High Performance Computing

Stefan Funken
Karsten Urban

Ulm University

Version 0.1, July 10, 2017

Contents

I Mathematical Model

1 Problem Formulation and Analysis 7

1.1 Governing equations 9
1.1.1 Maxwell's equations 9
1.1.2 Material laws 10
1.1.3 Reduction to 2D 12

1.2 Variational Formulation 13
1.2.1 The Variational Formulation 13

1.3 Existence and Uniqueness of the Solution 14
1.3.1 Conditions on the Reluctivity ν 15

2 The nonlinear Modell Problem 17

2.1 Material properties 18

2.2 Torque Calculation 19

2.3 Geometry and Initial Mesh of BLDC Motor 20

3 Finite Element Method 23

3.1 Short Matlab implementation 23
3.1.1 Model Problem 23
3.1.2 Galerkin Discretisation of the Linear Problem 24
3.1.3 Data-Representation of the Triangulation Ω 25
3.1.4 Assembling the Stiffness Matrix 27
3.1.5 Assembling the right-hand side 28
3.1.6 Incooperating Dirichlet conditions 29

3.1.7 Computation and Displaying the numerical solution 30
3.2 First Matlab Implementation for the Nonlinear Modell Problem 31
3.2.1 Newtons Method 32
3.2.2 Discrete Nonlinear Problem 33
3.2.3 Assembling the Nonlinear Function F 34
3.2.4 Assembling the Jacobian DF 35
3.2.5 Computation the numerical solution 36

Bibliography 39

Mathematical Model

1 Problem Formulation and Analysis 7
1.1 Governing equations
1.2 Variational Formulation
1.3 Existence and Uniqueness of the Solution

2 The nonlinear Modell Problem 17
2.1 Material properties
2.2 Torque Calculation
2.3 Geometry and Initial Mesh of BLDC Motor

3 Finite Element Method 23
3.1 Short Matlab implementation
3.2 First Matlab Implementation for the Nonlinear Modell Problem

Bibliography 39

1. Problem Formulation and Analysis

In this lecture, we will follow a realistic model problem in order to introduce the main topics of numerical methods for HPC.

For an optimal machine design with respect to individual requirements modelling and simulation of electrical machines play a crucial role. We will consider a model problem in electromagnetics, a brushless directed current motor (BLDC). To discuss and optimize the underlying mathematical model we will give a short introduction to electromagnetism and the system of nonlinear partial differential equations. Within the scope of the lecture we will consider the case of a stationary two-dimensional model problem, which we will introduce in the following section To demonstrate how such a simulation works in principle, also develop a short MATLAB program to get the solution of this problem.

When current flows through any wire it makes a magnetic field around the wire. Usually this magnetic field is very weak; it can be made stronger by adding more coils to the wire or adding an iron core through the coils. The current can also be increased to make the magnetism stronger. The direction of the magnetic field can be found by using the right-hand rule. This means that if a person wraps the fingers of their right hand around an iron core in the direction of the current, the magnetic field would go in the same direction as the thumb would point. The coils of a brushless directed current motor are located on the stator and the permanent magnet is the rotor. By applying DC power to the coil, the coil will energize and become an electromagnet. The operation of a BLDC is based on the simple force interaction between the permanent magnet and the electromagnet. When a coil is energized, the opposite poles of the rotor and stator are attracted to each other. As a result the rotor poles move near to the energized stator. As the rotor nears the green coil, the red coil is energized. As the rotor nears the red coil, the blue coil is energized. After that, the green coil is energized with the opposite polarity. This process is repeated, and the rotor continues to rotate. The DC current required in each coil and the rotating rotor is shown in the following picture Fig 1.2.

The current is controlled by using a micro controller and transistors to supply current into the correct phase. To determine which phase should be active, the angular position

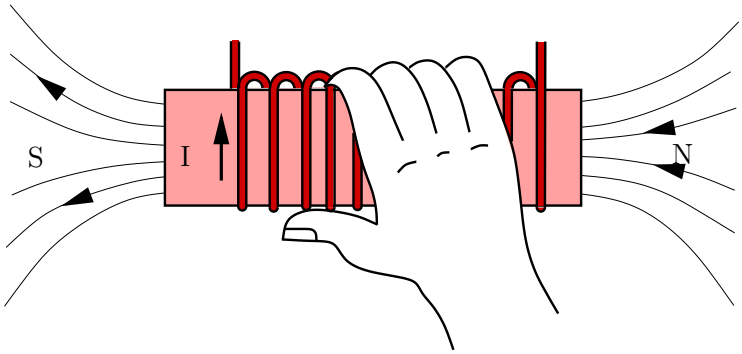


Figure 1.1: 'Right hand rule' of electromagnetism.

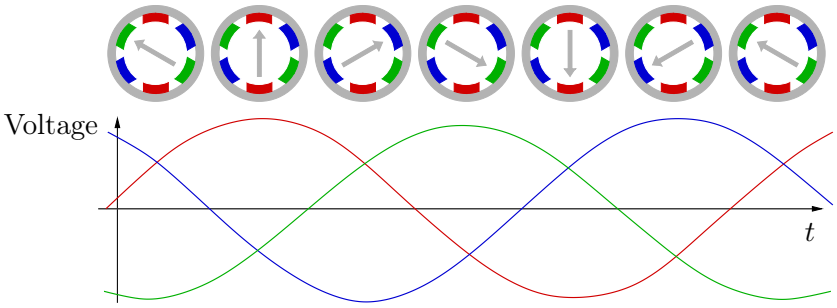


Figure 1.2: Brushless DC motor. Rotating magnetic field created by the sequential excitation of the pole pairs by a DC pulse.

of the rotor must be known at all times. To do this, most motors leverage the Hall Effect to determine where the rotor is in its rotation. A Hall Effect sensor works by detecting disturbances in its magnetic field caused by a moving rotor. The position is then relayed to the micro controller as a binary number based on which sensors are active and which are inactive. The micro-controller then decides which phase current should enter based upon the angular position.

From the famous Maxwell equations, which are used to describe electromagnetic phenomena, we derive the stationary magnetic field formulation. Under certain assumptions the system of nonlinear partial differential equations can be reduced to a single nonlinear differential equation in two dimensions.

The nonlinearity occurs in a material relation between the magnetic field $\mathbf{H}$ and the magnetic induction $\mathbf{B}$. The relation can be expressed by the so-called B-H-curve, or the closely related reluctivity ν (reciprocal of magnetic permeability μ). In general, none of these mappings are given analytically and must be approximated from measured data. With suitable boundary conditions, the magnetostatic 2D - problem can be formulated in a nonlinear variational formulation. Existence and uniqueness is shown by the theorem of Zarantonello. Usually one uses Newton's method to obtain the solution of the nonlinear variational problem.

1.1 Governing equations

1.1.1 Maxwell's equations

Maxwell's equations are used to describe macroscopic electromagnetic phenomena (cf. [3, 7]):

$$\operatorname{curl} \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \tag{1.1}$$

$$\operatorname{curl} \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{1.2}$$

$$\operatorname{div} \mathbf{B} = 0, \tag{1.3}$$

$$\operatorname{div} \mathbf{D} = \varrho. \tag{1.4}$$

The quantities involved are

$\mathbf{H}$ - magnetic field,	$\mathbf{D}$ - electric induction,
$\mathbf{E}$ - electric field,	$\mathbf{J}$ - electric current density,
$\mathbf{B}$ - magnetic flux,	ϱ - electric charge density.

All these quantities depend on the position in the space $x = (x_1, x_2, x_3)$ and on the time t. The boldface letters are vector fields. Via constitutive laws there exists a relation between the magnetic flux $\mathbf{B}$ and the magnetic field $\mathbf{H}$:

$$\mathbf{B} = \mu_0 \mu_r (\mathbf{H} + \mathbf{H}_0). \tag{1.5}$$

Here,

μ_0 denotes the permeability of the vacuum, $\mu_0 := 4\pi\, 10^{-7} \left[\frac{Vs}{Am}\right]$, and
μ_r denotes the relative permeability.

In the case of permanent magnetic materials, $-\mathbf{H}_0$ is called the magnetic field where the induction $\mathbf{B}$ disappears. For the quantities which are not permanent magnetic we assume that $\mathbf{H}_0 = 0$.

We neglect the effects of hysteresis, such that the relative permeability μ_r can be represented as a function of $|\mathbf{B}|$, such that

$$\mathbf{B} = \mu_0 \mu_r(|\mathbf{B}|)\,(\mathbf{H} + \mathbf{H}_0)\,. \tag{1.6}$$

Additionally we introduce another quantity, the so-called relative reluctivity

$$\nu(|B|) := \frac{1}{\mu_0 \mu_r(|\mathbf{B}|)} \tag{1.7}$$

such that following relation holds:

$$\mathbf{H} + \mathbf{H}_0 = \nu(|\mathbf{B}|)\,\mathbf{B}\,. \tag{1.8}$$

Since $\mathbf{B}$ is divergence free, we can find a vector potential $\mathbf{A}$ such that

$$\mathbf{B} = \operatorname{curl} \mathbf{A}\,. \tag{1.9}$$

Considering the low-frequency case of electromagnetism, displacement currents are negligible in comparison with the impressed currents, i.e.,

$$\left|\frac{\partial \mathbf{D}}{\partial t}\right| \ll |\mathbf{J}| \tag{1.10}$$

So we are left with the following reduced set of equations, the stationary (no dependence on t) magnetostatic formulation:

$$\begin{aligned} \operatorname{curl} \mathbf{H} &= \mathbf{J}\,, && (1.11)\\ \operatorname{div} \mathbf{B} &= 0\,, && (1.12)\\ \mathbf{H} + \mathbf{H}_0 &= \nu(|\mathbf{B}|)\,\mathbf{B} && (1.13) \end{aligned}$$

By writing $\mathbf{H}$ and $\mathbf{B}$ in terms of the vector potential $\mathbf{A}$ we arrive at the magnetostatic vector potential formulation,

$$\operatorname{curl}(\nu(|\operatorname{curl} \mathbf{A}|)\operatorname{curl} \mathbf{A}) = \mathbf{J} + \operatorname{curl} \mathbf{H}_0\,. \tag{1.14}$$

Before we perform further simplifications to the model we first inspect the reluctivity ν.

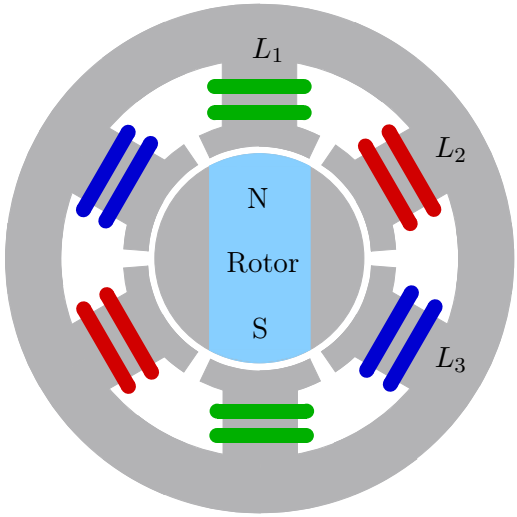


Figure 1.3: Brushless DC motor. Rotating magnetic field created by the sequential excitation of the pole pairs by a DC pulse.

1.1.2 Material laws

The material influence appears in form of the reluctivity ν which additionally depends on the position x. We can differentiate between three different kinds of materials: linear materials, where the reluctivity ν is a constant, permanent magnetic materials and nonlinear materials. Within one and the same material, ν is independent of the position.

Linear Materials

The most famous linear material is vacuum. It is well known, that vacuum behaves linearly, i.e.,

$$\begin{aligned} \mu_r &\equiv 1\,, && (1.15)\\ \nu &\equiv \frac{1}{\mu_0} =: \nu_0 && (1.16) \end{aligned}$$

where we recall that $\mu_0 = 4\pi \cdot 10^{-7}\left[\frac{Vs}{Am}\right]$ denotes the permeability of vacuum. For electromagnetic problems we can assume that air behaves just like vacuum.

Permanent magnetic Materials

For permanent magnetic materials we have the following material influence:

$$\mu_r \equiv 1\,. \tag{1.17}$$

From the previous subsection we know that for permanent magnetic materials we can assume

$$\mathbf{H}_0 \neq 0\,. \tag{1.18}$$

Nonlinear Materials

In our model problem we neglect hysteresis effects. In this case there exists a bijective mapping $f : \mathbb{R}_0^+ \to R_0^+$ such that

$$|\mathbf{B}| = f(|\mathbf{H}|)\,. \tag{1.19}$$

The function f is called B-H-curve or magnetization curve. For large values of $\mathbf{H}$ the degree of amplification behaves again like in vacuum - in this case we say the material is saturated.

These properties are summarized in the following assumption:

Assumption 1.1.1 Any B-H-curve

$$f : \mathbb{R}_0^+ \to R_0^+ \tag{1.20}$$

describing the relation $|\mathbf{B}| = f(|\mathbf{H}|)$ fulfills the following conditions:

1. f is continuously differentiable,
2. $f(0) = 0$,
3. $f'(s) \geq \mu_0 \quad (s \geq 0)$,
4. $f'(s) \overset{s\to\infty}{\longrightarrow} \mu_0$.

From (1.8), (1.19) and the assumption $\mathbf{H}_0 = 0$ for nonpermanent magnetic materials we know

$$\nu(|\mathbf{B}|) = \frac{|\mathbf{H}|}{|\mathbf{B}|} = \frac{f^{-1}(|\mathbf{B}|)}{|\mathbf{B}|} \tag{1.21}$$

So the reluctivity ν is related to f via

$$\nu(s) := \frac{f^{-1}(s)}{s} \tag{1.22}$$

We need several properties of f and ν that are summarized in the following lemma:

Lemma 1.1.2 With f fulfilling Assumption 1.1.1, the following statements holds:

1. The function $\nu(s) := \frac{f^{-1}(s)}{s}$ is well-defined and continuous in $[0;\infty)$ with
$$\nu(0) := (f^{-1})'(0) \tag{1.23}$$
and $\nu(s) \overset{s\to\infty}{\longrightarrow} \nu_0$.
2. The function $f^{-1}(s) = \nu(s)\,s$ is Lipschitz continuous with Lipschitz constant ν_0.
3. The function $f^{-1}(s) = \nu(s)\,s$ is strongly monotone with monotonicity constant $m > 0$, i.e.
$$(f^{-1}(s) - f^{-1}(t))(s-t) \geq m(s-t)^2 \quad (s,t \in \mathbb{R}_0^+)\,, \tag{1.24}$$
where
$$m := \min_{s\geq 0}(f^{-1})'(s)\,. \tag{1.25}$$
4. The function ν is continuously differentiable on $(0,\infty)$ and $\nu'(s) \overset{s\to\infty}{\longrightarrow} 0$.

1.1.3 Reduction to 2D

We consider a magnetic field problem on the x_1-x_2-plane. It is requested, that the electric current density $\mathbf{J}$ is perpendicular to the magnetic field $\mathbf{H}$, which should lie on the x_1-x_2-plane and that both fields are independend of x_3, i.e.

$$\mathbf{J} = \begin{pmatrix} 0 \\ 0 \\ J_3(x_1, x_2) \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} H_1(x_1, x_2) \\ H_2(x_1, x_2) \\ 0 \end{pmatrix}. \tag{1.26}$$

Additionally, we assume that

$$\mathbf{H}_0 = \begin{pmatrix} H_{01}(x_1, x_2) \\ H_{02}(x_1, x_2) \\ 0 \end{pmatrix}. \tag{1.27}$$

From this assumption we obtain

$$\operatorname{curl} \mathbf{H}_0 = \begin{pmatrix} 0 \\ 0 \\ -\frac{\partial}{\partial x_2} H_{01}(x_1, x_2) + \frac{\partial}{\partial x_1} H_{02}(x_1, x_2) \end{pmatrix}. \tag{1.28}$$

From the B-H-relation (1.6), we immediately get that B has the form

$$\mathbf{B} = \begin{pmatrix} B_1(x_1, x_2) \\ B_2(x_1, x_2) \\ 0 \end{pmatrix}. \tag{1.29}$$

Since the third component vanishes and $\mathbf{B} = \operatorname{curl} \mathbf{A}$, we find that

$$(\operatorname{curl}\mathbf{A})_3 = \frac{\partial A_2}{\partial x_1} - \frac{\partial A_1}{\partial x_2} = 0\,, \tag{1.30}$$

which leads to the following ansatz representing the vector potential

$$\mathbf{A} = \mathbf{A}(x_1, x_2) = \begin{pmatrix} 0 \\ 0 \\ A_3(x_1, x_2) \end{pmatrix}. \tag{1.31}$$

We conclude that

$$\mathbf{B} = \operatorname{curl} \mathbf{A} = \begin{pmatrix} \frac{\partial A_3}{\partial x_2} \\ -\frac{\partial A_3}{\partial x_1} \\ 0 \end{pmatrix}. \tag{1.32}$$

Combining (1.14), (1.26), (1.28), and (1.31), we obtain

$$-\operatorname{div}\left(\nu(|\nabla A_3|)\nabla A_3 = J_3 + (\operatorname{curl} H_0)_3\,. \right. \tag{1.33}$$

From now the unknown A_3 will be identified by u:

$$u := A_3\,. \tag{1.34}$$

To summarize, our 2D reduction yields to the scalar partial differential equation

$$-\operatorname{div}\left(\nu(|\nabla u|)\nabla u = J_3 - \frac{\partial}{\partial x_2} H_{01} + \frac{\partial}{\partial x_1} H_{02}\,. \right. \tag{1.35}$$

1.2 Variational Formulation

Before we derive the variational formulation of this problem, we define the Sobolev space $H^1(\Omega)$ (cf. [2]). Therefore we introduce a generalized concept of a derivative. Let Ω be a domain with boundary Γ. If for a function u there exists a continuous derivative $\partial u/\partial x_i$, we know with integration by parts that for every ∂x_i continuous differentiable function φ with $\varphi_\Gamma = 0$,

$$\int_\Omega u\frac{\partial \varphi}{\partial x_i}\,dx = -\int_\Omega \varphi\frac{\partial u}{\partial x_i}\,dx\,. \tag{1.36}$$

With this formula we define a derivative of functions, which are not necessarily differentiable in the classical sense. Let u and w be functions, which are integrable such that

$$\int_\Omega u\frac{\partial \varphi}{\partial x_i}\,dx = -\int_\Omega w\varphi\,dx \tag{1.37}$$

for every differentiable function φ with $\varphi_\Gamma = 0$, then we call w the weak derivative of u. With this new derivative it is possible to differentiate continuous, piecewise polynomial functions, which are used by the Finite Element Method. A function u of the space $L^2(\Omega)$ is an element of the space $H^1(\Omega)$, if all its partial derivatives of order one are in $L^2(\Omega)$.

1.2.1 The Variational Formulation

First, for simplicity, we consider the classical formulation: Let $\Omega \subset \mathbb{R}^2$ be a bounded domain with the boundary $\Gamma = \partial\Omega$. Find a function $u \in C^2(\Omega) \cap C(\overline{\Omega})$, such that the differential equation

$$-\text{div}\,(\nu(|\nabla u|)\nabla u = J_3 - \frac{\partial}{\partial x_2}H_{01} + \frac{\partial}{\partial x_1}H_{02} \quad \text{in } \Omega \tag{1.38}$$

and the homogeneous Dirichlet boundary condition

$$u = 0 \text{on } \Gamma \tag{1.39}$$

are satisfied. We are looking for a classical solution:

$$u \in C^2(\Omega) \cap C(\overline{\Omega}) \tag{1.40}$$

under classical assumptions imposed on the data

- $\nu \in C^1(\Omega)$
- $J_3 \in C(\Omega)$
- H_{01}, $H_{02} \in C^1(\Omega)$

We derive now the variational formulation of our problem. Under appropriate differentiability and integrability conditions the following steps can be performed:

1. Choose the space of test functions:

$$H^1_D(\Omega) = \{v \in H^1(\Omega) : v|_\Gamma = 0\} \tag{1.41}$$

2. Multiply the differential equation with an arbitrary test function $v \in H^1_D(\Omega)$ and integrate over the computational domain

$$\int_\Omega \left[-\text{div}\left(\nu(|\nabla u|)\nabla u + \begin{pmatrix} H_{02} \\ -H_{01}\end{pmatrix}\right)\right] v\,dx = \int_\Omega J_3\,v\,dx \tag{1.42}$$

3. Integration by parts in the principle part

$$
\begin{aligned}
&-\int_\Omega \left[\operatorname{div}\left(\nu(|\nabla u|)\nabla u + \begin{pmatrix} H_{02} \\ -H_{01}\end{pmatrix}\right)\right] v\, dx \\
&\quad = \quad -\int_\Omega v\left(\nu(|\nabla u|)\nabla u + \begin{pmatrix} H_{02} \\ -H_{01}\end{pmatrix}\right)\cdot \boldsymbol{n}\, ds \\
&\qquad\qquad\qquad + \int_\Omega \nabla v\left(\nu(|\nabla u|)\nabla u + \begin{pmatrix} H_{02} \\ -H_{01}\end{pmatrix}\right) dx
\end{aligned}
$$

The test-functions $u \in H^1_D(\Omega)$ vanish on the boundary Γ, such that

$$
\int_\Omega v\left(\nu(|\nabla u|)\nabla u + \begin{pmatrix} H_{02} \\ -H_{01}\end{pmatrix}\right)\cdot \boldsymbol{n}\, ds = 0\,. \tag{1.43}
$$

The result is the following nonlinear variational formulation of the boundary value problem: Find $u \in H^1_D(\Omega)$, such that

$$
a(u,v) = \langle F, v\rangle \quad (v \in H^1_D(\Omega)) \tag{1.44}
$$

where

$$
\begin{aligned}
a(u,v) &= \int_\Omega \nu(|\nabla u|)\nabla u\cdot\nabla v\, dx\,, \\
\langle F, v\rangle &= \int_\Omega \left(J_3\, v + H_{01}\frac{\partial v}{\partial y} - H_{02}\frac{\partial v}{\partial x}\right) dx\,.
\end{aligned}
$$

Here, $a(w;v)$ is only linear in the second argument v, but not necessarily in w. We equip the space V_0 with the norm $\|u\|_{V_0} := |u|_{H_1(\Omega)}$, which is indeed a norm due to Friedrichs' inequality (cf.[2]). It is relatively easy to show that $a(u,\cdot)$ is bounded for each fixed $u \in V_0$ and that F is a bounded linear form, in short $F \in V_0^*$. Then our problem can be rewritten as an operator equation in the dual space

$$
A(u) = F \in (H^1_D(\Omega))^*\,, \tag{1.45}
$$

with the nonlinear operator $A : H^1_D(\Omega) \to (H^1_D(\Omega))^*$ defined by the relation

$$
\langle A(u), v\rangle = a(u,v)\,, \tag{1.46}
$$

so it follows

$$
\langle A(u), v\rangle = \int_\Omega (\nu(x,|\nabla u|)\,\nabla u\cdot\nabla v\, dx\,. \tag{1.47}
$$

1.3 Existence and Uniqueness of the Solution

Existence and uniqueness of the solution to linear variational formulations is guaranteed by the Lax-Milgram theorem under suitable assumptions, in particular, ellipticity and boundedness of the bilinear form $a(\cdot,\cdot)$. In the nonlinear case, the operator equation (1.45) can be treated with a generalization of the Lax-Milgram theorem if the operator A is strongly monotone and Lipschitz continuous. Therefore, we first investigate these properties.

1.3.1 Conditions on the Reluctivity ν

Lemma 1.3.1 If $\nu(\cdot)\cdot : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is strongly monotone with monotonicity constant $m > 0$, i.e.

$$(\nu(t)t - \nu(s)s)(t-s) \geq m(t-s)^2 \quad (s,t \geq 0)\,, \tag{1.48}$$

then the nonlinear operator A defined by (1.46) is strongly monotone with monotonicity constant m, i.e.

$$\langle A(u) - A(v), u - v\rangle \geq m\|u-v\|^2_{H^1(\Omega)} \quad (u,v \in H^1_D(\Omega)) \tag{1.49}$$

Lemma 1.3.2 If $\nu(\cdot)\cdot : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ is Lipschitz continuous with Lipschitz constant $L > 0$, i.e.

$$|\nu(t)t - \nu(s)s| \leq L(t-s) \quad (s,t \geq 0)\,, \tag{1.50}$$

then the nonlinear operator A defined by (1.46) is Lipschitz continuous with Lipschitz constant $3L$, i.e.

$$\|A(u) - A(v)\|^2_{(H^1_D(\Omega))^*} \leq 3L\|u-v\|^2_{H^1(\Omega)} \quad (u,v \in H^1_D(\Omega)) \tag{1.51}$$

For existence and uniqueness we use the following Theorem , which is also known as the nonlinear Lax-Milgram theorem.

Theorem 1.3.3 — (8). Let $(V, (\cdot,\cdot)_V, \|\cdot\|_V)$ be a Hilbertspace, $F \in V^*$ and $A : V \to V^*$ a nonlinear operator, fulfilling the following conditions:

1. A is strongly monotone, i.e.

$$\exists c_1 > 0 : \langle A(u) - A(v), u - v\rangle \geq c_1\|u-v\|^2_V \quad (u,v \in V) \tag{1.52}$$

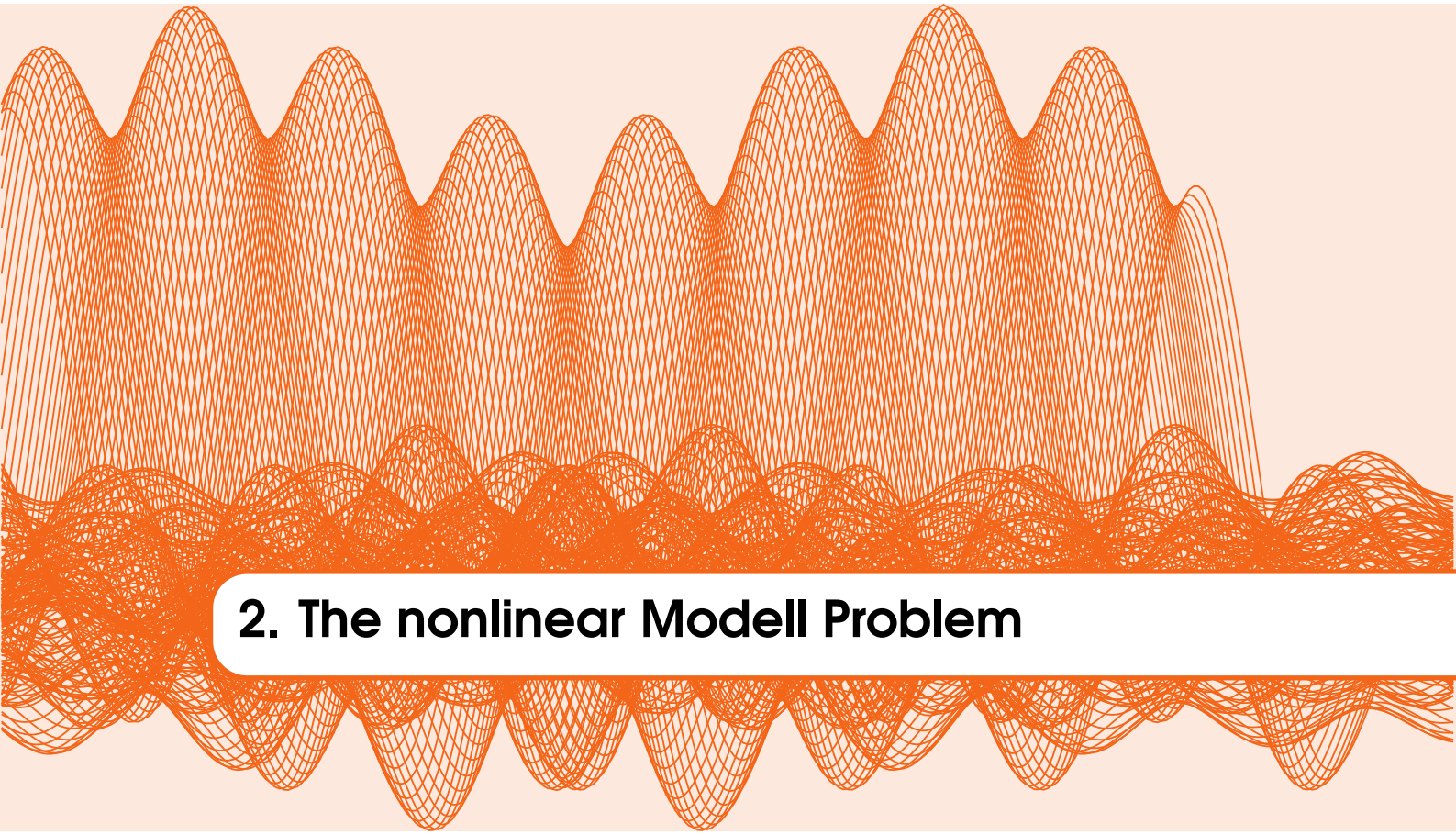
2. A is Lipschitz continuous, i.e.

$$\exists c_2 > 0 : \|A(u) - A(v)\|^2_{V^*} \leq c_2\|u-v\|^2_V \quad (u,v \in V) \tag{1.53}$$

Then the operator equation

$$A(u) = F \quad \text{in } V^* \tag{1.54}$$

has a uniquely determined solution $u^* \in V$.



2. The nonlinear Modell Problem

As discussed in Chapter 1 a two-dimensional stationary magnetic field problem involving the saturation effects of ferromagnetic materials can be written as a nonlinear boundary value problem in its variational formulation as follows:

Problem 2.0.1 Find $u \in H_0^D(\Omega)$ such that

$$a(u,v) = \langle f, v\rangle \quad (v \in H_0^D(\Omega)) \tag{2.1}$$

where

$$a(u,v) = \int_\Omega \nu(x, |\nabla u|)\, \nabla u \cdot \nabla v \, dx\,,$$

ans

$$\langle f, v\rangle = \int_\Omega \big(J\, v - H_{02}\frac{\partial v}{\partial x} + H_{01}\frac{\partial v}{\partial y}\big)\, dx\,.$$

Here, $\Omega \subset \mathbb{R}^2$ denotes a bounded domain.

The physical model has been developed from Maxwell's equation. We assume that Ω representing the cross section of some electromagnetic device lies in the x-y-plane of $\mathbb{R}^3$. Then, the solution u is the z- component of some vector potential A. The z- component of the current density is presented by J, and the vector $\mathbf{H}_0 = (H_{01}, H_{02}, 0)$ describes the magnetization of permanent magnets. The nonlinearity of the problem is represented by dependence of ν on the absolute value of the magnetic induction $B = |\mathrm{rot}\, A| = |\nabla u|$.

We assume that Ω consits of subdomains

$$\overline{\Omega} = \bigcup_{j=1}^{N_M} \overline{\Omega}_j, \text{ with } \Omega_j \cap \Omega_j = (i \neq j)$$

The Ω_j's represent materials with different magnetic properties (iron, copper, air, permanent magnetic materials) in the cross-section of an electromagnetic device. We assume that the function ν depends on the position $x \in \Omega$, but ν becomes independent of x inside

each subdomain Ω_j, i.e.

$$\nu(x, B) = \nu^{(j)}(B) \quad \text{if } x \in \Omega_j, \quad j = 1, \ldots, N_M \,.$$

The function $\nu^{(j)}(B)$ is constant, $\nu^{(j)}(B) \equiv \nu_1^{(j)}$, if the material in Ω_j is not ferromagnetic (e.g. copper, air, vacuum).

2.1 Material properties

Equation (2.1) denotes that we need to assign the values of the permeability for each material. For the nonlinear materials, such as the cobalt alloy and steel, the permeability as a function of the flux density is required. Manufacturers provide information about their material in the form of a B-H curve. In Tab. 2.1 data are given for an Cobalt-Nickel-Copper-Iron alloy. From these data we found the required function by nonlinear curve

Table 2.1: The B-H curve for the Cobalt-Nickel-Copper-Iron alloy

H (A/m)	20	60	80	95	105	120	140	160	180	200	240	2500
B (T)	0.19	0.65	0.87	1.04	1.18	1.24	1.272	1.3	1.32	1.34	1.36	1.45

fitting. For the Cobalt-Nickel-Copper alloy we will use the equation:

$$\mu_r(B) = \frac{1}{1e6}\left(8265\, e^{\frac{-(B-0.63)^{13}}{0.02}}\left(1 - \frac{e^{\frac{-B}{0.19}}}{2.4}\right) + 295\right). \tag{2.2}$$

From the data for the steel in Tab. 2.2 the corresponding equation takes the form:

$$\mu_r(B) = \frac{2.12146e4B.^2 - 7.248e4B + 6.167e4}{B^2 - 18.11B + 26.48}\,. \tag{2.3}$$

The data from Tab. 2.1 and 2.2 and the corresponding functions (2.2) and (2.3) are depicted in Fig. 2.1.

In the case of permanent magnets, manufacturers provide information in the form of a demagnetization curve, which lies in the second quadrant of the B-H plane. This curve indicates the remanent flux density B_r (for $H = 0$), the coercive field intensity H_c (for $B = 0$)), and the manner in which B and H vary between these two points. Rare-earth materials, such as Neodymium-Iron-Boron (NdFeB) magnets, exhibit an almost linear demagnetization curve and as a result the linear model used in (1.13) is sufficient for properly modeling them. From a manufacturer of NdFeB magnets datasheet we read $B_r = 1.16[T]$, $H_c = 883310[A/m]$ which according to:

$$B_r = \mu H_c$$

implies a relative permeability equal to 1.045.

Table 2.2: The B-H curve for the steel

H (A/m)	400	600	800	1000	1400	2000	3000	4000	6000
B (T)	0.73	0.92	1.05	1.15	1.28	1.42	1.52	1.58	1.60

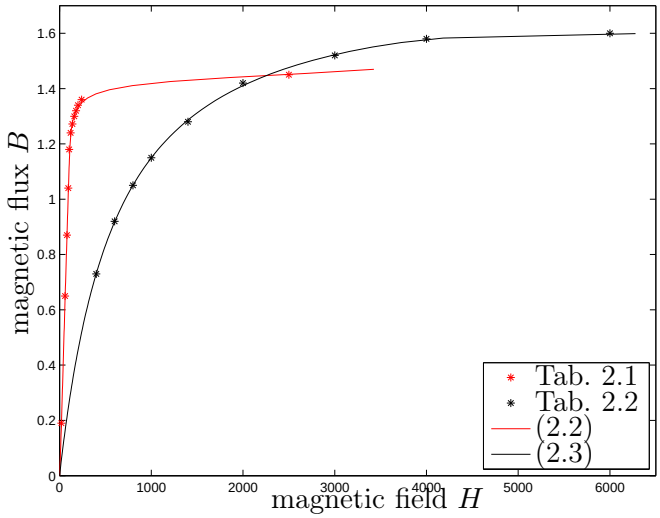


Figure 2.1: Data from the tables Tab. 2.1 and 2.2 and its fitted functions (2.2) and (2.3) .

2.2 Torque Calculation

The most important parameter to calculate is often the magnetically produced torque for a given rotor position and current excitation. For the torque calculation different methods exist. The most frequently used methods are those where the torque is calculated directly from the magnetic field solution in the motor. Such methods are the virtual work method, the magnetizing current method, and the Maxwell stress method (MSM). The accuracy of these methods depends to a great extend on the accuracy of the magnetic field calculation. In our case study we will use the MSM due to its simplicity.

In the MSM the torque is calculated on the basis of the magnetic field distribution on a closed surface in the air gap around the rotor [5, 6]. The differential torque produced is:

$$dT = \frac{1}{2}\big((r \times H)(n \cdot B) + (r \times B)(n \cdot H) - (r \times n)(H \cdot B)\big)$$

differential torque over that surface. Thus:

$$T = \frac{1}{2}\oint \big((r \times H)(n \cdot B) + (r \times B)(n \cdot H) - (r \times n)(H \cdot B)\big)\, dS$$

In the above equations r is the position vector of the point of integration and n denotes the unit vector normal to the surface. The torque is considered relative to the origin of the coordinate system located at the center of the stator.

If we consider the two-dimensional case and the origin of the rotor is $(0, 0)$ we have $r(\phi) = R(\cos(\phi), \sin(\phi), 0)$, $B = (\partial u/\partial x_2, -\partial u/\partial x_1, 0)$. In the air gap between rotor and stator we have $H_0 = 0$ and assume $\mu_r = 1$. Hence $H = \mu_0 B$ and $n = (\cos(\phi), \sin(\phi), 0)$, $(r \times n) = R(n \times n) = 0$. Furthermore

$$\begin{aligned}
(r \times H)(n \cdot B) &= (r \times B)(n \cdot H) = \mu_0 (r \times B)(n \cdot B) \\
&= -\mu_0 R\Big(\cos(\phi)\frac{\partial u}{\partial x_1} + \sin(\phi)\frac{\partial u}{\partial x_2}\Big)\Big(\cos(\phi)\frac{\partial u}{\partial x_2} - \sin(\phi)\frac{\partial u}{\partial x_1}\Big) \\
&= \mu_0 R \begin{pmatrix} \partial u/\partial x_1 \\ \partial u/\partial x_2 \end{pmatrix}^T \begin{pmatrix} \sin(\phi)\cos(\phi) & -\cos^2(\phi) \\ \sin^2(\phi) & -\sin(\phi)\cos(\phi) \end{pmatrix} \begin{pmatrix} \partial u/\partial x_1 \\ \partial u/\partial x_2 \end{pmatrix}.
\end{aligned}$$

Hence, with the axes length ℓ, we obtain

$$T = \mu_0 \ell R^2 \int_0^{2\pi} \begin{pmatrix} \partial u/\partial x_1 \\ \partial u/\partial x_2 \end{pmatrix}^T \begin{pmatrix} \sin(2\phi) & -1-\cos(2\phi) \\ 1-\cos(2\phi) & -\sin(2\phi) \end{pmatrix} \begin{pmatrix} \partial u/\partial x_1 \\ \partial u/\partial x_2 \end{pmatrix} d\phi$$

Using a midpoint rule we have to sum up the following term

$$\begin{aligned} & \int_\alpha^\beta \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}^T \begin{pmatrix} \sin(2\phi) & -1-\cos(2\phi) \\ 1-\cos(2\phi) & -\sin(2\phi) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} d\phi \\ \approx \quad & (\beta-\alpha)\Big((u_1^2 - u_2^2)\sin(\alpha+\beta) - 2u_1u_2\Big(\cos(\alpha+\beta)\Big) \end{aligned}$$

2.3 Geometry and Initial Mesh of BLDC Motor

We will use the following geometry for our BLDC motor excited by permanent magnets as real-life test example. Due to symmetries, we only need to fix a portion of the whole geometry. We describe the rotor and the stator separately. The edges are arcs or straight lines. The geometry of the stator is fixed by d_2, d_3, and $R_4, \dots R_9$, the geometry of the rotor as shown in Fig. 2.2 is given by d_1, $R_1, \dots R_3$.

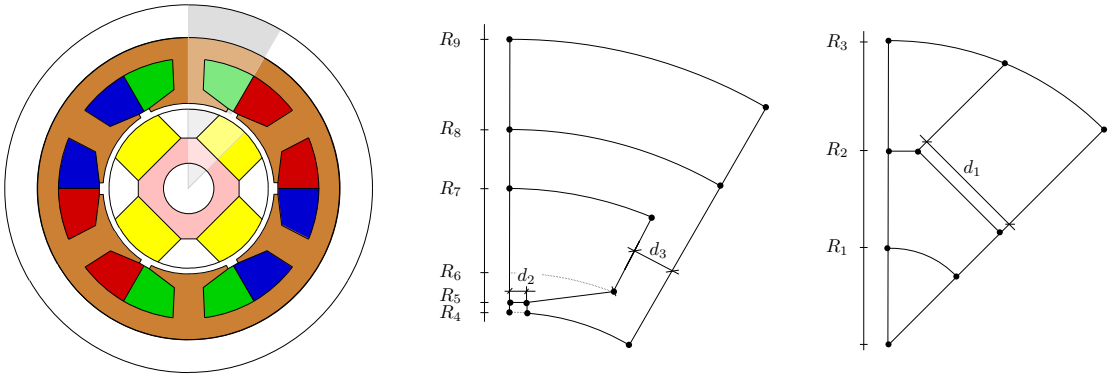


Figure 2.2: Geometry of BLDC motor, essential data of stator and rotor.

A coarse triangulation of stator and rotor as shown in Fig. 2.2 is presented in Fig. 2.3. For higher accuracy we will refine the coarse mesh uniformly. For our test example we

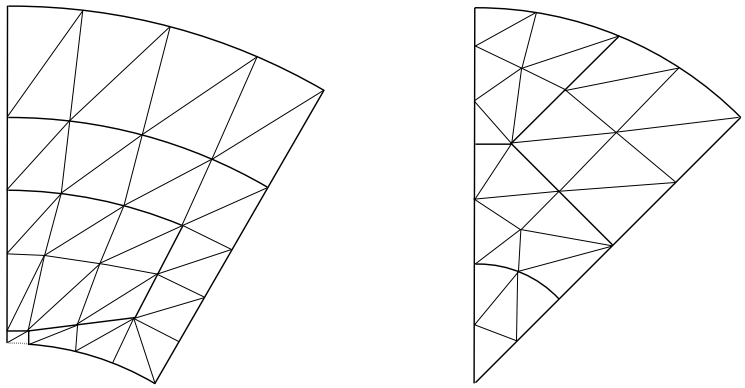


Figure 2.3: Coarse mesh of rotor and stator as shown in Fig. 2.2.

choose

Table 2.3: Geometry data for stator and rotor in test example.

d_1	d_2	d_3	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
1.3	0.2	0.3	1	2	3	3.3	3.5	3.6	4.5	5	6

A coarse mesh of the whole motor with paramater $d_1, \ldots, d_3$ and $R_1, \ldots R_9$ as given in Tab. 2.3 is shown in Fig. 2.4 . The colors indicate different materials or different functions J_3 and $\mathbf{H}_0$

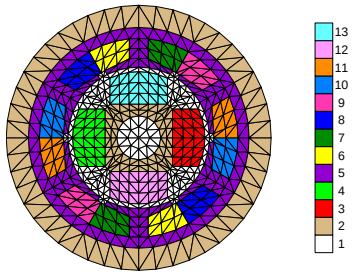


Figure 2.4: A coarse mesh of the whole motor and areas of different materials or load functions.

Table 2.4: Data sheet for first test example (Definitions in base units of S.I. Units)[1]

Data	Comment
$L = 3$	axes length of BLDC motor
$\mu_0 = 4\,\pi$ 1e-7 $\left[\frac{kg\,m}{s^2 A^2}\right]$	Permeability of free space
$\mu_r = 1$	Relative permeability of air (material 1)
$\mu_r = 1.045$	Relative permeability of magnets (materials 3,4,9,13)
$\mu_r(B)$	Relative permeability of Cobalt Alloy as a function of flux density given by (2.2)
$\mu_r(B)$	Relative permeability of steel as a function of flux density (2.3)
$J_1 = 0$	Current density in coils 6,7
$J_2 = \quad 1500000\ \left[\frac{A}{m^2}\right]$	Current density in coils 10, 11
$J_3 = -1500000\ \left[\frac{A}{m^2}\right]$	Current density in coils 8,9
$C = 883310\ \left[\frac{A}{m}\right]$	Coercive field intensity of NdFeB magnets
$\mathbf{H}_0 = (C, 0)$, $\mathbf{H}_0 = (-C, 0)$	Coercive field intensity vector of magnet 3 resp. 4
$\mathbf{H}_0 = (0, C)$, $\mathbf{H}_0 = (0, -C)$	Coercive field intensity vector of magnet 12 resp. 13

[1]The International System of Units consists of a set of base units, among others $[m]$, $[kg]$, $[s]$, and $[A]$.

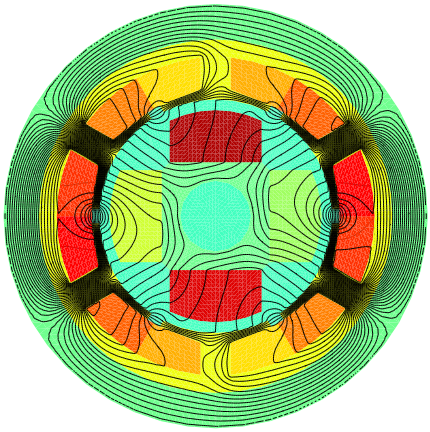


Figure 2.5: Equipotential lines of the magnetic vector potential for the motor and the data given in Tab. 2.4.

3. Finite Element Method

3.1 Short Matlab implementation

This chapter provides a simple and short open-box Matlab implementation of Courant's P_1 triangles for the numerical solutions of elliptic problems with mixed Dirichlet and Neumann boundary conditions. Based on four data files, arbitrary regular triangulations are determined. Instead of covering all kinds of possible problems in one code, the proposed tool aims to be simple, easy to understand and to modify. Therefore, only simple model examples are included to be adapted to whatever is needed.

The introduction to the finite element method and its Matlab implementation is organised as follows. As a model problem, a modified Poisson equation is described in Section 3.1.1. The discretisation is sketched in a mathematical language in Section 3.1.2. The heart of this contribution is the data representation of the triangulation, the Dirichlet and Neumann boundary as the three functions specifying f, g, uD, and OpA as described in Section 3.1.3 together with the discrete space. The main steps are the assembling procedure of the stiffness matrix in Section 3.1.4, the right-hand side in Section 3.1.5 and the incorporation of the Dirichlet boundary conditions in Section 3.1.6. A post-processing to preview the numerical solution is provided in Section 3.1.7. The application follows in Section **??** and illustrates the tool in a non-linear example.

3.1.1 Model Problem

The proposed Matlab program employs the finite element method to calculate a numerical solution U which approximates the solution u to the two dimensional Laplace problem (P) with mixed boundary conditions: Let $\Omega \subset \mathbb{R}^2$ be a bounded Lipschitz domain with polygonal boundary Γ.

The possibly nonlinear partial differential equation considered in Ω, the interior part

of the problem, is described by the operator $\mathcal{A}$ defined by

$$\mathcal{A} : L^2(\Omega;\mathbb{R}^2) \to L^2(\Omega;\mathbb{R}^2), \begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix} \mapsto \begin{pmatrix} a_{11}\left(\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix}\right)\cdot\varepsilon_1 + a_{12}\left(\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix}\right)\cdot\varepsilon_2 \\ a_{21}\left(\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix}\right)\cdot\varepsilon_1 + a_{22}\left(\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix}\right)\cdot\varepsilon_2 \end{pmatrix} \tag{3.1}$$

The coefficients $a_{ij} = a_{ji} \in L^\infty(\Omega)$ may or may not depend on the argument $\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\end{pmatrix} \in L^2(\Omega;\mathbb{R}^2)$ and may vary in Ω provided that $\mathcal{A}$ is uniformly bounded and monotone, i.e. there esists positive constants α_0 and α_1 with

$$\alpha_0|\delta-\varepsilon|^2 \le (\delta-\varepsilon)^T \begin{pmatrix} a_{11}(x,\delta)\delta_1 + a_{12}(x,\delta)\delta_2 - a_{11}(x,\varepsilon)\varepsilon_1 + a_{12}(x,\varepsilon)\varepsilon_2 \\ a_{21}(x,\delta)\delta_1 + a_{22}(x,\delta)\delta_2 - a_{21}(x,\varepsilon)\varepsilon_1 + a_{22}(x,\varepsilon)\varepsilon_2 \end{pmatrix} \le \alpha_1|\delta-\varepsilon|^2$$

for all $\delta, \varepsilon \in \mathbb{R}^2$ and for a.e. $x \in \Omega$.

■ **Example 3.1** As a typical example consider $\mathcal{A} = p \cdot I$ where I is the two-dimensional unit matrix. In the linear case $p \in L^\infty(\Omega)$ is a scalar function with $p_0 \le p(x) \le p_1$ for almost every $x \in \Omega$ and some global constants p_0, $p_1 > 0$:

$$(\mathcal{A}\varepsilon)(x) = p(x)\cdot\varepsilon \quad \text{for a.e. } x \in \Omega.$$

In particular $p = 1$ leads to the Laplacian operator. In the nonlinear case we consider p as a function of the argument $t := |\varepsilon|$ and may take e.g. $p(t) = 2 + \frac{1}{1+t}$ which gives

$$(\mathcal{A}\varepsilon)(x) = p(|\varepsilon|)\cdot\varepsilon \quad \text{for a.e. } x \in \Omega.$$

and $2 \le p(|\varepsilon|) \le 3$. ■

On some closed subset Γ_D of the boundary with positive length, we assume Dirichlet conditions, while we have Neumann boundary conditions on the remaining part $\Gamma_N := \Gamma \setminus \Gamma_D$. Given f_1, $\boldsymbol{f}_2 \in L^2(\Omega)$, $u_D \in H^1(\Omega)$, and $g \in L^2(\Gamma_N)$, seek $u \in H^1(\Omega)$ with

$$-\operatorname{div}\mathcal{A}(\nabla u) = f_1 + \operatorname{div}\boldsymbol{f}_2 \quad \text{in } \Omega, \tag{3.2}$$

$$u = u_D \quad \text{on } \Gamma_D, \tag{3.3}$$

$$\mathcal{A}(\nabla u)\cdot\boldsymbol{n} = g \quad \text{on } \Gamma_N, \tag{3.4}$$

where $\mathcal{A}$ is the (possibly nonlinear) operator considered above. According to the Zarantonellos Lemma, there always exists a weak solution to (3.2)-(3.4) which enjoys inner regularity (i.e. $u \in H^2_{loc}(\Omega)$), and envies regularity conditions owing to the smoothness of the boundary and the change of boundary conditions.

3.1.2 Galerkin Discretisation of the Linear Problem

We consider the solution of the linear problem first. Integration by parts gives rise to the induced form

$$a(u,v) := \int_\Omega \nabla v \cdot \mathcal{A}(\nabla u)\, dx\,. \tag{3.5}$$

The inhomogeneous Dirichlet conditions (3.3) are incorporated through the decomposition $u_0 = u - u_D$ so that $u_0 = 0$ on Γ_D, i.e., $v \in H^1_D(\Omega) := \{v \in H^1(\Omega) |\ v = 0 \text{ on } \Gamma_D\}$.

Then, the weak formulation of the boundary value problem (P) reads: Seek $u_0 \in H^1_D(\Omega)$, such that

$$a(u_0, v) = \int_\Omega f_1 v - \boldsymbol{f}_2 \cdot \nabla v \, dx + \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot \boldsymbol{n}) v \, ds - a(u_D, v) \quad (v \in H^1_D(\Omega)). \tag{3.6}$$

For the implementation, the linear problem (3.6) is discretised using the standard Galerkin–method, where $H^1(\Omega)$ and $H^1_D(\Omega)$ are replaced by finite dimensional subspaces S and $S_D = S \cap H^1_D$, respectively. Let $U_D \in S$ be a function that approximates u_D on Γ_D. (We define U_D as the nodal interpolant of u_D on Γ_D.) Then, the discretised problem (P_S) reads: Find $U_0 \in S_D$ such that

$$\int_\Omega \nabla V \cdot \mathcal{A}(\nabla U_0) \, dx = \int_\Omega f_1 V - \boldsymbol{f}_2 \cdot \nabla V \, dx + \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n) V \, ds - \int_\Omega \nabla V \cdot \mathcal{A}(\nabla U_D) \, dx \quad (V \in S_D). \tag{3.7}$$

Let $(\eta_1, \ldots, \eta_N)$ be a basis of the finite dimensional space S, and let $(\eta_{i_1}, \ldots, \eta_{i_M})$ be a basis of S_D, where $I = \{i_1, \ldots, i_M\} \subseteq \{1, \ldots, N\}$ is an index set of cardinality $M \leq N - 2$. Then, (3.7) is equivalent to

$$\int_\Omega \nabla \eta_j \cdot \mathcal{A}(\nabla U_0) \, dx = \int_\Omega f_1 \eta_j - \boldsymbol{f}_2 \cdot \nabla \eta_j \, dx + \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n) \eta_j \, ds - \int_\Omega \nabla \eta_j \cdot \mathcal{A}(\nabla U_D) \, dx \quad (j \in I). \tag{3.8}$$

Furthermore, let $U_0 = \sum_{k \in I} x_k \eta_k$ and $U_D = \sum_{k=1}^N U_k \eta_k$. Then, the equation (3.8) yields the linear system of equations

$$Ax = b. \tag{3.9}$$

The coefficient matrix $A = (A_{jk})_{j,k \in I} \in \mathbb{R}^{M \times M}$ and the right-hand side $b = (b_j)_{j \in I} \in \mathbb{R}^M$ are defined as

$$A_{jk} = \int_\Omega \nabla \eta_j \cdot \mathcal{A}(\nabla \eta_k) \, dx \tag{3.10}$$

and

$$b_j = \int_\Omega f_1 \eta_j - \boldsymbol{f}_2 \cdot \nabla \eta_j \, dx + \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n) \eta_j \, ds - \sum_{k=1}^N U_k \int_\Omega \nabla \eta_j \cdot \mathcal{A}(\nabla \eta_k) \, dx. \tag{3.11}$$

The coefficient matrix is sparse, symmetric and positive definite, so (3.9) has exactly one solution $x \in \mathbb{R}^M$ which determines the Galerkin solution $U = U_D + U_0 = \sum_{j=1}^N U_j \eta_j + \sum_{k \in I} x_k \eta_k$.

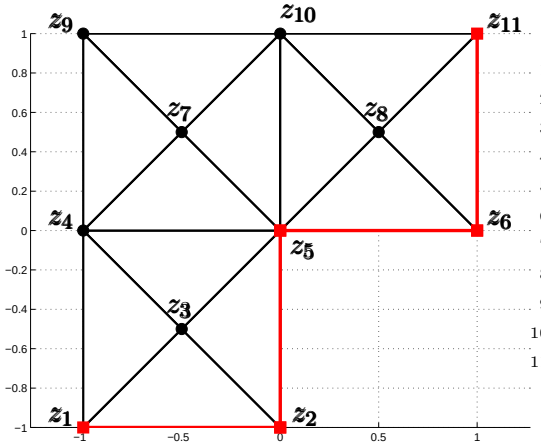
3.1.3 Data-Representation of the Triangulation Ω

Suppose the domain Ω has a polygonal boundary Γ, we cover $\bar{\Omega}$ by a regular triangulation $\mathcal{T}$ of triangles, i.e. $\bar{\Omega} = \bigcup_{T \in \mathcal{T}} T$ and each T is a closed triangle.

Regular triangulation in the sense of Ciarlet [4] means that the nodes $\mathcal{N}$ of the mesh lie on the vertices of the triangles, the elements of the triangulation do not overlap, no node lies on an edge of a triangle, and each edge $E \subset \Gamma$ of an element $T \in \mathcal{T}$ belongs either to $\bar{\Gamma}_N$ or to $\bar{\Gamma}_D$.

For the data representation of the set of all nodes $\mathcal{N} = \{z_1, \ldots, z_N\}$, the regular triangulation $\mathcal{T} = \{T_1, \ldots, T_J\}$, and the boundaries Γ_D and Γ_N, we follow [1]: We refer to Figure 3.1 for an exemplary triangulation $\mathcal{T}$ and corresponding data arrays, which are formally specified in the following:

The set of all nodes $\mathcal{N}$ is represented by the $N \times 2$ array `coordinates`. The ℓ-th row of `coordinates` stores the coordinates of the ℓ-th node $z_\ell = (x_\ell, y_\ell) \in \mathbb{R}^2$ as



`coordinates`

1	−1.0	−1.0
2	0.0	−1.0
3	−0.5	−0.5
4	−1.0	0.0
5	0.0	0.0
6	1.0	0.0
7	−0.5	0.5
8	0.5	0.5
9	−1.0	1.0
10	0.0	1.0
11	1.0	1.0

`elements`

1	1	2	3
2	2	5	3
3	5	4	3
4	4	1	3
5	4	5	7
6	5	10	7
7	10	9	7
8	9	4	7
9	5	6	8
10	6	11	8
11	11	10	8
12	10	5	8

`dirichlet`

1	1	2
2	2	5
3	5	6
4	6	11

`neumann`

1	11	10
2	10	9
3	9	4
4	4	1

Figure 3.1: Exemplary triangulation $\mathcal{T}$ of the L-shaped domain $\Omega = (-1,1)^2 \backslash ([0,1] \times [-1,0])$ into 12 triangles specified by the arrays `coordinates` and `elements`. The Dirichlet boundary, specified by the array `dirichlet`, consists of 4 edges which are plotted in red. The Neumann boundary is specified by the array `neumann` and consists of the remaining 4 boundary edges. The nodes $\mathcal{N} \cap \Gamma_D = \{z_1, z_2, z_5, z_6, z_{11}\}$ are indicated by red squares, whereas free nodes $\mathcal{N} \backslash \Gamma_D = \{z_3, z_4, z_7, z_8, z_9, z_{10}\}$ are indicated by black bullets.

$$\texttt{coordinates}(\ell,:) = [\ x_\ell \quad y_\ell\ ].$$

The triangulation $\mathcal{T}$ is represented by the $J \times 3$ integer array `elements`. The ℓ-th triangle $T_\ell = \operatorname{conv}\{z_i, z_j, z_k\} \in \mathcal{T}$ with vertices $z_i, z_j, z_k \in \mathcal{N}$ is stored as

$$\texttt{elements}(\ell,:) = [\ i \quad j \quad k\ ],$$

where the nodes are given in counterclockwise order, i.e., the parametrization of the boundary ∂T_ℓ is mathematically positive.

The Dirichlet boundary Γ_D is split into K affine boundary pieces, which are edges of triangles $T \in \mathcal{T}$. It is represented by a $K \times 2$ integer array `dirichlet`. The ℓ-th edge $E_\ell = \operatorname{conv}\{z_i, z_j\}$ on the Dirichlet boundary is stored in the form

$$\texttt{dirichlet}(\ell,:) = [\ i \quad j\ ].$$

It is assumed that $z_j - z_i$ gives the mathematically positive orientation of Γ, i.e.

$$n_\ell = \frac{1}{|z_j - z_i|} \begin{pmatrix} y_j - y_i \\ x_i - x_j \end{pmatrix},$$

gives the outer normal vector of Ω on E_ℓ, where $z_k = (x_k, y_k) \in \mathbb{R}^2$. Finally, the Neumann boundary Γ_N is stored analogously within an $L \times 2$ integer array `neumann`.

The spline spaces S and S_D are chosen globally continuous and affine on each triangular element and bilinear isoparametric on each quadrilateral element. In Fig. 3.2 we display a typical hat function η_j which are defined for every node (x_j, y_j) of the mesh by

$$\eta_j(x_k, y_k) = \delta_{jk} \quad (j,k = 1, \ldots, N)\,.$$

The subspace $S_D \subset S$ is the spline space which is spanned by all those η_j for which (x_j, y_j) does not lie on Γ_D. Then U_D, defined as the nodal interpolant of u_D lies in S_D.

With these spaces S and S_D and their corresponding basis, the integrals in (3.10), (3.11) can be calculated as a sum over all elements and a sum over all edges on Γ_N, i.e.,

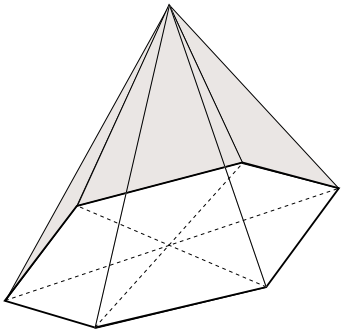


Figure 3.2: Piecewise affine hat Functions

$$A_{jk} = \sum_{T\in\mathcal{T}} \int_T \nabla\eta_j \cdot \mathcal{A}(\nabla\eta_k)\, dx\,, \tag{3.12}$$

$$b_j = \sum_{T\in\mathcal{T}} \int_T f_1\eta_j - \boldsymbol{f}_2 \cdot \nabla\eta_j\, dx + \sum_{E\subset\Gamma_N} \int_E (g + \boldsymbol{f}_2 \cdot n)\eta_j\, ds - \sum_{k=1}^{N} U_k \sum_{T\in\mathcal{T}} \int_T \nabla\eta_j \cdot \mathcal{A}(\nabla\eta_k)\, dx. \tag{3.13}$$

3.1.4 Assembling the Stiffness Matrix

The local stiffness matrix is determined by the coordinates of the vertices of the corresponding element and is calculated in the functions `stima.m`.

For a triangular element T let (x_1, y_1), (x_2, y_2) and (x_3, y_3) be the vertices and η_1, η_2 and η_3 the corresponding basis functions in S, i.e.

$$\eta_j(x_k, y_k) = \delta_{jk}, \quad j, k = 1, 2, 3.$$

A moments reflection reveals

$$\eta_j(x, y) = \det \begin{pmatrix} 1 & x & y \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{pmatrix} / \det \begin{pmatrix} 1 & x_j & y_j \\ 1 & x_{j+1} & y_{j+1} \\ 1 & x_{j+2} & y_{j+2} \end{pmatrix}, \tag{3.14}$$

whence

$$\nabla\eta_j(x, y) = \frac{1}{2|T|} \begin{pmatrix} y_{j+1} - y_{j+2} \\ x_{j+2} - x_{j+1} \end{pmatrix}. \tag{3.15}$$

Here the indices are to be understood modulo 3, and $|T|$ is the area of T, i.e.,

$$2\,|T| = \det \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix}. \tag{3.16}$$

Let $\bar{a}_{ij} := \int_T a_{ij}(x)\, dx / \int_T 1\, dx$ $(1 \le i, j \le 2)$. Since $\mathcal{A}$ is assumed to be linear, the resulting entry of the stiffness matrix is

$$M_{jk} = \int_T \nabla\eta_j \mathcal{A}(\nabla\eta_k)\, dx = \frac{|T|}{(2|T|)^2} (y_{j+1} - y_{j+2},\ x_{j+2} - x_{j+1}) \begin{pmatrix} \bar{a}_{11} & \bar{a}_{12} \\ \bar{a}_{21} & \bar{a}_{22} \end{pmatrix} \begin{pmatrix} y_{k+1} - y_{k+2} \\ x_{k+2} - x_{k+1} \end{pmatrix}$$

with indices modulo 3. This is written simultaneously for all indices as

$$M = \frac{|T|}{2} \cdot G \begin{pmatrix} \bar{a}_{11} & \bar{a}_{12} \\ \bar{a}_{21} & \bar{a}_{22} \end{pmatrix} G^T \quad \text{with} \quad G := \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

We approximate $\bar{a}_{ij}$ by $a_{ij}(m_T)$, where m_T is the center of gravity of T. Since we obtain similar formulae for three dimensions, the following Matlab routine works simultaneously for $d = 2$ and $d = 3$.

Listing 3.1: stima.m

```
function M = stima(vertices,material)
d = size(vertices,2);
G = [ones(1,d+1);vertices'] \ [zeros(1,d);eye(d)];
xM = sum(vertices)/(d+1);
A = OpA(xM,[],material);
M = det([ones(1,d+1);vertices']) * G * [A(1:2);A(2:3)] * G' / prod(1:d);
```

The values of a_{ij} of $\mathcal{A}$ are given by the function `OpA.m` which depends on the problem. The function is called with the coordinates of points in Ω and corresponding subdomain parameter. It returns the coeffitients a_{11}, $a_{12}(=a_{21})$, and a_{22} at these locations. For the numerical example shown in Figure 3.1 we used

Listing 3.2: OpA.m

```
function value = OpA(x,Du,material)
value = [ones(size(x,1),1),zeros(size(x,1),1),ones(size(x,1),1)];
```

3.1.5 Assembling the right-hand side

The volume forces are used for assembling the right hand side. Using the value of f_1 and $\boldsymbol{f}_2$ in the centre of gravity (x_S, y_S) of T the integrals $\int_T f_1 \eta_j \, dx$ and $\int_T \boldsymbol{f}_2 \cdot \nabla \eta_j \, dx$ in (3.13) are approximated by

$$\int_T f_1 \eta_j \, dx \approx \frac{|T|}{3} f_1(x_S, y_S),$$

and (using (3.15))

$$\int_T \boldsymbol{f}_2 \cdot \nabla \eta_j \, dx \approx \frac{1}{2} \left(y_{j+1} - y_{j+2}, x_{j+2} - x_{j+1}\right) \boldsymbol{f}_2(x_S, y_S).$$

with indices modulo 3. Therefore, with (3.16),

```
%*** Assembly of stiffness matrix
A = sparse(size(coordinates,1),size(coordinates,1));
for j = 1:size(elements,1)
  A(elements(j,:),elements(j,:)) = A(elements(j,:),elements(j,:)) ...
       + stima(coordinates(elements(j,:),:),material(j));
end
%*** Right hand side
b = zeros(size(coordinates,1),1);
for j = 1:size(elements,1)
```

The values of f_1, $\boldsymbol{f}_2$ are given by the function `f1.m`, `f2.m` which depend on the problem. The functions are called with the coordinates of points in Ω and it returns the volume forces at these locations. For the numerical example shown in Figure 3.1 we used

Listing 3.3: f1.m

```
function value = f1(x,material)
value = ones(size(x),1);
```

and

Listing 3.4: f2.m

```
function value = f2(x,material)
value = x;
```

Likewise, the Neumann conditions contribute to the right hand side. The integral $\int_E (g + \boldsymbol{f}_2 \cdot \boldsymbol{n})\eta_j \, ds$ in (3.13) is approximated using the value of the integrand in the centre (x_M, y_M) of E with length $|E|$ by

$$\int_E (g + \boldsymbol{f}_2 \cdot \boldsymbol{n})\eta_j \, ds \approx \frac{|E|}{2}(g(x_M, y_M) + \boldsymbol{f}_2(x_M, y_M) \cdot \boldsymbol{n}_E) \, .$$

```
  vertices = coordinates(elements(j,:),:);
  xM = sum(vertices)/3;
  b(elements(j,:)) = b(elements(j,:)) ...
   +det([1 1 1;vertices']) * f1(xM,material(j))/6 ...
   -1/2*(vertices([2,3,1],:)-vertices([3,1,2],:))*[0,-1;1,0]*f2(xM,material(j))';
end
%*** Neumann conditions
for j = 1 : size(neumann,1)
```

It is used here that in Matlab the size of an empty matrix is set equal to zero and that a loop of 1 through 0 is totally omitted. In that way, the question of the existence of Neumann boundary data is to be renounced.

The values of g are given by the function `g.m` which again depends on the problem. The function is called with the coordinates of points on Γ_N and returns the corresponding stresses. For the numerical example `g.m` was

Listing 3.5: g.m

```
function value = g(x,n)
value = zeros(size(x),1);
```

3.1.6 Incooperating Dirichlet conditions

With a suitable numbering of the nodes the system of linear equations resulting from the construction described in the previous section without incooperating Dirichlet conditions can be written as follows

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix} \cdot \begin{pmatrix} \vec{U} \\ \vec{U}_D \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{b}_D \end{pmatrix}, \tag{3.17}$$

with $\vec{U} \in \mathbb{R}^M$, $\vec{U}_D \in \mathbb{R}^{N-M}$. Here $\vec{U}$ are the values at the free nodes which are to be determined, $\vec{U}_D$ are the values at the nodes which are on the Dirichlet boundary and thus are known a priori. Hence, the first block of equations can be rewritten as

$$A_{11} \cdot \vec{U} = \vec{b} - A_{12} \cdot \vec{U}_D.$$

In fact, this is the formulation of (3.8) with $U_D = 0$ at non-Dirichlet nodes.

In the second block of equations in (3.17) the unknown is $\vec{b}_D$ but since it is not of our interest it is omitted in the following.

```
n = coordinates(neumann(j,2),:) - coordinates(neumann(j,1),:);
dist = norm(n);
n = ([0,1;-1,0]*n')/dist;
xM = sum(coordinates(neumann(j,:),:))/2;
b(neumann(j,:))=b(neumann(j,:)) + 0.5 * dist * (g(xM)+f2(xM,material(j))*n);
```

The values u_D at the nodes on Γ_D are given by the function `uD.m` which depends on the problem. The function is called with the coordinates of points in Γ_D and returns the values at the corresponding locations. For the numerical example `uD.m` was

Listing 3.6: uD.m

```
1 function value = uD(x)
2 value = zeros(size(x,1),1);
```

3.1.7 Computation and Displaying the numerical solution

The rows of (3.9) corresponding to the first M rows of (3.17) form a reduced system of equations with a symmetric, positive definite coefficient matrix A_{11}. It is obtained from the original system of equations by taking the rows and columns corresponding to the free nodes of the problem. The restriction can be achieved in Matlab through proper indexing.

The system of equations is solved by the binary operator \ installed in Matlab which gives the left inverse of a matrix.

```
end
%*** Prescribe values at Dirichlet nodes
u = zeros(size(coordinates,1),1);
```

Matlab makes use of the properties of a symmetric, positive definite and sparse matrix for solving the system of equations efficiently.

Using the data structure from above, we may visualize the solution $U = \sum_{j=1}^{N} \mathbf{u}_j V_j \in \mathcal{S}^1(\mathcal{T})$ by

```
dirichlet = unique(dirichlet);
u(dirichlet) = uD(coordinates(dirichlet,:));
```

Here, the column vector $\mathbf{u}_j = U(z_j)$ contains the nodal values of U at the j-th node $z_j \in \mathbb{R}^2$ given by `coordinates(j,:)`.

Here, the Matlab routine `trisurf(ELEMENTS,X,Y,U)` is used to draw triangulations for equal types of elements. Every row of the matrix `ELEMENTS` determines one polygon where the x-, y-, and z-coordinate of each corner of this polygon is given by the corresponding entry in `X`, `Y` and `U` resp. The colour of the polygons is given by values of `U`. The additional parameters `,'facecolor','interp'` lead to an interpolated colouring.

Figure 3.3 shows the solution for the mesh defined in Section 3.1.3 and the data files `f1.m`, `f2.m`, `g.m`, `uD.m` and `OpA.m` given in Sections 3.1.5 and 3.1.6.

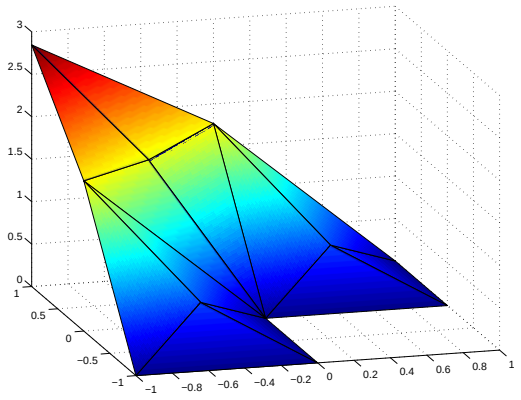


Figure 3.3: Solution for the Laplace problem

Listing 3.7: A First MATLAB Implementation

```
load coordinates.dat;
load elements.dat;
eval('load neumann.dat;','neumann=[];');
eval('load material.dat;','material=ones(size(elements,1),1);');
load dirichlet.dat;
%*** Assembly of stiffness matrix
A = sparse(size(coordinates,1),size(coordinates,1));
for j = 1:size(elements,1)
  A(elements(j,:),elements(j,:)) = A(elements(j,:),elements(j,:)) ...
       + stima(coordinates(elements(j,:),:),material(j));
end
%*** Right hand side
b = zeros(size(coordinates,1),1);
for j = 1:size(elements,1)
  vertices = coordinates(elements(j,:),:);
  xM = sum(vertices)/3;
  b(elements(j,:)) = b(elements(j,:)) ...
   +det([1 1 1;vertices']) * f1(xM,material(j))/6 ...
   -1/2*(vertices([2,3,1],:)-vertices([3,1,2],:))*[0,-1;1,0]*f2(xM,material(j))';
end
%*** Neumann conditions
for j = 1 : size(neumann,1)
   n = coordinates(neumann(j,2),:) - coordinates(neumann(j,1),:);
   dist = norm(n);
   n = ([0,1;-1,0]*n')/dist;
   xM = sum(coordinates(neumann(j,:),:))/2;
   b(neumann(j,:))=b(neumann(j,:)) + 0.5 * dist * (g(xM)+f2(xM,material(j))*n);
end
%*** Prescribe values at Dirichlet nodes
u = zeros(size(coordinates,1),1);
dirichlet = unique(dirichlet);
u(dirichlet) = uD(coordinates(dirichlet,:));
```

3.2 First Matlab Implementation for the Nonlinear Modell Problem

Before we discuss the solution of Eq. (3.2)-(3.4) if $\mathcal{A}$ is a nonlinear operator defined by (3.1), we will give some remarks on Newton's method.

3.2.1 Newtons Method

Let $F : \mathbb{R}^N \to \mathbb{R}^N$ be a continuously differentiable function. A nonlinear system is of the form

$$\mathbf{x} \in \mathbb{R}^N, \quad F(\mathbf{x}) = 0\,. \tag{3.18}$$

The the Newton method is

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \Big(DF(\mathbf{x}^k)\Big)^{-1} F(\mathbf{x}^k), \tag{3.19}$$

which can also be written in the form

$$DF(\mathbf{x}^k)\,\mathbf{s}^k = -F(\mathbf{x}^k), \quad \mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k\,. \tag{3.20}$$

So at each step, we solve a linear system. The method breaks down when $DF(\mathbf{x}^k)$ is singular or nearly singular. Under certain assumptions Newton's method converges locally and quadratically.

In order to save unnecessary iteration steps, we would like to know as early as possible whether the Newton iteration converges. Therefore we are looking for a convergence criterion that can be verified within the algorithm itself, and allows us to decide after one, or a few steps, if Newton's method converges. Often, the natural monotonicity test is suggested, i.e.

$$\|DF(\mathbf{x}^k)^{-1}F(\mathbf{x}^{k+1})\| \le \overline{\theta}\|DF(\mathbf{x}^k)^{-1}F(\mathbf{x}^k)\|\,. \tag{3.21}$$

On the right-hand side we recognize the Newton correction $\mathbf{s}^k = -DF(\mathbf{x}^k)^{-1}F(\mathbf{x}^k)$ that has to be computed anyway. On the left-hand side we detect the simplified Newton correction $\bar{\mathbf{s}}^{k+1}$ as the solution of the linear equation system

$$\Big(DF(\mathbf{x}^k)\Big)^{-1}\bar{\mathbf{s}}^{k+1} = -F(\mathbf{x}^{k+1})\,.$$

With this notation the natural monotonicity test (3.21) can be written as

$$\|\bar{\mathbf{s}}^{k+1}\| \le \overline{\theta}\|\mathbf{s}^k\|\,.$$

For the simplified Newton correction we obviously have to solve another system of linear equations with the same matrix $DF(\mathbf{x}^k)$, but with different right-hand side $F(\mathbf{x}^{k+1})$ evaluated at the next iterate

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k\,.$$

One possibility to globalize the local convergence of the ordinary Newton method is the damping of the Newton corrections $\mathbf{s}^k$. This leads to the modified iteration

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k\mathbf{s}^k, \quad \lambda_k \in (0,1]\,.$$

where λ_k is the damping factor. As a simple damping strategy we recommend choosing the damping factors λ_k such that the natural monotonicity test is satisfied with $\overline{\theta} = 1 - \lambda_k/2$, i.e.

$$\|\Big(DF(\mathbf{x}^k)\Big)^{-1}F(\mathbf{x}^k + \lambda_k s^k)\| \le \left(1 - \frac{\lambda_k}{2}\right)\|\Big(DF(\mathbf{x}^k)\Big)^{-1}F(\mathbf{x}^k)\|\,.$$

In the simplest implementation we choose a threshold value $\lambda_{\min} \ll 1$ and damping factors λ_k from some sequence, e.g.,

$$\{1, \frac{1}{2}, \frac{1}{4}, \ldots, \lambda_{\min}\}\,.$$

If $\lambda_k < \lambda_{\min}$ the iteration is terminated. In critical examples one will start with $\lambda_0 = \lambda_{\min}$, in harmless examples preferably with $\lambda_0 = 1$. Whenever λ_k was successful, we set

$$\lambda_{k+1} = \min\{1, 2\lambda_k\}$$

in the next iteration to attain asymptotically the quadratic convergence of the ordinary Newton method (with $\lambda_k = 1$ throughout). If the monotonicity test for λ_k fails, we try again with $\lambda_k/2$. A short Matlab implementation if given below.

Listing 3.8: Newton.m

```
function [u,nit] = newton(u,F,DF,tol,maxit,param)
Fu = F(u,param);
DFu = DF(u,param);
s = -DFu\Fu;
lam = 1;
tmp = max(tol,tol*norm(s));
nit = 0;
while norm(s) > tmp && nit <= maxit
  nit = nit + 1;
  u_old = u;
  lam = min(1,2*lam);
  for k=1:30
    u = u_old + lam * s; %*** Damping with parameter lam
    Fu = F(u,param);
    if norm(DFu\Fu) <= (1-lam/2) * norm(s)
      break               %*** Stop for-loop, if
    end                   %    convergencetest is satisfied
    lam = lam/2;          %*** lam too largen --> try lam/2
  end
  DFu = DF(u,param);
  s   = -DFu\Fu;
end
```

3.2.2 Discrete Nonlinear Problem

As in Subsection 3.1.2, let $(\eta_1, \ldots, \eta_N)$ be a basis of the finite dimensional space S, and let $(\eta_{i_1}, \ldots, \eta_{i_M})$ be a basis of S_D, where $I = \{i_1, \ldots, i_M\} \subseteq \{1, \ldots, N\}$ is an index set of cardinality $M \leq N - 2$.

The nonlinear problem reads now: Find $U_0 \in S_D$, s.t.

$$\int_\Omega \nabla\eta_j \cdot \mathcal{A}(\nabla U_0 + U_D) - f_1\eta_j + \boldsymbol{f}_2 \cdot \nabla\eta_j \, dx - \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n)\eta_j \, ds = 0 \quad (j \in I). \tag{3.22}$$

With a suitable numbering of the nodes described in the previous section let $U_0 = \sum_{k=1}^{M} x_k\eta_k$ and $U_D = \sum_{k=1}^{N-M} u_k\eta_{M+k}$. with $\mathbf{x} \in \mathbb{R}^M$, $\mathbf{u} \in \mathbb{R}^{N-M}$. Here $\mathbf{x}$ are the values at the free nodes which are to be determined, $\mathbf{u}$ are the values at the nodes which are on the Dirichlet boundary and thus are known a priori. Then, the equation (3.22) yields the nonlinear system of equations

$$\mathbf{x} \in \mathbb{R}^M\,, \quad F(\mathbf{x}) = 0\,, \tag{3.23}$$

with

$$\begin{aligned} F_j(\mathbf{x}) \quad := \quad & \int_\Omega \nabla\eta_j \cdot \mathcal{A}\left(\sum_{k=1}^{M} x_k\nabla\eta_k + \sum_{k=1}^{N-M} u_k\nabla\eta_{M+k}\right) \\ & -f_1\eta_j + \boldsymbol{f}_2 \cdot \nabla\eta_j \, dx - \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n)\eta_j \, ds \quad (j \in \{1, \ldots, M\}). \end{aligned} \tag{3.24}$$

3.2.3 Assembling the Nonlinear Function F

The vectorvalued nonlinear function F is determined by the coordinates of the vertices of the corresponding element, the function U, and and the operator $\mathcal{A}$. Its linear part, i.e.

$$\int_\Omega f_1\eta_j - \boldsymbol{f}_2 \cdot \nabla\eta_j \, dx + \int_{\Gamma_N} (g + \boldsymbol{f}_2 \cdot n)\eta_j \, ds \quad (j \in \{1, \dots, M\})$$

is calculated only once and stored in a structure field `para.b`. For further details see Subsection 3.1.5. See line 16-33 in `solveLaplaceNonLinear.m`.

```
16 %*** Right hand side
17 para.b = zeros(size(coordinates,1),1);
18 for j = 1:size(elements,1)
19   vertices = coordinates(elements(j,:),:);
20   xM = sum(vertices)/3;
21   para.b(elements(j,:)) = para.b(elements(j,:)) ...
22    +det([1 1 1;vertices']) * f1(xM,material(j))/6 ...
23    -1/2*(vertices([2,3,1],:)-vertices([3,1,2],:))*[0,-1;1,0]*f2(xM,material(j))';
24 end
25 %*** Neumann conditions
26 for j = 1 : size(neumann,1)
27    n = coordinates(neumann(j,2),:) - coordinates(neumann(j,1),:);
28    dist = norm(n);
29    n = ([0,1;-1,0]*n')/dist;
30    xM = sum(coordinates(neumann(j,:),:))/2;
31    para.b(neumann(j,:)) = para.b(neumann(j,:)) ...
32                         + 0.5 * dist * (g(xM)+f2(xM,material(j))*n);
33 end
```

The nonlinear term, i.e.

$$\int_\Omega \nabla\eta_j \cdot \mathcal{A}\left(\sum_{k=1}^{M} x_k\nabla\eta_k + \sum_{k=1}^{N-M} u_k\nabla\eta_{M+k}\right) \quad (j \in \{1, \dots, M\})$$

is calculated in the function `F.m`, where also the linear term is subtracted to compute F as given in (3.24).

```
function Fu = F(u,param)
%*** Assembly of stiffness matrix
d = zeros(size(param.coordinates,1),1);
for j = 1:size(param.elements,1)
  nodes = param.elements(j,:);
  vertices = param.coordinates(nodes,:);
  G = [ones(1,3);vertices'] \ [zeros(1,2);eye(2)];
  Du = G'*u(nodes);
  xM = sum(vertices)/3;
  cA = OpA(xM,Du,param.material(j));
  d(nodes) = d(nodes) ...
  + 1/2*det([ones(1,3);vertices']) * G * [cA(1:2);cA(2:3)] * Du;
end
%*** Computation of F(u) = d(u) - b
Fu = zeros(size(param.coordinates,1),1);
Fu(param.freenodes) = d(param.freenodes) - param.b(param.freenodes);
```

3.2.4 Assembling the Jacobian DF

The Jacobian matrix DF is determined by the coordinates of the vertices of the corresponding element, the function U, and the operator $\mathcal{A}$ and its partial derivatives. It is calculated in the function `DF.m`.

The nonlinearity for our test example of a BLDC motor has the form

$$(\mathcal{A}\varepsilon)(x) = p(|\varepsilon|)\cdot\varepsilon \quad \text{for a.e. } x\in\Omega \tag{3.25}$$

with $p:\mathbb{R}_0^+\to\mathbb{R}_0^+$ satisfying the assumptions of Theorem 1.3.3.

Let $U=\sum_j \alpha_j\varphi_j$ be given. To derive the Jabobian of F we have compute the derivative of $\mathcal{A}(\nabla U)$ with respect to α_k $(k=1,\dots,M)$. There holds

$$\begin{aligned}
&\frac{\partial}{\partial\alpha_k}\left\{p\left(x,|\textstyle\sum_j\alpha_j\nabla\varphi_j|\right)\sum_j\alpha_j\nabla\varphi_j\right\}\\
&= p\left(x,|\textstyle\sum_j\alpha_j\nabla\varphi_j|\right)\nabla\varphi_k + p_{,2}\left(x,|\textstyle\sum_j\alpha_j\nabla\varphi_j|\right)\frac{(\sum_j\alpha_j\nabla\varphi_j)^T\nabla\varphi_k}{|\sum_j\alpha_j\nabla\varphi_j|}\sum_j\alpha_j\nabla\varphi_j\\
&= \left(p\left(x,|\textstyle\sum_j\alpha_j\nabla\varphi_j|\right)I_{2\times2} + \frac{p_{,2}\left(x,|\sum_j\alpha_j\nabla\varphi_j|\right)}{|\sum_j\alpha_j\nabla\varphi_j|}(\textstyle\sum_j\alpha_j\nabla\varphi_j)(\sum_j\alpha_j\nabla\varphi_j)^T\right)\nabla\varphi_k\\
&=: G(x,\textstyle\sum_j\alpha_j\nabla\varphi_j;p,p_{,2})\nabla\varphi_k\,.
\end{aligned}$$

Notice, G is a symmetric operator. Hence, the resulting Jacobian DF is also symmetric. We have

$$(DF)_{ij}(\mathbf{x}) = \int_\Omega \nabla\varphi_j\cdot G(x,\textstyle\sum_k\alpha_k\nabla\varphi_k;p,p_{,2})\nabla\varphi_j\,dx$$

This can be computed in a similiar way as the stiffness matrix in Subsection 3.1.4. Do avoid difficulties with a vanishing denominator in G we expect functions p and $Dp(\delta) := p_{,2}(\cdot,\delta)/\delta$. The function values of G are given by the function `OpA.m` which depends on the problem. The function is called with coordinates of points in ω, the gradient of U, and the index of the material. It returns the values at the corresponding locations in the following way if `[v,Dv]=OpA(x,Du,m)`:

$$\mathrm{v} = \begin{vmatrix} a_{11}(x(1,:),Du(1,:),m(1)), & a_{12}(x(1,:),Du(1,:),m(1)), & a_{22}(x(1,:),Du(1,:),m(1))\\ a_{11}(x(2,:),Du(2,:),m(2)), & a_{12}(x(2,:),Du(2,:),m(2)), & a_{22}(x(2,:),Du(2,:),m(2))\\ a_{11}(x(3,:),Du(3,:),m(3)), & a_{12}(x(3,:),Du(3,:),m(3)), & a_{22}(x(3,:),Du(3,:),m(3))\\ \vdots & \vdots & \vdots \end{vmatrix}$$

and

$$\mathrm{Dv} = \begin{vmatrix} G_{11}(x(1,:),Du(1,:),m(1)), & G_{12}(x(1,:),Du(1,:),m(1)), & G_{22}(x(1,:),Du(1,:),m(1))\\ G_{11}(x(2,:),Du(2,:),m(2)), & G_{12}(x(2,:),Du(2,:),m(2)), & G_{22}(x(2,:),Du(2,:),m(2))\\ G_{11}(x(3,:),Du(3,:),m(3)), & G_{12}(x(3,:),Du(3,:),m(3)), & G_{22}(x(3,:),Du(3,:),m(3))\\ \vdots & \vdots & \vdots \end{vmatrix}$$

For the numerical example `OpA.m` was

```
function [val,Dval] = OpA(x,Du,material)
absDu = norm(Du);
val = [p(absDu,material),zeros(size(x,1),1),p(absDu,material)];
if nargout>1
```

```
  Dval = val;
  if absDu > 1e-12
    Dval = Dval + Dp(absDu,material)*...
                  [Du(1)^2,Du(1)*Du(2),Du(2)^2];
  end
end

function value = p(t,material)
if material ==1
  value = 1;
elseif material == 2
  value = 2-1/(1+t);
end

function value = Dp(t,material)
if material ==1
  value = 0;
elseif material == 2
  value = +1./(t*(1+t).^2);
end
```

```
function DFu = DF(u,para)
%*** Assembly of Jacobian matrix
DFu = sparse(size(para.coordinates,1),size(para.coordinates,1));
for j = 1:size(para.elements,1)
  vertices = para.coordinates(para.elements(j,:),:);
  G = [ones(1,3);vertices'] \ [zeros(1,2);eye(2)];
  [val,Dval] = OpA(sum(vertices)/3,G'*u(para.elements(j,:)),para.material(j));
  DFu(para.elements(j,:),para.elements(j,:)) = ...
    DFu(para.elements(j,:),para.elements(j,:))  ...
  + 1/2*det([ones(1,3);vertices']) * G * [Dval(1,1:2);Dval(1,2:3)] * G';
end
```

3.2.5 Computation the numerical solution

The system of nonlinear equations is solved by the subroutine `Newton.m` discussed in Subsection 3.2.1.

```
%*** Solve with Newton and variable step size
[u,nit] = newton(u,@F,@DF,1e-10,20,para);
fprintf('No. of Newton iterations = %3i\n',nit)
```

Figure 3.4 shows the solution for the mesh defined in Section 3.1.3 and the data files `f1.m`, `f2.m`, `g.m`, and `uD.m` given in Subections 3.1.5 and 3.1.6 and `OpA.m` given in Subections 3.2.4. Elements $1,\ldots,4$ and $9,\ldots,12$ belong to material 1 and elements $5,\ldots,9$ belong to material 2. The index of the material is represented by the $J \times 1$ array `material`. The material index k corresponding to the ℓ's element is stored as

$$\texttt{material}(\ell) = [\, k \,] \ .$$

In our first nonlinear example we have chosen p in (3.25)

$$p(t) = 1 \quad \text{for material 1 and} \qquad p(t) = 2 + \frac{1}{1+t} \quad \text{for material 2.}$$

Newton's method needed 4 iterations to satisfy the required tolerance of $1e-10$.

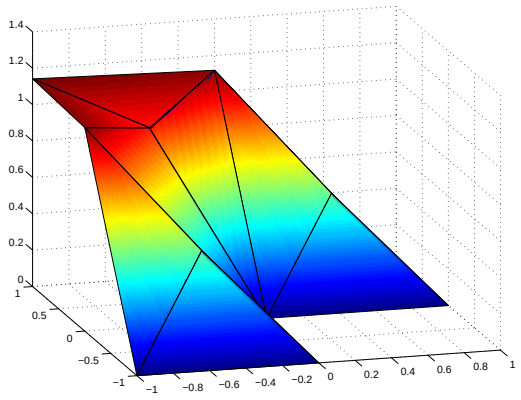


Figure 3.4: Solution for the nonlinear problem with 12 elements

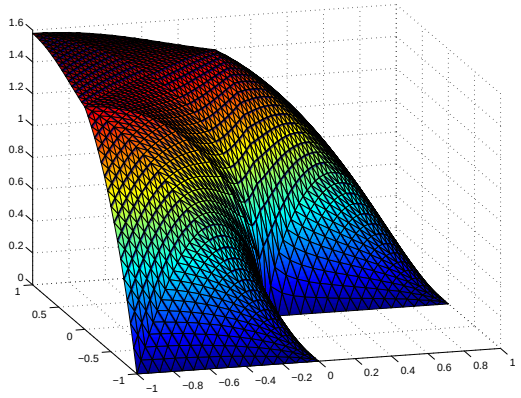


Figure 3.5: Solution for the nonlinear problem with 3073 elements

Notice, even for a finer mesh, there is still a jump of the gradient of the solution at $x = 0$ and $y = 0$ due to the jump in the function p for different materials.

Listing 3.9: A First MATLAB Implementation

```
% Initialisation
load coordinates.dat;
load elements.dat;
eval('load neumann.dat;','neumann=[];');
eval('load material.dat;','material=ones(size(elements,1),1);');
load dirichlet.dat;
%*** Prescribe values at Dirichlet nodes
para.dirichlet = unique(dirichlet);
u = zeros(size(coordinates,1),1);
u(para.dirichlet) = uD(coordinates(para.dirichlet,:));
%*** Store mesh as structure
para.elements = elements;
para.coordinates = coordinates;
para.material = material;
para.freenodes = setdiff(1:size(coordinates,1), para.dirichlet);
%*** Right hand side
para.b = zeros(size(coordinates,1),1);
for j = 1:size(elements,1)
  vertices = coordinates(elements(j,:),:);
  xM = sum(vertices)/3;
  para.b(elements(j,:)) = para.b(elements(j,:)) ...
```

```
22     +det([1 1 1;vertices']) * f1(xM,material(j))/6 ...
23     -1/2*(vertices([2,3,1],:)-vertices([3,1,2],:))*[0,-1;1,0]*f2(xM,material(j))';
24 end
25 %*** Neumann conditions
26 for j = 1 : size(neumann,1)
27     n = coordinates(neumann(j,2),:) - coordinates(neumann(j,1),:);
28     dist = norm(n);
29     n = ([0,1;-1,0]*n')/dist;
30     xM = sum(coordinates(neumann(j,:),:))/2;
31     para.b(neumann(j,:)) = para.b(neumann(j,:)) ...
32                          + 0.5 * dist * (g(xM)+f2(xM,material(j))*n);
33 end
34 %*** Solve with Newton and variable step size
35 [u,nit] = newton(u,@F,@DF,1e-10,20,para);
36 fprintf('No. of Newton iterations = %3i\n',nit)
37 %*** Graphic representation
38 figure(2)
39 trisurf(elements,coordinates(:,1),coordinates(:,2),u,'facecolor','interp')
```

Bibliography

[ACF94] Jochen Alberty, Carsten Carstensen, and Stefan A. Funken. "Remarks around 50 lines of Matlab: short finite element implementation." In: 20.2-3 (1994), pp. 117–137 (cit. on p. 25).

[Bra97] Dietrich Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge: Cambridge University Press, 1997 (cit. on pp. 13, 14).

[Che89] David K. Cheng. *Field and wave electromagnetics*. Addison Wesley, 1989 (cit. on p. 9).

[Cia78] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, 1978 (cit. on p. 25).

[Noq97] A. F. L. Noqueira. "Limitations of the Conventional Methods of Force and Torque Prediction." In: 12.1 (1997), pp. 74–79 (cit. on p. 19).

[THH97] M. Trlep, A. Hamler, and B. Hribernik. "Various Approaches to Torque Calculations by FEM and BEM." In: 12.2 (1997), pp. 127–130 (cit. on p. 19).

[Wan89] Ronald K. Wangsness. *Electromagnetic fields*. 2nd. John Wiley & Sons, 1989 (cit. on p. 9).

[Zar60] E. Zarantello. "Solving functional equations by contrative averaging". In: 160 (1960) (cit. on p. 15).