 \cdot 10^{-6}$
20	$1.99 \cdot 10^{-6}$	$5.88 \cdot 10^{-6}$	$1.63 \cdot 10^{-5}$	$4.56 \cdot 10^{-6}$
30	$1.44 \cdot 10^{-6}$	$9.27 \cdot 10^{-7}$	$6.26 \cdot 10^{-6}$	$1.86 \cdot 10^{-6}$

Table 5.4: Algebraic error for the conjugate gradient method preconditioned by multigrid with Gauss-Seidel smoother with u_e after 600 iterations

number of gridpoints in each direction	L2-norm of the residual			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	0.059	0.059	0.060	0.053
20	0.021	0.009	0.055	0.024
30	0.008	0.005	0.010	0.005

Table 5.5: L2-norm of the residual for the conjugate gradient method preconditioned by multigrid with Gauss-Seidel smoother after 250 iterations

number of gridpoints in each direction	convergence rate			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
10	0.98	0.97	0.94	0.95
20	0.99	0.97	0.98	0.99
30	0.98	0.97	0.98	0.99

Table 5.6: Convergence rate of the conjugate gradient method preconditioned by multigrid with Gauss-Seidel smoother measured after 40 iterations

5.4.2 Preconditioner: Multigrid with Damped Jacobi Smoother

Examination of the Damping Parameter

damping parameter ω	iterations			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	99 $(1.89 \cdot 10^{-3})$	138 $(1.89 \cdot 10^{-3})$	no convergence	no convergence
0.2	54 $(1.89 \cdot 10^{-3})$	39 $(1.27 \cdot 10^{-3})$	no convergence	no convergence
0.3	36 $(1.06 \cdot 10^{-3})$	30 $(1.20 \cdot 10^{-3})$	20 $(1.48 \cdot 10^{-3})$	19 $(1.67 \cdot 10^{-3})$
0.4	32 $(9.25 \cdot 10^{-4})$	27 $(7.66 \cdot 10^{-4})$	17 $(1.34 \cdot 10^{-3})$	16 $(1.56 \cdot 10^{-3})$
0.5	no convergence	no convergence	16 $(1.82 \cdot 10^{-3})$	16 $(1.26 \cdot 10^{-3})$
0.6	no convergence	no convergence	14 $(1.57 \cdot 10^{-3})$	22 $(2.45 \cdot 10^{-3})$

Table 5.7: Iterations of the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother for various damping parameters for depth of grid 3 with corresponding L2-norm of the residual in brackets

damping parameter ω	algebraic error			
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	$4.17 \cdot 10^{-8}$	$1.31 \cdot 10^{-8}$	no convergence	no convergence
0.2	$6.17 \cdot 10^{-8}$	$2.59 \cdot 10^{-8}$	no convergence	no convergence
0.3	$2.53 \cdot 10^{-7}$	$2.59 \cdot 10^{-8}$	$3.44 \cdot 10^{-8}$	$2.49 \cdot 10^{-8}$
0.4	$2.33 \cdot 10^{-7}$	$2.05 \cdot 10^{-8}$	$5.34 \cdot 10^{-8}$	$2.73 \cdot 10^{-8}$
0.5	no convergence	no convergence	$4.46 \cdot 10^{-8}$	$1.64 \cdot 10^{-8}$
0.6	no convergence	no convergence	$5.81 \cdot 10^{-8}$	$2.21 \cdot 10^{-8}$

Table 5.8: Algebraic error of the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother for various damping parameters for depth of grid 3 with u_e after 500 iterations

damping parameter ω	convergence rate			
	measured after 20 iterations		measured after 5 iterations	
	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	0.77	0.76	no convergence	no convergence
0.2	0.69	0.62	no convergence	no convergence
0.3	0.84	0.64	0.36	0.33
0.4	0.58	0.52	0.33	0.28
0.5	no convergence	no convergence	0.37	0.41
0.6	no convergence	no convergence	0.31	0.36

Table 5.9: Convergence rate of the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother for various damping parameters for depth of grid 3

Tests with Optimal Damping Parameter

depth of grid	iterations			
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.4$
3	32 ($9.25 \cdot 10^{-4}$)	27 ($7.66 \cdot 10^{-4}$)	14 ($1.57 \cdot 10^{-3}$)	16 ($1.56 \cdot 10^{-3}$)
4	37 ($2.77 \cdot 10^{-4}$)	37 ($2.41 \cdot 10^{-4}$)	16 ($4.91 \cdot 10^{-4}$)	21 ($4.73 \cdot 10^{-4}$)
5	47 ($4.72 \cdot 10^{-5}$)	46 ($4.79 \cdot 10^{-5}$)	16 ($7.16 \cdot 10^{-5}$)	25 ($1.81 \cdot 10^{-5}$)

Table 5.10: Number of iterations for the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother with corresponding L2-norm of the residual in brackets

depth of grid	algebraic error			
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.4$
3	$2.33 \cdot 10^{-7}$	$2.05 \cdot 10^{-8}$	$5.81 \cdot 10^{-8}$	$2.73 \cdot 10^{-8}$
4	$1.97 \cdot 10^{-8}$	$1.21 \cdot 10^{-8}$	$6.59 \cdot 10^{-8}$	$3.28 \cdot 10^{-8}$
5	$2.76 \cdot 10^{-8}$	$7.79 \cdot 10^{-9}$	$5.00 \cdot 10^{-8}$	$1.83 \cdot 10^{-8}$

Table 5.11: Algebraic error for the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother with u_e after 500 iterations

depth of grid	convergence rate			
	measured after 20 iterations		measured after 5 iterations	
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.4$
3	0.58	0.52	0.31	0.28
4	0.62	0.62	0.28	0.40
5	0.79	0.68	0.41	0.38

Table 5.12: Convergence rate of the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother

5.5 Multigrid with Gauss-Seidel Smoother

The iterations for this algorithm are fixed to 250 iterations to compare the results with the other multigrid algorithms.

	algebraic error			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	0.022	0.076	$2.66 \cdot 10^{-6}$	$4.20 \cdot 10^{-7}$
4	0.016	0.088	$2.78 \cdot 10^{-6}$	$2.71 \cdot 10^{-7}$
5	0.015	0.066	$2.73 \cdot 10^{-6}$	$2.06 \cdot 10^{-7}$

Table 5.13: Algebraic error for the multigrid method with Gauss-Seidel smoother with u_e after 600 iterations

	L2-norm of the residual			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	5.33	18.59	$4.42 \cdot 10^{-4}$	$6.94 \cdot 10^{-5}$
4	0.53	2.88	$6.25 \cdot 10^{-5}$	$6.08 \cdot 10^{-6}$
5	0.07	0.30	$7.97 \cdot 10^{-6}$	$6.02 \cdot 10^{-7}$

Table 5.14: L2-Norm of the residual for the multigrid method with Gauss-Seidel smoother after 250 iterations

	convergence rate			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	0.97	0.91	0.91	0.91
4	0.97	0.90	0.92	0.92
5	0.97	0.91	0.89	0.91

Table 5.15: Convergence rate of the multigrid method with Gauss-Seidel smoother measured after 30 iterations

5.6 Multigrid with Damped Jacobi Smoother

Examination of the Damping Parameter

The iterations for this algorithm are fixed to 250 iterations to compare the results with the other multigrid algorithms.

damping	algebraic error			
parameter ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	$6.05 \cdot 10^{-3}$	$5.58 \cdot 10^{-6}$	no convergence	no convergence
0.2	$1.22 \cdot 10^{-5}$	$5.14 \cdot 10^{-9}$	no convergence	no convergence
0.3	$3.21 \cdot 10^{-6}$	$6.82 \cdot 10^{-12}$	$1.88 \cdot 10^{-12}$	$1.71 \cdot 10^{-20}$
0.4	$9.58 \cdot 10^{-7}$	$9.01 \cdot 10^{-15}$	$4.36 \cdot 10^{-16}$	$3.46 \cdot 10^{-22}$
0.5	no convergence	no convergence	$6.70 \cdot 10^{-17}$	$5.99 \cdot 10^{-23}$
0.6	no convergence	no convergence	$5.07 \cdot 10^{-18}$	$3.10 \cdot 10^{-21}$

Table 5.16: Algebraic error for the multigrid method with damped Jacobi smoother for various damping parameters for depth of grid 3 with u_e after 600 iterations

damping	L2-norm of the residual			
parameter ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	0.953	0.101	no convergence	no convergence
0.2	0.012	$1.40 \cdot 10^{-4}$	no convergence	no convergence
0.3	$2.54 \cdot 10^{-3}$	$1.92 \cdot 10^{-7}$	$3.88 \cdot 10^{-9}$	$7.94 \cdot 10^{-13}$
0.4	$7.59 \cdot 10^{-4}$	$2.56 \cdot 10^{-10}$	$1.66 \cdot 10^{-12}$	$5.52 \cdot 10^{-13}$
0.5	no convergence	no convergence	$1.27 \cdot 10^{-12}$	$3.99 \cdot 10^{-13}$
0.6	no convergence	no convergence	$1.01 \cdot 10^{-12}$	$2.88 \cdot 10^{-13}$

Table 5.17: L2-norm of the residual of the multigrid method with damped Jacobi smoother for various damping parameters for depth of grid 3 after 250 iterations

damping	convergence rate			
parameter ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
0.1	0.94	0.93	no convergence	no convergence
0.2	0.95	0.91	no convergence	no convergence
0.3	0.94	0.89	0.90	0.85
0.4	0.92	0.87	0.88	0.81
0.5	no convergence	no convergence	0.85	0.76
0.6	no convergence	no convergence	0.82	0.75

Table 5.18: Convergence rate of the multigrid method with damped Jacobi smoother for various damping parameters for depth of grid 3 measured after 20 iterations

Tests with Optimal Damping Parameter

depth of grid	algebraic error			
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.6$
3	$9.58 \cdot 10^{-7}$	$2.56 \cdot 10^{-10}$	$5.07 \cdot 10^{-18}$	$3.10 \cdot 10^{-21}$
4	$4.45 \cdot 10^{-7}$	$2.64 \cdot 10^{-10}$	$5.19 \cdot 10^{-13}$	$4.18 \cdot 10^{-21}$
5	$5.81 \cdot 10^{-7}$	$5.61 \cdot 10^{-9}$	$3.34 \cdot 10^{-15}$	$8.10 \cdot 10^{-20}$

Table 5.19: Algebraic error for the multigrid method with damped Jacobi smoother with u_e after 600 iterations

depth of grid	L2-norm of the residual			
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.6$
3	$7.59 \cdot 10^{-4}$	$2.56 \cdot 10^{-10}$	$1.01 \cdot 10^{-12}$	$2.88 \cdot 10^{-13}$
4	$5.47 \cdot 10^{-5}$	$2.44 \cdot 10^{-6}$	$7.36 \cdot 10^{-16}$	$1.94 \cdot 10^{-13}$
5	$1.14 \cdot 10^{-5}$	$1.60 \cdot 10^{-5}$	$2.78 \cdot 10^{-13}$	$2.08 \cdot 10^{-13}$

Table 5.20: L2-norm of the residual for the multigrid method with damped Jacobi smoother after 250 iterations

depth of grid	convergence rate			
	Test Case 1 $\omega = 0.4$	Test Case 2 $\omega = 0.4$	Test Case 3 $\omega = 0.6$	Test Case 4 $\omega = 0.6$
3	0.92	0.87	0.82	0.75
4	0.92	0.88	0.82	0.79
5	0.93	0.88	0.83	0.79

Table 5.21: Convergence rate of the multigrid method with damped Jacobi smoother measured after 20 iterations

5.7 Multigrid with Conjugate Gradient Smoother

Examination of the Number of Smoothing Steps

The iterations for this algorithm are fixed to 250 iterations to compare the results with the other multigrid algorithms.

number of	algebraic error			
smoothing steps ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
2	$1.69 \cdot 10^{-9}$	$2.61 \cdot 10^{-12}$	$7.72 \cdot 10^{-6}$	$2.14 \cdot 10^{-7}$
6	$2.91 \cdot 10^{-17}$	$1.06 \cdot 10^{-16}$	$3.27 \cdot 10^{-14}$	$3.35 \cdot 10^{-15}$
10	$1.76 \cdot 10^{-17}$	$1.59 \cdot 10^{-16}$	$1.53 \cdot 10^{-16}$	$2.82 \cdot 10^{-16}$

Table 5.22: Algebraic error of the multigrid method with conjugate gradient smoother for various smoothing steps for depth of grid 3 with u_e after 600 iterations

number of	L2-norm of the residual			
smoothing steps ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
2	$2.59 \cdot 10^{-6}$	$3.97 \cdot 10^{-9}$	0.024	$6.64 \cdot 10^{-5}$
6	$2.29 \cdot 10^{-12}$	$4.19 \cdot 10^{-12}$	$1.20 \cdot 10^{-11}$	$1.90 \cdot 10^{-12}$
10	$2.34 \cdot 10^{-12}$	$4.50 \cdot 10^{-12}$	$2.09 \cdot 10^{-12}$	$1.59 \cdot 10^{-12}$

Table 5.23: L2-norm of the residual for the multigrid method with conjugate gradient smoother for various smoothing steps for depth of grid 3 after 250 iterations

number of	convergence rate			
smoothing steps ω	Test Case 1	Test Case 2	Test Case 3	Test Case 4
2	0.84	0.83	0.84	0.78
6	0.85	0.89	0.91	0.99
10	0.73	0.71	0.81	0.78

Table 5.24: Convergence rate of the multigrid method with conjugate gradient smoother for various smoothing steps for depth of grid 3 after 20 iterations

Tests with Optimal Number of Smoothing Steps

	algebraic error			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	$1.76 \cdot 10^{-17}$	$1.59 \cdot 10^{-16}$	$1.53 \cdot 10^{-16}$	$2.82 \cdot 10^{-16}$
4	$1.26 \cdot 10^{-16}$	$1.53 \cdot 10^{-16}$	$6.85 \cdot 10^{-17}$	$7.33 \cdot 10^{-17}$
5	$5.55 \cdot 10^{-17}$	$9.33 \cdot 10^{-17}$	$1.78 \cdot 10^{-16}$	$8.19 \cdot 10^{-17}$

Table 5.25: Algebraic error for the multigrid method with conjugate gradient smoother with 10 smoothing steps with u_e after 600 iterations

	L2-norm of the residual			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	$2.34 \cdot 10^{-12}$	$4.50 \cdot 10^{-12}$	$2.09 \cdot 10^{-12}$	$1.59 \cdot 10^{-12}$
4	$1.13 \cdot 10^{-12}$	$1.67 \cdot 10^{-12}$	$8.82 \cdot 10^{-13}$	$6.49 \cdot 10^{-13}$
5	$5.57 \cdot 10^{-13}$	$7.04 \cdot 10^{-13}$	$4.25 \cdot 10^{-13}$	$2.63 \cdot 10^{-13}$

Table 5.26: L2-norm of the residual for the multigrid method with conjugate gradient smoother with 10 smoothing steps after 250 iterations

	convergence rate			
depth of grid	Test Case 1	Test Case 2	Test Case 3	Test Case 4
3	0.73	0.71	0.81	0.78
4	0.85	0.42	0.87	0.99
5	0.86	0.50	0.77	0.61

Table 5.27: Convergence rate of the multigrid method with conjugate gradient smoother with 10 smoothing steps measured after 20 iterations

5.8 Special Test Problems

The conjugate gradient method preconditioned by multigrid with damped Jacobi smoother was tested under extreme geometric conditions. The damping parameter ω was set to 0.2.

	convergence rate	
after iteration	Test Case 5	Test Case 6
10	0.71	0.60
20	0.76	0.75
2000	0.99	0.99

Table 5.28: Convergence rate of the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother under extreme geometric conditions

	L2-norm of the residual	
after iteration	Test Case 5	Test Case 6
10 000	$3.43 \cdot 10^{-2}$	$1.47 \cdot 10^{-2}$

Table 5.29: L2-norm of the residual for the conjugate gradient method preconditioned by multigrid with damped Jacobi smoother under extreme geometric conditions

5.9 Visualisation of the Results

The following pictures show the results of the different temperature functions for the different geometries. Both are tested with a constant and a non-constant temperature function.
In general the deformations are infinitesimally small. Therefore large multiplication factors were used to show an exaggerated deformation of the laser crystals. Furthermore the exaggerated pictures show very clearly the correctness of the implemented formulas.
The applied constant function

$$\Delta T = 100\,000\,K \tag{5.7}$$

results in an enlargement of the initial volume. The Dirichlet boundary remains fixed and the rest of the body gets pumped up. Figure 5.1 shows this effect for `u_z` for the cylinder and figure 5.2 for `u_x` for the hexahedron.
In the case of the cylinder the applied non-constant function

$$\Delta T = x \cdot y \cdot 10\,000\,K \tag{5.8}$$

results in the symmetric image shown in figure 5.3, which shows the deformation of `u_z`.

In the case of the hexahedron the applied non-constant function

$$\Delta T = (2.0 - y) \cdot (2.0 - z) \cdot 10\,000\,K \tag{5.9}$$

results in the symmetric image depicted in figure 5.4 for `u_x`.
All other deformations are rotated respectively but the above mentioned pictures show the results clearer and more understandable.

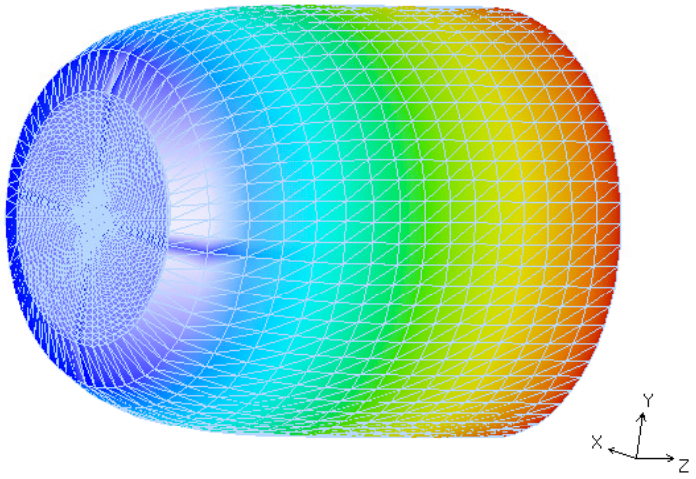


Figure 5.1: Deformation of the cylinder for a constant temperature function

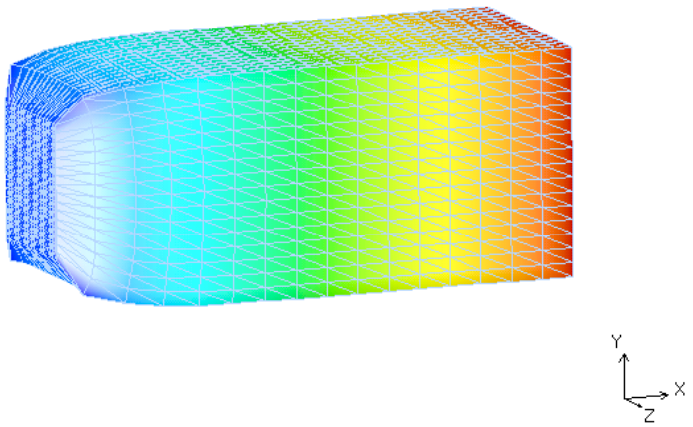


Figure 5.2: Deformation of the hexahedron for a constant temperature function

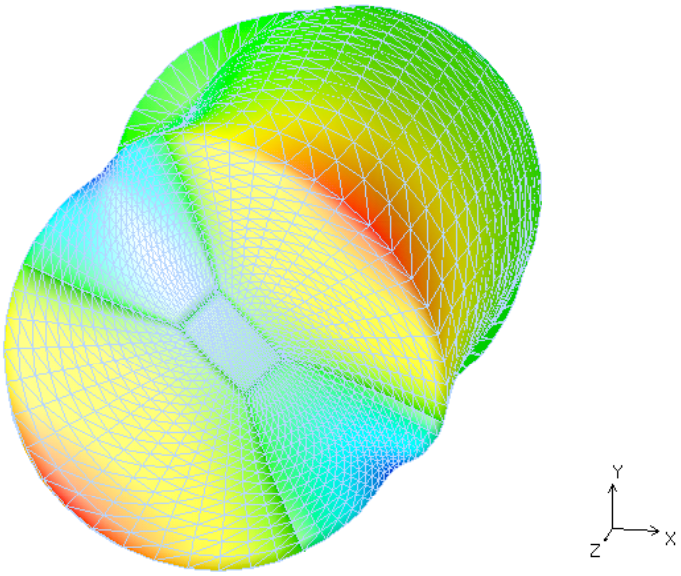


Figure 5.3: Deformation of the cylinder for a symmetric temperature function

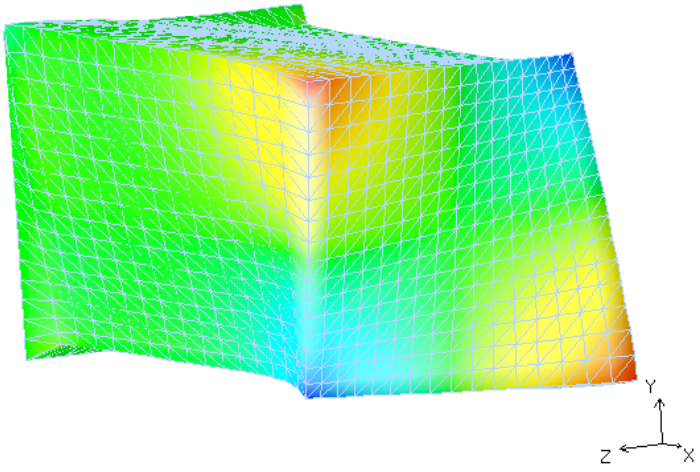


Figure 5.4: Deformation of the hexahedron for a symmetric temperature function

Chapter 6

Conclusions

In general the implemented algorithms for the cylinder geometry are slower than for the hexahedron geometry. This is based on the fact that despite the same parameters for the depth of grid or rather the number of gridpoints the produced grid of the unstructured block grids library for the cylinder geometry is considerable finer.
Moreover the implemented algorithms which base on a multigrid method with Gauss-Seidel smoother are worse than those with damped Jacobi smoother. This results from the fact that the Gauss-Seidel method is not symmetric in contrast to the damped Jacobi method (c.f. [Hac93, p. 115-122]).
The implemented multigrid method with conjugate gradient smoother seems to produce quite good results. But one has to consider that they are produced with ten very time-consuming pre and post-smoothing steps each. Thus the aim of a iterative solver is to smooth the high frequency errors in about two iterations is not fulfilled in this method. Therefore the good results of this method are mostly based on the iteration of the conjugate gradient method and not on the multigrid algorithm.
The best results can be achieved with the methods based on the multigrid method with damped Jacobi smoother. The pure multigrid method with damped Jacobi smoother produces good results for the algebraic error and the convergence rate. The large values for the algebraic error seem to be too good and therefore it could have been better to set the number of iterations to get the approximated exact solution larger than 600 iterations. Moreover the 250 iterations for the approximated solution could also be reduced because the convergence of the L2-norm of the residual is in most cases guaranteed before.
The even faster results regarding the iterations can be achieved by the preconditioned conjugate gradient method with multigrid with damped Jacobi smoother. This method has also been tested under extreme geometric conditions. Nevertheless the convergence rates at the beginning of the algorithm are good but decrease after around 1500 steps. For this further investigations have to be made.
Moreover the damping parameter ω is strongly dependent on the geometry, e.g. the preconditioned conjugate gradient method with multigrid with damped Jacobi smoother applied on Test Case 3 and 4 produces fastly good results for $\omega = 0.4$ but Test Case 5 and 6 do not converge for $\omega = 0.4$ but for $\omega = 0.2$.

Chapter 7

Acknowledgements

I would like to thank Professor Dr. Christoph Pflaum for his support and input. Furthermore I would like to thank Professor Dr. Kai Willner for his support with mechanical problems. I would also like to thank Jochen Härdtlein, Britta Heubeck, Matthias Wohlmuth and Christoph Freundl for their help with several problems. Additionally, I would like to thank Stefan Donath and Tobias Gradl for their help with LaTeX.

Bibliography

[Bra01] Braess, Dietrich: *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Second Edition. Cambridge: Cambridge University Press, 2001.

[Col07] *Colsamm Homepage*. `http://www10.informatik.uni-erlangen.de/~jochen/colsamm.html`, 06.04.2007.

[DR06] Dahmen, Wolfgang; Reusken, Arnold: *Numerik für Ingenieure und Naturwissenschaftler*. Berlin; Heidelberg; New York: Springer, 2006.

[GHSW04] Gross, Dietmar; Hauger, Werner; Schnell, Walter; Wriggers, Peter: *Technische Mechanik Band 4: Hydromechanik, Elemente der Höheren Mechanik, Numerische Methoden*. 5. Auflage. Berlin; Heidelberg; New York: Springer, 2004.

[Hac93] Hackbusch, Wolfgang: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*. 2. überarbeitete und erweiterte Auflage. Stuttgart: Teubner 1993.

[Iff90] Iffländer, Reinhard: *Festkörperlaser zur Materialbearbeitung*. Berlin; Heidelberg; New York: Springer 1990.

[Koe99] Koechner, Walter: *Solid-State Laser Engineering*. 5th revised and updated edition. Berlin; Heidelberg; New York: Springer 1999.

[Kro07] Kröher, Michael O. R.: *Aus Deutschlands Osten kommt das Licht.* `http://www.spiegel.de/wirtschaft/0,1518,475881,00.html`, 09.04.2007.

[Las07] *LASCAD Homepage.* `http://www.las-cad.com`, 19.04.2007.

[Pfl06] Pflaum, Christoph: *Simulation und wissenschaftliches Rechnen I.* Vorlesungsskript, 2006.

[SK04] Schwarz, Hans-Rudolf; Köckler, Norbert: *Numerische Mathematik*. 5., überarbeitete Auflage. Wiesbaden: Teubner, 2004.

[TOS01] Trottenberg, Ulrich; Oosterlee, Cornelis; Schüller, Anton: *Multigrid*. San Diego; San Francisco; New York; Boston; London; Sidney; Tokyo: Academic Press, 2001.

[Uns07] *Unstructured Block Grids Homepage.* `http://www10.informatik.uni-erlangen.de/~pflaum/expde/UG.html`, 06.04.2007.