Finite Element Solutions of Heat Conduction Problems in Complicated 3D Geometries Using the Multigrid Method

Diplomarbeit

Bastian Pentenrieder

Aufgabensteller: Prof. Dr. Christoph Zenger
Betreuer: Prof. Dr. Sergey Slavyanov
Abgabetermin: 29. Juli 2005
Hiermit erkläre ich, dass ich die Diplomarbeit selbständig angefertigt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 29. Juli 2005

...............................................................

Bastian Pentenrieder
# Contents

1 Introductory remarks ............................................. 7  
  1.1 Aims and motivation of this thesis .......................... 7  
  1.2 Overview of the individual chapters ......................... 9  
  1.3 Acknowledgements ............................................. 9  

2 Modelling .......................................................... 11  
  2.1 The different modes of heat transfer ......................... 11  
  2.2 Derivation of Fourier’s partial differential equation ....... 14  
  2.3 Boundary conditions ........................................... 16  
  2.4 Formulation of the problem ................................... 17  
  2.5 A possible generalization ..................................... 19  

3 Solution ............................................................. 21  
  3.1 Finite element method ......................................... 21  
    3.1.1 The basics ............................................... 21  
    3.1.2 Application to the posed problem ....................... 29  
    3.1.3 Application to the generalized problem ................. 34  
  3.2 Multigrid method .............................................. 36  
    3.2.1 Motivation ............................................... 37  
    3.2.2 Communication between the grids ....................... 43  
    3.2.3 Hierarchical generating system ......................... 46  
    3.2.4 Sketch of a multigrid cycle in peano3d ................. 47  
    3.2.5 Computation of the diagonal elements ................... 49  
  3.3 Physical interpretation of the algorithm .................... 53  
  3.4 Implementation ................................................ 55  
    3.4.1 Algorithm ............................................... 55  
    3.4.2 Geometry description ................................... 58  
    3.4.3 Postprocessing .......................................... 60  

4 Results ............................................................. 63  
  4.1 Graphical user interfaces for data I/O ....................... 63  
  4.2 Calculated heat fluxes and effective thermal conductivities 66  
  4.3 Visualization and interpretation ............................. 72  
  4.4 Findings from the generalized problem ....................... 82
5 Summary and outlook 85
  5.1 Achievements ......................................................... 85
  5.2 Outlook .............................................................. 85

A The element stiffness matrix for the tensor generalization 89
1 Introductory remarks

1.1 Aims and motivation of this thesis

Partial differential equations (PDEs) play an important role in a wide range of disciplines. They emerge as the governing equations of problems arising in such different fields of study as biology, chemistry, physics and engineering—but also economy and finance. In order to solve them and thus gain a deeper insight into the corresponding issue, one is in need of powerful algorithms and data structures.

A popular and widely used approach to the solution of partial differential equations is the finite element method (FEM), which, as often in numerical mathematics, reduces the initial problem to the task of solving a system of linear equations. For this purpose, especially when dealing with a large number of unknowns (e.g. \( \sim 10^6 \)), classical direct solvers turn out to be inappropriate, and more modern iterative schemes like the multigrid method come into play.

But not only a sophisticated algorithm is essential. If we fail to arrange the unknowns in the memory in an intelligent way, we will encounter a typical bottleneck related to contemporary computer architecture. It is caused by the fact that the speed of the CPU has become faster and faster over the last years, whereas the memory could not keep pace with that development. The following metaphor might give us an idea of the problem: We imagine a man working on an assembly line. He represents the processor. The pieces delivered by the line can be thought of as boxes (data packages), each one comprising a certain amount of units (values) that come from the depths of the factory (the memory). Since the man works extremely fast in comparison to the velocity of the assembly line, we have to ensure that only those units actually needed for the current job (computation) are put into the box. Otherwise, the worker will come across unnecessary parts instead of required ones, and he will have to wait for the next box to arrive doing nothing in the meantime (the processor performs idle cycles). Roughly speaking, this means that an efficient way has to be found how to store things inside the factory, take them out of the storage onto the assembly line and afterwards back again.

Of course, this conclusion sounds to some extent like a commonplace. However, the truth is that the problem of organizing the data in adequate memory structures in order to avoid the described bottleneck is not trivial. Within the scope of his doctoral thesis, Markus Pögl developed and implemented a method that traverses the computational domain on a space-filling curve of Peano type, while it moves the data values through a system of 27 stacks\(^1\) ([Pögl 04]). Thereby, it achieves perfect efficiency in terms of the

---

\(^1\)A stack is a simple structure within the memory which permits only two kinds of data access: *pop* (to take data from the stack) and *push* (to put data onto the stack).
1 Introductory remarks

so-called cache hit rate: Whenever the algorithm enters a new cell along the curve, the corresponding data can be found on top of a priori known stacks. After performing the computations, the data is pushed onto (in general) different stacks. When proceeding to the next cell, the situation there will be the same, i.e. we never have to pop any actually not required data from the stacks in order to get access to the elements beneath (for details see [Pögl 04]).

Since the main objective of the doctoral thesis consisted in demonstrating the cache-efficiency of the new concept, only the Poisson equation as a model problem was implemented in the original code:

$$- \Delta u(x) = f(x) \quad \forall x \in \Omega \subseteq (0,1)^3$$

$$u(x) = 0 \quad \forall x \in \partial \Omega$$

The aim of my thesis was to provide the necessary adjustments and extensions to overcome this restriction to a very small class of mathematical problems while maintaining the cache-efficiency. In the end, the algorithm should be able to solve PDEs of the type

$$-\text{div}(T(x)\nabla u(x)) = f(x) \quad \text{with } T(x) \in \mathbb{R}^{3,3}_{\text{positive definite}}.$$  

The major challenge was the adaptation of the existing multigrid method to coefficients that depend on the spatial coordinates, especially in terms of keeping the excellent convergence properties of the initial code from [Pögl 04].

Suitable real-life problems should be modelled and simulated by the resulting implementation, thus illustrating the power of the new concept with respect to practical issues.

Interplay with the research work of Prof. Slavyanov

It was in the year 2003 that I got to know Prof. Slavyanov during a student seminar in Saint Petersburg. Together with Prof. Zenger, he supervised the course Mathematical Modelling and Numerical Simulation of Problems in Physics. At that time, I myself participated in the program and contributed a talk about the principles of the finite element method.

One year later, Prof. Slavyanov told us that he was interested in the German Heat Insulation Ordinance and the succeeding EnergieEinsparVerordnung (EnEV). His aim was to establish an awareness of the necessity of energy-saving among Russian architects and construction engineers. Connections to the brick-manufacturing company Knauf finally gave rise to the idea of investigating the thermal characteristics of ceramic blocks\(^2\) with complicated geometry, as they are used in present-day buildings. Choosing this problem to be the first test for the algorithm, made particular sense since there already existed a paper with reference results which was published by V. Grikurov, R. Trepkov and Prof. Slavyanov ([Slavyanov et al. 04]).

Therefore, I decided to spend the main period of my diploma thesis on the spot—at the Faculty of Physics of the Saint Petersburg State University.

\(^2\)Throughout this thesis, ceramic block refers to a brick featuring gaps that are filled with air. When talking about the brick as a whole, we say ceramic block, in contrast to its two components brick and air.
1.2 Overview of the individual chapters

First of all, in chapter 2, a brief introduction to heat transfer is given. The different mechanisms of transportation are presented in some detail to help us understand the boundary conditions of Fourier’s PDE. Besides of that, you will find a derivation for the latter one, the exact formulation of the problem we are aiming to solve, and also a small discussion of alternative modelling approaches and a possible generalization.

After setting up the problem, chapter 3 then deals with its solution. It explains the basics of the finite element/multigrid method and shows how these techniques can be used for our simulation of heat conduction within ceramic blocks. By means of the knowledge from chapter 2, we will be able to recognize that the mathematical algorithm (at least to some extent) imitates the physical processes inside the material. The last subchapter is dedicated to details of the implementation.

Chapter 4 presents the calculated results in numbers and pictures. Both of these are discussed under various aspects. Particular attention will be paid to the promising convergence properties of the tensor generalization.

Finally, chapter 5 gives an outlook on possibilities how to continue the research on the topic, summarizes the achievements and points out how they can be used in related work.

1.3 Acknowledgements

A lot of people contributed to the success of this project. In the first instance, I am deeply indebted to Prof. Dr. Christoph Zenger and Prof. Dr. Sergey Slavyanov for the inspiring and patient supervision of my thesis and their great help in organizing the stay in Saint Petersburg.

Furthermore, I would like to thank Markus Pögl, Frank Günther and Andreas Krahnke for all the explanations during the initial phase of my work. Without them, the process of understanding the starting-point implementation of the algorithm would have taken much longer. The useful hints of Dr. Miriam Mehl concerning the multigrid method deserve a special mention. Wolfgang Herder and Ellen Maier were so kind to proof-read the final version of this thesis.

The pleasant atmosphere among the fellow graduands and scientists at the chair of Prof. Zenger and the amiability of Prof. Slavyanov and all the Russian people at his faculty with whom I became acquainted have been a very procreative environment for the whole time.

Last—but not least—the share of my parents and my girlfriend must not be forgotten. They always encouraged me in my plans. I am very grateful that my mother and my father made it possible for me to attend university, and I want them to know that I don’t take for granted all the support they have given me through the years.
1 Introductory remarks
2 Modelling

In *scientific computing*, the examination of a certain problem is always a three-stage process (figure 2.1). Foremost, we need to think about how to model the task in the language of mathematics. Then, the next step is to discretize the obtained (continuous) equations and solve them by an appropriate numerical method. Finally, we have to present the results in an adequate way. This may be done by giving some key values or by applying visualization techniques to the computed datasets.

According to this scheme, we start with introducing some fundamental laws of heat transfer that will help us to translate the heat conduction problem within ceramic blocks into mathematical equations. For profound studies on this branch of engineering, the interested reader is recommended the definitive textbooks [Incropera/DeWitt 02] and [Baehr/Stephan 03].

2.1 The different modes of heat transfer

By definition, heat is the energy that flows from the higher level of temperature to the lower (without any work being performed), whenever there exists a temperature diffe-
The equations which quantify the amount of transferred heat all fit into the pattern

\[
\text{flow} = \text{transport coefficient} \times \text{potential gradient},
\]

where the **potential gradient** represents a derivative or some difference expression. As flow value, we encounter

- **heat flux** \( \dot{q} \) \([W/m^2]\): energy per time and area or
- **heat transfer rate** \( \dot{Q} \) \([W]\): energy per time (passing through a fixed reference area).

The **transport coefficient** depends on the particular mode of transfer . . .

**Conduction**

Heat conduction is the diffusive transport of thermal energy. In liquids and gases, it is caused by the interaction of moving atoms and molecules, in solids by lattice oscillations and in electroconductive material additionally by unbound electrons.

For instance, consider a swimming pool that is warm on the left and cold on the right hand side (cf. figure 2.2). We suppose that the water is completely motionless. On a microscopic level, neighboring molecules are in mutual interaction and exchange kinetic energy—more or less chaotically. But if we change the scale of our observation to the macroscopic level, it is evident that, in total, more kinetic energy will be transferred from the left to the right than vice versa. The system tends towards an equilibrium state with the same temperature everywhere. During this levelling diffusion process, a transport of thermal energy occurs.

If we know the internal temperature distribution for a given moment in time, we can calculate the heat flux in every single point \( x \in \mathbb{R}^3 \) by **Fourier’s law**:

\[
\dot{q}(x) = -\lambda \nabla T(x), \quad [\lambda] = \frac{W}{mK} \tag{2.1}
\]
The coefficient $\lambda$ is called *thermal conductivity* and strongly depends on the material. Note that Fourier’s law gives us information about both direction and magnitude of the heat flux.

**Convection**
The exchange of heat between a moving fluid (gas or liquid) and an adjoining wall is named convection. Depending on the temperature proportions, heat is delivered to the wall ($T_f > T_w$) or taken away from it ($T_f < T_w$). If the current of the fluid is held up by a pump or blower, engineers denote that process *forced convection*. But also without such technical aids, a flow may emerge due to density differences in the fluid, which are brought forth on their part by temperature or concentration gradients (*free convection*). The forced convection usually clearly exceeds the free one.

Common examples for important convection processes are the cooling of a liquid that overflows a heat exchanging device (cf. figure 2.3) or the passing of hot gases through the stator vane of a turbine.

\[
\dot{Q} = \alpha A (T_f - T_w), \quad [\alpha] = \frac{W}{m^2 K}
\]

Figure 2.3: Cooling of a fluid by convection

Newton took the following approach in order to calculate the heat transfer rate from a fluid to an adjoining surface of area $A$:

\[
\dot{Q} = \alpha A (T_f - T_w), \quad [\alpha] = \frac{W}{m^2 K}
\]

The coefficient $\alpha$ is termed *convection heat transfer coefficient* or simply *Newton coefficient*. To determine it is a difficult task, as it depends—unlike the thermal conductivity from above—not only on the involved matter but also on the roughness of the surface and on the velocity and type of the flow (laminar or turbulent).

**Radiation**
All bodies emit thermal radiation. According to the *Stefan-Boltzmann law*, the *heat emission density* $q_{rad}$ of a body at temperature $T$ with emissivity $\varepsilon$ ($0 \leq \varepsilon \leq 1$) is calculated by

\[
q_{rad} = \varepsilon \sigma T^4, \quad \sigma = 5,67 \cdot 10^{-8} \frac{W}{m^2 K^4}.
\]
Between two objects of different temperature (e.g. between a sunbather and the sun, or a rooftop and the sky), there occurs a net heat exchange, which can be described by

\[ \dot{Q}_{1\rightarrow 2} = C_{12} A (T_1^4 - T_2^4), \quad [C_{12}] = \frac{W}{m^2 K^4}, \]

where \( A \) is a reference area and \( C_{12} \) a place holder for a more complex expression. Amongst others, this one accounts for the geometry of the two bodies, their mutual visibility and the emissivities.

![Figure 2.4: Two plates exchanging heat by radiation (electromagnetic waves)](image)

### 2.2 Derivation of Fourier’s partial differential equation

In the previous section, we have learned that knowledge of the temperature distribution would allow us to determine the heat flux in every single point (see Fourier’s law (2.1)). However, usually only information about surface values is provided. For this reason, the aim of the following paragraphs is to gain a PDE by the solution of which we shall obtain the internal temperature.

To this end, we imagine a tiny cube of volume \( dV = dx \, dy \, dz \), being part of a three-dimensional body (cf. figure 2.5). Under the influence of a temperature distribution \( T(\mathbf{x}) \) inside the body, there occur heat fluxes \( \dot{q}_k \) and \( \dot{q}_{k+dk} \) \((k = x, y, z)\) through the six corresponding surfaces of the cube. Using Taylor approximation of first order,

\[ \dot{q}_{k+dk} = \dot{q}_k + \frac{\partial \dot{q}_k}{\partial k} dk \quad (k = x, y, z), \]

we obtain the following net heat transfer rates into the cube (units in Watt):

- along the x-axis
  \[ (\dot{q}_x - \dot{q}_{x+dx}) \, dy \, dz = - (\partial \dot{q}_x/\partial x) \, dx \, dy \, dz = - (\partial \dot{q}_x/\partial x) \, dV \]
- along the y-axis
  \[ (\dot{q}_y - \dot{q}_{y+dy}) \, dx \, dz = - (\partial \dot{q}_y/\partial y) \, dy \, dx \, dz = - (\partial \dot{q}_y/\partial y) \, dV \]
- along the z-axis
  \[ (\dot{q}_z - \dot{q}_{z+dz}) \, dx \, dy = - (\partial \dot{q}_z/\partial z) \, dz \, dx \, dy = - (\partial \dot{q}_z/\partial z) \, dV \]
2.2 Derivation of Fourier’s partial differential equation

Adding up all three of them and substituting \(-\lambda (\partial T/\partial k)\) for \(\dot{q}_k\) (which is just the component form of equation (2.1)) yields the overall net heat entry into the volume \(dV\):

\[
\dot{Q}_{\text{net}} = \partial_x \left( \lambda \frac{\partial T}{\partial x} \right) dV + \partial_y \left( \lambda \frac{\partial T}{\partial y} \right) dV + \partial_z \left( \lambda \frac{\partial T}{\partial z} \right) dV = \text{div}(\lambda \nabla T) dV
\]

In the absence of heat sources and sinks respectively, \(\dot{Q}_{\text{net}}\) must be equal to the storage rate of thermal energy

\[
\dot{H}_{\text{st}} = c_p \, dm \frac{\partial T}{\partial t} = \rho \, c_p \, \frac{\partial T}{\partial t} \, dV,
\]

where

- \(c_p\) is the specific heat capacity ([\(c_p\] = \(kJ/kgK\))
- \(dm\) the mass of the volume \(dV\)
- \(\rho\) the density of the material and
- \(\frac{\partial T}{\partial t}\) the partial derivative of the temperature with respect to time.

This simple balance of \(\dot{Q}_{\text{net}}\) and \(\dot{H}_{\text{st}}\) finally gives us Fourier’s PDE

\[
\rho \, c_p \, \frac{\partial T}{\partial t} = \text{div}(\lambda \nabla T), \quad (2.2)
\]

which must be satisfied in every point \(\mathbf{x} = (x, y, z)\) for every moment \(t\).
2.3 Boundary conditions

Now that we have derived the underlying PDE of heat conduction, we need to make up our mind about boundary conditions, that, for a given point in time, specify the behavior of the solution on the border of the domain. In the following three paragraphs, the different types are introduced using the terms by which mathematicians (engineers) refer to them.

**Dirichlet boundary condition** (boundary condition of first type)
The temperature on the boundary of the body is prescribed by

\[ T(x, t) = T_w(x, t) \]

where \( T_w \) is a known function. In a lot of practical applications, \( T_w \) is simply a constant.

**Neumann boundary condition** (boundary condition of second type)
If the heat flux \( \dot{q}_w \) out of the body (perpendicular to the surface) is given, then Fourier’s law helps us in determining the partial derivative of the temperature with respect to the outward normal vector \( n \):

\[ \dot{q}_w(x, t) = -\lambda \frac{\partial T}{\partial n}(x, t) \Rightarrow \frac{\partial T}{\partial n}(x, t) = -\frac{\dot{q}_w(x, t)}{\lambda} \]

Notice that in the special case of perfect insulation (\( \dot{q}_w = 0 \)), the equation to the right becomes a homogeneous Neumann boundary condition

\[ \frac{\partial T}{\partial n}(x, t) = 0. \]

**Cauchy boundary condition** (boundary condition of third type)
Very often, there is a thermal interaction between the body and a surrounding fluid of temperature \( T_f \) (cf. figure 2.6). To quantify this, the boundary of the domain is considered as a control ”volume” for an energy balance (represented by dashed lines in the figure). Since the thickness of the boundary is zero, no energy can be stored within. This means that all the heat entering a surface increment from the interior (by conduction) has to leave outwards (by convection):

\[ -\lambda \frac{\partial T}{\partial n}(x, t) = \alpha (T(x, t) - T_f) \quad (2.3) \]

Sometimes, the heat transfer by radiation is taken into account as well. Engineers usually include it by simply introducing an additional transport coefficient \( \alpha_{rad} \) :

\[ -\lambda \frac{\partial T}{\partial n}(x, t) = (\alpha + \alpha_{rad}) (T(x, t) - T_f) \]

Note that the boundary condition of third type specifies neither the value nor the derivative of the solution, merely a correlation between both of them.
2.4 Formulation of the problem

The preceding subchapter has completed the preparation needed for setting up the model that describes the heat conduction problem within ceramic blocks. An example for the 3D geometry of such a block is shown in figure 2.7. We choose a coordinate system with axes perpendicular to the outer surfaces of the block. Then, the corners are given by the eight coordinate triples

\[(x_1, y_1, z_1), (x_2, y_1, z_1), (x_1, y_2, z_1), (x_2, y_2, z_1),
(x_1, y_1, z_2), (x_2, y_1, z_2), (x_1, y_2, z_2), (x_2, y_2, z_2)\].

All the ceramic blocks examined are cuboids, that we will denote by Ω:

\[Ω = (x_1, x_2) × (y_1, y_2) × (z_1, z_2)\]

They consist of several air cavities and the brick material itself:

\[Ω = Ω_{brick} ∪ Ω_{air} \land Ω_{brick} ∧ Ω_{air} = ∅\]

Since this thesis is restricted to steady-state problems (\(\frac{∂T}{∂t} = 0\)), the derived PDE of Fourier (2.2) becomes

\[\text{div}(\lambda(x)\nabla T(x)) = 0 \quad ∀x ∈ Ω\]  \hspace{1cm} (2.4)

with

\[\lambda(x) = \begin{cases} 
  \lambda_{brick} & \text{if } x ∈ Ω_{brick} \\
  \lambda_{air} & \text{if } x ∈ Ω_{air}
\end{cases} \quad (\lambda_{brick} \gg \lambda_{air}).\]  \hspace{1cm} (2.5)

**Dirichlet boundary conditions** are specified on

\[\{x ∈ ∂Ω : y = y_1 \lor y = y_2\} .\]
They symbolize the temperature proportions inside and outside of a room. On the remaining surfaces
\[ \{ x \in \partial \Omega : x = x_1 \lor x = x_2 \lor z = z_1 \lor z = z_2 \} , \]
we assume **homogeneous Neumann boundary conditions**. The aim is to calculate the temperature distribution, and hence

- the overall **heat transfer rates/mean heat fluxes** through the block, and
- a so-called **effective thermal conductivity** which will be defined later.

**First remark:** Using the model (2.4) and (2.5), we disregard all convection and radiation processes inside the ceramic block, whereas its thermal conductivity is represented by a piecewise constant function that exhibits jumps at the transitions \( \partial \Omega_{\text{brick}} \cap \partial \Omega_{\text{air}} \).

**Second remark:** In chapter 5.2, you can find a sketch of how to account for the more general boundary conditions of third type. Their implementation would entail some additional effort in the computation of the residual.
2.5 A possible generalization

**Alternative models**

Assuming the air cavities to be a perfect heat insulation, a simpler approach would be

\[
\text{div}(\lambda_{\text{brick}} \nabla T(x)) = \lambda_{\text{brick}} \Delta T(x) = 0 \quad \forall x \in \Omega_{\text{brick}}
\]

with

\[
\frac{\partial T}{\partial n}(x) = 0 \quad \forall x \in \partial \Omega_{\text{brick}} \cap \partial \Omega_{\text{air}}
\]

as boundary conditions at the (originally internal) transitions. Notice that the domain, on which the PDE shall be solved, is now \(\Omega_{\text{brick}}\) instead of \(\Omega\). This model underlies the computations in [Slavyanov et al. 04]. However, we use model (2.4) and (2.5) since we are interested in the question how to deal with jumps in \(\lambda\) from a numerical point of view.

Of course, there also exist more complex approaches taking into account micro-convection and cavity radiation. However, this thesis refrains from their consideration and establishes the major part of its examinations on the basis of (2.4) and (2.5).

2.5 A possible generalization

The thermal conductivity of layered materials like wood or rock formations depends on the direction. In order to account for that anisotropy, the scalar \(\lambda\) in Fourier’s law (2.1) needs to be replaced by a second order tensor:

\[
\dot{q}(x) = -\Lambda \nabla T(x), \quad \Lambda \in \mathbb{R}^{3 \times 3} \text{ symmetric positive definite}
\]

For that case, the same straightforward derivation of the governing PDE would lead to

\[
\text{div}(\Lambda(x) \nabla T(x)) = 0 \quad \forall x \in \Omega
\]  

(2.6)

instead of (2.4). Why is that interesting? You might object that our heat conduction problem does not include any anisotropic material. Well, you are right. But . . .

A synergetic effect

In the chapter about FEM, we shall see that (2.6) is closely related to the numerical solution of the Poisson equation

\[
-\Delta u(\xi) = f(\xi) \quad \forall \xi \in \mathcal{P}
\]

on a complex (physical) domain with curved boundary. The latter one is parameterized by a function \(\psi : \Omega \rightarrow \mathcal{P}\) that maps the regular (computational) domain \(\Omega\) to \(\mathcal{P}\). With the aid of \(\psi\), the originally cubic cells of \(\Omega\) can be stretched into certain directions. In some applications, this is extremely desirable: For instance, when considering the air flow along a wing, the quantities of interest usually exhibit much more vitality perpendicular to the profile than in tangential direction. Thus, a discretization of \(\mathcal{P}\) by stretched cells makes sense (cf. figure 2.8). Details about the grid generation can be found in the fourth chapter of [Dornseifer 97].
Figure 2.8: Example for a domain transformation

Because of its importance, we will include the above generalization, i.e. there will be a derivation of the corresponding *stiffness matrices* and also some results computed for simple model problems. However, you should keep in mind that this is an optional module within the developed program that accompanies this thesis. The heat conduction problem and its solution remain untouched by it.
3 Solution

3.1 Finite element method

3.1.1 The basics

Although the finite element method (FEM) serves for solving partial differential equations, its main features are actually explained best using a simple steady-state example in one space dimension. For this reason, let us consider the heat conduction inside a thin metal rod of length $L$ with an insulated surface:

\[
\begin{align*}
\text{differential equation} & \quad -\lambda T''(x) = f(x) \quad \forall x \in (0, L) \\
\text{boundary conditions} & \quad \begin{cases} 
T(x = 0) = 0 \\
T'(x = L) = 0
\end{cases}
\end{align*}
\]

(3.1)

The constant $\lambda$ denotes the thermal conductivity, $T(x)$ the temperature at point $x$ and $f$ the rate of heat generation per unit volume ($[f(x)] = W/m^3$). Such an internal heat generation may emerge due to chemical reactions or an electric current flowing through the rod. The left end is kept at temperature level 0, while at the right boundary no heat is allowed to enter or leave.

Remark: Of course, it would be an easy exercise to solve this particular problem (3.1) by simple integration. Applying finite elements in this context is, at least to some extent, like breaking a butterfly on a wheel. But after all, it will teach us the basics of FEM in a very clear and understandable way.

So, let’s start: The **first step** of the method consists in providing a new formulation. For this purpose, we multiply the differential equation in (3.1) with so-called test functions $v$ and integrate both sides over the whole domain $\Omega = (0, L)$:

\[-\int_0^L \lambda T''(x) v(x) \, dx = \int_0^L f(x) v(x) \, dx\]

(3.2)

Next, the second derivative of $T$ is shifted to the test function $v$ with the aid of integration by parts:

\[-\int_0^L \lambda T''(x) v(x) \, dx = -[\lambda T'v]_{x=0}^{x=L} + \int_0^L \lambda T'(x) v'(x) \, dx\]

(3.3)

If we require the test function $v$ to satisfy the same Dirichlet boundary condition as the temperature $T$,

\[v(x = 0) = 0,\]
and remember the homogeneous Neumann boundary condition from (3.1),

\[ T'(x = L) = 0, \]

then we can simplify (3.3) to

\[ - \int_0^L \lambda T''(x) v(x) \, dx = \int_0^L \lambda T'(x) v'(x) \, dx. \]  

(3.4)

Substitution of (3.4) into (3.2) finally yields:

\[ \lambda \int_0^L T'(x) v'(x) \, dx = \int_0^L f(x) v(x) \, dx \]  

(3.5)

At this point, although this section is not meant to become an excursion to functional analysis, it is necessary to spend some thoughts on the choice of a proper \textit{ansatz space}, which the test functions \( v \) are taken from. In equation (3.5), the first derivative of \( v \) appears inside an integral. Since the integration provides a smoothing effect, \( v' \) need not be continuous. Actually, \( v' \) may exhibit a quite “wild” behavior. In this context, the calculus of \textit{weak derivatives} comes into play:

\textbf{Definition}: A function \( w \) is termed weak derivative of a function \( v : (a, b) \to \mathbb{R} \), if

\[ \int_a^b w(x) \beta(x) \, dx = - \int_a^b v(x) \beta'(x) \, dx \]

holds for all \( \beta \in C^\infty([a, b]) \) with \( \beta(a) = \beta(b) = 0 \).

\( \square \)

\textbf{Remark}: Note that for a differentiable function the weak derivative is equal to the usual derivative. Hence, we refrain from the introduction of a new notation and simply refer to the weak derivative of a function \( v \) by \( v' \).

Furthermore, the \textit{Sobolev space} \( H^1((a, b)) \) is considered as the linear space of all functions \( v : (a, b) \to \mathbb{R} \), for which there exists a weak derivative \( v' \), and the squares of both function and weak derivative are integrable:

\[ \int_a^b v(x) v(x) \, dx < \infty, \quad \int_a^b v'(x) v'(x) \, dx < \infty \]

(In fact, the exact definition of Sobolev spaces is a little bit more complicated, as it requires the notion of the \textit{Lebesgue measure}. But within the scope of this thesis, an understanding of the concept in the above sense is sufficient.)

After this theoretical preparation, an adequate ansatz space \( V \) for the test functions can be determined:

\[ V := \{ v \in H^1((0, L)) \text{ with } v(0) = 0 \} \]  

(3.6)
3.1 Finite element method

Assuming that this functional space is also appropriate for the temperature function $T$, the problem may now be written in the following way:

Find $T \in V$, so that

$$\lambda \int_0^L T'(x) v'(x) \, dx = \int_0^L f(x) v(x) \, dx \quad \forall v \in V. \quad (3.7)$$

In comparison to (3.1), the temperature $T$ is not anymore required to be two times differentiable. The pointwise equality of both sides has been replaced by an integral expression. For this reason, (3.7) is called *weak formulation* and $T$ a *solution in the weak sense*. However, it can be shown that under certain circumstances a function $T$ that fulfills (3.7) also satisfies (3.1). For details, the reader is recommended the lecture [Brokate 02] or the excellent textbook [Braess 03].

**Remark:** In (3.7), the Dirichlet boundary condition from the original formulation (3.1) is still demanded ($T$ must be an element of $V$), whereas the theory tells us that the Neumann boundary condition will be fulfilled automatically. In this context, the first one is called an *essential* and the latter one a *natural* boundary condition.

The **second step** in FEM—as we apply it—is named *Galerkin projection*. The idea is as follows: Since we intend to solve (3.7) on the computer, the infinite-dimensional space $V$ is difficult to handle. Therefore, $V$ is replaced by a finite-dimensional subspace $S$. Postulation (3.7) then becomes:

Find $T_S \in S$, so that

$$\lambda \int_0^L T_S'(x) v'(x) \, dx = \int_0^L f(x) v(x) \, dx \quad \forall v \in S. \quad (3.8)$$

It has not been decided yet by which *ansatz functions* the vector space $S$ is to be spanned. Later on, we will make a special choice, namely linear finite elements (see below). For now, we just take an arbitrary basis $\{\phi_i : 1 \leq i \leq n\}$. An element $T_S \in S$ is then represented by

$$T_S = \sum_{i=1}^n T_i \phi_i.$$ 

Substitution of the basis functions $\phi_i$ into both sides of equation (3.8) results in the *equivalent* formulation

$$\sum_{i=1}^n T_i \int_0^L \phi_i'(x) \phi_j'(x) \, dx = \frac{1}{\lambda} \int_0^L f(x) \phi_j(x) \, dx \quad \forall j \in \{1, \ldots, n\}. \quad (3.9)$$

Defining the *stiffness matrix*

$$A := \begin{pmatrix} \int_0^L \phi_i'(x) \phi_j'(x) \, dx \end{pmatrix}_{i,j=1,\ldots,n} \quad (3.10)$$
3 Solution

and the right hand side

\[ \mathbf{b} := \left( \frac{1}{\lambda} \int_0^L f(x) \phi_j(x) \, dx \right)_{j=1,...,n}, \tag{3.11} \]

the system of linear equations (3.9) can be written as

\[ A \cdot \mathbf{t} = \mathbf{b}. \tag{3.12} \]

The matrix \( A \) is symmetric and invertible. Proof: Let us assume there exists a vector \( \mathbf{u} \neq \mathbf{0} \) with \( A \mathbf{u} = \mathbf{0} \). We define the function \( v_\mathbf{u} := \sum_{j=1}^n u_j \phi_j \). Then

\[ 0 = \sum_{j=1}^n A_{ij} u_j = \int_0^L v_\mathbf{u}'(x) \phi_i'(x) \, dx \quad (1 \leq i \leq n), \]

and furthermore

\[ 0 = \sum_{i=1}^n u_i \int_0^L v_\mathbf{u}'(x) \phi_i'(x) \, dx = \int_0^L (v_\mathbf{u}')^2 \, dx, \]

where the last equation implies \( v_\mathbf{u}' = 0 \), which means that \( v_\mathbf{u} \) has to be constant. Due to the boundary condition \( v_\mathbf{u}(0) = 0 \), it can only be zero: \( v_\mathbf{u} = 0 \). \( \square \)

This shows that the system (3.12), and thus problem (3.8), has a unique solution.

**Linear finite elements**

Now the time has come to exemplify the finite element method with a concrete choice of ansatz functions \( \phi_i \). For this purpose, we divide the interval \([0, L]\) into

\[ 0 = x_0 < x_1 < \ldots < x_n = L, \tag{3.13} \]

where the nodes \( x_i \) need not be equidistant. The ansatz space \( S \) shall be the set of all functions \( v \) satisfying

\[ v \in C^0([0, L]), \quad v|_{[x_{i-1}, x_i]} \text{ is linear}, \quad v(0) = 0 \text{ (essential boundary condition)}. \]

Obviously, \( S \) is a linear subspace of the previously defined \( V \) (cf. set 3.6). As a basis of \( S \), we choose the

**hat functions** \( \phi_i \) with \( \phi_i(x_j) = \delta_{ij} \quad (1 \leq i, j \leq n) \),

which are centered in the nodes \( x_i \) and linear in between. Because of the Neumann boundary condition, a"half hat" has been added on the right hand side (cf. figure 3.1). The set \( \{ \phi_i : 1 \leq i \leq n \} \) is called a **nodal basis**, since the coefficient \( T_j \) is just the value of the discrete solution in node \( x_j \):

\[ T_S(x_j) = \sum_{i=1}^n T_i \phi_i(x_j) = T_j \]
3.1 Finite element method

Assuming this basis, the task is to calculate stiffness matrix (3.10) and right hand side (3.11) for a previously fixed grid (3.13). Due to the local support of the hat functions, we will obtain a sparse—more precisely, a tridiagonal—matrix $A$:

$$A_{ij} = 0 \quad \text{for } |i - j| > 1$$

In addition to that, the non-zero entries can be computed easily because of the simple structure of the basis functions. How the computation is actually performed will be the subject of the following two paragraphs. But before we proceed, we fix two important advantages of FEM in general:

- The choice of basis functions with a local support yields a sparse matrix.
- Their shape permits an easy determination of the non-zero entries.

Global procedure: Let the index $i$ be fixed. The $i$-th row of the stiffness matrix $A$ features only three non-vanishing entries:

$$A_{i,i-1} = \int_0^L \phi'_i(x) \phi'_{i-1}(x) \, dx, \quad A_{i,i} = \int_0^L \phi'_i(x) \phi'_i(x) \, dx, \quad A_{i,i+1} = \int_0^L \phi'_i(x) \phi'_{i+1}(x) \, dx$$

Here, the individual basis functions are given by:

$$\phi_i(x) = \begin{cases} 
1 - (x_i - x)/(x_i - x_{i-1}) & \text{for } x_{i-1} \leq x \leq x_i \\
1 - (x - x_i)/(x_{i+1} - x_i) & \text{for } x_i < x \leq x_{i+1} \\
0 & \text{otherwise}
\end{cases}$$

Sometimes, the $\phi_i$ are denoted as global shape functions.

With the element lengths $h_i := x_i - x_{i-1}$, the evaluation of the integrals results in

$$A_{i,i-1} = -\frac{1}{h_i^2}, \quad A_{i,i} = \frac{1}{h_i} + \frac{1}{h_{i+1}}, \quad A_{i,i+1} = -\frac{1}{h_{i+1}}.$$
Hence, we finally obtain the stiffness matrix:

\[
A = \begin{bmatrix}
\frac{1}{h_1} & -\frac{1}{h_2} & 0 & \cdots & 0 \\
-\frac{1}{h_2} & \frac{1}{h_2} + \frac{1}{h_3} & -\frac{1}{h_3} & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
-\frac{1}{h_{n-1}} & \frac{1}{h_{n-1}} + \frac{1}{h_n} & -\frac{1}{h_n} & \cdots & 0 \\
-\frac{1}{h_n} & \frac{1}{h_n} & \cdots & \cdots & \cdots
\end{bmatrix}
\]  \tag{3.14}

The computation of the right hand side (3.11) is done in an analog way. There, again, the integrations span at most two subintervals. In case of a complicated function \( f \), an appropriate quadrature formula is needed.

**Local procedure:** In terms of practical applications, another approach to the determination of the stiffness matrix is usually preferred: It turns out that all the elements \([x_{i-1}, x_i] \) yield, apart from a transformation, the same contribution to \( A \).

The solution has been written as \( T_S = \sum_{i=1}^{n} T_i \phi_i \) with coefficients \( T_i \). Because of the small support of the basis functions, we can represent \( T_S \) inside the element \([x_{i-1}, x_i] \) in the following way:

\[
T_S|[x_{i-1}, x_i](x) = T_{i-1} \phi_{i-1}(x) + T_i \phi_i(x)
\]

There is no contribution from the other basis functions. Now, the interval \([x_{i-1}, x_i] \) is mapped to \([0, 1] \) using

\[
\xi = \frac{x - x_{i-1}}{h_i}.
\]  \tag{3.15}

On the unit interval, we define the two *local shape functions*

\[
N_1(\xi) := 1 - \xi, \quad N_2(\xi) := \xi, \quad 0 \leq \xi \leq 1.
\]

These functions \( N_1 \) and \( N_2 \) can be considered as standardized contributions of \( \phi_{i-1} \) and \( \phi_i \) within element \( i \) (cf. figure 3.2). We can decompose the integral expressions \( A_{ij} \) with respect to the grid (3.13):

\[
A_{ij} = \int_0^L \phi'_i(x) \phi'_j(x) \, dx = \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} \phi'_i(x) \phi'_j(x) \, dx = \sum_{k=1}^{n} A^{(k)}_{ij}
\]  \tag{3.16}

Each \( A^{(k)} \) contains only the contribution of the \( k \)-th element, i.e. of the interval \([x_{k-1}, x_k] \).

For this reason, the \( A^{(k)} \) are called *element stiffness matrices*, whilst \( A \) is termed *overall*
3.1 Finite element method

Figure 3.2: The local shape functions on \([x_{i-1}, x_i]\) and \([0, 1]\) respectively

**stiffness matrix.** Apparently, all but four entries of \(A^{(k)}\) are zero, which is the reason why we identify the element stiffness matrices with 2-by-2 blocks:

\[
A^{(k)} = \begin{bmatrix}
\int_{x_{k-1}}^{x_k} (\phi'_{k-1})^2 \, dx & \int_{x_{k-1}}^{x_k} \phi'_{k-1} \phi'_k \, dx \\
\int_{x_{k-1}}^{x_k} \phi'_k \phi'_{k-1} \, dx & \int_{x_{k-1}}^{x_k} (\phi'_k)^2 \, dx
\end{bmatrix}
\]

At this point, the transformation (3.15) comes into play. For instance, it holds:

\[
\int_{x_{k-1}}^{x_k} (\phi'_{k-1})^2 \, dx = \int_0^1 \left( \frac{dN_1}{d\xi} \frac{dN_1}{d\xi} \right) h_k \, d\xi = h_k \int_0^1 \left( N'_1(\xi) \frac{1}{h_k} \right)^2 \, d\xi
\]

And thus we find:

\[
A^{(k)} = \frac{1}{h_k} \begin{bmatrix}
\int_0^1 (N'_1)^2 \, d\xi & \int_0^1 N'_1 N'_2 \, d\xi \\
\int_0^1 N'_2 N'_1 \, d\xi & \int_0^1 (N'_2)^2 \, d\xi
\end{bmatrix}
\]  

(3.17)

This result is highly interesting, since the occurring integrals of the local shape functions are *independent* of the current element \(k\). We need to compute them only once, and multiply with the factors \(1/h_k\) afterwards in order to get the individual element stiffness matrices. Evaluation of (3.17) yields:

\[
A^{(k)} = \frac{1}{h_k} \begin{bmatrix}
1 & -1 \\
-1 & 1
\end{bmatrix}
\]  

(3.18)

Now that we know all the summands on the right hand side of (3.16), we build up the overall stiffness matrix. This process is named *assembly* or *compilation*:

\[
A = \begin{bmatrix}
\frac{1}{h_1} & -\frac{1}{h_1} \\
-\frac{1}{h_1} & \frac{1}{h_1}
\end{bmatrix} + \begin{bmatrix}
0 & 0 & 0 \\
0 & \frac{1}{h_2} & -\frac{1}{h_2} \\
0 & -\frac{1}{h_2} & \frac{1}{h_2}
\end{bmatrix} + \ldots
\]
In comparison to the result from the global approach (cf. 3.14), matrix (3.19) features one additional row and column. This is due to the fact that we have not accounted for the essential boundary condition \( T(x = 0) = 0 \) yet. The local procedure still involves a basis function \( \phi_0 \) with a corresponding coefficient \( T_0 \). This one must be set to zero explicitly, e.g. by cancelling the first row and column in (3.19).

According to the principle explained above, the right hand side vector (3.11) is calculated as a sum of element contributions:

\[
b^{(k)} = \lambda \left( \begin{array}{c}
\int_{x_{k-1}}^{x_k} f(x) \phi_{k-1}(x) \, dx \\
\int_{x_{k-1}}^{x_k} f(x) \phi_k(x) \, dx
\end{array} \right) = \lambda \left( \begin{array}{c}
\frac{1}{h_k} \int_0^1 f(x_{k-1} + \xi h_k) N_1(\xi) \, d\xi \\
\frac{1}{h_k} \int_0^1 f(x_{k-1} + \xi h_k) N_2(\xi) \, d\xi
\end{array} \right). \tag{3.20}
\]

One possibility to approximate the integrals in (3.20) is by linear interpolation of \( f \). We take the nodes of grid (3.13) as supporting points and define \( f_i := f(x_i) \). Then it holds:

\[
b^{(k)} \approx h_k \lambda \left[ \begin{array}{c}
\frac{1}{h_k} \int_0^1 N_1^2(\xi) \, d\xi \\
\frac{1}{h_k} \int_0^1 N_1 N_2(\xi) \, d\xi
\end{array} \right] \left( \begin{array}{c}
f_{k-1} \\
f_k
\end{array} \right)
\]

The 2-by-2 matrix is called element mass matrix. Its evaluation yields:

\[
b^{(k)} \approx \frac{h_k}{6} \lambda \left[ \begin{array}{cc}
2 & 1 \\
1 & 2
\end{array} \right] \left( \begin{array}{c}
f_{k-1} \\
f_k
\end{array} \right)
\]

Given this result, the vector \( \mathbf{b} \) can be assembled by the individual \( b^{(k)} \).

**Physical interpretation**

The method of finite elements has divided the metal rod into \( n \) parts. Each of them exhibits a linear temperature curve. In figure 3.3, we can spot the heat flux

\[
\dot{q}_{in} = -\lambda \frac{T_i - T_{i-1}}{h_i}
\]
entering temperature node $T_i$ from the left, whereas

$$\dot{q}_{\text{out}} = -\lambda \frac{T_{i+1} - T_i}{h_{i+1}}$$

is leaving to the right. Hence, the net heat flux out of node $i$ is:

$$\dot{q}_{\text{out}} - \dot{q}_{\text{in}} = -\lambda \frac{T_{i-1}}{h_i} + \left( \frac{\lambda}{h_i} + \frac{\lambda}{h_{i+1}} \right) T_i - \lambda \frac{T_{i+1}}{h_{i+1}} \tag{3.21}$$

Aside from the constant thermal conductivity, expression (3.21) is just the left part of the $i$-th equation in system (3.12). For this reason, it has to be equal to the product of $\lambda$ and the right hand side (3.11):

$$\dot{q}_{\text{out}} - \dot{q}_{\text{in}} = \lambda b_i = \int_0^L f(x) \phi_i(x) \, dx \tag{3.22}$$

Here, $\phi_i$ acts as a weighting function for the heat generation $f$ (cf. figure 3.3). From a physical point of view, the balance equation (3.22) tells us that the heat generation "input" from $[x_{i-1}, x_{i+1}]$ to node $i$ has to be equal to its net heat loss by diffusion. This fact is known as conservation of energy.

### 3.1.2 Application to the posed problem

We remember the governing PDE (2.4) from our model for the heat conduction within ceramic blocks:

$$\text{div}(\lambda(x)\nabla T(x)) = 0 \quad \forall x \in \Omega = (x_1, x_2) \times (y_1, y_2) \times (z_1, z_2)$$

The set $\Omega$ characterizes the whole block and comprises both the air cavities and the brick material itself. The boundary $\partial\Omega$ is decomposed with respect to the boundary conditions:

$$\partial\Omega = \Gamma_D \cup \Gamma_N \quad \land \quad \Gamma_D \cap \Gamma_N = \emptyset,$$
where \( \Gamma_D \) denotes the subset of \( \partial\Omega \) on which the Dirichlet boundary conditions have been declared and \( \Gamma_N \) corresponds to the homogeneous Neumann boundary condition, accordingly. The belonging values are given by

\[
T(x) = T_w(x) \quad \forall x \in \Gamma_D
\]

and

\[
\frac{\partial T}{\partial n}(x) = 0 \quad \forall x \in \Gamma_N.
\]

**Transition to the weak formulation:** In line with the one-dimensional example from the previous subsection, we commence with multiplication by an arbitrary test function \( v \) and subsequent integration over the domain \( \Omega \):

\[
\int_{\Omega} v \, \text{div}(\lambda \nabla T) \, dx = 0 \quad \forall v \in V
\]  \hspace{1cm} (3.25)

This time, *Green's formula*, which can be considered a multidimensional integration by parts, helps us in the further transformation:

\[
\int_{\Omega} v \, \text{div}(\lambda \nabla T) \, dx = \int_{\partial\Omega} v \lambda \frac{\partial T}{\partial n} \, ds - \int_{\Omega} \lambda \langle \nabla T, \nabla v \rangle \, dx = -\int_{\Omega} \lambda \langle \nabla T, \nabla v \rangle \, dx
\]  \hspace{1cm} (3.26)

where the surface integral is equal to zero due to (3.24) and the fact that we demand the test function \( v \) to vanish on \( \Gamma_D \) where the Dirichlet boundary conditions have been declared. Substituting (3.26) into (3.25), we obtain:

\[
\int_{\Omega} \lambda \langle \nabla T, \nabla v \rangle \, dx = 0 \quad \forall v \in V
\]

In order to determine the ansatz space \( V \), from which the test functions \( v \) are taken, we once again require the concept of weak derivatives and Sobolev spaces respectively. Now, in three space dimensions instead of one . . .

**Definition:** A function \( v : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R} \) with \( \int_{\Omega} v^2(x) \, dx < \infty \) holds a weak derivative \( w_k \) with respect to \( x_k \) \((k = 1, 2, 3)\), if there exists a function \( g : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R} \) with \( \int_{\Omega} g^2(x) \, dx < \infty \), so that

\[
\int_{\Omega} g(x) \beta(x) \, dx = -\int_{\Omega} v(x) \frac{\partial \beta}{\partial x_k} \, dx
\]

holds for all \( \beta \in C^\infty(\Omega) \) with \( \beta = 0 \) on \( \partial\Omega \). In this case, we set \( w_k = g \). \( \Box \)

The *Sobolev space* \( H^1(\Omega) \) shall be the linear space of functions \( v : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R} \) for which there exist all weak derivatives \( w_k \) \((k = 1, 2, 3)\) in the sense of the above definition.
This simple generalization to more than one space dimension allows us to choose the ansatz space $V$:

$$V := \{ v \in H^1(\Omega) \text{ with } v = 0 \text{ on } \Gamma_D \}$$

The inhomogeneous Dirichlet boundary condition (3.23) makes up an essential difference to the one-dimensional example (3.1) from the subsection about FEM basics. We can account for it by shifting the ansatz space for the temperature function $T$ by $T_w$. Then, the formulation of the problem may be written as:

Find $T \in T_w + V$, so that

$$\int_\Omega \lambda (\nabla T, \nabla v) \, dx = 0 \quad \forall v \in V.$$  \hspace{1cm} (3.27)

**Remark:** Strictly speaking, the function $T_w$ is defined only on $\Gamma_D$, of course. In (3.27), it is implicitly assumed that $T_w$ has been extended to the whole domain $\Omega$. In doing so, one has to make sure that

$$T_w + V = \{ v \in H^1(\Omega) \text{ with } v = T_w \text{ on } \Gamma_D \}$$

holds. This can be guaranteed by taking a continuous and bounded function for the shift. Later on, in chapter 3.4.1, we will make a special choice for $T_w$ and use it as an initial guess in the multigrid algorithm.

**Galerkin projection:** In terms of a concrete discretization of problem (3.27), the domain $\Omega$ will now be subdivided into elements (cf. the introduction of grid (3.13) in the one-dimensional case). To simplify matters, we suppose a decomposition of the cuboid $\Omega$,

$$\overline{\Omega} = \bigcup_k W_k, \quad \text{int}(W_m) \cap \text{int}(W_n) = \emptyset \quad (m \neq n),$$

into cubes $W_k$, which satisfies the three conditions:
All cubes are of the same size $h \times h \times h$.

The boundary $\partial \Omega$ is exactly resolved by them.

The function $\lambda(x)$, defined in (2.5), is constant within the individual cubes $W_k$.\(^1\)

Then, all the cubes $W_k$ can easily be mapped to the norm element (depicted in figure 3.4) by a translation $\tau_k \in \mathbb{R}^3$:

$$\xi = \tau_k(x) := x - \tau_k$$

(Projection $\tau_k$ is just the position vector of the lower left front corner of the $k$-th element.)

By means of the functions $\tilde{N}_0$ and $\tilde{N}_1$, defined by

$$\tilde{N}_0(\cdot) := 1 - \frac{1}{h}(\cdot), \quad \tilde{N}_1(\cdot) := \frac{1}{h}(\cdot),$$

and using the node numbering shown in figure 3.4, we set up the eight local shape functions $N_i (i = 0, \ldots, 7)$ on the norm element as tensor products:

$$N_{4d_2+2d_1+d_0}(\xi_0, \xi_1, \xi_2) = \tilde{N}_{d_0}(\xi_0) \cdot \tilde{N}_{d_1}(\xi_1) \cdot \tilde{N}_{d_2}(\xi_2) \quad \forall d_0, d_1, d_2 \in \{0, 1\} \quad (3.29)$$

Remark: Since it is not possible to print the graph of the trilinear functions (3.29) on a sheet of paper (they map $[0, h]^3$ to $[0, 1]$), their two-dimensional analogon is displayed in figure 3.5. For the one-dimensional case, the shape functions on the unit interval could already be seen in the right part of figure 3.2.

Figure 3.5: The four bilinear ansatz functions on the unit square

The next task is to calculate the stiffness matrix of an element $e^{(k)} = \tau_k + [0, h]^3$:

$$A_{ij}^{(k)} = \int_{e^{(k)}} \lambda^{(k)} \langle \nabla (N_i \circ t_k)(x), \nabla (N_j \circ t_k)(x) \rangle \, dx$$

$$= \lambda^{(k)} \int_{[0,h]^3} \langle \nabla N_i(\xi), \nabla N_j(\xi) \rangle \, d\xi$$

\(^1\)In general, the grid structure will not coincide with the material transitions. Chapter 3.4.2 explains what to do in this case.
⇒ \[ A^{(k)} = \frac{\lambda^{(k)} h}{12} \begin{bmatrix}
4 & 0 & 0 & -1 & 0 & -1 & -1 & -1 \\
0 & 4 & -1 & 0 & -1 & 0 & -1 & -1 \\
0 & -1 & 4 & 0 & -1 & 0 & -1 & -1 \\
-1 & 0 & 0 & 4 & -1 & -1 & -1 & 0 \\
0 & -1 & -1 & -1 & 4 & 0 & 0 & -1 \\
-1 & 0 & -1 & -1 & 0 & 4 & -1 & 0 \\
-1 & -1 & 0 & -1 & 0 & -1 & 4 & 0 \\
-1 & -1 & -1 & 0 & -1 & 0 & 0 & 4
\end{bmatrix} \]

In principle, it would be possible to compile the overall stiffness matrix \( A \) from the individual \( A^{(k)} \). We refrain from doing so and keep everything on the element level, since the computation of the residuals \( \text{res}_i \) in the multigrid method (see chapter 3.2) involves only the element stiffness matrices of the eight cubes in immediate vicinity of the \( i \)-th node. Thus, with regard to the algorithm, it makes no sense to assemble the complete matrix \( A \).

**Remark:** In case of a non-vanishing right hand side function \( f \) in the original PDE, the weak formulation features the expression

\[
\int_{\Omega} f(x) v(x) \, dx,
\]

and with respect to the chosen discretization, the integrals

\[
b^{(k)}_i = \int_{e^{(k)}} f(x) (N_i \circ t_k)(x) \, dx \quad (i = 0, \ldots, 7) \tag{3.31}
\]

need to be evaluated. For this purpose, \( f \) is approximated by trilinear interpolation: Denoting its values at the corners of the \( k \)-th element by \( f^{(k)}_0, \ldots, f^{(k)}_7 \) (according to the node numbering introduced in figure 3.4), we replace \( f(x) \) in (3.31) with

\[
\sum_{j=0}^{7} f^{(k)}_j (N_j \circ t_k)(x).
\]

Thus, we obtain:

\[
b^{(k)}_i = \sum_{j=0}^{7} f^{(k)}_j \int_{e^{(k)}} (N_j \circ t_k)(x) (N_i \circ t_k)(x) \, dx = \sum_{j=0}^{7} f^{(k)}_j \int_{[0,h]^3} N_j(\xi) N_i(\xi) \, d\xi
\]
With the mass matrix $M_{ij} := \int_{[0,h]^3} N_j(\xi) N_i(\xi) \, d\xi,$

$$M = \left( \frac{h}{6} \right)^3 \begin{bmatrix}
8 & 4 & 4 & 2 & 4 & 2 & 2 & 1 \\
4 & 8 & 2 & 4 & 2 & 4 & 1 & 2 \\
4 & 2 & 8 & 4 & 2 & 1 & 4 & 2 \\
2 & 4 & 4 & 8 & 1 & 2 & 2 & 4 \\
4 & 2 & 2 & 1 & 8 & 4 & 4 & 2 \\
2 & 4 & 1 & 2 & 4 & 8 & 2 & 4 \\
2 & 1 & 4 & 2 & 4 & 2 & 8 & 4 \\
1 & 2 & 2 & 4 & 2 & 4 & 4 & 8
\end{bmatrix},$$  \hspace{1cm} (3.32)

we can write

$$b^{(k)} = M f^{(k)} \quad \text{with} \quad f^{(k)} = (f_0^{(k)}, \ldots, f_7^{(k)})^T$$

for the contribution of $f$ to the discretized right hand side.

### 3.1.3 Application to the generalized problem

Let us suppose we want to solve the Dirichlet problem

$$- \Delta u(\xi) = f(\xi) \quad \forall \xi \in P$$

$$u(\xi) = 0 \quad \forall \xi \in \partial P$$  \hspace{1cm} (3.33)

on a (complex-shaped) physical domain $P$ featuring curved boundary surfaces. We assume that we have constructed a bijective $C^1$-function

$$\psi : \Omega \rightarrow P, \quad x \mapsto \xi := \psi(x)$$

mapping a (cuboidal) computational domain $\Omega$ to $P$: $\psi(\Omega) = P$ (cf. figure 3.6). The inverse $\psi^{-1}$ shall exist and be a $C^1$-function on its part. Thus, in particular:

$$\det J_\psi(x) \neq 0 \quad \forall x \in \Omega,$$

where $J_\psi = (\partial \psi_i/\partial x_j)_{1 \leq i,j \leq 3}$ denotes the Jacobian matrix. The function $\psi$ provides a parametrization of $P$, i.e. a generalized coordinate system matching the curved boundary. It is a simple exercise to transform (3.33) into the corresponding weak formulation:

Find $u \in V^* := \{ \tilde{v} \in H^1(P) : \tilde{v} = 0 \text{ on } \partial P \}$, so that

$$\int_{P} \langle \nabla u, \nabla \tilde{v} \rangle \, d\xi = \int_{P} f \tilde{v} \, d\xi \quad \forall \tilde{v} \in V^*. \hspace{1cm} (3.34)$$

Our objective is to write down equation (3.34) with regard to $\Omega$ instead of $P$. In order to achieve this, we need to make use of the chain rule

$$\nabla_x(g(\psi(x))) = J_{\psi}^T (\nabla_\xi g)(\psi(x))$$
and the integral transformation
\[ \int_{\mathbf{P}} g(\xi) \, d\xi = \int_{\Omega} g(\psi(x)) |\det J_{\psi}(x)| \, dx, \]
where \( g \) is a function mapping \( \mathbb{R}^3 \) to \( \mathbb{R} \). Substituting these two formulas into (3.34), we find:

\[
\begin{align*}
\int_{\Omega} \langle (\nabla_{\xi} u)(\psi(x)), (\nabla_{\xi} \tilde{v})(\psi(x)) \rangle |\det J_{\psi}(x)| \, dx &= \int_{\Omega} f(\psi(x)) \tilde{v}(\psi(x)) |\det J_{\psi}(x)| \, dx \\
\int_{\Omega} \langle J_{\psi}^{-T} \nabla x \hat{u}(x), J_{\psi}^{-T} \nabla x \hat{v}(x) \rangle |\det J_{\psi}(x)| \, dx &= \int_{\Omega} \hat{f}(x) \hat{v}(x) |\det J_{\psi}(x)| \, dx \\
\int_{\Omega} \langle J_{\psi}^{-1}(J_{\psi}^{-1})^T \nabla \hat{u}(x), \nabla \hat{v}(x) \rangle |\det J_{\psi}(x)| \, dx &= \int_{\Omega} \hat{f}(x) \hat{v}(x) |\det J_{\psi}(x)| \, dx
\end{align*}
\]
(3.35)

The hats in the last two lines are an abbreviation for the concatenation with \( \psi \). Notice that the matrix \( J_{\psi}^{-1}(J_{\psi}^{-1})^T \) in (3.35) is symmetric positive definite. Since \( |\det J_{\psi}(x)| > 0 \), equation (3.35) shows us that problem (3.34) is—in principle—of the following type:

Find \( T \in V := \{ v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega \} \), so that

\[ \int_{\Omega} \langle U(x) \nabla T(x), \nabla v(x) \rangle \, dx = \int_{\Omega} f(x) v(x) \, dx \quad \forall v \in V, \quad \tag{3.36} \]

where \( U(x) \in \mathbb{R}^{3,3} \) is a symmetric positive definite matrix. We would have obtained exactly the same problem (3.36) by deriving the weak formulation for:

\[
\begin{align*}
- \text{div}(\Lambda(x) T(x)) &= f(x) \quad \forall x \in \Omega \\
T(x) &= 0 \quad \forall x \in \partial\Omega,
\end{align*}
\]
(3.37)
where $\Lambda(x) \in \mathbb{R}^{3,3}$ is a (symmetric) second order tensor representing an anisotropic thermal conductivity. This underlines, what was already mentioned in chapter 2, that problems (3.33) and (3.37) are closely related.

**Computation of the stiffness matrix**
Assuming the same discretization as before, i.e. a decomposition of $\Omega$ into cube elements of size $h \times h \times h$ with trilinear ansatz functions, we will now determine the element stiffness matrices that belong to (3.36). To this end, the symmetric matrix $U(x)$ is approximated by a constant $U^{(k)}$ within each single element $e^{(k)}$:

$$U(x)|_{e^{(k)}} \rightarrow U^{(k)} = \begin{bmatrix} U^{(k)}_{11} & U^{(k)}_{12} & U^{(k)}_{13} \\ U^{(k)}_{12} & U^{(k)}_{22} & U^{(k)}_{23} \\ U^{(k)}_{13} & U^{(k)}_{23} & U^{(k)}_{33} \end{bmatrix}$$

The replacement may be done by evaluation at the center of the corresponding element or by taking the average values of the entries at the eight corners of the cell. With the aid of this simplification, we can easily compute the stiffness matrix:

$$A_{ij}^{(k)} = \int_{[0,h]^3} \langle U^{(k)} \nabla N_i(\xi), \nabla N_j(\xi) \rangle \, d\xi$$

$$= U^{(k)}_{11} \int_{[0,h]^3} \frac{\partial N_i}{\partial \xi_1} \frac{\partial N_j}{\partial \xi_1} \, d\xi + U^{(k)}_{12} \int_{[0,h]^3} \frac{\partial N_i}{\partial \xi_2} \frac{\partial N_j}{\partial \xi_2} \, d\xi +$$

$$U^{(k)}_{13} \int_{[0,h]^3} \left( \frac{\partial N_i}{\partial \xi_3} \frac{\partial N_j}{\partial \xi_3} + \frac{\partial N_i}{\partial \xi_3} \frac{\partial N_j}{\partial \xi_3} \right) \, d\xi + U^{(k)}_{23} \int_{[0,h]^3} \left( \frac{\partial N_i}{\partial \xi_2} \frac{\partial N_j}{\partial \xi_2} + \frac{\partial N_i}{\partial \xi_2} \frac{\partial N_j}{\partial \xi_2} \right) \, d\xi$$

$$= U^{(k)}_{11} B_{11}^{(k)} + U^{(k)}_{22} B_{22}^{(k)} + U^{(k)}_{33} B_{33}^{(k)} + U^{(k)}_{12} B_{12}^{(k)} + U^{(k)}_{13} B_{13}^{(k)} + U^{(k)}_{23} B_{23}^{(k)}$$

(3.38)

**Remark:** The matrices $B^{lm}$ are independent of the current element and can be computed a priori. You can find them in appendix A. It is important to notice that the determination of $A^{(k)}$ according to (3.38) involves a huge amount of identical products so that it is possible to keep the computational effort to a minimum.

### 3.2 Multigrid method

In the preceding chapter, we have learned how to transform a PDE into a system of linear equations. Now, our task is to solve this system. The following section starts with an extensive analysis of a conventional iterative method that could be used for this purpose in principle. By means of the investigation, we will realize an important insufficiency of the method and thus motivate the usage of multiple grids.
3.2 Multigrid method

3.2.1 Motivation

Just like in the section about FEM, it is instructive to start with the examination of a one-dimensional model problem. As an example, we consider once more the steady-state temperature distribution in a long uniform rod. It was described by

\[-\lambda T''(x) = f(x) \quad \forall x \in (0, L). \quad (3.39)\]

This time, the boundary conditions shall be \( T(0) = T(L) = 0 \). Instead of finite elements, we apply the finite difference method. For that purpose, the domain \( \Omega = (0, L) \) is partitioned into \( n \) subintervals by introducing equidistant nodes \( x_i = ih \), where \( h = L/n \). The corresponding grid is denoted \( \Omega^h \). Referring to the approximate solution in node \( x_i \) by \( T_i \) and substituting \( T''(x) \) with the second order approximation \( T(x - h) - 2T(x) + T(x + h) \), we obtain from (3.39) the following system of linear equations:

\[-\frac{T_{i-1} + 2T_i - T_{i+1}}{h^2} = \frac{1}{\lambda} f(x_i), \quad 1 \leq i \leq n - 1 \quad (3.40)\]

with \( T_0 = T_n = 0 \). Without loss of generality, we set \( \lambda = 1 \). The definitions

\[A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & \\ & & & & -1 & 2 \end{bmatrix}, \quad t = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_{n-2} \\ T_{n-1} \end{bmatrix}, \quad f = h^2 \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{n-2}) \\ f(x_{n-1}) \end{bmatrix}\]

allow us to write down the system (3.40) in a more compact form:

\[At = f \quad (3.41)\]

The matrix \( A \) is sometimes called Poisson matrix. We now turn to the question how (3.41) might be treated using a conventional iterative method. In this context, the approximation shall be denoted by \( \mathbf{v} \). An important measure for the quality of \( \mathbf{v} \) as an approximation to \( \mathbf{t} \) is the error

\[e := \mathbf{t} - \mathbf{v}. \quad (3.42)\]

Unfortunately, this quantity is as inaccessible as the exact solution itself. If we would know \( e \), we could immediately calculate \( \mathbf{t} \), since \( \mathbf{t} = \mathbf{v} + e \). Before we proceed, we introduce the residual

\[r := f - Av \quad (3.43)\]

as another measurement of how well \( \mathbf{v} \) approximates the solution \( \mathbf{t} \). Notice that \( r \), in contrast to \( e \), is a quantity which can be computed at any time during the iteration process. However, it is in general not true that a small residual implies a small error.
Rewriting the definition (3.43) as $A\mathbf{v} = \mathbf{f} - \mathbf{r}$ and subtracting this expression from (3.41), we find an interesting relationship between the error and the residual:

$$A\mathbf{e} = \mathbf{r}$$  \hspace{1cm} (3.44)

In the remainder of this subchapter, it will be called the residual equation. The next step consists in the derivation of the iterative method. For that purpose, the matrix $A$ is partitioned as

$$A = D - L - U,$$

where $D$ denotes the diagonal and $-L$ and $-U$ the strictly lower and upper triangular parts of $A$, respectively. The original equation (3.41) then becomes

$$(D - L - U)\mathbf{t} = \mathbf{f},$$

and further

$$D\mathbf{t} = (L + U)\mathbf{t} + \mathbf{f} \quad \Leftrightarrow \quad \mathbf{t} = D^{-1}(L + U)\mathbf{t} + D^{-1}\mathbf{f}.$$  \hspace{1cm} (3.45)

The last line suggests the following fixed point iteration:

$$\mathbf{v}^{(j+1)} = D^{-1}(L + U)\mathbf{v}^{(j)} + D^{-1}\mathbf{f}, \quad j \geq 0$$  \hspace{1cm} (3.46)

We commence with an initial guess $\mathbf{v}^{(0)}$ and compute the iterate $\mathbf{v}^{(1)}$ according to (3.46). After that, the next iterate $\mathbf{v}^{(2)}$ is determined from $\mathbf{v}^{(1)}$, and so on. These iteration sweeps are continued until (ideally) we obtain satisfactory convergence to the solution.

**Remark:** By now, it is not evident whether our hope for convergence is justified or not. However, later we will provide a proof that the method works fine.

What is actually done by (3.46), becomes clearer when taking a look at its corresponding component form:

$$v_i^{(j+1)} = \frac{1}{2}(v_i^{(j)} + v_{i+1}^{(j)} + h^2 f_i), \quad 1 \leq i \leq n - 1$$  \hspace{1cm} (3.47)

From (3.47), we can learn that the derived iterative scheme solves the $i$-th equation of (3.41) for the $i$-th unknown $u_i$. All components of $\mathbf{v}^{(j)}$ are updated “at the same time”, meaning that the new values $v_i^{(j+1)}$ are not used as soon as they are available; the complete iteration step has to be finished first. For this reason, the above scheme—known as Jacobi method—is sometimes also termed the simultaneous displacement method.

There is a simple but important possibility to modify (3.47) and thus generate an entire family of iterations called the weighted or relaxed Jacobi method. To this end, the right
hand side in (3.47) is considered an intermediate value only. The new iterate is then computed as the weighted average of that intermediate value and the old \( v_i^{(j)} \):

\[
v_i^{(j+1)} = (1 - \omega) v_i^{(j)} + \omega \frac{1}{2} (v_{i-1}^{(j)} + v_{i+1}^{(j)} + h^2 f_i), \quad \omega \in \mathbb{R}
\]

Notice that \( \omega = 1 \) yields the original Jacobi iteration. In matrix form, the weighted Jacobi method is given by

\[
v^{(j+1)} = [(1 - \omega) I + \omega D^{-1}(L + U)] v^{(j)} + \omega D^{-1} f \tag{3.48}
\]

or equivalently

\[
v^{(j+1)} = v^{(j)} + \omega D^{-1} r^{(j)}, \tag{3.49}
\]

where \( r^{(j)} \) denotes the residual that belongs to the corresponding approximation \( v^{(j)} \). The matrix \( (1 - \omega) I + \omega D^{-1}(L + U) \) in (3.48) is termed iteration matrix.

**Remark:** Recalling definition (3.42) of the error \( e = t - v \) and the residual equation \( A e = r \), we can write:

\[
t = v + A^{-1} r
\]

From this point of view, the matrix \( \omega D^{-1} \) in (3.49) can be interpreted as an approximation to the inverse of \( A \).

Until now, it is not clear whether the obtained iteration methods will converge or not. In addition to that question, the influence of the parameter \( \omega \) needs to be examined. We start our analysis by defining an abbreviation for the iteration matrix:

\[
P_\omega := (1 - \omega) I + \omega D^{-1}(L + U)
\]

With its aid the relaxation scheme (3.48) can be rewritten as

\[
v^{(j+1)} = P_\omega v^{(j)} + \omega D^{-1} f.
\]

We also note that, if the iteration converges, it does not change the exact solution, i.e.

\[
t = P_\omega t + \omega D^{-1} f.
\]

By subtracting the last two equations, we find \( e^{(j+1)} = P_\omega e^{(j)} \), and thus

\[
e^{(j)} = P_\omega^j e^{(0)} \tag{3.50}
\]

This is an interesting starting point for further examination, since (3.50) provides information about how the error develops during the iteration process. If we manage to determine the eigenvalues and corresponding eigenvectors of \( P_\omega \), a detailed convergence analysis will become possible. Fortunately, the complex looking iteration matrix \( P_\omega \) can be simplified:

\[
P_\omega = I - \frac{\omega}{2} A \tag{3.51}
\]
It is an easy exercise to show that (3.51) holds. We look up the eigenvalues $\lambda_k$ and belonging eigenvectors $w_k$ of the Poisson matrix $A$ in the literature:

$$\lambda_k(A) = 4 \sin^2 \left( \frac{k\pi}{2n} \right) \quad (1 \leq k \leq n - 1)$$

$$w_{k,i} = \sin \left( \frac{ki\pi}{n} \right) \quad (1 \leq k \leq n - 1)$$

Figure 3.7: The modes $w_k = \sin \left( \frac{k\pi}{n} \right)$, $0 \leq i \leq n$, with wave numbers $k = 1, 3, 6$

The vectors $w_k$ are called *discrete Fourier modes*, because their components are the sampling points of waves. The $k$-th mode consists of $k/2$ full sine waves. In this context, $k$ is termed *wave number*. Notice that the higher the wave number, the more oscillatory the mode (cf. figure 3.7).

Due to (3.51), the eigenvalues of $P_\omega$ are

$$\lambda_k(P_\omega) = 1 - \frac{\omega}{2} \lambda_k(A) = 1 - 2 \omega \sin^2 \left( \frac{k\pi}{2n} \right) \quad (1 \leq k \leq n - 1), \quad (3.52)$$

whereas the eigenvectors of $P_\omega$ are the same as the eigenvectors of $A$.

**Observation:** The smooth modes (modes with small wave numbers) always belong to eigenvalues close to 1, while the eigenvalues of the high-frequency modes exhibit a strong dependence on $\omega$. This circumstance is illustrated in figure 3.8.

Keeping this in mind, we now go back to equation (3.50):

$$e^{(j)} = P^j e^{(0)}$$

Since the set of eigenvectors $w_k$ of $A$ (and $P_\omega$ respectively) is a basis of $\mathbb{R}^{n-1}$, it is possible to represent the error in an initial guess in the form:

$$e^{(0)} = \sum_{k=1}^{n-1} c_k w_k, \quad c_k \in \mathbb{R}$$
3.2 Multigrid method

Figure 3.8: Eigenvalues of the matrix $P_\omega$ for different choices of $\omega$. The eigenvalues $\lambda_k(P_\omega)$ (see 3.52) are plotted as if the wave number $k$ were a continuous variable on $[0, n]$.

Substitution of this eigenvector expansion for $e^{(0)}$ into equation (3.50) yields:

$$e^{(j)} = \sum_{k=1}^{n-1} c_k P_\omega^j w_k = \sum_{k=1}^{n-1} c_k \lambda_k^j(P_\omega) w_k$$  \hspace{1cm} (3.53)

The second equality holds since $P_\omega w_k = \lambda_k(P_\omega) w_k$. With (3.53), we have finally obtained an eigenvector expansion for $e^{(j)}$—the desired instrument for analyzing the convergence properties of the weighted Jacobi method: We can see that $j$ iterations reduce the $k$-th mode of the initial error by a factor of $\lambda_k^j(P_\omega)$. The weighted Jacobi method is not able to "mix modes". It can only change the amplitudes of them. From (3.52) and figure 3.8, it is obvious that $0 < \omega < 1$ implies $|\lambda_k(P_\omega)| < 1$ for all $k$ and thus because of (3.53) convergence of the scheme for an arbitrary initial guess $v^{(0)}$. However, the eigenvalues belonging to the smooth modes are always close to 1, meaning a slow reduction of these modes in the error. For instance:

$$\lambda_1 = 1 - 2\omega \sin^2 \left(\frac{\pi}{2n}\right) = 1 - 2\omega \sin^2 \left(\frac{\pi h}{2L}\right) \approx 1 - \frac{\omega}{2L^2}$$

The finer the grid spacing $h$, the more iteration steps become necessary due to the bad convergence of the smooth components in the error. On the other hand, it is possible to choose values for $\omega$ so that the elimination of the oscillatory modes works fine independent of the meshsize $h$. In particular, setting $\omega = 2/3$ guarantees

$$|\lambda_k(P_\omega)| < \frac{1}{3}$$
for all high-frequency modes $w_k$ ($n/2 \leq k \leq n - 1$).

**Conclusion:** In general, we can expect that the first few iterations of the weighted Jacobi method will decrease the error rapidly. This is because of the effective reduction of its oscillatory components. Then, the convergence will become slower and slower due to the remaining low-frequency modes. This kind of selective elimination of the error components provides a smoothing effect on the shape of the error.

**First remark:** Although our observations were carried out for a special system of linear equations, they are also valid in a more general setup. The observed deficiency is typical for all classical iterative solvers.

**Second remark:** The relaxed Jacobi method provides the smoothing effect described above not only for the error, but also for the residual. To show this, we make use of the residual equation $Ae = r$:

$$r^{(j)} = Ae^{(j)} = AP_\omega r^{(0)} = A(I - \frac{\omega}{2}A)^j e^{(0)} = (I - \frac{\omega}{2}A)^j r^{(0)} = P_\omega r^{(0)}$$

The last equation means that the iteration matrix $P_\omega$ also applies for the residual.

**Multigrid method**

Now the multigrid method comes into play. Its basic idea is to use several coarser grids besides a fine one and thus overcome the convergence problems related to the low-frequency modes of the error: The key point of this strategy is that modes with small wave numbers appear more oscillatory (with respect to the density of sampling points!) when being represented on a coarser grid (cf. figure 3.9). For this reason, an iterative solver will eliminate the low-frequency error components more efficiently on a coarser grid than it would on the finest one.

![Figure 3.9: Mode with wave number 3 represented on grids $\Omega^{1/16}$ (left) and $\Omega^{1/8}$ (right)](image-url)

There are different designs in the multigrid method: For instance, one can start with a very coarse grid and successively generate improved initial guesses for iterative schemes.
on finer grids. To this end, we solve the system of linear equations associated with a widely spaced grid $\Omega^{h}$. Then, the obtained solution $v^{h}$ is transferred to a finer grid $\Omega^{h-1}$ by some interpolation operator that still has to be specified. Doing this, we get a good initial guess $\tilde{v}^{h-1}$ for an iterative solver on $\Omega^{h-1}$ (good because the low-frequency error components should be small since $v^{h}$ solved the system on $\Omega^{h}$). After some iteration sweeps on the grid $\Omega^{h-1}$, possible high-frequency error modes in $\tilde{v}^{h-1}$ should be damped so that we can interpolate again and thus create an initial guess for the next finer grid $\Omega^{h-2}$, and so on. The process is stopped as soon as the spacing has reached a sufficient degree of refinement. This concept of incorporating coarser grids to create improved initial guesses is the basis of a strategy termed nested iteration. We do not want to go into its details. However, it should be mentioned that – although this very simple approach already looks rather promising – there might be problems arising if smooth error components remain when ”descending” to finer and finer grid levels.

A more common example for the multigrid method is the so-called correction grid scheme: On a fine grid $\Omega^{h}$, iteration steps are performed until the error $e^{h}$ has become smooth and the convergence slow. Performing further iterations would not be wise since the low-frequency modes of the error $e^{h}$ cannot be eliminated efficiently on $\Omega^{h}$. We refer to the last iterate on $\Omega^{h}$ by $\hat{v}^{h}$. The corresponding residual $\hat{r}^{h}$ is now transferred to a coarse grid $\Omega^{2h}$ by some restriction operator that still has to be specified. We use the obtained vector $r^{2h}$ as the right hand side in the residual equation (3.44) on $\Omega^{2h}$: $A^{2h}e^{2h} = r^{2h}$.

(How to get the ”$\Omega^{2h}$ version” from the original matrix $A^{h}$ will be explained later.) With the zero vector as an initial guess, an iterative method for the residual equation on $\Omega^{2h}$ is started. This yields an approximation $\hat{e}^{2h}$ which can be transferred back to the grid $\Omega^{h}$ by interpolation (the result shall be denoted $\hat{\hat{e}}^{h}$) in order to correct (improve) the last iterate $\hat{v}^{h}$ there: $\hat{v}^{h} \leftarrow \hat{v}^{h} + \hat{\hat{e}}^{h}$. Of course, this principle can be embedded into a recursion, i.e. the result $\hat{e}^{2h}$ from the residual equation iteration on $\Omega^{2h}$ can be corrected on its part with aid of the residual equation on $\Omega^{4h}$, and so on. At each level of such a recursion, the correction step helps us to eliminate the low-frequency error components which could not be damped sufficiently by the iteration scheme on the respective level.

Remark: The multigrid method used in our program is of another type and will be presented later. In addition to principle differences, peano3d changes the spacing—due to technical reasons—by a factor of three instead of two when going from one grid level to the next. This means that a cube which is to be refined will be subdivided into 27 (and not eight) smaller cubes.

### 3.2.2 Communication between the grids

In the previous section, we were talking about restricting the residual to a coarser level, interpolating an approximation to a finer grid and about a so-called ”$\Omega^{2h}$ version” of the system matrix $A$. Until now, it is not clear how to perform the corresponding tasks of transferring information between the grid levels. In order to create a basic understanding of these processes, this part of the chapter gives simple one-dimensional examples for the interpolation and restriction operators in case of refinement by bisection.
The interpolation operator

Let us assume we hold a vector \( \mathbf{v}^{2h} \) containing values that are associated with a coarse grid \( \Omega^{2h} \). You might think of this vector as the error approximation on \( \Omega^{2h} \) in the coarse grid correction scheme from above. The task is to transfer \( \mathbf{v}^{2h} \) to a fine grid \( \Omega^{h} \) (see figure 3.10). At nodes existing on both levels, we can simply take the values from the coarse to the fine grid:

\[
v_h^{2h} = v_i^{2h}, \quad i = 1, 2, \ldots, n - 1
\]

(3.54)

If no special information is provided about \( \mathbf{v}^{2h} \), it is reasonable to use linear interpolation at the intermediate nodes in order to compute their belonging vector entries:

\[
v_h^{2h-1} = \frac{1}{2}(v_i^{2h} + v_i^{2h}), \quad i = 1, 2, \ldots, \frac{n}{2},
\]

(3.55)

where \( v_0^{2h} = v_n^{2h/2} = 0 \). The equations (3.54) and (3.55) can be rewritten in matrix vector form:

\[
\mathbf{v}^h = I_{2h}^h \mathbf{v}^{2h}, \quad I_{2h}^h \in \mathbb{R}^{n-1, \frac{n}{2}-1}
\]

The matrix \( I_{2h}^h \) is called interpolation or prolongation operator. In the case which is depicted in figure 3.10, we have \( n = 8 \) and

\[
I_{2h}^h = \frac{1}{2} \begin{bmatrix}
1 & 0 & 0 \\
2 & 0 & 0 \\
1 & 1 & 1 \\
0 & 2 & 0 \\
0 & 1 & 1 \\
0 & 0 & 2 \\
0 & 0 & 1
\end{bmatrix}
\]

The restriction operator

Now, the situation is vice versa: We are given a vector \( \mathbf{v}^h \) containing values with respect to the fine grid \( \Omega^h \). (You might think of this vector as the residual \( \mathbf{r}^h \) on \( \Omega^h \) in the coarse grid correction scheme, just before we communicate it to \( \Omega^{2h} \) for solving the residual...
3.2 Multigrid method

Figure 3.11: Full weighting restriction from $\Omega^h$ to $\Omega^{2h}$

equation there.) The task is to restrict $v^h$ to $v^{2h}$. A very simple approach to do this would be to skip the odd indeces in $v^h$ and take only the entries with even ones:

$$v_i^{2h} = v_{2i}^h, \quad i = 1, 2, \ldots, \frac{n}{2} - 1$$

Hence, the coarse grid vector gets its values directly from the corresponding fine grid points. This procedure is named *injection*. Of course, we lose information by using it since the values $v_{2i-1}^h$ are not taken into account. Sometimes, this is not desirable. Another possibility for restriction is the so-called *full weighting*:

$$v_i^{2h} = \frac{1}{4} (v_{2i-1}^h + 2v_{2i}^h + v_{2i+1}^h), \quad i = 1, 2, \ldots, \frac{n}{2} - 1 \quad (3.56)$$

Here, an entry of the coarse grid vector is a weighted average of the belonging fine grid value and its neighbors (see figure 3.11). Once again, we can write down the equations (3.56) in matrix vector form:

$$v^{2h} = I_h^{2h} v^h, \quad I_h^{2h} \in \mathbb{R}^{n-1,n-1}$$

The matrix $I_h^{2h}$ is termed *restriction* or *projection operator*. For $n = 8$, we would obtain:

$$v^{2h} = I_h^{2h} v^h = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \end{bmatrix} v^h$$

The interesting observation is that–apart from a scalar factor–the restriction operator is the transpose of the interpolation operator: $I_h^{2h} = c \left( I_{2h}^h \right)^T$.

**The coarse grid matrix $A^{2h}$**

This topic involves quite a lot of theory so that we refrain from a derivation. The textbook [Briggs 87] motivates the following (plausible) definition of the coarse grid operator $A^{2h}$:

$$A^{2h} = I_h^{2h} A^h I_{2h}^h \quad (3.57)$$
Remark: Using the system matrix $A$ from the one-dimensional example (3.41) as well as the interpolation and full weighting restriction operators from above for the case $n = 8$, we obtain:

$$A^{2h} = \frac{1}{4} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Notice that—apart from the scalar in front—$A^{2h}$ is a Poisson matrix just like $A^h$ was. The factor $1/4$ is no surprise since it can be explained by the double-sized grid spacing in $\Omega^{2h}$ and the fact that the approximation of the second derivative of the temperature entailed the square of the meshsize in the denominator (cf. 3.40).

### 3.2.3 Hierarchical generating system

In order to describe the current approximation of the solution during the iteration process, the *peano3d* algorithm makes use of a *hierarchical generating system* instead of a nodal basis. For reasons of clearness, figure 3.12 illustrates the principle by means of a one-dimensional element—an interval of length $3h$, which has been subpartitioned into three smaller intervals of size $h$. Besides the blue hat functions $\phi_i^h$ on the fine grid $\Omega^h$, we can see further ansatz functions $\phi_i^{3h}$ on the coarse grid $\Omega^{3h}$ (depicted in red). Assuming the $\phi_i^h$ as a (nodal) basis, we could represent an arbitrary piecewise-linear function $v$ (the green line in the lower part of the figure) by a unique linear combination:

$$v = \sum_{i=0}^{3} \tilde{q}_i^h \phi_i^h$$
3.2 Multigrid method

However, our program takes another approach: The set of the small hat functions $\phi^h_i$ is extended to a generating system by involving the additional hat functions $\phi^{3h}_3$, which hold a support that is three times wider than that of the $\phi^h_i$. They are centered at each third node of the fine grid $\Omega^h$. In this way, we lose the uniqueness in the representation of $v$. In exchange, we now have six degrees of freedom in total—four at the fine grid nodes (depicted as points) and two at the coarse grid nodes (depicted as circles). Notice the coexistence of degrees of freedom at nodes "living" on both the coarse and the fine level. Using the coefficients from figure 3.12, the function $v$ is described by:

$$v = \sum_{i=0}^{1} q_i^{3h} \phi_i^{3h} + \sum_{i=0}^{3} q_i^h \phi_i^h$$

Since all information about the functional characteristics of $v$ is contained in its values $v_i$, another representation on the interval is

$$v = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} = r_{3h} \begin{pmatrix} q_0^{3h} \\ q_1^{3h} \end{pmatrix} + \begin{pmatrix} q_0^h \\ q_1^h \\ q_2^h \\ q_3^h \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 2/3 & 1/3 \\ 1/3 & 2/3 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} q_0^{3h} \\ q_1^{3h} \end{pmatrix} + \begin{pmatrix} q_0^h \\ q_1^h \\ q_2^h \\ q_3^h \end{pmatrix}, \quad (3.58)$$

where we have used the linear interpolation operator in an element-wise form adapted to the trisection. In the context of (3.58), the coefficients $q_i^h$ are called hierarchical surpluses.

It is clear that the explained principle can be generalized to more than two grids by recursive refinement. This is what peano3d does. To build up the current approximation to the solution, the algorithm starts on a very coarse grid and then performs interpolation and accumulation of hierarchical surpluses in alternating order, until the finest level has been reached. Since the code works in three dimensions, cubes with trilinear ansatz functions (3.29) take the place of the intervals with hat functions from the example above; but the principle remains the same.

3.2.4 Sketch of a multigrid cycle in peano3d

The program peano3d solves a system of linear equations $At = b$ by taking the weighted Jacobi method (cf. 3.49),

$$\tilde{t}^{(j+1)} = \tilde{t}^{(j)} + \omega D^{-1} \left( b - A \tilde{t}^{(j)} \right) = \tilde{t}^{(j)} + \omega D^{-1} r^{(j)}, \quad (3.59)$$

as the underlying iterative method for a special kind of multigrid algorithm. In (3.59), the vector $\tilde{t}^{(j)}$ denotes the approximation to the solution $t$ in the $j$-th iteration, $\omega$ is the parameter of the relaxation and $D$ the diagonal of the system matrix $A$. The expression in brackets is the residual $r^{(j)}.$

The multigrid algorithm implemented in peano3d is of the so-called additive type. It performs smoothing steps on all levels of the grid simultaneously. The course of an individual iteration cycle looks as follows: Starting on a coarse grid $\Omega^{h\cdot 3p}$, information is
processed by a trilinear interpolation operator $I_{3h}^h$ to the next finer grid $\Omega^{h-3p-1}$ in the hierarchy. This procedure is repeated recursively, until the finest grid $\Omega^h$ has been reached, where we compute the residual $r_h$ by means of the nodal values $\tilde{t}_h$. Using a restriction operator $I_h^h$, the residual is transported to all coarser levels by recursion. As soon as $r_{h,3^m}$ ($m = 0, 1, \ldots, p$) is determined on grid $\Omega^{h-3^m}$, we can perform the corresponding iteration step (3.59) there. When this has been done on all levels, the recursion has ascended to the top again, and the multigrid cycle is complete.

**Remark:** There are two interesting things to notice in this setup: (a) Computations take place only on the finest level. The residuals on the coarser grids are not really calculated, but determined with aid of the restriction operator. (b) The updated node values are not used before a new cycle begins. In this sense, the additive multigrid method can be considered a simultaneous displacement method working on all grids at the same time.

![Figure 3.13: Information transport between two grid levels by interpolation and restriction, based on the hierarchical generating system (source: [Pögl 04])](image)

Figure 3.13 shows the weighting factors involved in the interpolation and restriction as they are implemented in the peano3d code—once again for the one-dimensional example case. In the picture, we can spot two adjacent coarse grid elements $E_{g0}$ and $E_{g1}$ sharing a common node. When subdividing e.g. $E_{g0}$ into $E_{f0}$, $E_{f1}$ and $E_{f2}$, the two values associated with the interval ends of $E_{g0}$ are transferred to the fine grid nodes below using the displayed weighting factors. There, the hierarchical surpluses are added. Vice versa, the restriction operator puts the residual, that has been determined with respect to the fine grid, to the nodes on the coarse level. Before performing the smoothing step (3.59) there, all the individual parts of the residual, stemming from the support of the coarse grid basis function, need to be accumulated. For example: The correction of the coarse grid node in the middle requires the residuals of the five central fine grid nodes. Notice that in contrast to the full-weighting operator introduced earlier, the restriction factors in figure 3.13 do not sum up to 1, but to 3—in two dimensions, it would be 9, and in three dimensions 27. (The weighting factors for the multi-dimensional case are

2In this section, subscripts instead of superscripts are used to denote the corresponding grid level of a vector, in order to avoid confusion with the iterate number.
obtained by tensor product from the one-dimensional ones.) For this reason, the order of magnitude of the residuals computed in peano3d will heavily increase when ascending to the coarser levels. Since one can show that the diagonal elements grow only by a factor of 3 at the same time, the effect is that corrections (3.59) are in general larger on the coarse levels than on the finer ones, so that—after some cycles—the hierarchical surpluses belonging to the coarse grid ansatz functions should be larger than those of the fine grid ansatz functions.

Now that we have gathered all the major ideas, let us recall that we were aiming to solve a system of linear equations \( A \mathbf{t} = \mathbf{b} \) arising from the finite element discretization and the usage of a hierarchical generating system. The residual in our FE approach is computed by

\[
\mathbf{r}_h^{(j)} = M_h \mathbf{f}_h - A_h \tilde{\mathbf{t}}_h^{(j)},
\]

where \( M_h \) denotes the overall mass and \( A_h \) the overall stiffness matrix. The vector \( \mathbf{f}_h \) contains the nodal values from the right hand side of the PDE and \( \tilde{\mathbf{t}}_h^{(j)} \) refers to the approximate solution in the \( j \)-th iteration. We can fix an individual cycle of the algorithm as follows:

### Multigrid cycle with finite elements on a hierarchical generating system

```plaintext
processGrid(\tilde{\mathbf{t}}_h^{(j)}) \{
    \tilde{\mathbf{t}}_h^{(j)} = I_{3h} \cdot \mathbf{t}_h^{(j)} /* interpolation */
    \tilde{\mathbf{t}}_h^{(j)} += \hat{\mathbf{v}}_h^{(j)} /* adding hierarchical surpluses */
    if (finest grid level) then
        \mathbf{r}_h^{(j)} = M_h \mathbf{f}_h - A_h \tilde{\mathbf{t}}_h^{(j)} /* compute residual */
    else
        \mathbf{r}_h^{(j)} = processGrid(\tilde{\mathbf{t}}_h^{(j)}) /* recursion */
    endif
    \hat{\mathbf{v}}_h^{(j+1)} = \hat{\mathbf{v}}_h^{(j)} + \omega D^{-1} \mathbf{r}_h^{(j)} /* correction step */
    \mathbf{r}_{3h}^{(j)} = I_{3h} \mathbf{r}_h^{(j)} /* restriction */
    return \mathbf{r}_{3h}^{(j)}
}
```

### 3.2.5 Computation of the diagonal elements

In the original implementation, that was written by Markus Pögl within the frame of his doctoral thesis [Pögl 04], the peano3d code solved the Poisson equation

\[
- \Delta u = - \text{div}(1 \cdot \nabla u) = f,
\]

which is equivalent to having a thermal conductivity of \( \lambda = 1.0 \) in the whole domain \( \Omega \). For this reason, the diagonal entries in the element stiffness matrix (3.30) become \( h/3 \). Since in three space dimensions, a node belongs to eight surrounding cubes, the diagonal entries in the overall stiffness matrix add up to \( 8h/3 \). (Keep in mind that this matrix...
is not assembled by the numerical code. However, knowledge of its diagonal elements is essential.) Thus, in the correction step of the multigrid cycle outlined above, we can simply replace $D^{-1}$ by the value $\frac{3}{8h}$, where $h$ denotes the meshsize of the corresponding grid level.

So far so good. What will be the situation in terms of the heat conduction problem within ceramic blocks? With respect to the chosen hierarchical discretization of the domain, we have to draw an important distinction in the treatment of the different levels. On the finest grid, each element exhibits one out of two thermal conductivities $\lambda_{\text{air}}$ and $\lambda_{\text{brick}}$. Hence, the diagonal element in matrix (3.30) becomes $h\lambda_{\text{air}}/3$ or $h\lambda_{\text{brick}}/3$, respectively. In line with the consideration from above, these entries sum up to

$$d_i = s \frac{h\lambda_{\text{air}}}{3} + (8 - s) \frac{h\lambda_{\text{brick}}}{3}, \quad s \in \{0, 1, \ldots, 8\}$$

as the $i$-th diagonal element in the overall stiffness matrix ($s$ denotes the number of air cells around the node). On the coarse grids, i.e. on all grids coarser than the finest one, the elements are partitioned into 27 cubes, who might be further subdivided on their part. How do we compute the diagonal elements for the correction steps there?

The stiffness matrix on coarse level

In order to determine the diagonal elements on the coarse grids, we must calculate the stiffness matrices of the corresponding levels. This process is—to some extent—in accordance with the compilation of the overall stiffness matrix. We shall exemplify it by means of two space dimensions. In figure 3.14, you can see a coarse grid square consisting of four cells with thermal conductivities $\lambda_1, \lambda_2, \lambda_3$ and $\lambda_4$. For reasons of simplicity, we assume bisection (remember that the peano3d code performs trisection). Bilinear ansatz functions on the squares (cf. figure 3.5) result in element stiffness matrices

$$A_k = \frac{\lambda_k}{6} \begin{bmatrix} 4 & -1 & -1 & -2 \\ -1 & 4 & -2 & -1 \\ -1 & -2 & 4 & -1 \\ -2 & -1 & -1 & 4 \end{bmatrix}, \quad k = 1, 2, 3, 4. \quad (3.60)$$

![Figure 3.14: Node numbering on the finest and the next coarser grid](image)
Using the fine grid numbering introduced in figure 3.14, these four matrices $A_k$ are now assembled to a 9-by-9 matrix $A^{(intermediate)}$:

$$A^{(intermediate)} = \begin{bmatrix}
4 & -1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 \\
-1 & 4 & 0 & -2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -2 & 0 & 4 & -1 & 0 & 0 & 0 & 0 \\
-2 & -1 & 0 & -1 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} + \frac{\lambda_1}{6} \begin{bmatrix}
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} + \frac{\lambda_2}{6} \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & -1 & 0 & -1 & -2 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & -2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -2 & 0 & 4 & -1 & 0 & 0 & 0 \\
0 & -2 & -1 & 0 & -1 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} + \frac{\lambda_3}{6} \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} + \frac{\lambda_4}{6} \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
$$

This intermediate stiffness matrix is communicated to the coarse grid level by multiplying with the interpolation operator $I_{2h}^h$ from the right and with the restriction operator $I_{h}^{2h}$ from the left (cf. the coarsening rule 3.57):

$$A^{(coarse)} := I_{2h}^h \cdot A^{(intermediate)} \cdot I_{h}^{2h} \quad (3.61)$$

Because of the bilinear ansatz functions and the defined node numbering, the interpola-
Solution operator is given by:

\[
I_h^{2h} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1/2 & 0 & 1/2 & 0 \\
1/4 & 1/4 & 1/4 & 1/4 \\
0 & 1/2 & 0 & 1/2 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1/2 & 1/2 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Notice that the \(i\)-th column of \(I_h^{2h}\) contains the values of the \(i\)-th coarse grid ansatz function at the fine grid nodes—in the order defined by the red numbers in figure 3.14. The restriction operator is simply the transpose of the interpolation matrix: \(\tilde{I}_h^{2h} = (I_h^{2h})^T\). Why is that? If we would redraw figure 3.13 for the bisection instead of trisection, we would encounter weighting factors \(0, \frac{1}{2}, 1\) instead of \(0, \frac{1}{3}, \frac{2}{3}, 1\). With the two-dimensional tensor product of the vector \([0, \frac{1}{2}, 1]\), we obtain

\[
I_h^{2h} = \begin{bmatrix}
1 & 1/2 & 0 & 1/2 & 1/4 & 0 & 0 & 0 & 0 \\
0 & 1/2 & 1 & 0 & 1/4 & 1/2 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/4 & 0 & 1 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/2 & 0 & 1/2 & 1
\end{bmatrix},
\]

which is just the transpose of the interpolation operator. Notice that the \(i\)-th row of \(I_h^{2h}\) gives us the weighting factors of the values at the fine grid nodes for the residual restriction to the \(i\)-th coarse grid node.

After these considerations, all quantities on the right hand side of (3.61) are known and we are able to evaluate \(A^{(coarse)}\). The diagonal elements required in the correction step of the multigrid cycle can then be accumulated from the diagonals of such 4-by-4 matrices \(A^{(coarse)}\). For instance: For the update of the coefficient of the third coarse grid degree of freedom (depicted as a blue circle in the upper left corner of figure 3.14), the examined coarse grid element would contribute \(A_{3,3}^{(coarse)}\).

**Some remarks:** (a) The above example has explained how to go from the very finest level to the next coarser one. If there exist further levels in the hierarchy—as it is the case in general—we simply ascend to them by recursion: The calculated matrices \(A^{(coarse)}\) then take over the role of the \(A_k\) from the lowest grid level. (b) Since the computation of all the intermediate stiffness matrices on the different levels of the grid is very expensive, the numerical code performs it only once during an initial run. The obtained cell contributions to the diagonal elements are stored in special variables, from which they can be read during later iterations. (c) Of course, the illustrated principle remains the same in three space dimensions with trisection instead of bisection.
3.3 Physical interpretation of the algorithm

Let us take a closer look at the computation of the cell residual \( r_h^{(cell)} \), which takes place on the finest grid with meshsize \( h \). We know that a cube on the lowest level in the hierarchy is not refined any further. Associated with its eight corners are temperature values representing the current approximation of the solution. These ones derive from a previous iterate, the initial guess or the Dirichlet boundary condition. We combine them to a vector \( \tilde{t}_h^{(cell)} \). Their order shall be in correspondence with figure 3.4. Then, peano3d computes the cell residual \( r_h^{(cell)} \) in the following way (cf. the sketch of the multigrid cycle):

\[
\begin{align*}
    r_h^{(cell)} &= M_h \tilde{t}_h^{(cell)} - A_h^{(cell)} \tilde{t}_h^{(cell)} = -A_h^{(cell)} \tilde{t}_h^{(cell)} \\
    &\quad \text{(3.62)}
\end{align*}
\]

Here, \( M_h \) denotes the element mass matrix (3.32) and \( A_h^{(cell)} \) the element stiffness matrix (3.30) with its corresponding thermal conductivity \( \lambda^{(cell)} \). The second equality in (3.62) holds, since the right hand side in (2.4) is zero (no internal heat sources or sinks). Vector \( r_h^{(cell)} \) has length 8 and contains the contribution of the cell to the residuals of its belonging nodes. For instance, the last entry of \( r_h^{(cell)} \) gives us the cell’s contribution to the residual in the rear upper right corner of the cube. We shall examine this statement more profoundly: Using the definition of the element stiffness matrix and the trilinear ansatz functions (3.29), we can write out equation (3.62) for the eighth component:

\[
(r_h^{(cell)})_8 = - (A_h^{(cell)} \tilde{t}_h^{(cell)})_8 = \frac{1}{h^3} \int_{[0,1]^3} \langle \begin{pmatrix} yz \\ xz \\ xy \end{pmatrix}, - \lambda^{(cell)} \nabla \left( \sum_{i=1}^{8} (\tilde{t}_h^{(cell)})_i N_{i-1} \right) \rangle \, dx
\]

The expression \( \sum_{i=1}^{8} (\tilde{t}_h^{(cell)})_i N_{i-1} \) is the trilinear approximation to the temperature distribution inside the cell. With the aid of Fourier’s law (2.1), we are able to simplify (3.63):

\[
(r_h^{(cell)})_8 = \frac{1}{h^3} \int_{[0,1]^3} \langle \begin{pmatrix} yz \\ xz \\ xy \end{pmatrix}, \dot{q}(x) \rangle \, dx
\]

where \( \dot{q}(x) \) refers to the heat flux in position \( x \). The vector \( [yz, xz, xy]^T \) points towards the rear upper right corner of the cube (not exactly—but approximately!) and, by means of the scalar product, provides a projection of \( \dot{q} \) onto the corresponding directions. Fur-

Figure 3.15: Vector field \( [yz, xz, xy]^T \) inside the plane \( z = 1 \)
thermore, it gives us a weighting effect, in a way that the value of the heat flux is taken
the more into account the longer the vector \([yz, xz, xy]^T\), i.e.–roughly spoken–the nearer
\(x\) is located to the rear upper right corner. Figure 3.15 shows this weighting and projection effect of \([yz, xz, xy]^T\) within the upper bounding plane of the unit cube. Bringing all the ideas and observations together, the cell residual \((r_{h(\text{cell})})_8\) can be considered the heat flow from the cell towards its rear upper right corner. Notice in this context that the unit of the residual is Watt, which can be verified by \((3.64)\).

The overall residual in a node is accumulated from all the cells surrounding the node
(cf. figure 3.16). In this respect, it can be interpreted as the net heat flow from the

![Diagram](image)

Figure 3.16: Assembly of the residual in two dimensions. All four cells contribute to the
residual in the central node. The displayed weighting factors correspond with the entries of the element stiffness matrices \((3.60)\).

surrounding cells to the node in their middle. If the algorithm should lead to a situation
in which all the assembled residuals vanish, all nodes are in thermal equilibrium (i.e. the
entering heat flows are equal to the heat flows leaving them) and we have reached the steady-state temperature distribution—the solution. No further iterations are required.
In practice, this is very unlikely to happen. Instead, we will stop the algorithm if the
residuals have become "small enough".

In case of a positive/negative overall residual in a particular node, the correction step
of the multigrid cycle will rise/lower the belonging temperature value, which is in line
with the above interpretation: A positive/negative net heat flow to a node will, from a
physical point of view, increase/decrease its temperature. Thus, we can conclude that the
algorithm emulates—at least to some extent—the physical processes within the ceramic
block. However, it should not be concealed that this analogy isn’t perfect since the
correction step neglects two important quantities: density and specific heat capacity.
Indeed, it is the product of them that determines the velocity of the temperature change
whenever there occurs a non-vanishing net heat flow to a point. Instead of this product,
the correction steps in the multigrid cycle involve the diagonal elements, which relate to
the thermal conductivities. For that reason, we could say that the development of the
temperature distribution during the course of iterations takes place on some artificial
time scale.
3.4 Implementation

The coarse levels

The *peano3d* code computes the residual only on the finest grid. On the coarse levels, it is determined by the restriction operator. In figure (3.17), we can see the weighting factors for two space dimensions: As soon as we have calculated the residuals of the 16 fine grid nodes (some of them are fully assembled, others not!), we multiply them with the red numbers and sum up. In this way, we obtain the heat flow from the coarse cell to its upper right corner. It is interesting to notice in this context that, in order to build the overall residual of the upper right coarse grid node, we would encounter 25 weighting factors around it, the sum of which is 9. This is no surprise since the area of a coarse grid cell is larger than the area of a fine grid cell by exactly that factor. Thus, the interpretation of the residual makes also sense for the coarse levels and—from the point of view of this interpretation—the chosen restriction operator becomes more plausible in contrast to the full-weighting, where the sum was always 1.

3.4 Implementation

3.4.1 Algorithm

All the computations performed by *peano3d* must take place within the unit cube $[0, 1]^3$. This constraint is due to a special kind of traversing technique associated with the cache efficiency mentioned in chapter 1.1.

Every algorithm needs to address the cells of the domain in a linear order that determines the sequence in which they are processed. To this end, the programs of our chair make use of a space filling curve of Peano type (for details on space filling curves see [Sagan 94]). Its construction principle shall be outlined in a nutshell: We subdivide the unit cube $[0, 1]^3$ into 27 identical cubes of size $1/3 \times 1/3 \times 1/3$. These ones are put into a linear order with the first one being in front lower left position and the last one diagonally opposite—in rear upper right (cf. figure 3.18). Now, the small cubes are refined on their part. Thus, we obtain 729 cells of size $1/9 \times 1/9 \times 1/9$. What about their traversing sequence? In
fact, it is possible to take the pattern described by the red line in figure 3.18, scale it
down, rotate/mirror it (where needed) and put it into each of the 27 coarse cubes so
that the transitions from one of these to the next match. This construction process is
continued, until we have reached the desired level of depth. Figure 3.19 shows the first
two steps of this recursion for the unit square. In principle, the construction process
is not unique. However, choosing a more elegant way of description, we can enforce
uniqueness by means of grammars (see [Pögl 04]). For our purpose, the details are not
important.

Because of the multigrid method used, the algorithm needs to process cells not only on
the finest but also on the coarser grids. This circumstance gives rise to the question how
to merge the sequences of the cells on different levels with respect to the correction step.
The answer is: Walking through the order of the smallest cubes, we correct the values
of a coarse one as soon as all the cells it comprises have been visited.

**Remark:** In the construction process explained above, it is not necessary always to
refine the whole volume. Instead, we can select only some of the cubes for refinement.
This might be desirable for reasons of geometry or expected behavior of the solution, e.g.
at material transitions. The instruments introduced so far (finite elements and multigrid
method on a hierarchical generating system) are perfectly compatible with that kind of
adaptivity. However, in this context we must pay attention to the occurrence of *hanging
nodes*—nodes at a particular grid level that are surrounded by less than eight cubes of
the respective level. They must not be treated as degrees of freedom, i.e. their hierarchical
surpluses are set to zero. Otherwise, the approximation to the solution would become
discontinuous.
3.4 Implementation

Figure 3.19: Recursive construction of the 2D Peano curve (source: [Günther 04])

Importing the ceramic block
What is now the correlation between the unit cube \([0, 1]^3\) and the ceramic block? We have learned that the cache efficiency of the numerical code is achieved (for details see [Pögl 04]) by the usage of a Peano curve which is embedded into \([0, 1]^3\). From this point of view, the unit cube can be considered a required structure holding the actual domain \(\Omega\) on which the PDE is to be solved. Hence, we position the ceramic block \(\Omega\) in the center of \([0, 1]^3\). Its orientation to the coordinate axes shall be in accordance with the problem formulating section 2.4. The discretization is chosen with respect to the cross section of the block (see subchapter 3.4.2), meaning a higher resolution near the transitions between air and brick material. Outside the block, a lower degree of refinement is sufficient.

Dirichlet boundary conditions and initial guess
Because of the Dirichlet boundary conditions, the two regions \(\{x \in [0, 1]^3 : y \leq y_1\}\) and \(\{x \in [0, 1]^3 : y \geq y_2\}\) are marked as obstacles: We may think of them as air volumes adjoining an imaginary wall \(W := \{x \in [0, 1]^3 : y_1 < y < y_2\}\). No computations are performed there and the hierarchical surpluses on the different levels are all kept at zero, except for those at the corners of the very coarsest cell—the unit cube. These ones are set in a way so that their interpolating function satisfies the inhomogeneous Dirichlet boundary conditions. Taking into account, that, at the launch of the algorithm, all the hierarchical surpluses between the two Dirichlet surfaces vanish, we get an initial guess which is linear in y- and constant in x- and z-direction. This obviously makes sense.

Homogeneous Neumann boundary condition
Around the four block surfaces on which the homogeneous Neumann boundary condition was declared, i.e. within the set \(W\backslash\Omega\), we define an artificial thermal conductivity of zero. This represents a perfect heat insulation and thus assures the fulfillment of the homogeneous Neumann condition.

The controlling framework
The course of an individual iteration is given by the multigrid cycle scheme and the processing order of the cells according to the Peano curve. The first iteration plays a
special role: It computes the diagonal elements (expensive!) and stores them for later reference. Furthermore, the initial residual is determined by taking the absolute maximum of all assembled residuals at the degrees of freedom. This one is required for the stop criterion: As soon as either the maximum norm of the residual has become smaller than a fraction $\varepsilon$ of the initial residual (e.g. $\varepsilon = 10^{-5}$) or a predetermined limit of iterations has been reached, the algorithm will perform a last output iteration and exit.

The relaxation parameter $\omega$

In the introducing part of the multigrid chapter (section 3.2.1), we were able to find an optimum value for the relaxation parameter $\omega$ in terms of the damped Jacobi iteration. With respect to the application of the multigrid method to the heat conduction problem inside ceramic blocks, such a theoretical derivation would go beyond the scope of this thesis due to the complicated structure of the discontinuous coefficients. By experiments, it turned out that a choice of $\omega \in (0.42, 0.45)$ works fine for our computations. Choosing larger values results in an oscillating development of the residual or even divergence. Too small values are no problem for the convergence, but increase the amount of iterations needed.

3.4.2 Geometry description

The cross section of a ceramic block is represented by a B/W bitmap. A pixel in this graphics file corresponds with a certain area, e.g. $0.5 \text{ mm} \times 0.5 \text{ mm}$. The scale is passed to the program via a special input variable. The height of the ceramic block completes the geometry information required by the code.

Remark: In fact, peano3d internally changes the dimensions of the block before placing it into the unit cube of volume $1 \text{ m}^3$, within which all the computations of the algorithm take place. In this way, it avoids negative effects on the convergence rate that are related to the fact that all hierarchical surpluses in the Dirichlet obstacle region are kept at zero. Notice that the temperature distribution is not affected by this transformation. The correct calculation of the mean heat fluxes and the effective thermal conductivities is guaranteed by a scaling module in the postprocessing part of the program, so that we shall neglect the change of the dimensions in the remainder of this thesis for reasons of simplicity.

![Figure 3.20: Bit representation of the cross section (detail view of the area marked red)](image-url)
3.4 Implementation

In figure 3.20, you can see a detail view of the bitfield that belongs to the picture on the left. Air pixels are represented by 0, brick pixels by 1.

Material property

On the finest grid level, the algorithm needs to determine the thermal conductivity of each cell $k$ inside the ceramic block, because this $\lambda^{(k)}$ occurs in the element stiffness matrix (3.30) and is thus necessary in order to compute the cell residual. Since there are no cell-associated variables coming from the stacks, the information must be derived from the node-associated data: For every node of the cube $k$, we can calculate the coordinates and then look up its material property (brick or air) at the corresponding position in the bitfield. If we encounter eight times the same value, the problem how to treat the cell is trivial. In case of a mixed cell, a so-called priority switch decides, whether the cell is to be considered as air or brick cell. By means of two runs of the algorithm—first one with brick, second one with air priority—we obtain a hopefully small interval for the mean heat fluxes and the effective thermal conductivity.

Remark: Notice that the grid structure provided by the unit cube and its recursive refinement does not need to coincide with the material transitions inside the ceramic block—even if these ones are parallel to the coordinate axes. This fact is the reason for the necessity of the above priority rule. The physical meaning of it is an artificial enlargement/diminishment of the brick region inside the ceramic block. We expect larger values for the mean heat fluxes in case of brick priority and smaller ones in case of air priority, since $\lambda_{\text{brick}} \gg \lambda_{\text{air}}$. The values of the actual ceramic block should lie somewhere in between. Of course, we will try to obtain intervals as small as possible. In this context, it is reasonable to use a higher resolution near the material transitions.

Refinement strategy

Whenever the algorithm enters a cube, it has to decide whether this one is to be refined or not. To this end, there exists a so-called depth function which can be evaluated at every node and delivers an integer. If the size of the current cube is $3^{-r} \times 3^{-r} \times 3^{-r}$ with a natural number $r$, and at least one of the eight return values of the depth function exceeds $r$, the cell will be refined. That’s all. So, what are the return values of the depth function at the nodes inside the ceramic block?

Throughout the whole block, we define a basic resolution $r_b$. For example: Basic resolution 5 would entail fine grid cubes of size $1/243 \times 1/243 \times 1/243$ or smaller. In the surroundings of material transitions, we would like to refine once compared to $r_b$, very close to them even twice. In figure 3.20, you can spot two blue boxes. During the pre-processing part of the program, these ones are moved through the whole field—bit by bit. In every position, we check if there are different numbers inside the frame, i.e. ones and zeros simultaneously. If yes, all the sixteen/four entries are marked for refinement. The 4-by-4 frame selects for stage-1-refinement ($\text{depth function} = r_b + 1$), the 2-by-2 for stage-2-refinement ($\text{depth function} = r_b + 2$). In this simple way, a reasonable refinement strategy is realized.
3.4.3 Postprocessing

When the residual has become small enough (or the maximum number of iterations is reached), the algorithm will perform a final iteration in order to output the results. During this last cycle, it computes the heat transfer rates through the surfaces of the (cuboidal) ceramic block $\Omega$ on the basis of the obtained temperature distribution. According to the principle of cellwise processing, we determine the heat flow through a surface as the sum of the heat flows through all cells adjoining to this surface. The direction we are interested in is always perpendicular to the surface under consideration.

For example: Let us assume we want to compute the heat transfer through the rear surface $y = y_2$ of the ceramic block. The fine grid cell depicted in figure 3.21 shall lie within $\Omega$ and border on the surface $y = y_2$, i.e. the four nodes with indeces 3, 4, 7 and 8 are located inside the plane $y = y_2$, the remaining ones in front of it. How can we derive an approximation for the heat transfer $\dot{Q}_y$ through this cube into positive direction of the y-axis by means of the meshsize $h$, the thermal conductivity $\lambda^{(k)}$ and the eight temperature values $T_1, \ldots, T_8$?

**First possibility** For the front and the back face of the depicted cube, average temperatures are introduced:

$$T_{\text{front}} := \frac{T_1 + T_2 + T_3 + T_6}{4}, \quad T_{\text{back}} := \frac{T_3 + T_4 + T_7 + T_8}{4}$$

With the aid of these ones, we obtain:

$$\dot{Q}_y = \dot{q}_y h^2 = \lambda^{(k)} \frac{T_{\text{front}} - T_{\text{back}}}{h} h^2 = \lambda^{(k)} h \left( \frac{T_1 + T_2 - T_3 - T_4 + T_5 + T_6 - T_7 - T_8}{4} \right)$$
Second possibility  From section 3.3, we recall the physical interpretation of the product of $-1.0$, the element stiffness matrix $A^{(k)}$ and the temperature vector $\mathbf{t}^{(k)}$ as the heat flow from the cell into its eight nodes (the result of the product was an 8-vector!). In order to compute the heat flow from the cell into one of its six faces, we simply take the sum of the heat flows into the four nodes that belong to this face:

$$
\dot{Q}_y = -(A^{(k)} \mathbf{t}^{(k)})_3 - (A^{(k)} \mathbf{t}^{(k)})_4 - (A^{(k)} \mathbf{t}^{(k)})_7 - (A^{(k)} \mathbf{t}^{(k)})_8
$$

$$
= -\frac{\lambda^{(k)} h}{12} \left( +0 T_1 - 1 T_2 + 4 T_3 + 0 T_4 - 1 T_5 - 1 T_6 + 0 T_7 - 1 T_8 \right)
$$

$$
- \frac{\lambda^{(k)} h}{12} \left( -1 T_1 + 0 T_2 + 0 T_3 + 4 T_4 - 1 T_5 - 1 T_6 - 1 T_7 + 0 T_8 \right)
$$

$$
- \frac{\lambda^{(k)} h}{12} \left( -1 T_1 - 1 T_2 + 0 T_3 - 1 T_4 + 0 T_5 - 1 T_6 + 4 T_7 + 0 T_8 \right)
$$

$$
- \frac{\lambda^{(k)} h}{12} \left( -1 T_1 - 1 T_2 - 1 T_3 + 0 T_4 - 1 T_5 + 0 T_6 + 0 T_7 + 4 T_8 \right)
$$

$$
= + \frac{\lambda^{(k)} h}{4} \left( +1 T_1 + 1 T_2 - 1 T_3 - 1 T_4 + 1 T_5 + 1 T_6 - 1 T_7 - 1 T_8 \right)
$$

Remark: Obviously, both approaches yield the same result for the heat transfer rate through the cell in y-direction. If the heat flow in x-direction was to be determined, we would have obtained the sign sequence $(+, -, +, -, +, +, -)$ for the multiplication with the temperatures at the nodes, and in case of the z-direction $(+, +, +, +, -, -, -, -)$. Notice that the approximation is of first order only. Of course, it would be better to use the central difference for the temperature derivative and gain an additional order, thus. However, this is not that easy since the cells outside the ceramic block do not hold valid temperature values, and the framework of the stacks makes random access to cells lying further inside (required for a higher approximation order!) a complicated matter.

The mean heat flux through a particular surface of the block is now computed by adding together the heat flows through all the fine grid cells bordering on this surface and then dividing by its area.

**Effective thermal conductivity**

The ceramic block is not homogeneous. It consists of air cavities with a thermal conductivity $\lambda_{\text{air}}$ and of brick material with thermal conductivity $\lambda_{\text{brick}}$. The insulating effect of the air depends on the number, size and position of the cavities. In order to measure it, we introduce the effective thermal conductivity $\lambda_{\text{eff}}$ which is defined by the following equation:

$$
\frac{1}{(x_2 - x_1)(z_2 - z_1)} \int_{x_1}^{x_2} \int_{z_1}^{z_2} \hat{q}_y(x, \hat{y}, z) \, dx \, dz = \lambda_{\text{eff}} \frac{T(y = y_1) - T(y = y_2)}{y_2 - y_1}, \quad (3.65)
$$

where $T(y = y_1)$ and $T(y = y_2)$ are the constant temperatures at the front and back face of the ceramic block according to the specified Dirichlet boundary conditions. The
coordinate \( \hat{y} \) is arbitrary within the interval \([y_1, y_2]\), because the result of the integral expression on the left (i.e., the heat transfer rate perpendicular to the y-axis) must be the same for all choices of \( \hat{y} \), since we assume steady-state conditions. The physical meaning of (3.65) can be put into a nutshell: What must be the thermal conductivity \( \lambda_{\text{eff}} \) of a homogeneous material that shall replace the ceramic block and produce, for the same temperature values in the Dirichlet boundary conditions, the same heat transfer rates perpendicular to the y-axis?
4 Results

4.1 Graphical user interfaces for data I/O

An important objective of the work was to provide architects and construction engineers with a tool that should help them to investigate the thermal characteristics of ceramic blocks on their own. For this reason, the adjustment and extension of the existing numerical code in terms of non-constant coefficients was not sufficient. Interfaces for the convenient input and output of the data had to be written.

![Diagram](image)

Figure 4.1: The process of examination from a user’s point of view

In figure 4.1, you can see the concept of the investigation process on the whole: The user enters all the parameters needed by the algorithm into a data input interface. From a bitmap library that holds the geometries of different blocks, the one that is to be analyzed must be chosen. (Of course, one can also design new blocks in a graphics editor and import them into the program via this channel.) Then, the data input interface collects the required information and writes them into a text file which is read by the
numerical code prior to launching the iterations. For the user, the numerical code is a black box that communicates with the outside world only via files and the output of some numbers to the console. Among other things, these numbers comprise the calculated mean heat fluxes and effective thermal conductivities. For the graphical processing of the information stored in the output files (in the first instance the computed steady-state temperature distribution), there exists a second interface window, through which the visualization in OpenDX can be controlled. In the remainder of this section, the two user interfaces are presented in some detail.

The data input interface

![Figure 4.2: The data input interface window](image)

The window in figure 4.2 provides the user with the possibility to enter all required data concerning the algorithm:

- resolution level for the computation, i.e. the basic resolution \( r_b \) within the block,
- relaxation parameter \( \omega \),
- priority switch for mixed cells,
• maximum number of iterations and

• the factor $\varepsilon$ by which the initial residual is to be reduced.

Next, a ceramic block is selected for examination by choosing a B/W bitmap from the disc. After it was loaded, the user can see a graphical representation of this file in the upper left preview area. Furthermore, the scaling and the thermal conductivities of brick material and air cavities need to be specified. Together with the temperature values for the surfaces where a Dirichlet boundary condition is declared, all the data is written into an ASCII text file. A push on the button "Calculate" executes the numerical code, which reads the generated input file and then starts the actual computations. The data input interface was implemented in Delphi by the Russian student Nikolai Zavyalov.

The visualization interface
After the algorithm has performed the last multigrid cycle, it writes the calculated approximation to the steady-state temperature distribution along with the node-associated material properties (brick/air) into a set of output files. Depending on the resolution level that was chosen for the computation, their size varies from less than one megabyte up to some gigabytes. Of course, there is no sense in analyzing these datasets with the aid of a text editor. Instead, we make use of the mighty visualization tool OpenDX, that helps us to produce graphics from the numerical values.

The way in which OpenDX turns the information from the files into a picture is controlled by means of a so-called network that needs to be programmed visually by connecting certain modules and defining how these ones act on the data that is passing through them. Since the idea of our group was to keep everything as simple as possible for the user, an adequate network file is already provided. The important parameters in this visual network can be changed by the user via the window in figure 4.3, so that there is no need to learn anything about the programming concepts of OpenDX.

In the upper part of the interface, the location of the output files can be set by choosing the DX header that was generated by the numerical code. It comprises references to the actual files. In this way of organizing the datasets, it is a simple matter to hold results from different program runs simultaneously on the disk and select a particular one of them for visualization.

Having completed this task, the next step for the user is to decide, whether OpenDX shall display either the temperature distribution or the heat flux. The latter one can be calculated for every point $x$ inside the block by Fourier's law (2.1). Notice that the graphical representation always takes place within a plane $z = const$. In case of examining the heat flux, there are different options listed in the interface window. We will see examples later, in chapter 4.3.

Independent of the above choice, it is possible to show isothermal lines within the mentioned plane $z = const$. These ones are places of equal temperature. The provided visual network distributes them uniformly. For instance: If we examine the output of a computation with Dirichlet boundary conditions of 10/20 degrees centigrade on the front/back surface of the block, a selection of 19 lines would result in isotherms of the temperature
levels 10.5, 11.0, 11.5, . . . , 19.0 and 19.5 degrees centigrade. Their color can be set to a particular one for all—or to "temperature dependent". Finally, a push on the execute button in the menu calls the visual network in the background and generates an image window with a cross-sectional view of the temperature distribution or heat flux, respectively.

4.2 Calculated heat fluxes and effective thermal conductivities

Prior to taking a look at some nice graphics, the results of the computations shall be presented in numbers. Program runs were carried out for four different geometries. The last one of them coincides, apart from a 90° clockwise rotation, the third. All ceramic blocks examined are produced in reality by the company Knauf.

The basic resolution $r_b$ of the discretization inside $\Omega$ (cf. section 3.4.2) was chosen to be five for blocks 1NF and 2NF with both stages of refinement activated, and six for the computations of block 4NF with only the stage-1-refinement enabled.

In the following tables, every line containing numerical values corresponds with one program run. For each block, two different values for the thermal conductivity $\lambda_{\text{brick}}$ have been considered, the lower one belonging to a more porous brick material with better heat insulation properties. The air cavities were assumed to exhibit a constant
4.2 Calculated heat fluxes and effective thermal conductivities

The thermal conductivity $\lambda_{\text{air}} = 0.025 \text{W/(mK)}$. For the meaning of the priority switch, refer to section 3.4.2.

Remember that we had specified Dirichlet boundary conditions on the front and back surfaces of the block. They were meant to symbolize the temperature proportions inside and outside of an imaginary room. For the computations, the value at the back was held fixed at $T_i = 20^\circ\text{C}$ according to non-changing conditions inside, whereas the temperature on the front surface assumes different values $T_o$, as you can see in the first column of the tables.

**Important remark:** Choosing constants as Dirichlet boundary conditions on both surfaces instead of $z$-dependent functions, leads in effect to a two-dimensional problem, of course. However, notice that all the adjustments and extensions explained in this thesis and implemented in the accompanying numerical code fully work in three space dimensions.

The second column in the tables contains information about the value used for the relaxation parameter $\omega$ in the correction steps of the multigrid cycle. Furthermore, $n_{it}$ is the number of iterations that were necessary in order to reduce the initial residual by a factor of $\varepsilon = 10^{-6}$. The maximum norm $\|r^{(n_{it})}\|_\infty$ of the final residual is given alongside. Be aware that the error itself is not accessible and might be larger than the residual by orders of magnitude. In exchange for that, the mean heat fluxes $\bar{q}$ through the surfaces of the block provide some possibility to verify the trustability of the results: Since the underlying calculated approximation to the temperature distribution is ideally steady-state,

$$\bar{q}(y = y_1) = \bar{q}(y = y_2)$$

should hold. In addition to that, we can check the results by means of the homogeneous Neumann boundary conditions:

$$\bar{q}(x = x_1) = \bar{q}(x = x_2) = 0$$

The effective thermal conductivity $\lambda_{\text{eff}}$ is finally shown in the last column. Heat transfer theory tells us that it is an invariant with respect to different outside temperatures $T_o$, which is perfectly fulfilled.
4 Results

Ceramic block 1NF

Figure 4.4: 1NF (250 mm × 120 mm)

Thermal conductivity $\lambda_{\text{brick}} = 0.61 \, \frac{W}{mK}$

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>$\omega$</th>
<th>$n_d$</th>
<th>$|r^{(n_s)}|_{\infty}$</th>
<th>$\bar{q}(y = y_1)$</th>
<th>$\bar{q}(y = y_2)$</th>
<th>$\bar{q}(x = x_1)$</th>
<th>$\bar{q}(x = x_2)$</th>
<th>$\lambda_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>118</td>
<td>8.91e-09</td>
<td>-51.109926</td>
<td>-51.109925</td>
<td>-0.000529</td>
<td>0.000529</td>
<td>0.408879</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>118</td>
<td>5.88e-09</td>
<td>-34.073284</td>
<td>-34.073283</td>
<td>-0.000352</td>
<td>0.000352</td>
<td>0.408879</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>118</td>
<td>2.95e-09</td>
<td>-17.036642</td>
<td>-17.036642</td>
<td>-0.000176</td>
<td>0.000176</td>
<td>0.408879</td>
</tr>
</tbody>
</table>

$\text{priority switch} = \text{brick material}$

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>$\omega$</th>
<th>$n_d$</th>
<th>$|r^{(n_s)}|_{\infty}$</th>
<th>$\bar{q}(y = y_1)$</th>
<th>$\bar{q}(y = y_2)$</th>
<th>$\bar{q}(x = x_1)$</th>
<th>$\bar{q}(x = x_2)$</th>
<th>$\lambda_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>105</td>
<td>9.69e-09</td>
<td>-49.917677</td>
<td>-49.917676</td>
<td>-0.000490</td>
<td>0.000490</td>
<td>0.399341</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>105</td>
<td>6.44e-09</td>
<td>-33.278451</td>
<td>-33.278451</td>
<td>-0.000327</td>
<td>0.000327</td>
<td>0.399341</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>105</td>
<td>3.22e-09</td>
<td>-16.639226</td>
<td>-16.639226</td>
<td>-0.000163</td>
<td>0.000163</td>
<td>0.399341</td>
</tr>
</tbody>
</table>

$\text{priority switch} = \text{air}$

Thermal conductivity $\lambda_{\text{brick}} = 0.43 \, \frac{W}{mK}$

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>$\omega$</th>
<th>$n_d$</th>
<th>$|r^{(n_s)}|_{\infty}$</th>
<th>$\bar{q}(y = y_1)$</th>
<th>$\bar{q}(y = y_2)$</th>
<th>$\bar{q}(x = x_1)$</th>
<th>$\bar{q}(x = x_2)$</th>
<th>$\lambda_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>108</td>
<td>6.46e-09</td>
<td>-36.492185</td>
<td>-36.492184</td>
<td>-0.000463</td>
<td>0.000463</td>
<td>0.291937</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>108</td>
<td>4.32e-09</td>
<td>-24.328124</td>
<td>-24.328123</td>
<td>-0.000309</td>
<td>0.000309</td>
<td>0.291937</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>108</td>
<td>2.16e-09</td>
<td>-12.164062</td>
<td>-12.164061</td>
<td>-0.000154</td>
<td>0.000154</td>
<td>0.291937</td>
</tr>
</tbody>
</table>

$\text{priority switch} = \text{air}$
### 4.2 Calculated heat fluxes and effective thermal conductivities

**Ceramic block 2NF**

![Diagram of 2NF block](image)

Figure 4.5: 2NF (250 mm × 120 mm)

**Thermal conductivity** $\lambda_{\text{brick}} = 0.61 \frac{W}{mK}$

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>$\omega$</th>
<th>$n_{it}$</th>
<th>$[r^{(n_i)}]_\infty$</th>
<th>$\bar{q}(y = y_1)$</th>
<th>$\bar{q}(y = y_2)$</th>
<th>$\bar{q}(x = x_1)$</th>
<th>$\bar{q}(x = x_2)$</th>
<th>$\lambda_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority switch = brick material</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>129</td>
<td>1.75e-08</td>
<td>-46.934678</td>
<td>-46.934678</td>
<td>0.000581</td>
<td>-0.000581</td>
<td>0.375477</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>129</td>
<td>1.15e-08</td>
<td>-31.289785</td>
<td>-31.289785</td>
<td>0.000388</td>
<td>-0.000388</td>
<td>0.375477</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>129</td>
<td>5.79e-09</td>
<td>-15.644893</td>
<td>-15.644893</td>
<td>0.000194</td>
<td>-0.000194</td>
<td>0.375477</td>
</tr>
<tr>
<td>priority switch = air</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>110</td>
<td>1.69e-08</td>
<td>-45.856455</td>
<td>-45.856455</td>
<td>0.000337</td>
<td>-0.000337</td>
<td>0.366852</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>110</td>
<td>1.12e-08</td>
<td>-30.570970</td>
<td>-30.570970</td>
<td>0.000225</td>
<td>-0.000225</td>
<td>0.366852</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>110</td>
<td>5.62e-09</td>
<td>-15.285485</td>
<td>-15.285485</td>
<td>0.000112</td>
<td>-0.000112</td>
<td>0.366852</td>
</tr>
</tbody>
</table>

**Thermal conductivity** $\lambda_{\text{brick}} = 0.43 \frac{W}{mK}$

<table>
<thead>
<tr>
<th>$T_o$</th>
<th>$\omega$</th>
<th>$n_{it}$</th>
<th>$[r^{(n_i)}]_\infty$</th>
<th>$\bar{q}(y = y_1)$</th>
<th>$\bar{q}(y = y_2)$</th>
<th>$\bar{q}(x = x_1)$</th>
<th>$\bar{q}(x = x_2)$</th>
<th>$\lambda_{\text{eff}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority switch = brick material</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>116</td>
<td>1.15e-08</td>
<td>-33.699168</td>
<td>-33.699168</td>
<td>0.000506</td>
<td>-0.000506</td>
<td>0.269593</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>115</td>
<td>8.38e-09</td>
<td>-22.466113</td>
<td>-22.466113</td>
<td>0.000337</td>
<td>-0.000337</td>
<td>0.269593</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>116</td>
<td>3.81e-09</td>
<td>-11.233056</td>
<td>-11.233056</td>
<td>0.000169</td>
<td>-0.000169</td>
<td>0.269593</td>
</tr>
<tr>
<td>priority switch = air</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.45</td>
<td>105</td>
<td>1.16e-08</td>
<td>-32.955609</td>
<td>-32.955609</td>
<td>0.000343</td>
<td>-0.000343</td>
<td>0.263645</td>
</tr>
<tr>
<td>10.0</td>
<td>0.45</td>
<td>105</td>
<td>7.87e-09</td>
<td>-21.970406</td>
<td>-21.970406</td>
<td>0.000229</td>
<td>-0.000229</td>
<td>0.263645</td>
</tr>
<tr>
<td>15.0</td>
<td>0.45</td>
<td>105</td>
<td>3.91e-09</td>
<td>-10.985203</td>
<td>-10.985203</td>
<td>0.000114</td>
<td>-0.000114</td>
<td>0.263645</td>
</tr>
</tbody>
</table>
4 Results

Ceramic block 4NF

Figure 4.6: 4NF (250 mm × 250 mm)

Thermal conductivity $\lambda_{\text{brick}} = 0.61 \frac{W}{mK}$

| $T_o$ | $\omega$ | $n_{id}$ | $|r^{(n_{id})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|---------|----------------|------------------|------------------|------------------|------------------|------------------|
| priority switch = brick material |
| 5.0   | 0.42   | 216     | 1.52e-08       | -12.365594       | -12.365597       | -0.000452       | 0.000217        | 0.206093        |
| 10.0  | 0.45   | 201     | 1.03e-08       | -8.243730        | -8.243731        | -0.000302       | 0.000145        | 0.206093        |
| 15.0  | 0.42   | 216     | 5.05e-09       | -4.121865        | -4.121866        | -0.000151       | 0.000073        | 0.206093        |
| priority switch = air |
| 5.0   | 0.42   | 210     | 1.50e-08       | -11.678877       | -11.678879       | -0.000474       | 0.000243        | 0.194648        |
| 10.0  | 0.42   | 210     | 1.00e-08       | -7.785918        | -7.785920        | -0.000316       | 0.000162        | 0.194648        |
| 15.0  | 0.42   | 210     | 4.99e-09       | -3.892959        | -3.892960        | -0.000158       | 0.000081        | 0.194648        |

Thermal conductivity $\lambda_{\text{brick}} = 0.43 \frac{W}{mK}$

| $T_o$ | $\omega$ | $n_{id}$ | $|r^{(n_{id})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|---------|----------------|------------------|------------------|------------------|------------------|------------------|
| priority switch = brick material |
| 5.0   | 0.42   | 196     | 1.04e-08       | -9.416845        | -9.416847       | -0.000305       | 0.000146        | 0.156947        |
| 10.0  | 0.42   | 195     | 7.03e-09       | -6.277897        | -6.277898       | -0.000203       | 0.000097        | 0.156947        |
| 15.0  | 0.42   | 195     | 3.63e-09       | -3.138948        | -3.138949       | -0.000102       | 0.000049        | 0.156947        |
| priority switch = air |
| 5.0   | 0.42   | 191     | 1.01e-08       | -8.904137        | -8.904138       | -0.000320       | 0.000165        | 0.148402        |
| 10.0  | 0.42   | 191     | 6.79e-09       | -5.936091        | -5.936092       | -0.000214       | 0.000110        | 0.148402        |
| 15.0  | 0.42   | 191     | 3.40e-09       | -2.968046        | -2.968046       | -0.000107       | 0.000055        | 0.148402        |
4.2 Calculated heat fluxes and effective thermal conductivities

Ceramic block 4NF (rotated)

Figure 4.7: rotated 4NF (250 mm × 250 mm)

Thermal conductivity $\lambda_{\text{brick}} = 0.61 \ \frac{W}{mK}$

| $T_o$ | $\omega$ | $n_{\ell}$ | $|R^{(n_{\ell})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 5.0   | 0.42   | 198      | 1.55e-08        | -22.790085      | -22.790084      | -0.002166       | 0.002444        | 0.379835        |
| 10.0  | 0.42   | 198      | 1.03e-08        | -15.193390      | -15.193390      | -0.001444       | 0.001629        | 0.379835        |
| 15.0  | 0.42   | 198      | 5.16e-09        | -7.596695       | -7.596695       | -0.000722       | 0.000815        | 0.379835        |

priority switch = brick material

| $T_o$ | $\omega$ | $n_{\ell}$ | $|R^{(n_{\ell})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 5.0   | 0.42   | 195      | 1.47e-08        | -21.628927      | -21.628927      | -0.002527       | 0.002812        | 0.360482        |
| 10.0  | 0.42   | 195      | 9.81e-09        | -14.419285      | -14.419285      | -0.001685       | 0.001875        | 0.360482        |
| 15.0  | 0.42   | 195      | 4.91e-09        | -7.209642       | -7.209642       | -0.000842       | 0.000937        | 0.360482        |

priority switch = air

Thermal conductivity $\lambda_{\text{brick}} = 0.43 \ \frac{W}{mK}$

| $T_o$ | $\omega$ | $n_{\ell}$ | $|R^{(n_{\ell})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 5.0   | 0.42   | 179      | 1.04e-08        | -16.304018      | -16.304018      | -0.001428       | 0.001602        | 0.271734        |
| 10.0  | 0.42   | 179      | 6.94e-09        | -10.869346      | -10.869346      | -0.000952       | 0.001068        | 0.271734        |
| 15.0  | 0.42   | 179      | 3.47e-09        | -5.434673       | -5.434673       | -0.000476       | 0.000534        | 0.271734        |

priority switch = brick material

| $T_o$ | $\omega$ | $n_{\ell}$ | $|R^{(n_{\ell})}|_{\infty}$ | $\bar{q}(y = y_1)$ | $\bar{q}(y = y_2)$ | $\bar{q}(x = x_1)$ | $\bar{q}(x = x_2)$ | $\lambda_{\text{eff}}$ |
|-------|--------|----------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| 5.0   | 0.42   | 176      | 1.02e-08        | -15.501469      | -15.501469      | -0.001657       | 0.001838        | 0.258358        |
| 10.0  | 0.42   | 176      | 6.79e-09        | -10.334313      | -10.334313      | -0.001105       | 0.001225        | 0.258358        |
| 15.0  | 0.42   | 176      | 3.39e-09        | -5.167156       | -5.167156       | -0.000552       | 0.000613        | 0.258358        |

priority switch = air

71
Discussion of the results

Let us consider the priority switch related relative discrepancies

\[ d_{rel} = \frac{(\lambda_{eff})_{\text{brick prior}}}{} \frac{(\lambda_{eff})_{\text{air prior}}}{\lambda_{\text{brick}}} - 1 \]

in the effective thermal conductivities:

<table>
<thead>
<tr>
<th>geometry</th>
<th>1NF</th>
<th>2NF</th>
<th>4NF</th>
<th>4NF (90°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda_{\text{brick}})</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
</tr>
<tr>
<td>(d_{rel})</td>
<td>2.39%</td>
<td>2.35%</td>
<td>2.26%</td>
<td>5.88%</td>
</tr>
</tbody>
</table>

Especially for both 4NF geometries, the relative discrepancies are larger than one would expect, and desire, them to be. Different reasons can be imagined for that circumstance: The complex structure of the block might worsen the condition of the problem in a way, so that the residual at the end of the iterations is, indeed, small—but the error not. On the other hand, we should not forget that the mean heat fluxes \(\bar{q}\) and thus also \(\lambda_{eff}\) are computed using an approximation of first order only. This problem was due to the technical peculiarities of the code and must also be taken into account when checking (4.1) and (4.2).

A comparison with findings from the Russian computations in [Slavyanov et al. 04], that were carried out assuming perfect heat insulation by the air cavities \((\Leftrightarrow \lambda_{\text{air}} = 0.0 \frac{W}{mK})\), produced reasonable qualitative results, i.e. the \(\lambda_{eff}\) values in the above tables are larger than the Russian ones—but not too much. Here also, ceramic block 4NF played a special role to some extent.

4.3 Visualization and interpretation

In this section, we want to analyze the results on the basis of their graphical representation. Don’t be confused about the parameters of the underlying computations not being provided alongside the individual pictures. \(T_o\) is 10°C and \(\lambda_{\text{brick}} = 0.61 \frac{W}{mK}\) in all of them. Beyond that, the examination is meant to be merely qualitative.

The temperature distribution

The temperature is displayed by means of colors. Red represents the highest temperature, blue the lowest. In between, values are spread according to the color spectrum, i.e. the redder a point the warmer—the bluer the cooler. Since it is rather challenging to detect small differences with the naked eye, all the figures feature isothermal lines, along which the temperature is constant. From Fourier’s law (2.1), we can deduce that the heat flux must be high where the distance between adjacent lines is small. However, inside the air cavities, the heat flux is nevertheless low, in spite of small distances between the isothermal lines, because the thermal conductivity \(\lambda_{\text{air}}\) is almost zero. Perpendicular to the lines, \(\dot{q}\) is maximum, whereas along them it vanishes. This is an important property of equipotential surfaces and entails that the isothermal lines should flow at right angle into those parts of the boundary where the homogeneous Neumann condition was specified.
4.3 Visualization and interpretation

Figure 4.8: Temperature distribution in block 1NF (75 isothermal lines)

Figure 4.9: Temperature distribution in block 2NF (85 isothermal lines)
Figure 4.10: Temperature distribution in block 4NF (100 isothermal lines)

Figure 4.11: Temperature distribution in the rotated block 4NF (100 isothermal lines)
We observe that the main heat flow is from top to bottom. Especially in geometry 4NF, but also above and below the huge cavities in figures 4.9 and 4.11, strong deviations from this vertical direction are encountered. They indicate the insulating effect of the air. At the material transitions, isothermal lines exhibit sharp bends due to the jumps in $\lambda(x)$. The homogeneous Neumann boundary conditions are apparently fulfilled in all the pictures.

**Heat flux magnitude**

We can calculate the heat flux $\dot{q}$ by means of Fourier’s law (2.1),

$$\dot{q}(x) = -\lambda(x) \nabla T(x),$$

since both temperature distribution $T(x)$ and thermal conductivity $\lambda(x)$ are known from the data in the output files. Notice that $\dot{q}$ is a vector quantity and contains information about direction and magnitude at the same time. First, we will visualize only the latter one in order to spot the points where the peaks occur at. To this end, $|\dot{q}|_2$ is displayed by colors. Once again, red marks large values and blue small ones. Important: The underlying colormaps of the following figures were adjusted individually, so that there is no sense in comparing the colors in two different pictures with each other.

![Figure 4.12: Magnitude of the heat flux in block 1NF (75 isothermal lines)](image-url)
Figure 4.13: Magnitude of the heat flux in block 2NF (85 isothermal lines)
4.3 Visualization and interpretation

Figure 4.14: Magnitude of the heat flux in block 4NF (100 isothermal lines)
Figure 4.15: Magnitude of the heat flux in the rotated block 4NF (100 isothermal lines)
About figure 4.12: The peak values appear around the air cavities, next to their corners. They are rather small—but distinctive. The vertical bars of brick material act as thermal bridges, along which the main heat flow takes place. In contrast, the heat hardly manages to use the horizontal connections between them. Their stretch in y-direction is too little.

About figure 4.13: What was said about the heat flux in figure 4.12, in principle also holds for 2NF. The big difference is, of course, the huge air cavity in the center of the block. The visualization shows that little heat flows through it. The major amount tries to evade it, which is the reason for the two crimson regions sideways. It is interesting to study the central cavity’s influence on the magnitude of heat flux above and below it by means of the picture.

About figure 4.14: Notice that the color distribution in this figure is not symmetric. We encounter a larger heat flow to the right of the two big cavities than to the left of them. This circumstance can be made plausible from the alignment of the small cavities in the upper part of the block: They are staggered in a way that the distances (along the brick material!) to the sideway regions of the first huge cavity differ from each other. For the heat flow, entry to the right hand side of the big cavity is easier than to its left hand side.

Furthermore, we can observe that the large deviation of the brick material bars from the vertical direction (along which heat must flow due to the underlying boundary conditions) prevents them very well from acting as thermal bridges.

About figure 4.15: In contrast to figure 4.14, the rotated alignment of block 4NF now turns the inner brick structure into a pattern of several thermal bridges. It was for this reason that the computation of $\lambda_{\text{eff}}$ yielded to almost the double values in comparison with the unrotated block (cf. section 4.2).

**Detail views of the heat flux direction**

In principle, information about the direction of the heat flux can be deduced from the isothermal lines, since $\dot{q}$ must always be perpendicular to these ones. In this respect, the graphical discussion of the results is yet exhaustive. Nevertheless, it will be interesting to take a closer look at the direction of $\dot{q}$ by means of some detail views. This is what we shall do finally. Note: In the following figures, all the arrows have been scaled to equal length. Their original size is represented by colors.
Figure 4.16: Detail view of the horizontal connection element between two of the vertical bars that are posing as thermal bridges in the ceramic block 1NF. One can clearly observe how the heat first enters the element (upper part of the figure) and then evades again due to the cavity in front (lower part). The refinement near the transitions between air and brick material is reflected by the higher density of the arrows in the corresponding regions.
4.3 Visualization and interpretation

Figure 4.17: Detail view of the central air cavity in block 2NF

Figure 4.18: Left boundary in block 4NF. The arrows close to the thick white bar are vertical, i.e. the homogeneous Neumann boundary conditions are fulfilled.
4.4 Findings from the generalized problem

The tensor generalization has been tested numerically by means of two simple example problems. Both of them were of the following type:

\[- \text{div}(T(x) \nabla u(x)) = f(x) \quad \forall x \in \Omega = (0,1)^3 \]
\[u(x) = 0 \quad \forall x \in \partial \Omega \quad (4.3)\]

where \(T(x) \in \mathbb{R}^{3,3}\) is a symmetric positive definite matrix. Here, in contrast to the simulation of heat conduction within ceramic blocks, the domain \(\Omega\) is the whole unit cube. Since we specify homogeneous Dirichlet boundary conditions, there has to be a non-vanishing right hand side \(f\) in (4.3). Otherwise, the solution \(u\) would be zero. For the different program runs, maximum resolutions from 2 to 5 were used, i.e. the number of smallest cubes in the discretization ranged from \(9^3\) to \(243^3\). By experiment, the relaxation parameter was set to 0.55. The algorithm should terminate whenever the initial residual had been reduced by a factor of \(\varepsilon = 10^{-5}\).

**First example: constant tensor**

The tensor was chosen to be

\[
T(x) \equiv \begin{bmatrix}
1.0 & 0.1 & 0.0 \\
0.1 & 1.2 & 0.0 \\
0.0 & 0.0 & 1.4
\end{bmatrix}
\]

and the right hand side function

\[
f(x) = 3.6 \pi^2 \prod_{i=1}^{3} \sin(\pi x_i) - 0.2 \pi^2 \cos(\pi x_1) \cos(\pi x_2) \sin(\pi x_3).
\]

Then, the exact solution of problem (4.3) is

\[
u(x) = \prod_{i=1}^{3} \sin(\pi x_i).
\]

The following table shows the number of iterations that were necessary in order to reduce the initial residual by the factor \(\varepsilon\) for the different resolutions:

<table>
<thead>
<tr>
<th>resolution</th>
<th>degrees of freedom</th>
<th>iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 \times 9 \times 9</td>
<td>520</td>
<td>25</td>
</tr>
<tr>
<td>27 \times 27 \times 27</td>
<td>18,096</td>
<td>25</td>
</tr>
<tr>
<td>81 \times 81 \times 81</td>
<td>530,096</td>
<td>25</td>
</tr>
<tr>
<td>243 \times 243 \times 243</td>
<td>14,702,584</td>
<td>26</td>
</tr>
</tbody>
</table>

We can observe that the numbers in the right column are more or less independent of the resolution and thus of the degrees of freedom. Since this typical convergence property of the multigrid method occurs, we can state that the implemented tensor generalization
apparently works fine. In figure 4.19, the residuals have been plotted in logarithmic scale versus the iterations. The profiles should be straight lines, ideally, which is almost fulfilled. For the highest resolution, we encounter small oscillations in the second part of the curve. These ones have been encountered in [Pögl 04] as well and are due to a non-optimal choice of the relaxation parameter \( \omega \). In principle, \( \omega \) needs to be derived theoretically, but as this would be beyond the scope of this thesis, we refrain from it.

**Second example: non-constant tensor**

This time, the tensor shall be

\[
T(x) = \begin{bmatrix}
1 & \frac{x_1 x_2}{4} & \frac{x_1 x_3}{4} \\
\frac{x_2 x_1}{4} & 1 & \frac{x_2 x_3}{4} \\
\frac{x_3 x_1}{4} & \frac{x_3 x_2}{4} & 1
\end{bmatrix}
\]

and the right hand side

\[
f(x) \equiv 1.
\]

The following table presents the obtained results:

<table>
<thead>
<tr>
<th>resolution</th>
<th>degrees of freedom</th>
<th>iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 \times 9 \times 9</td>
<td>520</td>
<td>33</td>
</tr>
<tr>
<td>27 \times 27 \times 27</td>
<td>18,096</td>
<td>32</td>
</tr>
<tr>
<td>81 \times 81 \times 81</td>
<td>530,096</td>
<td>32</td>
</tr>
<tr>
<td>243 \times 243 \times 243</td>
<td>14,702,584</td>
<td>32</td>
</tr>
</tbody>
</table>
Once again, we find multigrid convergence. The residual curves can be studied in figure 4.20. It is interesting to see that all of them feature a sharp bend around the thirteenth iteration revealing a sudden slowdown in the convergence. This one is due to the (development of the) spectral composition of the residual in the concrete example. The fact that it occurs at the same position for all resolutions indicates that the multigrid method works fine.
5 Summary and outlook

5.1 Achievements

With the aid of the approaches and ideas presented in this thesis, it was possible to turn the original peano3d code from the dissertation [Pögl 04] into a useful tool for the simulation of heat conduction inside complicated three-dimensional geometries. We have seen that the introduced principle of determining the proper diagonal elements for the correction steps in the multigrid cycle is valid in a more general setup than only in terms of the ceramic blocks. Actually, it allows us to deal with an arbitrary number of different materials inside the domain Ω and with anisotropic heat conduction as well. In these cases, all necessary changes in the numerical code are restricted to the computation of new element stiffness matrices. The part of the algorithm which delivers the diagonal elements does not need to be altered and remains the same. Thus, finite element and multigrid method are two independent modules in the code. Such modularity is very desirable with respect to the objective of merging the achievements of this work into the related research projects [Dieminger 05], [Herder 05], [Krahnke 04], [Langlotz 04] and [Wagner 05] at the chair of Prof. Zenger.

From a user’s point of view, the two interface windows for data input prior to the computation and for visualization of its results afterwards are very welcome. By means of them, engineers may now carry out numerical experiments on their own without a profound knowledge of the underlying mathematical theories. Especially the possibility to represent the calculated solutions graphically provides a highly interesting insight. The experienced engineer might use the pictures to draw conclusions how to improve the heat insulating effect of the blocks.

5.2 Outlook

The effective thermal conductivities λ_{eff} were determined assuming boundary conditions of first type. However, concerning practical problems, it is more common to consider boundary conditions of third type. The reason is that we usually don’t know neither the temperature on nor the heat flux through the corresponding surface. What we know in exchange, is a correlation between both of them which is described by equation (2.3) and couples the occurring convection and conduction processes with each other (see section 2.3). The involved Newton coefficients α can be looked up in tables, e.g. in the German industry norm DIN.

To implement the changed model within the frame of the existing algorithm is not very difficult: The original Dirichlet nodes are turned into degrees of freedom, i.e. their hier-
archical surpluses are not any more kept at zero but are updated during the course of iterations. As a consequence, we need to compute their residuals for the correction steps. Here, we remember that the overall residual of a node was assembled from the eight cells surrounding it. In general, a "third type node" in a former Dirichlet surface belongs to eight cells as well. But only four of them lie inside the block $\Omega$. Our understanding of the cell residual as the heat flow from a cell into its nodes now helps us in determining the residual of a third type node: The four cells inside $\Omega$ contribute to the overall residual by conduction—in the way we have encountered so far. The remaining (outside) cells also transfer heat to the node. This takes place by convection and is calculated by means of the current temperature approximation in the node, the fluid temperature, the dimensions of the cell and the Newton coefficient $\alpha$. Of course, the adaptation of the diagonal elements and especially the coarser grid levels are the main challenge in terms of this approach.

A second possibility to continue the work started with this thesis would be to adjust the interpolation operator with respect to the jumps in the coefficient $\lambda$. Notice in this context that the linear interpolation implemented in the algorithm completely neglects the material property and thus cannot be the best choice. Using the continuity of the heat flux at the transitions, we can derive a better interpolation operator. To understand the principle, we imagine a one-dimensional interval (representing e.g. a heat conducting rod) with temperature values $T_l$ and $T_r$ at the left and right end respectively. For reasons of simplicity, it shall be partitioned by bisection (instead of trisection). We suppose that the left part exhibits a thermal conductivity of $\lambda_l$ and the right part of $\lambda_r$. Linear interpolation would result in a central temperature node whose value $T_{c,lin}$ is just the mean value of $T_l$ and $T_r$:

$$T_{c,lin} = \frac{T_l + T_r}{2}$$

The thermal conductivities $\lambda_l$ and $\lambda_r$ are not at all taken into account. Without adding the hierarchical surplus to $T_{c,lin}$, we cannot expect a correct representation of the steady-state temperature distribution (only if $\lambda_l = \lambda_r$). In contrast, evaluation of the heat flux balance in the central node yields:

$$T_c = \frac{\lambda_l}{\lambda_l + \lambda_r} T_l + \frac{\lambda_r}{\lambda_l + \lambda_r} T_r$$

Using this equation in the definition of the one-dimensional interpolation operator, the interpolating function through $T_l$, $T_c$ and $T_r$ automatically fits the steady-state temperature distribution we are looking for. No hierarchical surplus needs to be added. In more than one space dimension, this desirable property unfortunately cannot be achieved any more. However, it is possible to adapt the multi-dimensional operators in a way that the interpolation contributes at least a major part to the correct approximation and the required hierarchical surpluses become small. For the two-dimensional case, you can find details about the construction of such an interpolation operator in [Alcouffe/Brandt 81]. The generalization of the idea presented there towards the usage
5.2 Outlook

in three space dimensions and especially the implementation within the framework of the existing cache-efficient code is not trivial. The overall effect that one can expect from an adaptation of the interpolation operator is, that jumps in $\lambda$ may exhibit even higher orders of magnitude than so far and the choice of the relaxation parameter $\omega$ becomes less critical. Larger values could be taken for it and the algorithm should converge faster. Всё будет хорошо.
5 Summary and outlook
A The element stiffness matrix for the tensor generalization

For the generalized problem, we had derived the following element stiffness matrix:

\[ A^{(k)} = U^{(k)}_{11} B^{11} + U^{(k)}_{22} B^{22} + U^{(k)}_{33} B^{33} + U^{(k)}_{12} B^{12} + U^{(k)}_{13} B^{13} + U^{(k)}_{23} B^{23} \]

In this formula, the individual \( B^{lm} \) are given by:

\[
B^{11} = \frac{h}{36} \begin{bmatrix}
4 & -4 & 2 & -2 & 2 & -2 & 1 & -1 \\
-4 & 4 & -2 & 2 & -2 & 2 & -1 & 1 \\
2 & -2 & 4 & -4 & 1 & -1 & 2 & -2 \\
-2 & 2 & -4 & 4 & -1 & 1 & -2 & 2 \\
2 & -2 & 1 & -1 & 4 & -4 & 2 & -2 \\
-2 & 2 & -1 & 1 & -4 & 4 & -2 & 2 \\
1 & -1 & 2 & -2 & 2 & -2 & 4 & -4 \\
-1 & 1 & -2 & 2 & -2 & 2 & -4 & 4
\end{bmatrix}
\]

\[
B^{22} = \frac{h}{36} \begin{bmatrix}
4 & 2 & -4 & -2 & 2 & 1 & -2 & -1 \\
-4 & -2 & 4 & 2 & -2 & -1 & 2 & 1 \\
-2 & -4 & 2 & 4 & -1 & -2 & 1 & 2 \\
2 & 1 & -2 & -1 & 4 & 2 & -4 & -2 \\
1 & 2 & -1 & -2 & 2 & 4 & -2 & -4 \\
-2 & -1 & 2 & 1 & -4 & -2 & 4 & 2 \\
-1 & -2 & 1 & 2 & -2 & -4 & 2 & 4
\end{bmatrix}
\]

\[
B^{33} = \frac{h}{36} \begin{bmatrix}
4 & 2 & 2 & 1 & -4 & -2 & -2 & -1 \\
2 & 4 & 1 & 2 & -2 & -4 & -1 & -2 \\
2 & 1 & 4 & 2 & -2 & -1 & -4 & -2 \\
1 & 2 & 2 & 4 & -1 & -2 & -2 & -4 \\
-4 & -2 & -2 & -1 & 4 & 2 & 2 & 1 \\
-2 & -4 & -1 & -2 & 2 & 4 & 1 & 2 \\
-2 & -1 & -4 & -2 & 2 & 1 & 4 & 2 \\
-1 & -2 & -2 & -4 & 1 & 2 & 2 & 4
\end{bmatrix}
\]
A The element stiffness matrix for the tensor generalization

\[
B_{12} = \frac{h}{12} \begin{bmatrix}
2 & 0 & 0 & -2 & 1 & 0 & 0 & -1 \\
0 & -2 & 2 & 0 & 0 & -1 & 1 & 0 \\
0 & 2 & -2 & 0 & 0 & 1 & -1 & 0 \\
-2 & 0 & 0 & 2 & -1 & 0 & 0 & 1 \\
1 & 0 & 0 & -1 & 2 & 0 & 0 & -2 \\
0 & -1 & 1 & 0 & 0 & -2 & 2 & 0 \\
0 & 1 & -1 & 0 & 0 & 2 & -2 & 0 \\
-1 & 0 & 0 & 1 & -2 & 0 & 0 & 2
\end{bmatrix}
\]

\[
B_{13} = \frac{h}{12} \begin{bmatrix}
2 & 0 & 1 & 0 & 0 & -2 & 0 & -1 \\
0 & -2 & 0 & -1 & 2 & 0 & 1 & 0 \\
1 & 0 & 2 & 0 & 0 & -1 & 0 & -2 \\
0 & -1 & 0 & -2 & 1 & 0 & 2 & 0 \\
0 & 2 & 0 & 1 & -2 & 0 & -1 & 0 \\
-2 & 0 & -1 & 0 & 0 & 2 & 0 & 1 \\
0 & 1 & 0 & 2 & -1 & 0 & -2 & 0 \\
-1 & 0 & -2 & 0 & 0 & 1 & 0 & 2
\end{bmatrix}
\]

\[
B_{23} = \frac{h}{12} \begin{bmatrix}
2 & 1 & 0 & 0 & 0 & 0 & -2 & -1 \\
1 & 2 & 0 & 0 & 0 & 0 & -1 & -2 \\
0 & 0 & -2 & -1 & 2 & 1 & 0 & 0 \\
0 & 0 & -1 & -2 & 1 & 2 & 0 & 0 \\
0 & 0 & 2 & 1 & -2 & -1 & 0 & 0 \\
0 & 0 & 1 & 2 & -1 & -2 & 0 & 0 \\
-2 & -1 & 0 & 0 & 0 & 0 & 2 & 1 \\
-1 & -2 & 0 & 0 & 0 & 0 & 1 & 2
\end{bmatrix}
\]
TER, The multigrid method for the diffusion equation with strongly discontinuous

[Baehr/Stephan 03] HANS D. BAEHR, KARL STEPHAN, Wärme- und Stoffübertragung,

[Braess 03] DIETRICH BRAESS, Finite Elemente, Theorie, schnelle Löser und Anwen-
dungen in der Elastizitätstheorie; 3., korrigierte und ergänzte Auflage, Springer,
Berlin, 2003

[Briggs 87] WILLIAM L. BRIGGS, A Multigrid Tutorial, printed for the Society for In-

[Brokate 02] MARTIN BROKATE, Partielle Differentialgleichungen 1, Vorlesungsskript,
Zentrum Mathematik, TU München, 2002

[Dieminger 05] NADINE DIEMINGER, Kriterien für die Selbstadaption cache-effizienter
Mehrgitteralgorithmen, diploma thesis, TU München, 2005

[Dornseifer 97] THOMAS DORNSEIFER, Diskretisierung allgemeiner elliptischer Diffe-
rentialgleichungen in krummlinigen Koordinatensystemen auf dünnen Gittern, Diss-
sertation, Institut für Informatik der TU München, 1997

[Grigull/Sandner 90] U. GRIGULL, H. SANDNER, Wärmeleitung, zweite Auflage,
Springer-Verlag, Berlin, 1990

[Günther 04] FRANK GÜNThER, Eine cache-optimale Implementierung der Finite-
Element-Methode, Dissertation, Institut für Informatik der TU München, 2004

[Herder 05] WOLFGANG HERDER, Lastverteilung und parallelisierte Erzeugung von Ein-
gabetabellen für ein paralleles cache-optimales Finite-Element-Verfahren, diploma the-
esis, TU München, 2005

[Incropera/DeWitt 02] FRANK P. INCroPERA, DAVID P. DEWITT, Fundamentals of
Heat and Mass Transfer, 5th edition, John Wiley & Sons Ltd., Chichester, 2002
Bibliography


[Simeon 04] Bernd Simeon, Numerik partieller Differentialgleichungen, Skriptum zur Vorlesung, Zentrum Mathematik, TU München, 2004

[Slavyanov et al. 04] В. Грикуров, С. Славинов, Р. Трепков, Расчёт теплопроводящих характеристик керамических блоков с внутренними полостями, paper, Department of Mathematics and Computational Physics, Saint Petersburg State University, 2004