Master Thesis: Adaptive Timestepping for the Simulation of Electric Machines

Supervised by:
Dr. Jörg Ostrowski, Siemens Digital Industries Software
Prof. Dr. Ralf Hiptmair, SAM, D-MATH, ETH Zürich

Luca Wolfart

April 2025

Abstract

In this thesis we investigate the feasibility of using Rosenbrock Wanner (ROW) methods for the computation of magnetic fields in electric machines. We then compare the performance to established timestepping methods. We discover that ROW methods are suitable for the simulation of electrical machines, but special attention has to be given to the time-derivative of the mesh deformation. Adaptive timestepping with ROW posed unresolved challenges, which needs to be addressed. The performance of ROW with fixed timestep-size and established methods has been compared on two geometries: firstly, a static geometry, which represents a transformer, and then on a rotating geometry which represents an electric machine. We provide a complexity analysis for both methods based on a set of selected elementary operations, which allows to determine which method is better suited given the cost of the elementary operations.

Contents

Dedication

To mom and dad, and to all the friends I made along the way.

Acknowledgements

I like to acknowledge Robert Hahn, for the time he took to help me solving GMSH related issues, and to decide whether to use GetDP or LehrFEM++; Ryan Galagusz, for patiently helping me with the implementation of Newton iterations; Peter Herlig and Martin Schütz, for helping me use the MAGNET solver in SimCenter.

Goes without saying, the daily supervision by Jörg Ostrowski was invaluable and the meetings with Prof. Ralf Hiptmar were greatly insightful.

I also like to thank Siemens AG for giving me the opportunity to join them for this project and get an insight into the industry.

Chapter 1

Introduction

The simulation of magnetic fields is an important component of the development of low-frequency electric machines, in order to optimize the desired properties, such as torque, and minimize undesired effects such as heating of the machine. A popular approach to compute magnetic fields is the finite element method (FEM) [Logan, 2011], which provides a discrete approximation to the magnetic field in space. Magnetic fields change in time, due to changes in the current applied to the electric machine, motion of the electric machines and transient effects dictated by the governing equations. In order to approximate the changing magnetic field, so called timestepping schemes are needed. In such schemes, the time interval in which we compute the magnetic field is discretized into time steps, at which an approximation of the solution is provided. A widely used timestepping scheme is Implicit Euler, which will be considered the reference in this work. In this scheme, a non-linear system has to be solved at every timestep, which means that Newton-Raphson iterations need to be used. This is an expensive operation, and the aim of this work is to replace Implicit Euler timestepping with an iteration free method. The chosen method is called Rosenbrock-Wanner (ROW), which linearizes the equation and promises high order of convergence. ROW methods have already been used to simulate magnetic fields in static electronic components such as transformers, and the contribution of this work is to apply ROW to rotating machines and explore whether it provides the same advantages as in static components.

In Chapter 2 we explain concepts on which this work builds, in Chapter 3 we report what the previous works have achieved, in Chapter 4 we explain how Implicit Euler and ROW have been implemented to compute the magnetic fields, and in Chapter 6 we report the performance of the two methods, which is then discussed in 7.

Chapter 2

Preliminaries

2.1 Runge-Kutta methods

Runge-Kutta methods are a family of iterative timestepping methods used to approximately solve ordinary differential equations (ODEs). They are defined by a general formula, which is parametrized by a set of coefficients. The coefficients can be computed such that the method satisfies a certain order of convergence, which is a measure of how well the numerical solution approximates the exact solution, based on the step-size. A method of order x has an error of $O(h^{x+1})$, where h is the step size. A general initial value problem (IVP) is given by an ordinary differential equation (ODE) and the initial value, which read as follows:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0 \tag{2.1}$$

A Runge-Kutta method approximately solves such a problem by discretizing it in time, meaning that instead of providing a solution at every possible time, it computes the solution at a finite set of time-points. The solutions are computed in the following way [Süli and Mayers, 2003, pp. 351–352]:

$$y^{(j+1)} = y^{(j)} + h \sum_{i=1}^{s} b_i k_i \tag{2.2}$$

where j indices the timestep, k_i is the so called increment, h is the size of the timestep , and s is the number of stages. The coefficients b_i are tabulated for every Runge-Kutta method. For this work we need implicit Runge-Kutta methods, since the nature of our problem requires stability guarantees, which we will further discuss in a later section. The increments for such methods are computed by solving the equations

$$k_i = f(x^{(j)} + hc_i, y^{(j)} + h \sum_{j=1}^{s} a_{ij} k_j), \quad 1 \leq i \leq s. \tag{2.3}$$

The argument $x^{(j)}$ indicates the time at the previous timestep and a_{ij} and c_i are coefficients which are tabulated for every method. A property of these equations that is central to this work is that f can be nonlinear, which means that we need to solve a nonlinear system of equations in every timestep, which entails expensive Newton-Raphson iterations.

2.2 Rosenbrock-Wanner methods

A detailed introduction to Rosenbrock-Wanner methods can be found in [Lang, 2021]. The original idea of Rosenbrock [Rosenbrock, 1963], on which Wanner builds [Hairer and Wanner, 1987], is to apply a single Newton-Raphson step to the Crank-Nicholson timestepping method [Crank and Nicolson, 1947]. Rosenbrock then expands this idea to general Runge-Kutta methods, and the general form of a Rosenbrock method reads as:

$$\left(I - h\gamma_{ii}J(y^{(j)} + \sum_{j=1}^{i-1}\gamma_{ij}k_j)\right)k_i = hf(y^{(j)} + \sum_{j=1}^{i-1}\alpha_{ij}k_j), \quad i = 1, ..., s, \quad (2.4)$$

$$y^{(j+1)} = y^{(j)} + \sum_{i=1}^{s} b_i k_i, \quad (2.5)$$

where J is the Jacobian matrix of f, and γ_{ij} are method-specif coefficients, which are tabulated for every method. Wanner modifies this method by adding a term and only evaluating the Jacobian matrix once per timestep, which leads to the following formula for a Rosenbrock-Wanner (ROW) method:

$$\left(I - h\gamma_{ii}J(y^{(j)})\right)k_i = hf(y^{(j)} + \sum_{j=1}^{i-1}\alpha_{ij}k_j) + hJ(y^{(j)})\sum_{j=1}^{i-1}\gamma_{ij}k_j, \quad i = 1, ..., s, \quad (2.6)$$

$$y^{(j+1)} = y^{(j)} + \sum_{i=1}^{s} b_i k_i, \quad (2.7)$$

High order ROW methods have been found, which means that even when performing only one Newton-Raphson step, high accuracy can be maintained.

2.3 Stability of the Timestepping Methods

A property of numerical integration methods that is relevant for our problem is stability. The ODE which describes our problem is stiff, which means that it requires special attention to the size of the timesteps, since for some integration methods a too large timestep may cause the numerical solution to diverge from the true solution. The stability of integration methods is based on how it

behaves when applied to the initial value problem (IVP)

$$\frac{d}{dt}y(t) = k \cdot y(t), \quad y(0) = 1, \quad k \in \mathbb{C} \tag{2.8}$$

Two types of stability are relevant for this work: A-stability and L-stability. A method is called A-stable [Dahlquist, 1963], if, when applied to Equation (2.8), has the following property:

$$\lim_{j\to\infty} y^{(j)} = 0 \quad \text{for} \quad Re(k) < 0. \tag{2.9}$$

This is desirable since it reflects the behavior of the true solution. A method is called L-stable if is A-stable and it converges to 0 in one timestep, for an infinitely big timestep [Ehle, 1969]. This property means that unwanted oscillations are avoided. The methods BDF-1 and BDF-2, which we will use and will be introduce later, are both L-stable, while the ROW method we will use is A-stable.

2.4 Adaptive timestepping and Embedded Methods

Regarding the choice of the step size there are two options, either one chooses a fixed step size, meaning the solution is computed at regular time-intervals, or one changes the step-size between the timesteps, depending on an estimate of the error of the solution. The latter is called adaptive timestepping. Adaptive timestepping is a feature needed in this project, since variation in the prescribed current in the coils and saturation of the magnetic material means that a different amount of change may happen in different moments, so an adaptive step size is needed.

In order to control the step size, we need to know how good the solution is with the current step size, and reduce the size if the error is too big. In order to do so, an estimate of the error is needed. Such an estimate can be provided by embedded methods. These are constituted by a pair of Runge-Kutta methods, in which the two methods only differ by b_i coefficients, and have a different order. By taking the difference of the solution provided by the two methods, an estimate of the error is computed. If the error is smaller than a tolerance, the solution is accepted and the next timestep is computed. If the error is above the tolerance the solution is discarded and a new one is computed with half the step size. An adaptive timestepping method has two parameters: a relative tolerance and an absolute tolerance, which are used as following. If est is the difference between the solution using the methods of high and low order, then the step is accepted if $est < \max(\text{reltol} \cdot \text{norm}(y^{(j)}), \text{abstol})$ [Hiptmair, 2023a], where reltol and abstol are the relative and absolute tolerances respectively, and $\text{norm}(y^{(j)}$ is the norm of the previous timestep.

2.5 Maxwell's equations

In this work we are interested in the magnetic fields generated inside of the electrical machine. These are governed by Maxwell's equations, which are given by

$$\begin{aligned} \mathbf{curl}\,\mathbf{E} &= -\partial_t \mathbf{B}\,, && (2.10)\\ \mathbf{curl}\,\mathbf{H} &= \mathbf{j} + \partial_t \mathbf{D}\,, && (2.11)\\ \operatorname{div}\mathbf{B} &= 0\,, && (2.12)\\ \operatorname{div}\mathbf{D} &= \rho\,. && (2.13) \end{aligned}$$

Where E is the electric field, B is magnetic flux, H is the magnetic field, and D is the displacement field. ρ is the electric charge density, and j is the electric current. The fields are related by the following equations:

$$\begin{aligned} \mathbf{D} &= \epsilon \cdot \mathbf{E} = \epsilon_0 \cdot \epsilon_r \cdot \mathbf{E} && (2.14)\\ \mathbf{B} &= \mu \cdot \mathbf{H} = \mu_0 \cdot \mu_r \cdot \mathbf{H}\,, && (2.15) \end{aligned}$$

where ϵ, ϵ_0 and ϵ_r are the absolute, vacuum and relative permittivity respectively, while μ, μ_0 and μ_r are the respective permeabilities

We assume ohmic conductors

$$\mathbf{j} = \sigma \mathbf{E}\,, \tag{2.16}$$

where σ is the conductivity.

Chapter 3

Previous Works

The idea of applying ROW methods to the Eddy current problem has been explored in [Kähne and Clemens, 2025]. One difference between their approach and ours is that they compute the Jacobian on the right hand side of the ROW increment equation 2.6 for every increment, while we only compute it once per timestep. According to the authors, computing the Jacobian for every increment improves accuracy, so this could be a possible way to improve our method. They show that the improved ROW method produces accurate results on a static geometry.

In [Clemens et al., 2009] adaptivity in space and time is explored, meaning that in every timestep the mesh is refined based on local error estimates, with a ROW method. This underlines how influential the mesh quality is on the performance of ROW methods, and suggests paying particular attention to the mesh quality when using ROW methods.

Chapter 4

Methods

4.1 Notation

We provide the definition of the following space:

$$H^1(\Omega) := \left\{ v : \Omega \to \mathbb{R} \mid \|v\|_{H^1(\Omega)} < \infty \right\}, \tag{4.1}$$

where the norm is defined as following:

$$\|u\|^2_{H^m(\Omega)} := \sum_{k=0}^{m} \sum_{\alpha \in \mathbb{N}^d, |\alpha|=k} \int_\Omega |D^\alpha u|^2 \, dx, \quad \text{where} \quad D^\alpha u := \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}. \tag{4.2}$$

This is called a Sobolev space.

4.2 The Eddy Current Model in A* formulation

The Eddy current model is a simplification of the full Maxwell's equations that results from neglecting the displacement current $\partial_t \mathbf{D}$. In time-domain it writes

$$\begin{aligned} \mathbf{curl}\, \mathbf{E} &= -\partial_t \mathbf{B}\,, & (4.3)\\ \mathbf{curl}\, \mathbf{H} &= \mathbf{j}\,, & (4.4)\\ \operatorname{div} \mathbf{B} &= 0\,. & (4.5) \end{aligned}$$

The charge density ρ can be determined in a post-processing step, and is therefore neglected. We use an potential approach in temporal gauge ($\varphi = 0$), i.e.,

$$\begin{aligned} \mathbf{B} &= \mathbf{curl}\, \mathbf{A}\,, & (4.6)\\ \mathbf{E} &= -\partial_t \mathbf{A}\,. & (4.7) \end{aligned}$$

Some communities call this an A*-formulation. Inserting this approach into the eddy current model by using the material relations

$$\mathbf{j} = \sigma\mathbf{E} = -\sigma\partial_t\mathbf{A} + \mathbf{j}_0\,, \tag{4.8}$$
$$\mathbf{B} = \mu\mathbf{H} \tag{4.9}$$

finally yields

$$\mathbf{curl}\left(\frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\right) + \sigma\partial_t\mathbf{A} = \mathbf{j}^0\,, \qquad \text{in } \Omega. \tag{4.10}$$

with the solenoidal exciting current density $\mathbf{j}^0$. The function of the excitation current $\mathbf{j}_0$ can e.g. be to model windings of coils.

The excitation current density $\mathbf{j}^0$ is constant in the windings, and the conductivity σ in the exciting winding is set to be zero there.

4.3 3D Weak Formulation with Electric-Circuit-Element (ECE) Boundary Conditions

In order to solve the Eddy-current equations using FEM, we describe the boundary conditions that are used, and then derive the weak formulation of the equations which results from these conditions.

Stokes's formula for the curl operator is given by:

$$\int_\Omega \mathbf{curl}\,\mathbf{v}\cdot\mathbf{u}\,\mathrm{dx} = \int_\Omega \mathbf{curl}\,\mathbf{u}\cdot\mathbf{v}\,\mathrm{dx} - \int_\Gamma (\mathbf{v}\times\mathbf{n})\cdot\mathbf{u}\,\mathrm{dS} \quad \forall\mathbf{u}\in\mathbf{H}(\mathbf{curl};\Omega), \mathbf{v}\in\mathbf{H}_0^1(\Omega), \tag{4.11}$$

where Ω is the computational domain, u and v are functions defined on Ω and n is the normal to the domain.

The weak formulation of (4.10) is derived in the following steps. First the equation is multiplied by a test function $\mathbf{A}'$ and integrated over the domain Ω. Then Stokes's formula is applied to the first term.

$$\int_\Omega \mathbf{curl}\left(\frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\right)\mathbf{A}'\mathrm{dx} + \int_\Omega \sigma\partial_t\mathbf{A}\mathbf{A}'\mathrm{dx} = \int_\Omega j_0\mathbf{A}'\mathrm{dx} \quad \forall\mathbf{A}'\in\mathbf{H}^1(\Omega) \tag{4.12}$$

$$\Leftrightarrow \int_\Omega \frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\,\mathbf{curl}\,\mathbf{A}'\mathrm{dx} - \int_\Gamma (\frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\times\mathbf{n})\cdot\mathbf{A}'\,\mathrm{dS} + \int_\Omega \sigma\partial_t\mathbf{A}\mathbf{A}'\mathrm{dx} = \int_\Omega \mathbf{j}^0\mathbf{A}'\mathrm{dx} \quad \forall\mathbf{A}'\in\mathbf{H}^1(\Omega). \tag{4.13}$$

Now we consider the ECE boundary conditions. They read as following:

(i) There is no inductive coupling with the exterior:

$$\partial_t\mathbf{B}\cdot\boldsymbol{n} = 0 \quad \Longleftrightarrow \quad \mathbf{curl}\,\mathbf{E}\cdot\boldsymbol{n} = 0 \quad \text{on the entire boundary } \partial\Omega\,,$$

where $\boldsymbol{n}$ is the outer unit normal vector-field on $\partial\Omega$.

(ii) No electric currents can penetrate the boundary of the non-conductive domain:

$$\mathbf{curl}\,\mathbf{H}\cdot\boldsymbol{n}=0 \quad\Longleftrightarrow\quad \mathbf{E}\cdot\boldsymbol{n}=0 \quad \text{on } \partial\Omega_N.$$

Analogously to the **curl** operator, it can be shown that when the surface curl of a vector field ($\mathbf{curl}\,\boldsymbol{\phi}\cdot\mathbf{n}$) is 0, then the tangential component trace $\boldsymbol{\phi_t} := (\boldsymbol{n}\times\boldsymbol{\phi})\times\boldsymbol{n}|_{\partial\Omega}$ is the surface gradient of a scalar potential. For the electric field it means that we can make the following ansatz:

$$\mathbf{A} = \mathbf{A}_0 + \mathbf{grad}\,\eta \tag{4.14}$$
$$\mathbf{E} = -\partial_t\mathbf{A}_0 - \mathbf{grad}(\partial_t\eta), \tag{4.15}$$

$$\text{where } \mathbf{A}_0 = 0 \text{ on the boundary } \partial\Omega, \tag{4.16}$$
$$\eta \text{ is only defined on the boundary } \partial\Omega. \tag{4.17}$$

Plugging this ansatz into (4.13), for the boundary term we get:

$$\int_\Gamma (\frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\times\mathbf{n})\cdot\mathbf{A}'\;\mathrm{dS} \tag{4.18}$$
$$= \int_\Gamma (\frac{1}{\mu}\,\mathbf{curl}(\mathbf{A}_0+\mathbf{grad}\,\eta)\times\mathbf{n})\cdot\mathbf{A}'\;\mathrm{dS} \tag{4.19}$$
$$= \int_\Gamma (\frac{1}{\mu}\,\mathbf{curl}(\mathbf{grad}\,\eta)\times\mathbf{n})\cdot\mathbf{A}'\;\mathrm{dS} \tag{4.20}$$
$$= 0 \tag{4.21}$$

where we used (4.17) in the first step and $\mathbf{curl}\,\mathbf{grad}(\boldsymbol{\phi}) = 0$ for any $\boldsymbol{\phi}$, in the second step.

Equation (4.13) then simplifies to

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}\,\mathbf{A}\,\mathbf{curl}\,\mathbf{A}'\mathrm{dx} + \int_\Omega \sigma\partial_t\mathbf{A}\mathbf{A}'\mathrm{dx} = \int_\Omega \mathbf{j}^0\mathbf{A}'\mathrm{dx}. \tag{4.22}$$

Plugging in the Ansatz again:

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}(\mathbf{A}_0+\mathbf{grad}\,\eta)\,\mathbf{curl}\,\mathbf{A}'\mathrm{dx} + \int_\Omega \sigma\partial_t(\mathbf{A}_0+\mathbf{grad}\,\eta)\mathbf{A}'\mathrm{dx} = \int_\Omega \mathbf{j}^0\mathbf{A}'\mathrm{dx}. \tag{4.23}$$

The conductivity σ is set to 0 on the boundary in all examples, since the electrical devices are set in an air box. Thus, $\mathbf{grad}\,\eta$, which is only defined on the boundary, drops out in the second term. It also drops out of the first term for the same reason as before.

The resulting weak formulation for $\mathbf{A}_0$ is:

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}(\mathbf{A}_0)\,\mathbf{curl}\,\mathbf{A}'\mathrm{dx} + \int_\Omega \sigma\partial_t(\mathbf{A}_0)\mathbf{A}'\mathrm{dx} = \int_\Omega \mathbf{j}^0\mathbf{A}'\mathrm{dx}.. \tag{4.24}$$

4.4 Simplification to 2D

Following the derivation in [Casagrande, 2013], the weak 3D formulation is reduced to a 2D formulation. To do so, a so-called TM model is used. It assumes that the excitation current $\mathbf{j}^0$ is aligned with the z-axis. The fields follow:

Field Qantity	TM Model
$\mathbf{j}^0$	$(0, 0, j_z^0)$
$\mathbf{E}$	$(0, 0, E_z)$
$\mathbf{B}$	$(B_x, B_y, 0)$
$\mathbf{A}$	$(0, 0, A_z)$

Table 4.1: Field quantities for the TM model

The curl operator of a field is defined as following:

$$\mathbf{curl}\,\mathbf{A} = \begin{bmatrix} \frac{\partial A_z}{\partial y} - \frac{\partial A_y}{\partial z} \\ \frac{\partial A_x}{\partial z} - \frac{\partial A_z}{\partial x} \\ \frac{\partial A_y}{\partial x} - \frac{\partial A_x}{\partial y} \end{bmatrix}. \tag{4.25}$$

Plugging in $\mathbf{A}$ from the TM model, (4.25) becomes

$$\mathbf{curl}\,\mathbf{A} = \begin{bmatrix} \frac{\partial A_z}{\partial y} \\ -\frac{\partial A_z}{\partial x} \\ 0 \end{bmatrix}, \tag{4.26}$$

thus the z-component of the curl is known and the operator can be reduced to

$$\mathbf{curl}_{2D}\,\mathbf{A} := \begin{bmatrix} \frac{\partial A_z}{\partial y} \\ -\frac{\partial A_z}{\partial x} \end{bmatrix} \tag{4.27}$$

in equation (4.24), resulting in

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}(t)\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}'\mathrm{dx} + \int_\Omega \sigma(t)\partial_t A_\mathrm{z}(t)\cdot A'_\mathrm{z}\mathrm{dx} = \int_\Omega \mathbf{j}^0\cdot A'_\mathrm{z}\mathrm{dx} \quad \forall\mathbf{A}' \in \mathbf{H}^1 \tag{4.28}$$

4.5 Stationary Finite Element Discretization of the 2D Formulation

In the stationary problem, the current and magnetic field do not change with time. This means that in this setting the time derivative in weak form (4.28) becomes zero, resulting in the weak form:

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}'\mathrm{d}\mathrm{x} = \int_\Omega \mathbf{j}^0\cdot A'_\mathrm{z}\mathrm{d}\mathrm{x} \quad \forall \mathbf{A}'\in\mathbf{H}^1. \tag{4.29}$$

Following the principles of FEM, as presented in [Hiptmair, 2023b], a discrete formulation of the 2D weak form (4.28) is derived. The test and trial spaces, which are the spaces in which the test functions and solution live, are substituted by a finite-dimensional space. Here the space of 1-st degree Lagrangian finite element functions on a simplicial mesh $\mathcal{M}$, $\mathcal{S}_1^0$, is used.

To compute the matrices corresponding to the various terms of (4.28), the element matrices need to be computed, from which the "LehrFEM++" finite element software can construct the system matrix and vector. Element matrices are the restriction of the Galerkin matrices to mesh triangles.

The stiffness matrix, corresponding to the term

$$a(\mathbf{A},\mathbf{A}') = \int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}'\mathrm{d}\mathrm{x}, \tag{4.30}$$

has the following form:

$$\mathbf{A}_k = \left[\int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\lambda_i\cdot\mathbf{curl}_{2\mathrm{D}}\,\lambda_j\mathrm{d}\mathrm{x}\right]_{i,j=1}^{3} \in \mathbb{R}^{3,3}, \tag{4.31}$$

where λ_i is the barycentric coordinate function corresponding to point i.

The barycentric coordinate function can be represented in the form

$$\lambda_i(x) = \alpha_i + \boldsymbol{\beta}^i\cdot x. \tag{4.32}$$

Consider a triangle with vertices $\mathbf{a}_K^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix}, \mathbf{a}_K^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix}, \mathbf{a}_K^1 = \begin{bmatrix} a_1^3 \\ a_2^3 \end{bmatrix}$. In In the Lecture Script [Hiptmair, 2023b] it is shown that the coefficients β_i^j can be found by solving the following equation:

$$\begin{bmatrix} 1 & a_1^1 & a_2^1 \\ 1 & a_1^2 & a_2^2 \\ 1 & a_1^3 & a_2^3 \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 \\ \beta_1^1 & \beta_1^2 & \beta_1^3 \\ \beta_2^1 & \beta_2^2 & \beta_2^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.33}$$

From (4.32) it can be seen that

$$\beta_j^i = \frac{\partial \lambda^i}{\partial j}. \tag{4.34}$$

Considering that μ is constant over the cell, and inserting the definition of 2D curl, (4.28) becomes:

$$\mathbf{A}_K = \frac{1}{\mu}|K|\left[\frac{\partial\lambda_i}{\partial y}\frac{\partial\lambda_j}{\partial y} + \frac{\partial\lambda_i}{\partial x}\frac{\partial\lambda_j}{\partial x}\right]_{i,j=1}^{3}. \tag{4.35}$$

The element matrix then becomes:

$$\mathbf{A}_K = \frac{1}{\mu}|K| \begin{bmatrix} \beta_2^1 & \beta_1^1 \\ \beta_2^2 & \beta_1^2 \\ \beta_2^3 & \beta_1^3 \end{bmatrix} \begin{bmatrix} \beta_2^1 & \beta_2^2 & \beta_2^3 \\ \beta_1^1 & \beta_1^2 & \beta_1^3 \end{bmatrix}. \tag{4.36}$$

The element vector, corresponding to the term

$$\mathbf{l}(v) := \int_\Omega \mathbf{j}^0 \cdot A'_z \mathrm{dx}, \tag{4.37}$$

has the form

$$\vec{\phi}_K := \left[\int_K j_z^0 \lambda_i \mathrm{dx} \right]_{i=1}^3 = j_z^0 \left[\int_K \lambda_i \mathrm{dx} \right]_{i=1}^3 = \frac{1}{3}|K| j_{z,K}^0 \left[1\right]_{i=1}^3 \in \mathbb{R}^3, \tag{4.38}$$

since we consider the current to be constant over mesh elements. The last step follows from the midpoint quadrature rule.

4.6 Verifying the Stationary Solver on a Cylindrical Domain with Known Analytical Solution

In order to test a solver implementation, the result can be compared to the analytical solution of a problem with known solution. In the case of the Eddy-current problem, the chosen problem is a conducting cylinder with infinite extension in z direction in an air box, which can be seen in Figure 4.1. The problem is solved in cylinder coordinates, so the solution is expressed in terms of the radius r, angle ϕ and z-coordinate. Since the cylinder is infinitely long, there is no dependence from z. Radial symmetry also implies that the solution is not dependent from the angle, so it is only dependent from r. Consider the air-box to be infinitely big. For the first part we are interested in testing the stationary solver, which will later be extended to a time-dependent solver, so the second part in (4.10) is zero, and the equation becomes:

$$\mathbf{curl}_{2\mathrm{D}} \left(\frac{1}{\mu} \mathbf{curl}_{2\mathrm{D}} \mathbf{A} \right) = \mathbf{j}^0, \quad \text{in } \Omega. \tag{4.39}$$

Further, we solve the equation in the two domains separately, first in the interior of the cylinder and then in the outside.

Inside of the cylinder:
In the interior of the cylinder there is a constant current j_z^0 in the z-direction, constant conductivity σ greater than 0, which will become relevant in the time-dependent part, and constant permeability μ. Since μ is constant on the inside, it can be pulled in front of the first **curl** operator, and (4.39) becomes

$$\frac{1}{\mu} \Delta \mathbf{A} = \mathbf{j}^0, \quad \text{in } \Omega, \tag{4.40}$$

where $\Delta = \mathbf{grad}^2$, and the equality $\mathbf{curl}^2\, u = \Delta u$ has been used.

We will proceed to solve the equation in cylinder coordinates. The Laplace operator (Δ) in cylinder coordinates is

$$\Delta f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2} \tag{4.41}$$

(4.40) then becomes

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial A}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 A}{\partial \theta^2} = \mu j_z^0 \tag{4.42}$$

Since the domain is rotationally symmetric, the second term is 0

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial A}{\partial r}\right) = \mu j_z^0. \tag{4.43}$$

Integrating both sides over r we get

$$r\frac{\partial A}{\partial r} = \frac{r^2}{2}\mu j_z^0 + C_1, \tag{4.44}$$

$$\frac{\partial A}{\partial r} = \frac{r}{2}\mu j_z^0 + \frac{C_1}{r}, \tag{4.45}$$

$$A(r) = \frac{r^2}{4}\mu j_z^0 + C_1 \log(r) + C_2. \tag{4.46}$$

Since we are interested in the magnetic field B, which we will later argue is the radial derivative of A, we are not interested in C_2, and can consider it to be zero. A inside of the cylinder then becomes:

$$A(r) = \frac{r^2}{4}\mu j_z^0 \tag{4.47}$$

Outside of the Cylinder
Outside of the cylinder no current is flowing, so the right hand side is 0:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial A}{\partial r}\right) = 0. \tag{4.48}$$

Integration then delivers:

$$r\frac{\partial A_z}{\partial r} = C_3, \tag{4.49}$$

$$A_z(r) = C_3 \log(r) + C_4, \tag{4.50}$$

where again C_4 can be ignored since A_z will be derived with respect to r to get B_θ, resulting in the following equation outside the cylinder:

$$A_z(r) = C_3 \log(r) \tag{4.51}$$

As introduced earlier, the relation between A_z and B is $B = \mathbf{curl}\, A_z$. In Cylinder coordinates the **curl** operator corresponds to:

$$\mathbf{curl}\, A_z = \left(\frac{1}{r}\frac{\partial A_z}{\partial \rho} - \frac{\partial A_z}{\partial z}\right)\hat{\mathbf{r}} + \left(\frac{\partial A_r}{\partial z} - \frac{\partial A_z}{\partial r}\right)\hat{\boldsymbol{\varphi}} + \frac{1}{r}\left(\frac{\partial(\rho A_\varphi)}{\partial r} - \frac{\partial A_r}{\partial \varphi}\right)\hat{\mathbf{z}} \tag{4.52}$$

We know the magnetic field will only have a φ component, and $\frac{\partial A_r}{\partial z} = 0$, so

$$B(r) = -\frac{\partial A_z}{\partial r}. \tag{4.53}$$

Inside of the cylinder we get

$$B(r) = \frac{r}{2}\mu j_z^0, \tag{4.54}$$

while outside we get

$$B(r) = \frac{C_3}{r}. \tag{4.55}$$

To get the coefficient C_3 the equations need to be compared at the boundary, and the value of B at from the inside has to be the same as at the outside. Assume the cylinder has radius R. The the following has to hold at $r = R$:

$$\frac{R}{2}\mu j_z^0 = \frac{C_3}{R} \tag{4.56}$$

$$\implies C_3 = \frac{R^2}{2}\mu j_z^0, \tag{4.57}$$

and the final formula for the magnetic field outside the cylinder reads as

$$B(r) = \frac{R^2}{2}\mu j_z^0 \frac{1}{r}. \tag{4.58}$$

After we computed this formula, we computed a static FEM solution with the previously derived matrices, and sampled the solution along the radius of the cylinder, which had a perfect correspondence with the analytical solution. This allowed us to be confident about the accuracy of the stiffness matrix $\underline{\underline{A}}$ and proceed with the implementation of the transient solver. In order to solve the system of equations arising from the FEM method, we built a preconditioner, which is described in Appendix B.

4.7 Time-Dependent Finite Element Discretization of the 2D Formulation

After having verified the static solver, we move on to implementing the time dependent equation.

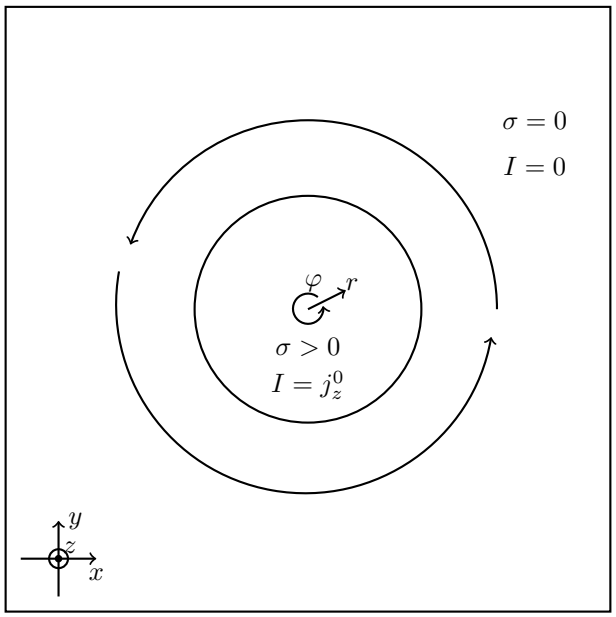


Figure 4.1: Simple static test geometry. A current is applied round cylinder made of a conductive material, and placed in air. The magnitude of the magnetic field grows linearly inside of the magnetic field and decays with $1/r$ outside the cylinder.

The following matrix definitions will be useful:

$$\underline{\underline{A}} := \left[\int_\Omega \frac{1}{\mu(x)} \, \mathbf{curl}_{2\mathrm{D}} \, \mathbf{b}_h^i(\mathbf{x}) \cdot \mathbf{curl}_{2\mathrm{D}} \, \mathbf{b}_h^j(\mathbf{x}) \mathrm{dx} \right]_{i,j=1}^{N}, \tag{4.59}$$

$$\underline{\underline{M}} := \left[\int_\Omega \sigma(x,t) \mathbf{b}_h^i(\mathbf{x}) \cdot \mathbf{b}_h^j(\mathbf{x}) \mathrm{dx} \right]_{i,j=1}^{N}, \tag{4.60}$$

$$\vec{\phi}(t) = \left[\mathbf{j}^0(t)(\mathbf{b}_h^i) \right]_{i=1}^{N}, \tag{4.61}$$

where b_h^i is the i^{th} basis function of the finite dimensional subspace, and N is the dimension of the subspace

4.7.1 Time Discretization using Implicit Euler

To solve the equation we discretize it in time, meaning that instead of having a value of the solution for every time-point $A_z(t)$, we compute it for discrete timesteps j: $A_z^{(j)}$, where j is a given timestep. When using implicit Euler, the time derivative of A_z is approximated as

$$\frac{\partial A^{(j)}}{\partial t} \approx \frac{A^{(j)} - A^{(j-1)}}{\Delta t}. \tag{4.62}$$

Inserting this approximation into (4.28) we get

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}^{(j)}\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}'\mathrm{d}x+\int_\Omega \sigma(t)\frac{A^{(j)}-A^{(j-1)}}{\Delta t}\cdot A'_z\mathrm{d}x$$
$$=\int_\Omega \mathbf{j}^0\cdot A'_z\mathrm{d}x\quad\forall\mathbf{A}'\in\mathbf{H}^1. \tag{4.63}$$

Bringing all the known quantities to the right hand side:

$$\int_\Omega \frac{1}{\mu}\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}^{(j)}\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}'\mathrm{d}x+\int_\Omega \sigma(t)\frac{A^{(j)}}{\Delta t}\cdot A'_z\mathrm{d}x=\int_\Omega\left(\mathbf{j}^0(t)+\sigma(t)\frac{A^{(j-1)}}{\Delta t}\right)A'_z\mathrm{d}x\quad\forall\mathbf{A}'\in\mathbf{H}^1. \tag{4.64}$$

For the coefficient expansion vector $\mathbf{x}$ it then follows that:

$$\underline{\underline{A}}\mathbf{x}^{(j)}+\underline{\underline{M}}\mathbf{x}^{(j)}/\Delta t=\vec{\phi}(t)+\underline{\underline{M}}\mathbf{x}^{(j-1)}/\Delta t. \tag{4.65}$$

Solving for $\mathbf{x}$:

$$\left(\underline{\underline{A}}\Delta t+\underline{\underline{M}}\right)\mathbf{x}^{(j)}=\vec{\phi}(t)\Delta t+\underline{\underline{M}}\mathbf{x}^{(j-1)}. \tag{4.66}$$

The mass element matrix in this problem setting is given by

$$\underline{\underline{M}}_K=\left[\int_K \sigma\lambda_i\cdot\lambda_j\mathrm{d}x\right]_{i,j=1}^{3}\in\mathbb{R}^{3,3} \tag{4.67}$$
$$=\frac{1}{3}\sigma|K|I_3, \tag{4.68}$$

4.8 Assembling the Element and Stiffness Matrix in a Rotating Mesh

An electric motor typically consists of three parts, which are the stator, the rotor which turns around the stator, and an air-gap which separates the two, shown in Figure 4.2. When creating a mesh, we can mesh the rotor and stator in advance, since their geometry is constant, and simply rotate the mesh of the rotor. The only part that needs to be re-meshed at every timestep is the air-gap. Practically, this means that in every timestep the meshes of the different regions are handled in the following ways:

(i) The mesh of the rotor is kept the same

(ii) The mesh of the stator is rotated by a constant angle, which is determined by a constant rotation speed. This means that every node is rotated by the same angle, but the edges between the nodes remain the same

(iii) The air-gap is re-meshed in every timestep, meaning there is no relation between the nodes of one timestep and the next.

From equation (4.66) we can see that to get the solution at timestep j, we need to know the solution at the position of the same nodes, at timestep $j-1$, except for those in the air-gap. This is because in the air-gap the conductivity is 0, so the term corresponding to the previous timesteps cancels out. In the stator finding the value of the solution at a node at timestep $j-1$ is trivial, since the mesh is static. In the rotor on the other side it is not, since the node was at a different position in the previous timestep. Still [Casagrande, 2013] proved that the solution at the same node from the previous timestep can be used when using implicit Euler method for timestepping. The arrow in Figure 4.2 illustrates that the solution at node k from the previous timestep, which was at a different position, is used in a given timestep to solve equation (4.66).

This means that we know which at which nodes the solution needs to be evaluated at for all the regions. The only block missing is how to map the node indices from one timestep to the other. In fact, the matrix is changing size in every timestep, because the number of nodes in the air-gap may change. To make sure the indices of the rotor correspond between timesteps, you can construct the solution vector by putting the nodes corresponding to stator and rotor first, and the solution corresponding to the air-gap last. In this way even when the number of nodes in the air-gap changes, the solution-indices of rotor and stator nodes will stay the same.

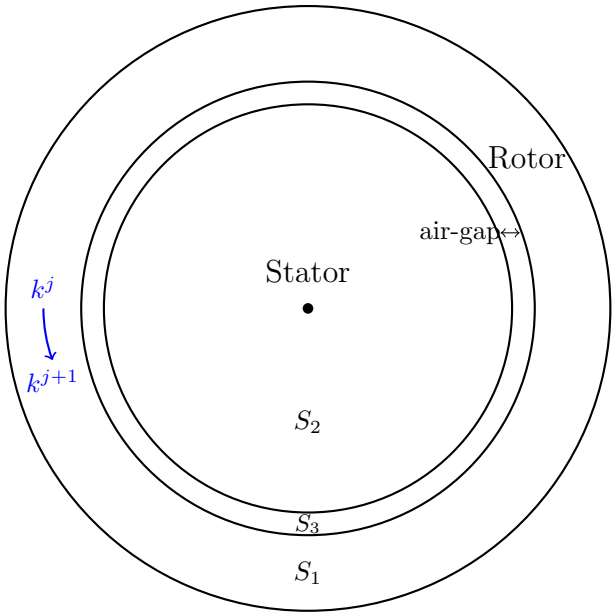


Figure 4.2: Simplified geometry of a motor. The rotor is the mobile part of the domain, and is rotated around the stator. A thin airgap separates the rotor and stator.

To ensure that the solution vector will have the above described structure,

we have to construct the system matrix in the corresponding way. Let us say that the solution vector has the form

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{4.69}$$

where the indices 1, 2, and 3 indicate nodes of the rotor, stator, and air-gap respectively, as shown in Figure 4.2. Then the matrix A has the form

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, \tag{4.70}$$

and the same holds for M. A_{ii} is a sub-matrix, and the subscript indicates the interaction between the different parts of the domain. A_{13} for example indicates the interaction between basis functions in the rotor and air-gap. Since there is no overlap between the basis functions in the rotor and stator, the matrix entries corresponding to those interactions are 0. The sub-matrices A_{11} and A_{22} can be precomputed since the meshes do not change between timesteps.

The right hand side vector also has to be assembled to follow the structure described by (4.69), i.e.

$$\vec{\phi} = \begin{pmatrix} \vec{\phi}_1 \\ \vec{\phi}_2 \\ \vec{\phi}_3 \end{pmatrix} \tag{4.71}$$

Details about the implementation of matrix assembly in a rotating mesh can be found in Appendix A.

4.9 Nonlinear Materials

Earlier we assumed for simplicity that there is a linear relationship between magnetic field $\mathbf{H}$ and magnetic flux $\mathbf{B}$. In reality this is not the case, and the magnetic permeability is dependent on the strength of the magnetic flux: $\mu = \mu(\mathbf{B})$. We know that $\mathbf{B} = \mathbf{curl}\,\mathbf{A}$, and $\mathbf{B} = \mu\mathbf{H}$. Thus $\mathbf{H} = \frac{\mathbf{B}}{\mu} = \frac{\mathbf{curl}\,A}{\mu}$.

Following [Bachinger et al., 2006] (4.28) then becomes

$$\int_\Omega \mathbf{H}(\mathbf{curl}_{2D}\, A_z(t)) \cdot \mathbf{curl}_{2\mathrm{D}}\, \mathbf{v}\mathrm{dx} + \int_\Omega \sigma(t)\partial_t A_\mathrm{z}(t) \cdot \mathbf{v}\mathrm{dx} = \int_\Omega \mathbf{j}^0 \cdot \mathbf{v}\mathrm{dx} \quad \forall \mathbf{v} \in \mathbf{H}^1, \tag{4.72}$$

and (4.64) becomes:

$$\int_\Omega H(\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}^{(j)})\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{dx}+\int_\Omega\sigma(t)\frac{A^{(j)}}{\Delta t}\cdot\mathbf{v}\mathrm{dx}=\int_\Omega\left(\mathbf{j}^0(t)+\frac{A^{(j-1)}}{\Delta t}\right)\mathbf{v}\mathrm{dx}\quad\forall\mathbf{v}\in\mathbf{V}_h \tag{4.73}$$

$$\Leftrightarrow\int_\Omega H(\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}^{(j)})\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{dx}+\int_\Omega\sigma(t)\frac{A^{(j)}}{\Delta t}\cdot\mathbf{v}\mathrm{dx}-\int_\Omega\left(\mathbf{j}^0(t)+\frac{A^{(j-1)}}{\Delta t}\right)\mathbf{v}\mathrm{dx}=0\quad\forall\mathbf{v}\in\mathbf{V}_h. \tag{4.74}$$

which is a nonlinear system of equations in standard form $F(\mathbf{A}^{(j)})=0$. This suggests using Newton's iterations to find the solution. To do so, we need the Jacobian of F.

Following the derivation in [Bachinger et al., 2006], we write the operator $\mathcal{A}$ as

$$\langle\mathcal{A}(\mathbf{A})|\mathbf{v}\rangle:=\int_\Omega\mathbf{H}(\mathbf{curl}\,\mathbf{A})\cdot\mathbf{curl}\,\mathbf{v}\mathrm{dx}, \tag{4.75}$$

which has derivative

$$\langle\mathcal{A}'(\mathbf{A})\mathbf{w}|\mathbf{v}\rangle=\int_\Omega\left[\frac{\partial\mathbf{H}}{\partial\mathbf{B}}(\mathbf{curl}\,\mathbf{A})\,\mathbf{curl}\,\mathbf{w}\right]\cdot\mathbf{curl}\,\mathbf{v}\mathrm{dx}, \tag{4.76}$$

by using the chain rule when evaluated at $\mathbf{w}$.

For the second term of (4.74) we write the operator $\mathcal{M}$ defined as

$$\langle\mathcal{M}\mathbf{A}|\mathbf{v}\rangle:=\int_\Omega\sigma(t)\frac{\mathbf{A}}{\Delta t}\mathbf{v}\mathrm{dx}, \tag{4.77}$$

with derivative

$$\langle\mathcal{M}'\mathbf{w}|\mathbf{v}\rangle=\int_\Omega\sigma(t)\frac{\mathbf{w}}{\Delta t}\mathbf{v}\mathrm{dx}. \tag{4.78}$$

The derivative of $\mathcal{M}$ does not depend from $\mathbf{A}$, since it is linear.

A Newton iteration is defined as following. Assume you have a function $F(x):\mathbb{R}^N\to\mathbb{R}^N$ and want to find the root. Then you can start with an initial guess x_0, and iterate using the formula [Strang, 2017]:

$$x_{n+1}=x_n-J(x_n)^{-1}F(x_n), \tag{4.79}$$

where J(x) is the Jacobian of $F(x)$.

Moving things around:

$$J(x_n)(x_n-x_{n+1})=F(x_n). \tag{4.80}$$

So we are solving for $\delta = x_n - x_{n-1}$.
Using the derivatives (4.76) and (4.78), (4.80) becomes:

$$\int_\Omega \left[\frac{\partial \mathbf{H}}{\partial \mathbf{B}_n}(\mathbf{curl}_{2\mathrm{D}}\, x_n)\,\mathbf{curl}_{2\mathrm{D}}\,\delta\right]\cdot \mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{d}x + \int_\Omega \sigma(t)\frac{\delta}{\Delta t}\mathbf{v}\mathrm{d}x = \tag{4.81}$$

$$\int_\Omega H(\mathbf{curl}_{2\mathrm{D}}\,\mathbf{A}^{(j)})\cdot \mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{d}x + \int_\Omega \sigma(t)\frac{A^{(j)}}{\Delta t}\cdot\mathbf{v}\mathrm{d}x - \int_\Omega \left(\mathbf{j}^0(t) + \frac{A^{(j-1)}}{\Delta t}\right)\mathbf{v}\mathrm{d}x \quad \forall \mathbf{v}\in\mathbf{V}_h. \tag{4.82}$$

Since x_n represents a step to find the solution of $\mathbf{A}^{(j)}$, we can say that $\mathbf{B}_n = \mathbf{curl}_{2\mathrm{D}}\, x_n$ is a solution step for the magnetic flux, and rewrite (4.81) as

$$\int_\Omega \left[\frac{\partial \mathbf{H}}{\partial \mathbf{B}_n}(\mathbf{B}_n)\,\mathbf{curl}_{2\mathrm{D}}\,\delta\right]\cdot \mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{d}x + \int_\Omega \sigma(t)\frac{\delta}{\Delta t}\mathbf{v}\mathrm{d}x = \tag{4.83}$$

$$\int_\Omega H(\mathbf{curl}_{2\mathrm{D}}\, x_n)\cdot \mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{d}x + \int_\Omega \sigma(t)\frac{x_n}{\Delta t}\cdot\mathbf{v}\mathrm{d}x - \int_\Omega \left(j^0(t) + \frac{x^{(j-1)}}{\Delta t}\right)\mathbf{v}\mathrm{d}x \quad \forall \mathbf{v}\in\mathbf{V}_h. \tag{4.84}$$

The last piece of the puzzle is the Jacobian $\frac{\partial \mathbf{H}}{\partial \mathbf{B}}$. We will take it directly from [Gyselinck et al., 2004], where it reads:

$$\frac{\partial \mathbf{H}}{\partial \mathbf{B}} = \nu \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + 2\frac{d\nu}{db^2}\begin{bmatrix} b_x b_x & b_x b_y \\ b_y b_x & b_y b_y \end{bmatrix}. \tag{4.85}$$

Also, the integrand of the first term has been already computed in the paper and reads as:

$$\nu\,\mathbf{grad}\,\delta\cdot\mathbf{grad}\,\mathbf{v} + 2\frac{d\nu}{db^2}(\mathbf{grad}\,\delta\cdot\mathbf{grad}\,x_n)(\mathbf{grad}\,\mathbf{v}\cdot\mathbf{grad}\,x_n). \tag{4.86}$$

Putting everything together, we arrive at

$$\int_\Omega \nu\,\mathbf{grad}\,\delta\cdot\mathbf{grad}\,\mathbf{v} + 2\frac{d\nu}{db^2}(\mathbf{grad}\,\delta\cdot\mathbf{grad}\,x_n)(\mathbf{grad}\,\mathbf{v}\cdot\mathbf{grad}\,x_n)\mathrm{d}x + \int_\Omega \sigma(t)\frac{\delta}{\Delta t}\mathbf{v}\mathrm{d}x =$$

$$\int_\Omega H(\mathbf{curl}_{2\mathrm{D}}\, x_n)\cdot \mathbf{curl}_{2\mathrm{D}}\,\mathbf{v}\mathrm{d}x + \int_\Omega \sigma(x)\frac{x_n}{\Delta t}\cdot\mathbf{v}\mathrm{d}x - \int_\Omega \left(j^0(t) + \sigma(x)\frac{x^{(j-1)}}{\Delta t}\right)\mathbf{v}\mathrm{d}x \quad \forall \mathbf{v}\in\mathbf{V}_h. \tag{4.87}$$

The second term on the left-hand side gives rise to a new type of matrix, by inserting a basis expansion for the increment δ and test function $\mathbf{v}$:

$$\underline{\underline{N}}(x_n) := \left[\int_\Omega 2\frac{d\nu}{db^2}(\mathbf{grad}_{2\mathrm{D}}\,\mathbf{b}_h^i(\mathbf{x})\cdot\mathbf{grad}\,x_n)(\mathbf{grad}_{2\mathrm{D}}\,\mathbf{b}_h^j(\mathbf{x})\cdot\mathbf{grad}\,x_n)\mathrm{d}x\right]_{i,j=1}^{N}. \tag{4.88}$$

Similarly to (4.36), the matrix for a single triangle K looks like:

$$\underline{\underline{N}}_K = 2\frac{d\nu}{db^2}|K| \left(\begin{bmatrix} \beta_1^1 & \beta_2^1 \\ \beta_1^2 & \beta_2^2 \\ \beta_1^3 & \beta_2^3 \end{bmatrix} \mathbf{grad}\, x_n|_K \right) \left(\begin{bmatrix} \beta_1^1 & \beta_2^1 \\ \beta_1^2 & \beta_2^2 \\ \beta_1^3 & \beta_2^3 \end{bmatrix} \mathbf{grad}\, x_n|_K \right)^T. \quad (4.89)$$

For the first term in the right-hand side we can write the vector

$$\vec{\boldsymbol{\rho}}(n) = \left[\int_\Omega H(\mathbf{curl}_{2D}\, x_n) \cdot \mathbf{curl}_{2D}\, \mathbf{b}_h^i(x) \mathrm{dx} \right]_{i=1}^N, \quad (4.90)$$

which for a single element leads to:

$$\begin{aligned} \vec{\boldsymbol{\rho}}_K(n) = \left[\int_K H(\mathbf{curl}_{2D}\, x_n) \cdot \mathbf{curl}_{2D}\, \lambda^i(x) \mathrm{dx} \right]_{i=1}^3 = \\ \left[\int_K H_x(\mathbf{curl}_{2D}\, x_n) \lambda_y^i(x) - H_y(\mathbf{curl}_{2D}\, x_y) \lambda_y^i(x) \mathrm{dx} \right]_{i=1}^3. \end{aligned} \quad (4.91)$$

Where H_x and H_y are the x and y components of the field $\mathbf{H}$. Since the gradients are constant on the mesh elements, we get:

$$\vec{\boldsymbol{\rho}}_K(n) = \frac{1}{3}|K| \left[H_x(\mathbf{curl}_{2D}\, x_n) \frac{\partial \lambda^i}{\partial y}(x) - H_y(\mathbf{curl}_{2D}\, x_y) \frac{\partial \lambda^i}{\partial x}) \right]_{i=1}^3 = \quad (4.92)$$

$$= \frac{1}{3}|K| \left[H_x(\mathbf{curl}_{2D}\, x_n) \beta_2^i - H_y(\mathbf{curl}_{2D}\, x_y) \beta_1^i) \right]_{i=1}^3 \quad (4.93)$$

The remaining terms lead to straight-forward matrices and vectors:

$$\underline{\underline{A}} := \left[\int_\Omega \nu\, \mathbf{curl}_{2\mathrm{D}}\, \mathbf{b}_h^i(\mathbf{x}) \cdot \mathbf{curl}_{2\mathrm{D}}\, \mathbf{b}_h^j(\mathbf{x}) \mathrm{dx} \right]_{i,j=1}^N \quad (4.94)$$

$$\underline{\underline{M}} := \left[\int_\Omega \sigma(x,t) \mathbf{b}_h^i(\mathbf{x}) \cdot \mathbf{b}_h^j(\mathbf{x}) \mathrm{dx} \right]_{i,j=1}^N \quad (4.95)$$

$$\vec{\boldsymbol{\phi}} := \left[\mathbf{j}^0(t) \mathbf{b}_h^i \right]_{i=1}^N \quad (4.96)$$

The Newton iteration in matrix form then looks like

$$\underline{\underline{N}}(n)\delta_h + \underline{\underline{A}}(n)\delta_h + \underline{\underline{M}}(n)\delta_h/\Delta t = \vec{\boldsymbol{\rho}} + \underline{\underline{M}}(n)x_n/\Delta t - \vec{\boldsymbol{\phi}} - \underline{\underline{M}}(n)x^{(j-1)}/\Delta t \quad (4.97)$$

where the index $(j-1)$ indicates the previous timestep.
Solving for δ_h:

$$(\underline{\underline{N}}(n)\Delta t + \underline{\underline{A}}(n)\Delta t + \underline{\underline{M}}(n))\delta_h = \vec{\boldsymbol{\rho}}\Delta t + \underline{\underline{M}}(n)x_n - \vec{\boldsymbol{\phi}}\Delta t - \underline{\underline{M}}(n)x^{(j-1)} \quad (4.98)$$

4.10 Backwards Differentiation Formula 2

In practice a method called Backwards Differentiation formula 2 (BDF-2) is often used instead of the Implicit Euler method. In order to compare to this method we derive the formula of a Newton step for it.

The formula of a timestep of the BDF-2 method is given by:

$$x^{(j+2)} - \frac{4}{3}x^{(j+1)} + \frac{1}{3}x^{(j)} = \frac{2}{3}hf(t_{j+2}, x^{(n+2)}) \tag{4.99}$$

$$\Leftrightarrow x^{(j+2)} - \frac{4}{3}x^{(j+1)} + \frac{1}{3}x^{(j)} - \frac{2}{3}hf(t_{j+2}, x^{(j+2)}) = 0. \tag{4.100}$$

We are interested in the basis expansion coefficient vector x^{j+2}, where the exponent indicates the timestep, and are given the previous two timesteps.

The function f in the case of the Eddy-current equation is given by

$$f(x^{(n+2)}, t_{n+2}) = \underline{\underline{M}}^{-1}(\varphi(t_{n+2}) - \underline{\underline{A}}(x^{(n+2)})(x^{(n+2)})), \tag{4.101}$$

which we insert into Equation (4.100):

$$\underline{\underline{M}}x^{(j+2)} - \frac{4}{3}\underline{\underline{M}}x^{(j+1)} + \frac{1}{3}\underline{\underline{M}}x^{(j)} - \frac{2}{3}h(\varphi(t_{n+2}) - \underline{\underline{A}}(x^{(n+2)})(x^{(n+2)})) = 0 \tag{4.102}$$

$$\Leftrightarrow (\underline{\underline{M}} + \frac{2}{3}h\underline{\underline{A}}(x^{(j+2)}))x^{(j+2)} - \frac{4}{3}\underline{\underline{M}}x^{(j+1)} + \frac{1}{3}\underline{\underline{M}}x^{(j)} - \frac{2}{3}h\varphi(t_{n+2}) = 0. \tag{4.103}$$

Again we use the Newton-Raphson iteration to find the solution of this nonlinear system. To do so, we need the Jacobian of the left hand side with respect to $x^{(j+2)}$, which as before is given by

$$J(x) = (\underline{\underline{M}} + \frac{2}{3}h(\underline{\underline{A}}(x) + \underline{\underline{N}}(x))). \tag{4.104}$$

A Newton-Raphson step then looks like:

$$(\underline{\underline{M}} + \frac{2}{3}h(\underline{\underline{A}}(x_n) + \underline{\underline{N}}(x_n)))\delta = (\underline{\underline{M}} + \frac{2}{3}h\underline{\underline{A}}(x_n))x_n - \frac{4}{3}\underline{\underline{M}}x^{(j+1)} + \frac{1}{3}\underline{\underline{M}}x^{(j)} - \frac{2}{3}h\varphi(t_{n+2}). \tag{4.105}$$

4.11 Termination Criterion for the Newton-Raphson Iteration

An important issue when using Newton-Raphson iterations is when to terminate the iteration, such that the iterate is close enough to the solution of the nonlinear

system of equations. The correct way to do this is a so called correction-based termination, which evaluates how much one iterate is different from the previous one and stops when the difference is small enough. For this project on the other hand, we used a residual-based termination criterion, which evaluates the relative residuum $\frac{||\text{lhs}(x^{(k)})-\text{rhs}||}{||\text{rhs}||}$ and terminates when this value is small enough, lhs is the left hand side of the system of equation and rhs is the right hand side. For example in BDF2 the non linear system of equations is:

$$F(x^{(j+2)}) = (\underline{\underline{M}} + \frac{2}{3}h\underline{\underline{A}}(x^{(j+2)}))x^{(j+2)} - \frac{4}{3}\underline{\underline{M}}x^{(j+1)} + \frac{1}{3}\underline{\underline{M}}x^{(j)} - \frac{2}{3}h\varphi(t_{n+2}), \tag{4.106}$$

and the left hand side and right hand side are given by:

$$lhs = (\underline{\underline{M}} + \frac{2}{3}h\underline{\underline{A}}(x^{(j+2)})), \tag{4.107}$$

$$rhs = \frac{4}{3}\underline{\underline{M}}x^{(j+1)} - \frac{1}{3}\underline{\underline{M}}x^{(j)} + \frac{2}{3}h\varphi(t_{n+2}). \tag{4.108}$$

For this project the tolerance was set to 10^{-6}.

4.12 Rosenbrock Wanner Timestepping for the Eddy Current Model

The following section follows the notes kindly provided by Professor Hiptmair.

Consider the A-based formulation of the Eddy current model with ECE boundary conditions, which reads as following:

$$\begin{aligned}&\text{Seek } \mathbf{A} : [0,T] \to \mathbf{H}_0(\mathbf{curl},\Omega)\\ &\int_\Omega \underline{\underline{\sigma}}(x)\partial_t\mathbf{A}\cdot\mathbf{A}'\mathrm{dx} + \int_\Omega \underline{\underline{\nu}}(x,\mathbf{curl}\,\mathbf{A})\,\mathbf{curl}\,\mathbf{A}\cdot\mathbf{curl}\,\mathbf{A}'\mathrm{dx}\\ &= \int_\Omega j^0\mathbf{A}'\mathrm{dx}\ \forall A' \in \mathbf{H}_0(\mathbf{curl},\Omega)\end{aligned} \tag{4.109}$$

where $\underline{\underline{\nu}}$ is the B-dependent magnetic susceptibility:

$$\underline{\underline{\nu}} : \Omega \times \mathbb{R}^3 \to \mathbb{R}^{3,3}_{\text{spd}}, \tag{4.110}$$

and $\underline{\underline{\omega}}$ is the electric conductivity, which is defined separately in a conductive and non-conductive domain:

$$\underline{\underline{\sigma}}(x) = \begin{cases} 0, x \in \Omega_I, \\ \underline{\underline{\sigma}}_0(x), x \in \Omega_c := \Omega \setminus \bar{\Omega}_I, \end{cases} \tag{4.111}$$

where Ω_I is the non-conductive medium.

Furthermore, we will separate the domain Ω in two types of domain: one which is defined by a rigid body, and one which is deforming, such as an air-gap:

$$\Omega = \Omega_d \bigcup_{i=1}^{P} \Omega_i, \tag{4.112}$$

where i are the multiple rigid domains while d references the deformable domains.

As an essential part of this work we assume that the conductive domain is contained in the union of rigid bodies:

$$\Omega_c \subset \bigcup_{i=1}^{P} \Omega_i. \tag{4.113}$$

One should note that formulation (4.109) a so-called ungauged formulation, which means that the solution $\mathbf{A}$ is unique only up to a kernel contribution, and

$$\mathbf{A} \in \{\vec{z} \in \mathbf{H}_0(\mathbf{curl}, \Omega) : \mathbf{curl}\, \vec{z} = 0\}. \tag{4.114}$$

Now we will proceed to solve the problem using a Finite Element discretization, where we substitute the space $\mathbf{H}_0(\mathbf{curl}, \Omega)$ with a finite dimensional subspace V_h.

We assume that the mesh used by the FE discretization resolves the domain $\overline{\Omega} = \overline{\Omega}_I \,\dot\cup\, \overline{\Omega}_C$

Since the conducting and non-conducting domain have different physical properties, that can be exploited to simplify the solution, we split the ungauged discrete variational formulation, and define two subspaces of V_h

$$V_h^C := \text{span of all basis functions of } V_h, \text{ whose supports overlap with } \Omega_c, \tag{4.115}$$

$$V_H^I := \text{span of all basis functions of } V_h, \text{ whose supports overlap with } \overline{\Omega}_I. \tag{4.116}$$

From these definitions, follows that

$$V_h = V_h^C + V_h^I. \tag{4.117}$$

Using these two new subspaces we define a new problem formulation:

$$\begin{aligned}
&\text{seek } \mathbf{A}_h^C : [0,T] \to V_h^C, \\
&\quad \mathbf{A}_h^I : [0,T] \to V_h^I, \\
&\int_\Omega \underline{\underline{\sigma}}(x)\partial_t \mathbf{A}_h^C \cdot \mathbf{W}_h^C \mathrm{d}x + \int_\Omega \underline{\underline{\nu}}(x, \mathbf{curl}(\mathbf{A}_h^C + \mathbf{A}_h^I)\, \mathbf{curl}(\mathbf{A}_h^C + \mathbf{A}_h^I) \cdot \mathbf{curl}\, \mathbf{W}_h^C \mathrm{d}x \\
&\qquad = \int_\Omega j^0 \mathbf{W}_h^C \mathrm{d}x \;\forall \mathbf{W}_h^C \in V_h^C, \\
&\int_\Omega \underline{\underline{\nu}}(x, \mathbf{curl}(\mathbf{A}_h^C + \mathbf{A}_h^I)\, \mathbf{curl}(\mathbf{A}_h^C + \mathbf{A}_h^I) \cdot \mathbf{curl}\, \mathbf{W}_h^I \mathrm{d}x = \int_\Omega j^0 \mathbf{W}_h^I \mathrm{d}x \;\forall \mathbf{W}_h^I \in V_h^I.
\end{aligned} \tag{4.118}$$

This problem gives rise to the following abstract algebraic structure:

$$\underline{\underline{M}}\frac{d}{dt}\vec{\mu} + \underline{\underline{A}}_C(\vec{\mu}(t) + \vec{\xi}(t)))(\vec{\mu}(t) + \vec{\xi}(t))) = \vec{\varphi}_C(t), \tag{4.119}$$

$$\underline{\underline{A}}_I(\vec{\mu}(t) + \vec{\xi}(t)))(\vec{\mu}(t) + \vec{\xi}(t))) = \vec{\varphi}_I(t). \tag{4.120}$$

The matrices $\underline{\underline{A}}_C$ and $\underline{\underline{A}}_I$ both depend on the magnetic flux B, and they map vectors in the whole discrete space to vectors defined on the conductive and non-conductive domain respectively:

$$\underline{\underline{A}}_C : \mathbb{R}^N \to \mathbb{R}^{N_C,N} \ , \ N_C := |V_h^C|, \tag{4.121}$$

$$\underline{\underline{A}}_I : \mathbb{R}^N \to \mathbb{R}^{N_I,N} \ , \ N_I := |V_h^I|, \tag{4.122}$$

$$N = N_C + N_I. \tag{4.123}$$

The equations (4.119) and (4.120) form a Differential Algebraic Equation (DAE), which is a system which contains a differential and algebraic equation. It is a so-called index-1 DAE.

The general form of a DAE is:

$$\dot{y} = d(t, y, z) \ , \ d : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \tag{4.124}$$

$$0 = c(t, y, z) \ , \ c : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \tag{4.125}$$

and a DAE is of index-1 when you can solve c for z.

In [Rang and Angermann, 2005] the authors propose Rosenbrock W-methods to solve DAEs of index 1. A DAE becomes an initial value problem (IVP) when you add the condition $y(t_0) = y_0$:

$$IVP: \ \dot{y} = d(t, y, z) \ , \ d : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \tag{4.126}$$

$$0 = c(t, y, z) \ , \ c : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n, \tag{4.127}$$

$$y(t_0) = y_0. \tag{4.128}$$

Assume you are at time t_0, know the approximate solution at that time given by y_0 and z_0, and want to compute the approximate solution y_1, z_1 at time t_1.

The stage equations used to solve a DAE defined by the Equations (4.126) to (4.128) are given by

$$\begin{pmatrix} t_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} t_0 \\ y_0 \\ z_0 \end{pmatrix} + \sum_{j=1}^{i-1} a_{ij} \begin{pmatrix} \tau \\ k_j^y \\ k_j^z \end{pmatrix}, \tag{4.129}$$

while the increments k_i^y and k_i^z are given by the equations

$$\begin{pmatrix} Mk_i^y \\ 0 \end{pmatrix} = \tau \begin{pmatrix} d(t_i, y_i, z_i) \\ c(t_i, y_i, z_i) \end{pmatrix} + \tau \underline{\underline{J}} \sum_{j=1}^{i} \gamma_{ij} \begin{pmatrix} k_j^y \\ k_j^z \end{pmatrix} + \tau^2 \gamma_i \begin{pmatrix} \partial_t d \\ \partial_t c \end{pmatrix}. \tag{4.130}$$

The matrix J is given by:

$$\underline{\underline{J}} = \begin{pmatrix} \partial_y d & \partial_z d \\ \partial_y c & \partial_z c \end{pmatrix}. \tag{4.131}$$

The update equation to compute y_1 and z_1 is given by

$$\begin{pmatrix} y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} y_0 \\ z_0 \end{pmatrix} + \tau \sum_{i=0}^{s} b_i \begin{pmatrix} k_i^y \\ k_i^z \end{pmatrix}. \tag{4.132}$$

For the purpose of this work, we will focus on stiffly accurate methods, which provide good performance for stiff problems [Michels et al., 2017]. Stiffly accurate methods satisfy the following condition by definition:

$$a_{si} + \gamma_{si} = b_i \ , \ i = 1, \dots, s. \tag{4.133}$$

The coefficients for such a method are presented in [Rang and Angermann, 2005]:

γ	=	4.3586652150845900e-1		
α_{21}	=	8.7173304301691801e-1	γ_{21}	= −8.7173304301691801e-1
α_{31}	=	8.4457060015369423e-1	γ_{31}	= −9.0338057013044082e-1
α_{32}	=	−1.1299064236484185e-1	γ_{32}	= 5.4180672388095326e-2
α_{41}	=	0.0000000000000000e+00	γ_{41}	= 2.4212380706095346e1
α_{42}	=	0.0000000000000000e+00	γ_{42}	= −1.2232505839045147e+00
α_{43}	=	1.0000000000000000e+00	γ_{43}	= 5.4526025533510214e-1
b_1	=	2.4212380706095346e-1	$\hat{b}_1$	= 3.7810903145819369e-1
b_2	=	−1.2232505839045147e+00	$\hat{b}_2$	= −9.6042292212423178e-2
b_3	=	1.5452602553351020e+00	$\hat{b}_3$	= 5.0000000000000000e-1
b_4	=	4.3586652150845900e-1	$\hat{b}_4$	= 2.1793326075422950e-1

The diagonal coefficients γ_{ii} are equal to γ. This method is of order 3 with 4 stages, and it is A-stable.

Now we proceed to compute the matrix $\underline{\underline{J}}$. To this end, we explicitly write out the functions c and d.

$$d(t, \vec{\mu}, \vec{\xi}) = (\varphi_C(t) - A_C(\vec{\mu} + \vec{\xi})(\vec{\mu} + \vec{\xi})), \tag{4.134}$$

$$c(t, \vec{\mu}, \vec{\xi}) = (\varphi_I(t) - A_I(\vec{\mu} + \vec{\xi})(\vec{\mu} + \vec{\xi})). \tag{4.135}$$

Since we already computed the derivative of A in Section 4.9, we infer that

$$d_{\vec{\mu}} = -(A_{C,C}(\vec{\mu} + \vec{\xi}) + N_{C,C}(\vec{\mu} + \vec{\xi})), \tag{4.136}$$

$$d_{\vec{\xi}} = -(A_{I,C}(\vec{\mu} + \vec{\xi}) + N_{I,C}(\vec{\mu} + \vec{\xi})), \tag{4.137}$$

$$c_{\vec{\mu}} = -(A_{C,I}(\vec{\mu} + \vec{\xi}) + N_{C,I}(\vec{\mu} + \vec{\xi})), \tag{4.138}$$

$$c_{\vec{\xi}} = -(A_{I,I}(\vec{\mu} + \vec{\xi}) + N_{I,I}(\vec{\mu} + \vec{\xi})), \tag{4.139}$$

where $A_{*,\dagger}(\vec{\mu})$ and $N_{*,\dagger}(\vec{\mu})$ are given by

$$\underline{\underline{A}}_{*,\dagger}(\vec{\mu}) := \left[\int_\Omega \nu(\vec{\mu})\,\mathbf{curl}_{2\mathrm{D}}\,\mathbf{b}_h^i(\mathbf{x})\cdot\mathbf{curl}_{2\mathrm{D}}\,\mathbf{b}_h^j(\mathbf{x})\mathrm{dx}\right]_{i,j=1}^{N_*,N_\dagger}, \mathbf{b}_h^i \in V_h^*, \mathbf{b}_h^j \in V_h^\dagger, \tag{4.140}$$

$$\underline{\underline{N}}_{*,\dagger}(\vec{\mu}) := \left[\int_\Omega 2\frac{d\nu}{db^2}(\vec{\mu})(\mathbf{grad}_{2\mathrm{D}}\,\mathbf{b}_h^i(\mathbf{x})\cdot\mathbf{grad}\,\vec{\mu})(\mathbf{grad}_{2\mathrm{D}}\,\mathbf{b}_h^j(\mathbf{x})\cdot\mathbf{grad}\,\vec{\mu})\mathrm{dx}\right]_{i,j=1}^{N}, \mathbf{b}_h^i \in V_h^*, \mathbf{b}_h^j \in V_h^\dagger, \tag{4.141}$$

and $*, \dagger$ are either C or I.

The time derivatives of d and c outside of the air-gap are computed as following:

$$\frac{d}{dt}c(t,\vec{\mu},\vec{\xi}) = \dot{\vec{\varphi}}_I, \tag{4.142}$$

$$\frac{d}{dt}d(t,\vec{\mu},\vec{\xi}) = \dot{\vec{\varphi}}_C. \tag{4.143}$$

Since the matrix A is constant outside of the airgap.

4.12.1 Time-Parametric Finite Element Methods (FEM) for the Air-Gap

Equation (4.130) presents a problem for the setting of a rotating machine. If we re-mesh the Air-Gap to compute the increments corresponding to different times, we lose correspondence of degrees of freedom (DOFs) between the increments, which invalidates the solution.

To overcome this issue, we deform the mesh for increment timesteps, instead of reconstructing it for each increment. This allows to keep the same number of DOFs and to keep track of them between the increments. See Figure 4.3 for a visual representation of mesh deformation.

In this work we assume constant angular velocity of the rotor

Assumption 1. $\omega = C$.

This allows to parametrize the matrix, and use the paradigm of Parametric FEM to compute the time-dependent deformation of the mesh.

One alternative to using parametric FEM is to rotate mesh nodes for every increment and then assembling the stiffness matrix using the modified mesh. The reason that this could be undesirable is that in the equation for the increments (4.130) a time derivative of the stiffness matrix is needed, and using a numerical approximation could interfere with the order of the method.

Let a triangle of the original mesh be $\hat{K}$, $K(t)$ the deformed triangle at time t, $\Phi_D(t)$ a transformation $\Phi_D(t) : \hat{K} \mapsto K(t)$, and $(\Phi_D^*(t)u)(\hat{x}) := u(\Phi_D(t)(\hat{x}))$ the pullback of a function to $\hat{K}$. As proven in the Lecture Script [Hiptmair, 2023b] the following relation holds:

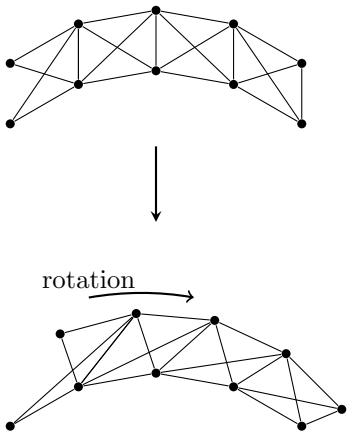


Figure 4.3: This figure depicts the deformation of the air-gap for one of the increments in the ROW equations

$$(\mathbf{grad}_{\hat{x}}(\Phi_d^* u))(\hat{x}) = (D\Phi(\hat{x}))^T (\mathbf{grad}_x\, u)(\Phi(\hat{x})). \tag{4.144}$$

A entry of the stiffness matrix for the deformed mesh reads as

$$[A(t)]_{i,j}^N = \int_K \mu(t)\, \mathbf{grad}\, b_K^j(t)(x) \cdot \mathbf{grad}\, b_K^i(t)(x) dx \tag{4.145}$$

$$= \int_{\hat{K}} \mu(t) \Phi_D^*(t)(\mathbf{grad}\, b_{\hat{K}}^j))(\hat{x}) \cdot (\Phi_D^*(t)(\mathbf{grad}\, b_{\hat{K}}^i))(\hat{x}))| \det D\Phi_D(t)(\hat{x})| \mathrm{d}\hat{x} \tag{4.146}$$

$$= \int_{\hat{K}} \mu(t)(D\Phi_D(t))^{-T}(\mathbf{grad}\, b_K^j)) \cdot (D\Phi_D(t))^{-T}(\mathbf{grad}\, b_k^i))| \det D\Phi_D(t)(\hat{x})| \mathrm{d}\hat{x}, \tag{4.147}$$

where to get to Equation (4.147) we used the gradient formula (4.144) and the fact that the gradients of first order Lagrangian FEM are constant, so x dependency falls away.

Now we present the time-dependent transformation $\Phi_D(t)$.

Figure 4.4 shows how points in the air-gap are transformed, along with the parameters which depend on the machine geometry, which are the inner and outer radius of the air-gap (r and R), and the angular velocity ω. Without loss of generality, we make the following assumption:

Assumption 2. *Only the outer boundary is moved, so the rotor is outside of the stator.*

Since only the outer boundary is moved, points in the air-gap are rotated by an amount which is linearly proportional to the distance to the outer boundary.

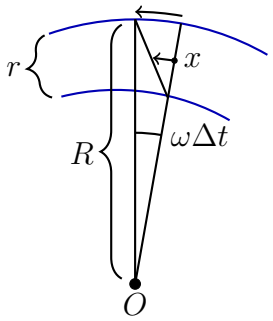


Figure 4.4: This figure shows the transformation $\Phi_D(t)$ in the air-gap. The outer boundary moves, while the inner boundary is stationary. Thus every point in the air-gap is moved along the rotational direction by an amount dependent on the proximity to the outer boundary. ω is the angular speed and Δt is the time interval.

In polar coordinates this translates to:

$$\begin{cases} r_x(t) = r_x(0) \\ \varphi_x(t) = \frac{r_x(t)-r}{R-r} + \varphi_x(0), \end{cases} \tag{4.148}$$

where r_x and φ_x are the polar coordinates of the point x with respect to it's initial position.

In Cartesian coordinates it becomes

$$\Phi_D(t)(\vec{x}) = \sqrt{x_1^2 + x_2^2} \begin{bmatrix} \cos\left(\frac{\sqrt{x_1^2+x_2^2}-r}{R-r}\omega t + \varphi(0)\right) \\ \sin\left(\frac{\sqrt{x_1^2+x_2^2}-r}{R-r}\omega t + \varphi(0)\right) \end{bmatrix}, \tag{4.149}$$

with Jacobian matrix

$$D\Phi_D(t)(\vec{x}) = \begin{bmatrix} \frac{x_1\cos(\theta)}{r_x} - \frac{tx_1\omega\sin(\theta)}{R-r} & \frac{x_2\cos(\theta)}{r_x} - \frac{tx_2\omega\sin(\theta)}{R-r} \\ \frac{x_1\sin(\theta)}{r_x} + \frac{tx_1\omega\cos(\theta)}{R-r} & \frac{x_2\sin(\theta)}{r_x} + \frac{tx_2\omega\cos(\theta)}{R-r}) \end{bmatrix}, \tag{4.150}$$

where $\theta = \frac{t\omega(r_x-r)}{R-r} + \phi$ and $r_x = \sqrt{x_1^2 + x_2^2}$.

Next, we compute the time derivative of the stiffness matrix in the air-gap. Recalling the definition of stiffness matrix entries (4.147), we see that only the Jacobian and the determinant are time-dependent, which simplifies

the computation resulting in

$$\left[\frac{d}{dt}\underline{\underline{A}}(\vec{\xi})\right]_{i,j}^{N} = \int_{\hat{K}} \mu(\vec{\xi})(\frac{d}{dt}(D\Phi_D(t)))^{-T}(\mathbf{grad}\, b_K^j)) \cdot (\frac{d}{dt}(D\Phi_D(t)))^{-T}(\mathbf{grad}\, b_k^i))|\frac{d}{dt}(\det D\Phi_D(t)(\hat{x}))|\mathrm{d}\hat{x} \tag{4.151}$$

The time derivative of the Jacobian is

$$\frac{d}{dt}D\Phi_D(t)(\vec{x}) = \begin{bmatrix} \frac{x_1\omega\sin(\theta)}{R-r} & \frac{x_2\omega\sin(\theta)}{R-r} \\ \frac{x_1\omega\cos(\theta)}{R-r} & \frac{x_2\omega\cos(\theta)}{R-r}) \end{bmatrix} \tag{4.152}$$

For the computation of the load vector on the deformed mesh we rely on the fact that the prescribed current is constant for a given cell:

$$[\varphi_i(t)]_{i=1}^N = \int_K j^0(t)(x)b_i(x)dx = \int_{\hat{K}} (\Phi_D^*(t)j^0(t))(\hat{x})b_i(\hat{x})|\det D\Phi_D(t)(\hat{x})|\mathrm{d}\hat{x} \tag{4.153}$$

$$= j^0(t)\int_{\hat{K}} b_i(\hat{x})|\det D\Phi_D(t)(\hat{x})|\mathrm{d}\hat{x}, \tag{4.154}$$

so only a change of surface area of the cell needs to be taken into account.

The derivative is:

$$[\dot{\varphi}_i(t)]_{i=1}^N = \dot{j^0}(t)\int_{\hat{K}} b_i(\hat{x})|\det D\Phi_D(t)(\hat{x})|\mathrm{d}\hat{x} + j^0(t)\int_{\hat{K}} b_i(\hat{x})|\frac{d}{dt}(\det D\Phi_D(t)(\hat{x}))|\mathrm{d}\hat{x}. \tag{4.155}$$

Finally, we need to compute the time derivative of d and c.

$$\frac{d}{dt}c(t,\vec{\mu},\vec{\xi}) = \frac{d}{dt}\left(\varphi_C(t) - A_C(\vec{\mu}+\vec{\xi})\right), \tag{4.156}$$

$$\tag{4.157}$$

where we computed both quantities in (4.151) and (4.155).

4.13 Implementation Note

Time-parametric are the exact way to compute the time derivative of the mesh deformation. Due to it's high complexity and time constraints, this was not implemented, and we opted for a numerical estimation of the deformation derivative. This was done with the forward finite difference method, where the derivative is estimated as:

$$\frac{d}{dt}\underline{\underline{A}}(t) = \frac{\underline{\underline{A}}(t+dt) - \underline{\underline{A}}(t)}{dt}, \tag{4.158}$$

and $\underline{\underline{A}}(t+dt)$ is obtained from $\underline{\underline{A}}(t)$ by applying the transformation (4.149) and reassembling the matrix.

4.14 Re-Computing the Solution in the airgap

In the equation for the temporary solutions y_i, z_i, (4.129) we notice that even in the algebraic (deformable) domain, the solution at the previous timestep (z_0) is needed. This is problematic, since we remesh the airgap, so we don't have correspondence of DOFs between timesteps. Thankfully, the solution in the algebraic is not time dependent, and can be computed instantly using the solution at the non-deformable domain boundary. We solve a stationary problem, as described in Section 4.5, on the deformable domain, and use the solution of the previous timestep at the boundary of the non deformable domain as boundary values.

4.15 Adaptive Timestepping

Adaptive timestepping is especially important in ROW timestepping, since the error for a given timestep depends on the error of the linearization, which changes over the simulation. For example when the magnetic field is strong the magnetic material becomes saturated, and the B-H curve becomes linear.

We implemented adaptive timestepping according to Section 2.4. As the error norm we chose the dissipative power, which quantifies how much power is dissipated on the domain. It is defined exactly later in Section 6.1.

Chapter 5

Performance Comparison Between BDF-2 and ROW Methods

In order to compare the performance of BDF-2 and ROW, we proceed as following: first we define the compute intensive operations in the timestepping scheme, and for each of the three methods we estimate how many of these operations are needed per timestep and Newton-Raphson step. Then we define an error norm, which describes how much the solution found by a method deviates from the exact solution. For each method we impose that the error norm is below a threshold. In the Newton-iteration methods (implicit Euler and BDF-2) there are two parameters which influence the accuracy: the timestep size and the threshold for the residual of the Newton-iteration. Thus the combination which is optimal in terms of total cost and is below the required accuracy has to be found. For the Rosenbrock-Wanner method on the other side only the step-size has to be optimized. Finally, the cost of the BDF-2 method with the optimal Newton-threshold and timestep size will be compared with the cost of the Rosenbrock-Wanner method with the optimal step-size.

5.1 Theoretical Performance Estimates

We give the performance estimates based on how many times the most expensive computations have to be performed. The cost of the specific operations can depend on how much this operations has been optimized, so a user of the timestepping method can discern which method will be more efficient for his

implementation of the operations. The operations are defined as following:

q_* = "cost of quadrature for element of matrix *, where * can be A, M, N"
P = "cost of computing the preconditioner of the system matrix"
S = "cost of solving the LSE when the preconditioner has already been decomposed"

And the number of elements per domain is defined as following:

- N: total number of elements in the domain
- N_L: number of elements in linear material domain.
- N_{nnL}: number of elements in nonlinear domain.
- N_R: in rigid domain.
- N_D : in deformable domain.

By elements we mean the 2D mesh elements, which in our case are triangles.

5.1.1 Implicit Euler and BDF-2

The performance of BDF-1 and BDF-2 methods are equivalent in terms of the operations defined above, since the same matrices are assembled, and general structure is the same. The number of operations can be directly read from Equation (4.98), which describes a Newton step for the implicit Euler method. The following operations are needed per program execution, per timestep, and Newton iteration:

1. Once per program execution: $N_L \cdot q_N + N_L \cdot q_A + N \cdot q_M$.
2. Once per timestep: $N_D \cdot q_N + N_D \cdot q_A + N_D \cdot q_M$.
3. Once per Newton step: $N_{nnL} \cdot q_N + N_{nnL} \cdot q_A + P + S$.

Since the stiffness matrix depends on the solution in the domain where the material is nonlinear, it has to be reassembled there in every Newton step. The mass matrix only has to be reassembled in the airgap, since it is remeshed in every timestep.

The total time needed for one timestep depends on the time required per operation, so we define Δt_* as the time needed for operation $*$. The time needed for one timestep is:

$$\begin{aligned} &N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M \\ &+ N_{Newton} \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + P \cdot \Delta t_P + S \cdot \Delta t_S), \end{aligned} \quad (5.1)$$

where N_{Newton} is the number of Newton iterations needed to reach the desired residuum in a given timestep.

5.1.2 Rosenbrock-Wanner

This method uses the following operations:

1. Once per program execution: $N_L \cdot q_N + N_L \cdot q_A + N \cdot q_M$.
2. Once per timestep: $N_D \cdot q_N + N_D \cdot q_A + N_D \cdot q_M + P$.
3. Once per increment: $N_{nnL} \cdot q_N + N_{nnL} \cdot q_A + N_D \cdot q_N + N_D \cdot q_A + N_D \cdot q_M + S$.

Since the left-hand side is the same for all four increments, the preconditioner can be computed once per timestep (P), and the system can be solved quickly for different right hand sides (S). We need to recompute the matrices in the airgap for every increment, since we are deforming it there

The cost per Rosenbrock Wanner timestep is:

$$\begin{aligned} &N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M + LU \cdot \Delta t_{LU} \\ &+ 4 \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + S \cdot \Delta t_S). \end{aligned} \tag{5.2}$$

Chapter 6

Results

In the previous chapter we introduced a theoretical comparison on the cost per timestep of Newton iteration-bound methods and ROW. Now we test both methods on a model geometry, and while keeping the error below a critical threshold, we record the number of timesteps, to propose a suggestion on which method is more efficient.

6.1 Defining the Error Norm

In order to determine the quality of a solution, be it the solution obtained with iteration-based methods, or with ROW methods, we need to define an error norm with respect to a baseline solution, which represents the exact solution. For the Eddy-current problem, the quantity of interest is the energy dissipation in the domain, since often the objective of the design of an electrical machine is to minimize such energy dissipation. The energy dissipation in a spacial domain is given by Poyntings theorem [Poynting and Strutt, 1997]. Let Ω be the computational domain. The energy dissipation per time, which corresponds to dissipative power, is given by

$$P := \int_{\Omega} \mathbf{H} \cdot \dot{\mathbf{B}} + j \cdot \mathbf{E} \mathrm{dx}, \tag{6.1}$$

where $\mathbf{B}$ is the magnetic field, j is the electric current and $\mathbf{E}$ is the electric field. The normalized difference between the baseline solution u_b and a computed solution u is given by

$$||u - u_b||_P := \frac{\int_{\Omega} ||(\mathbf{H} \cdot \dot{\mathbf{B}} + j \cdot \mathbf{E}) - (\mathbf{H}_b \cdot \dot{\mathbf{B}}_b + j_b \cdot \mathbf{E}_b)|| \mathrm{dx}}{\int_{\Omega} ||(\mathbf{H}_b \cdot \dot{\mathbf{B}}_b + j_b \cdot \mathbf{E}_b)|| \mathrm{dx}}, \tag{6.2}$$

where the fields are derived from the solutions u_b and u. In the discrete solution, the fields $\mathbf{E}$, $\mathbf{B}$ and the current are defined on the cells. We chose first order Lagrangian elements, so the fields and current are constant on the mesh

elements. The error norm for a solution u with corresponding magnetic field B and electric field E then becomes:

$$||u - u_b||_P = \sum_K |K| \cdot \frac{||(\mathbf{H}_K \cdot \dot{\mathbf{B}}_K + j_K \cdot \mathbf{E}_K) - (\mathbf{H}_{b,K} \cdot \dot{\mathbf{B}}_{b,K} + j_{b,K} \cdot \mathbf{E}_{b,K})||}{||\mathbf{H}_{b,K} \cdot \dot{\mathbf{B}}_{b,K} + j_{b,K} \cdot \mathbf{E}_{b,K})||}, \tag{6.3}$$

where K is a mesh element, and $|K|$ is the area of such an element.

6.2 Computing a Baseline

In order to compute the error for a given approximate solution, we need a reference solution. Except for very simple geometries it is not possible to compute an exact solution. Thus we computed a discrete solution that converges to the exact solution. To find such a discrete solution we computed a solution using fixed timestepping, and decreased the step size by a factor of 2, until the norm as described by (6.3) was small enough:

$$\text{Repeat: } h \leftarrow h/2 \tag{6.4}$$

$$\text{until } ||u_h - u_{h/2}||_P < tol, \tag{6.5}$$

where h is the step size, and we set tol to 1e-4.

6.3 Comparison Between Iteration-Based Methods and ROW on a Static Geometry

We chose a transformer as a model geometry, where a prescribed current is set on the primary coils, and a current is induced in the secondary coils. A transformer core goes through the primary and secondary coils, in order to transport the magnetic flux from the primary to secondary coils. A diagram is shown in Figure 6.1 The current applied to the primary coils is sinusoidal, with frequency 50hz and an amplitude which is changed for different experiment, and we simulate a whole period, which corresponds to 0.02 seconds. The H-B curve of the core has been set to a parametrized analytical function, where the parameters have been changed in different experiments. The formula for the relative magnetic permeability is:

$$\mu_r(B) := \frac{\mu}{\mu_0} = \frac{\max}{1 + B^4 * \max/c} + 1. \tag{6.6}$$

For the parameters max = 5000, $c = 100$, the function is shown in Figure 6.2

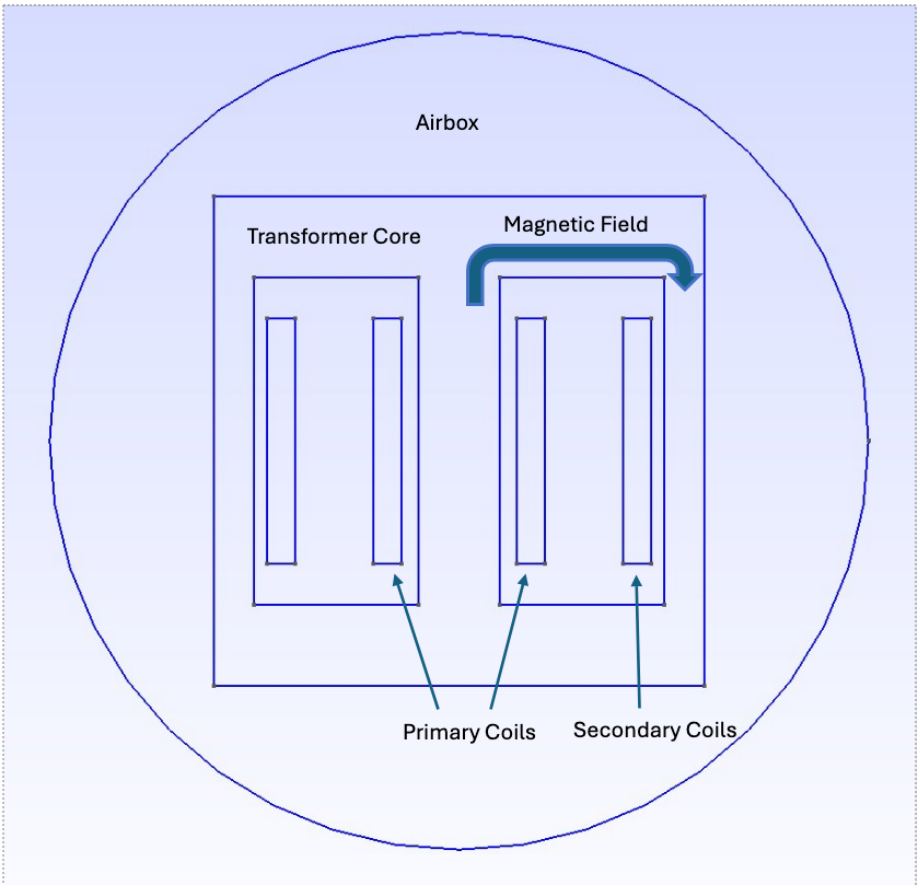
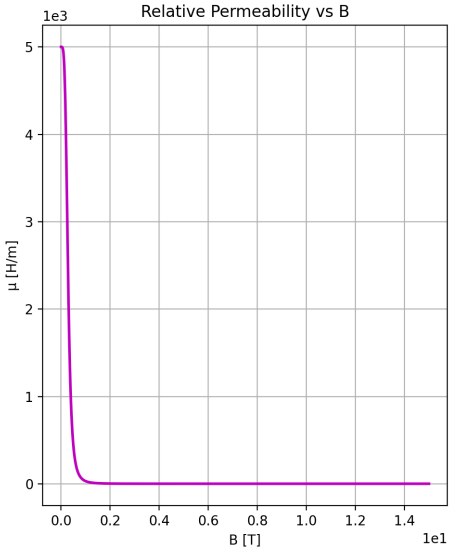
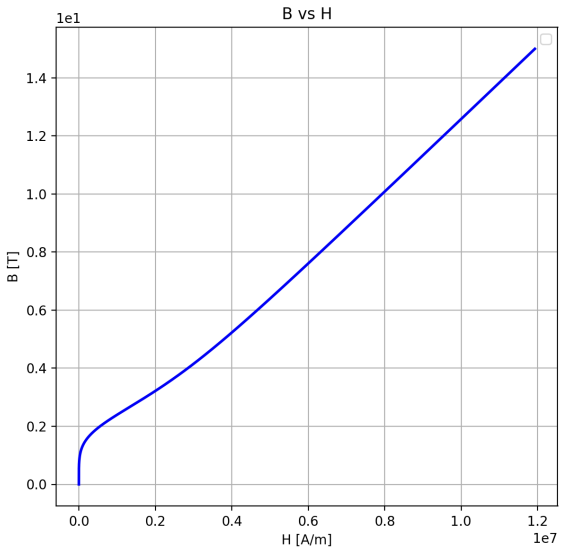


Figure 6.1: Diagram for a transformer. A current is applied to the primary coils, and a current is induced in the secondary coils. The transformer core transports the magnetic flux from the primary to secondary coils. Two coils are missing on the external part of the core but the mechanics remain the same. The dimensions of the transformer core are: width = 60cm, heidth = 60cm.



(a) Relative Permeability



(b) B-H curve

Figure 6.2: Relative permeability and B-H curve for the analytical function $\mu_r(B) := \frac{\mu}{\mu_0} = \frac{\max}{1+B^4 * \max/c} + 1, \quad \max = 5000,\ c = 100.$

6.3.1 Fixed timestep

To avoid issues which arise from adaptive timestep size, we first analyze the performance using fixed timestep. We try different sets of conditions to evaluate how they influence the performance. The parameters we evaluate are the current amplitude and nonlinearity. We will change the parameters of the nonlinearity in order to assess how the strength of the curvature will influence the results. In a nonlinear setting, ROW methods can degrade, because the nonlinearity is linearized. Thus if the timestep is too big, this approximation leads to a diverging solution. In order to understand what the largest possible timestep is we perform a simulation with a large timestep and decrease it until the solution doesn't diverge.

We try two sets of parameters:

- Current density j: $2 \cdot 10^7 A/m^2$, nonlinearity parameters: max $= 1000$, $c = 100$ (Set 1).
- Current density j: $1 \cdot 10^7 A/m^2$, nonlinearity parameters: max $= 1000$, $c = 100$ (Set 2).
- Current density j: $1 \cdot 10^7 A/m^2$, nonlinearity parameters: max $= 500$, $c = 100$ (Set 3).

For each of the sets of parameters, we compute a baseline solution at $5 \cdot 10\text{e}{-6}s$, evaluate what the largest possible timestep for the ROW method is,

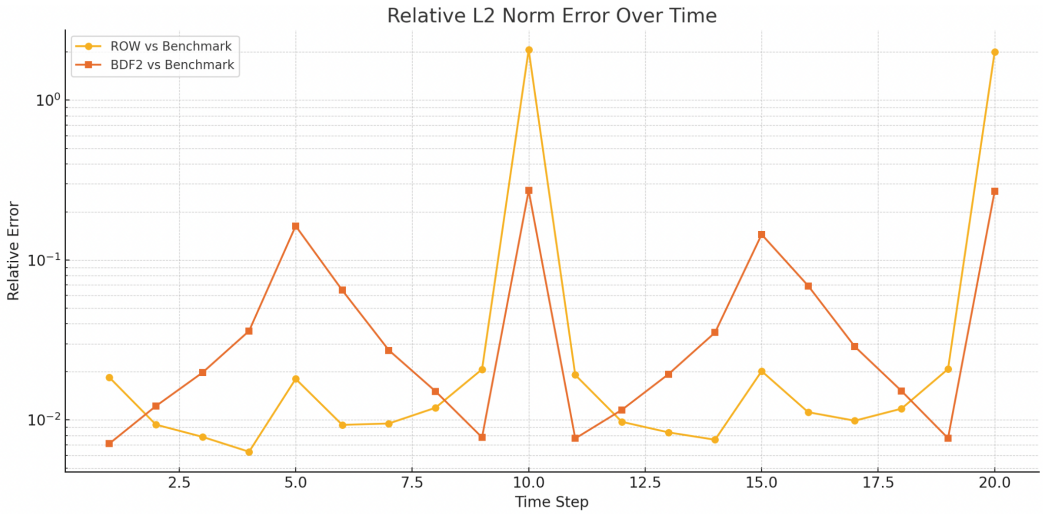


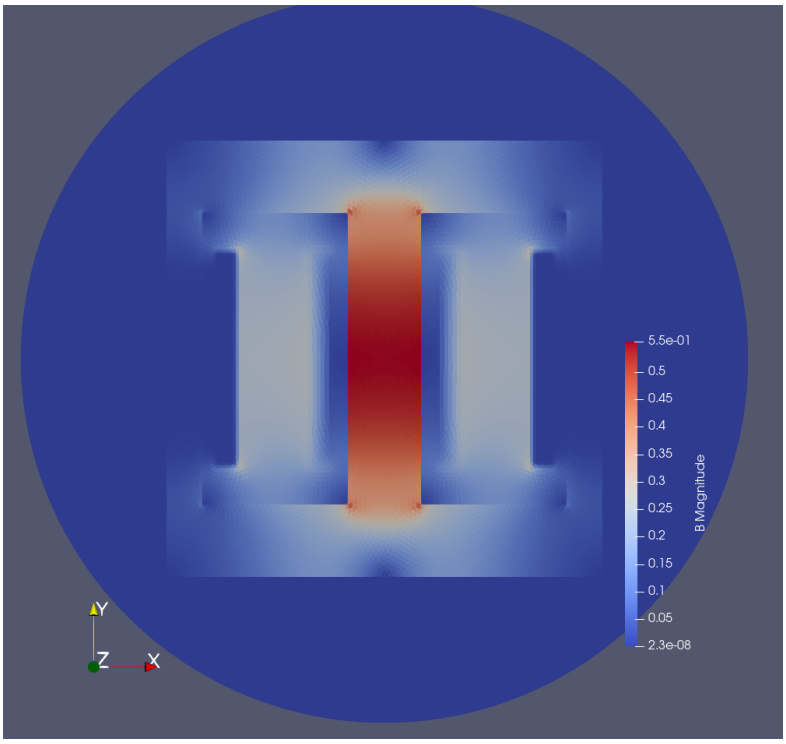
Figure 6.3: This is the L2 norm of the error over the computational domain over time. For every timestep, we ccompare the solution with the baseline solution, which was computed using a smaller timestep. We divide the norm of difference between the two solutions by the norm of the baseline solution for every timestep. ROW has a better accuracy for all timesteps except two.

and compare the error norm and number of Newton steps/timesteps of BDF-2 and ROW respectively. We compute the error for every timestep, of we there are 20 in total. In Table 6.1 median of the errors is shown, while Figure 6.3 shows the error for all the timesteps.

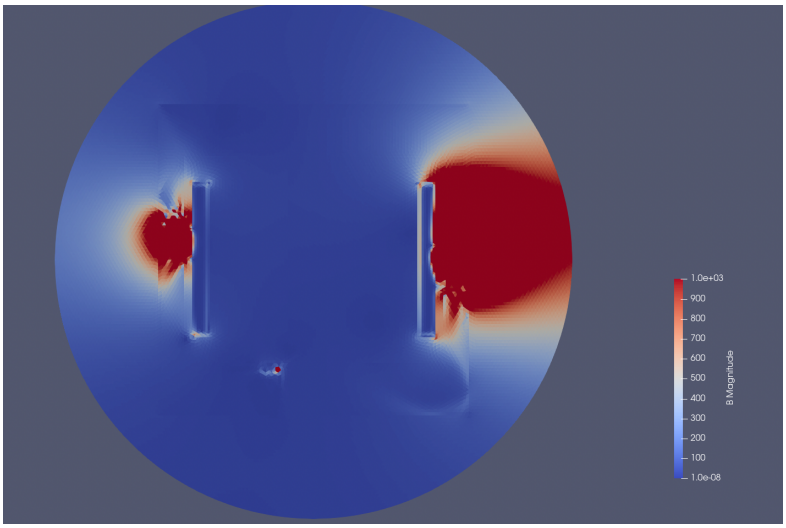
From the table you can see that ROW always takes less timesteps than BDF-2 Newton steps. The non-linearity of the B-H curve doesn't have an influence of the

Set	Method	Step Size	Number of Newton/time Steps	Error
1	ROW	3e-5s	681	0.011
	BDF-2	1e-4s	958	0.038
2	ROW	6e-5	341	0.011
	BDF-2	1e-4	412	0.024
3	ROW	6e-5	341	—
	BDF-2	1e-4	412	—

Table 6.1: This table compares the performance of the two methods on different sets of parameters, on a static mesh. The mesh has 19802 DOFs.



(a) This is a stable solution



(b) Diverged solution, which results from a timestep size which is too large

Figure 6.4: A comparison between a converged solution and a diverged solution. When a solution diverges the timestep size needs to be reduced.

6.3.2 Adaptive timestepping

We didn't manage to get a simulation of the full time-period using adaptive timestepping. The issue was that the relative tolerance needed to be small enough not to create an instability, see Figure 6.4, but with the tolerance too small the error couldn't be brought below the tolerance at a later time, even with small timesteps. For example if we set the relative tolerance to 1.2, meaning that the estimated error may be at most 1.2 times larger than the norm of the previous timestep solution, the simulation runs until time 0.007s, where the error is too big and the method doesn't manage to bring it below the critical threshold.

6.4 Comparison on a Rotating Geometry

6.4.1 Error Norms on a Rotating Geometry

The norm derived from the dissipative power (6.1) is not easily applicable to a rotating mesh, since the energy dissipation per time includes a time derivative of the magnetic field, which is difficult to compute in the air-gap. The magnetic field is defined on the mesh cells, and the air-gap is remeshed, so the correspondence between the cells is lost. There are two norms which we use as an alternative: the magnetic field energy, which is doesn't depend on any time derivative, and the density of electric power dissipated by the electric current, which is only defined on the conducting domain.

The magnetic field energy reads as:

$$E(B) = \mathbf{H} \cdot \mathbf{B}, \tag{6.7}$$

and the dissipative current power reads as:

$$P(j, E) = j\mathbf{E}. \tag{6.8}$$

6.4.2 Setup and Results

To assess how the method performs on a rotating geometry, we constructed a test geometry which includes all the defining features of an electrical motor, which are: a stator, a rotor, and an airgap which separates the two. Conversely to a real motor, in our test geometry the rotor is outside of the stator, but this doesn't influence the physical phenomena we are interested in simulating, and preserves the numerical challenges of the rotating motion. See Figure 4.1 for a diagram of the test geometry. We again applied a sinusoidal current of 50Hz to the coil and simulated for the duration of one period, and rotated the rotor at a constant speed of 10 rotations per second. The steel rotor has the same nonlinear behavior as the transformer core.

The sets of parameters used to test the performance for the rotating ring are:

- Current density j: $2 \cdot 10^7 A/m^2$, nonlinearity parameters: max = 1000, $c = 30$ (Set 1).
- Current density j: $1 \cdot 10^7 A/m^2$, nonlinearity parameters: max = 500, $c = 30$ (Set 2).

Set	Method	Step Size	Number of Newton/time Steps	Error — E(B)	Error — P(j, E)
1	ROW	3e-5s	667	0.0061	0.023
	BDF-2	1e-4s	1678	0.017	0.033
2	ROW	6e-5	333	0.03178	0.02299
	BDF-2	1e-4s	1257	0.01680	0.03248

Table 6.2: This table compares the performance of the two methods on different sets of parameters, on a static mesh. The mesh has 3321 DOFs.

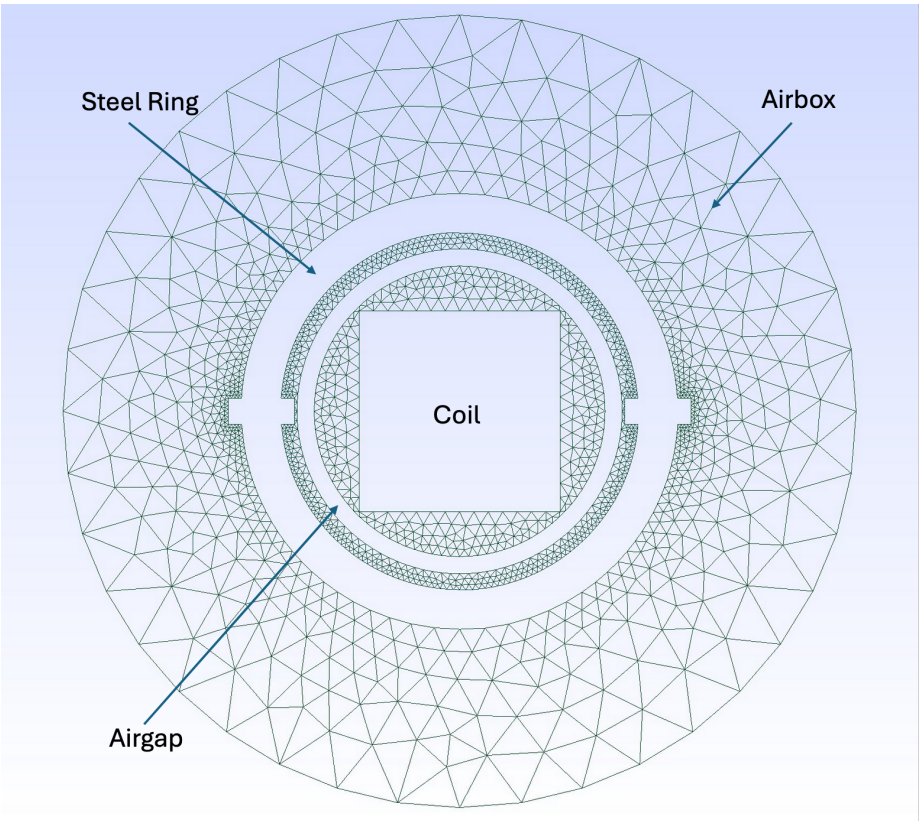


Figure 6.5: Diagram for the rotating ring. A current is applied to the coil in the center, which has a quadratic form in order to break the circular symmetry. This means that when in the irregularity present on the ring will move through this field the magnetic field changes and a current is induced.

Chapter 7

Discussion

7.1 Numerical Performance Comparison Between Iteration-Based Methods and ROW with Fixed Timestep

In Chapter 5 we provided a detailed comparison between the performance of iterative methods and ROW, based on the number of timesteps and Newton iterations. Now that we have applied both methods to a variety of experimental setups, and have recorded the number of timesteps/Newton-iterations, we can deduce which method is more efficient, in terms of basic operations.

We recall the total time per timestep for BDF-2 and ROW respectively:

$$\begin{aligned} t_{BDF-2} = N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M \\ + N_{Newton} \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + \\ P \cdot \Delta t_P + S \cdot \Delta t_S), \end{aligned} \tag{7.1}$$

$$\begin{aligned} t_{ROW} = & N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M + P \cdot \Delta t_P \\ & + 4 \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + S \cdot \Delta t_S), \end{aligned} \tag{7.2}$$

with the operations are defined as following:

q_* = "cost of quadrature for element of matrix *, where * can be A, M, N"
P = "cost of computing the preconditioner of the system matrix"
S = "cost of solving the LSE when the preconditioner has already been decomposed"

And the number of elements per domain is defined as following:

- N: total number of elements in the domain
- N_L: number of elements in linear material domain.

- N_{nnL}: number of elements in nonlinear domain.
- N_R: in rigid domain.
- N_D : in deformable domain.

Thus the total time is:

$$\begin{aligned} t_{BDF-2} =& N_{timesteps,BDF-2} \cdot (N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M) \\ &+ N_{Newton} \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + \\ &\quad P \cdot \Delta t_P + S \cdot \Delta t_S), \end{aligned} \tag{7.3}$$

$$\begin{aligned} t_{ROW} =& N_{timesteps,ROW} \cdot (N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M + P \cdot \Delta t_P) \\ &+ 4 \cdot (N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A} + S \cdot \Delta t_S). \end{aligned} \tag{7.4}$$

As a simplification, we assume that assembly of the matrices in the linear and nonlinear domain takes the same time:

Assumption 3. $A := N_D \cdot q_N \cdot \Delta t_{q_N} + N_D \cdot q_D \cdot \Delta t_{q_D} + N_D \cdot q_M + P \cdot \Delta t_P = N_{nnL} \cdot q_N \cdot \Delta t_{q_N} + N_{nnL} \cdot q_A \cdot \Delta t_{q_A}$.

The difference in time between the two methods then becomes:

$$\begin{aligned} \Delta t := t_{BDF-2} - t_{ROW} =& N_{timesteps,BDF-2} \cdot A + N_{Newton} \cdot (A + P \cdot \Delta t_P + S \cdot \Delta t_S) \\ &- N_{timesteps,ROW} \cdot ((A + P \cdot \Delta t_P) + 4 \cdot (A + S \cdot \Delta t_S)). \end{aligned} \tag{7.5}$$

As a further simplification, we neglect the matrix assemblies which are computed in every timestep of BDF-2, since the number of timesteps is much smaller than the number of Newton iterations:

$$\begin{aligned} \Delta t = t_{BDF-2} - t_{ROW} =& N_{Newton} \cdot (A + P \cdot \Delta t_P + S \cdot \Delta t_S) \\ &- N_{timesteps,ROW} \cdot ((A + P \cdot \Delta t_P) + 4 \cdot (A + S \cdot \Delta t_S)). \end{aligned} \tag{7.6}$$

Some algebraic manipulations provide the following expression:

$$\begin{aligned} \Delta t =& A \cdot (N_{Newton} - 5 \cdot N_{timesteps,ROW}) + P \cdot \Delta t_P \cdot (N_{Newton} - N_{timesteps,ROW}) + \\ & S \cdot \Delta t_S \cdot (N_{Newton} - 4 \cdot N_{timesteps,ROW}). \end{aligned} \tag{7.7}$$

We simplify the analysis to three cases, and from Equation (7.7) we deduce a condition on the number of total Newton iterations and ROW timesteps for which ROW is more efficient than BDF-2.

1. The cost of computing the preconditioner (P) dominates over the other two ($P \gg S$, $P \gg A$). ROW is more efficient than BDF-2 if $N_{Newton} > N_{timesteps,ROW}$.

2. The cost of solving the system (S) dominates. ROW is more efficient if $N_{Newton} > 4 \cdot N_{timesteps,ROW}$.

3. The cost of assembling the matrix (A) dominates. ROW is more efficient if $N_{Newton} > 5 \cdot N_{timesteps,ROW}$

We look at the experimental results from Tables 6.1 and 6.2, and see that in all parameter sets ROW is more efficient than BDF-2 only in Case 1, when the cost of computing the preconditioner dominates over the other operations. For example with the static geometry (transformer), and Parameter Set 1, BDF-2 takes 958 Newton iterations, while ROW takes 681 timesteps. This is enough for ROW to be more efficient in Case 1, but falls short of the 4 or 5 times more Newton iterations BDF-2 would need for ROW to be more efficient, in Cases 2 and 3 respectively.

Ultimately, a case in between the ones analyzed could occur, where one operation doesn't dominate over the others, and you need to apply formulas (7.7), or formulas (7.3) and (7.4) for an accurate comparison, with the measured times for the various operations.

7.2 Acceptable Relative Error

Together with the number of timesteps and Newton iterations, we always computed the error with respect to a given norm, when compared to the baseline. Our philosophy is that as long as the error stays below an acceptability threshold, lower errors are not an advantage. We choose an error threshold of 5% (0.05 in Tables 6.1 and 6.2). For example if ROW has a lower error because we used smaller timesteps due to stability constraints, we don't consider it an advantage.

7.3 Performance in a Rotating vs Static Geometry

In Table 6.2 you can see that ROW performs much better relatively to BDF-2 in the rotating geometry than in the static geometry (6.1. This might be due to the higher order of ROW, or due to the fact that the geometry of the rotating setup is simpler.

7.4 Issues with Adaptive Timestepping

The issue with adaptive timestepping means that the method, as it was developed in this project, isn't suitable for industrial application. Adaptive timestepping is essential, because a method needs to automatically recover from instability, and it can't produce diverged solutions when used in a practical setting. This issue probably can be solved, and may be the subject of future work.

Another issue of adaptive timestepping is the setting of tolerances. During the testing of adaptive timestepping, we tried to find an absolute tolerance which doesn't lead to instabilities, but also doesn't make the timestep too small. This was only possible experimentally, which is not optimal in an industrial setting.

Chapter 8

Conclusion

In this work we compared the performance of ROW methods to BDF-2 on static electronic devices and rotating electric machines. We performed numerical experiments to compare the amount of elementary operations needded for both methods, which we used to conclude which method is more efficient in three simplified cases. We provided formulas to predict the total time needed for both methods, based on the time needed for the elementary operations. While ROW worked using a fixed timestep, it failed with an adaptive timestep, and further work is needed to find a solution.

From this investigation, we conclude that using ROW on rotating machines is feasible, since the rotation of mesh doesn't pose any conflict with the formulation of the ROW method, but it poses some challenges for use in an industrial setting. On the one hand, the failure of adaptive timestepping means that the user needs to know in advance what timestep size will avoid instability, on the other hand, even if the adaptive timestepping had worked, there would still be the challenge of finding a suitable tolerance for the estimated error. Under this perspective, BDF-2 seems a much simpler method, since when selecting a small tolerance it never diverges. Thus the two methods can be compared as following: ROW is a better performing method, but it still needs some work and is more difficult to use; BDF-2 is less performing, but is very reliable and can be used without any domain knowledge.

The implementation of ROW on a rotating geometry posed some challenges, and this manuscript together with the provided code repository can be used to improve the adaptive timestepping, reproduce the results, or implement ROW for Eddy-current simulations in another framework.

Appendix A

Implementation

In this chapter implementation details which posed difficulties will be explained. The code repository can be found at `https://github.com/luca-commits/ROW_engines`

A.1 Mesh Operations using GMSH

In this chapter we explain the techniques we used to ensure mesh conformity at the airgap-stator and airgap-rotor interfaces. Indeed, GMSH doesn't provide tools to ensure conformity of meshes that are merged together, which means that this has to be done manually. Another issue is the numbering of vertices when merging the different meshes together. It is important that DOFs maintain the same index between timesteps, in order to compute (numerical) time derivatives. Since LehrFEM++ uses mesh indices to assign DOF indices, we need to ensure that nodes in the rotor and stator have the same index between timesteps, while the number of vertices in the airgap may change.

A.1.1 Rotating the Mesh

To understand how we ensured correspondence of nodes at the rotor-airgap boundary, we need to explain how geometry definition and mesh generation works in GMSH. First the geometry of the object needs to be defined. We focus on the rotor-airgap boundary, which is the only mesh interface which is moved. In order to define a circle, we need to define four arcs, for which we need four points. After the geometry is defined, the arcs are meshed in 1D, meaning that nodes and edges are placed on the curves, which which are subsequently used to create a 2D mesh. The nodes on the rotor boundary are generated pre-compilation time, and are rotated at every timestep together with all rotor nodes, a shown below:

```
1 void rotateAllNodes_alt(const std::string& input_file, const std::string& output_file, double angle) {
2     gmsh::initialize();
```

```
    gmsh::option::setNumber("General.Terminal", 0);

    // Calculate rotation matrix coefficients once
    double cosA = std::cos(angle);
    double sinA = std::sin(angle);

    // Open the mesh file
    gmsh::merge(input_file);

    // Get all nodes
    std::vector<std::size_t> nodeTags;
    std::vector<double> coords;
    std::vector<double> parametricCoords;

    gmsh::model::mesh::getNodes(nodeTags, coords, parametricCoords);

    for (std::size_t i = 0; i < nodeTags.size(); ++i) {

        double x = coords[i * 3];
        double y = coords[i * 3 + 1];
        double z = coords[i * 3 + 2];

        double new_x = x * cosA - y * sinA;
        double new_y = x * sinA + y * cosA;

        gmsh::model::mesh::setNode(
            nodeTags[i],
            {new_x, new_y, z} ,
            parametricCoords
        );
    }

    gmsh::model::mesh::rebuildNodeCache(false);
    gmsh::write(output_file);

    gmsh::finalize();
} .
```

The nodes on the airgap boundary, on the other hand, are generated at every timestep. The points that are used to define the arcs which compose the outer airgap boundary are rotated by the same angle as which the rotor is rotated. Since the nodes are placed uniformly between the arc extremities, this guarantees that the nodes on the airgap have the same position as the rotor. The following code shows how the boundary-defining points are rotated and the airgap is remeshed:

```
void remeshAirgap(std::string airgap_geo_file,  const std::string& output_file, double angle) {
    gmsh::initialize();
    gmsh::option::setNumber("General.Terminal", 0);

```

```
    gmsh::open(airgap_geo_file);


    gmsh::model::geo::rotate({{0, 139}, {0, 140}, {0, 141}, {0, 142}},
                             0,
                             0,
                             0,
                             0,
                             0,
                             1,
                             angle);


    gmsh::model::mesh::generate(2);

    std::vector<std::size_t> nodeTags;
    std::vector<double> nodeCoords, nodeParams;
    gmsh::model::mesh::getNodes(nodeTags, nodeCoords, nodeParams);

    // Number of nodes is the size of nodeTags
    unsigned airgap_nodes = nodeTags.size();

    std::cout << "airgap nodes : " << airgap_nodes << std::endl;

    // Save and show
    gmsh::write(output_file);

    gmsh::finalize();
} ,
```

where the points 139, 140, 141 and 142 define the outer airgap boundary.

A.1.2 Ensuring DOF Correspondence between Timesteps

In order to generate the whole motor mesh, the meshes of the rotor, stator and airgap are merged together. This is a simple operation in GMSH. A nuance of this operation is that the index of nodes in the merged mesh is assigned in order of inclusion. This means that the meshes that have a constant number of nodes need to be merged first, and the mesh of the airgap needs to be merged last, as following:

```
unsigned mergeEverything(const std::string& input_file_stator,
                         const std::string& input_file_rotor,
                         const std::string& input_file_airgap,
                         const std::string& output_file) {
    gmsh::initialize();
    gmsh::option::setNumber("General.Terminal", 0);
    unsigned numStableDof;

```

```
    try {

        gmsh::merge(input_file_rotor);
        gmsh::model::geo::synchronize();
        gmsh::model::mesh::removeDuplicateNodes();
        gmsh::model::mesh::removeDuplicateElements();

        gmsh::merge(input_file_stator);
        gmsh::model::geo::synchronize();
        gmsh::model::mesh::removeDuplicateNodes();
        gmsh::model::mesh::removeDuplicateElements();

        std::vector<std::size_t> nodeTags;
        std::vector<double> nodeCoords, nodeParams;
        gmsh::model::mesh::getNodes(nodeTags, nodeCoords, nodeParams);

        numStableDof = nodeTags.size();

        gmsh::merge(input_file_airgap);
        gmsh::model::geo::synchronize();
        gmsh::model::mesh::removeDuplicateNodes();
        gmsh::model::mesh::removeDuplicateElements();


        gmsh::write(output_file);
        gmsh::finalize();

    } catch (const std::runtime_error& e) {
        gmsh::logger::write("Error during merge operation: " + std::string(e.what()));
        gmsh::finalize();
        throw;
    }

    return numStableDof;
} .
```

Appendix B

Computing a Preconditioner and Solving the Linear System of Equations (LSE)

In all the solver settings, which are stationary, transient, static geometry, rotating geometry, using BDF-2 or ROW, a linear system of equations has to be solved, where the left hand side always consist of a linear combination of the stiffness matrix ($\underline{\underline{A}}$), mass matrix ($\underline{\underline{M}}$) and Jacobian matrix. A closer inspection of the definition of the mass and stiffness matrix reveals that the stiffness matrix has some 0 diagonal elements, and the mass matrix has 0 diagonal entries for DOFs corresponding to the non-conducting domain. A linear combination of these matrices results in a matrix which has entries 0 on the diagonal, which leads to bad conditioning. For example when we set the conductivity to 0 on the whole domain, the solver failed.

In order to deal with bad conditioning, we constructed a preconditioner, where we set the conductivity to a small number everywhere the conductivity was 0, and used it to construct a mass matrix, which we then added to the left hand side matrix.

We used the bi conjugate gradient method together with this preconditioner to solve the LSEs which arose in the various solvers.

Bibliography

Daryl L Logan. *A first course in the finite element method*, volume 4. Thomson, 2011.

Endre Süli and David F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, August 2003. ISBN 978-0-521-00794-8. Google-Books-ID: hj9weaqJTbQC.

Jens Lang. Rosenbrock-Wanner Methods: Construction and Mission. In Tim Jax, Andreas Bartel, Matthias Ehrhardt, Michael Günther, and Gerd Steinebach, editors, *Rosenbrock—Wanner–Type Methods: Theory and Applications*, pages 1–17. Springer International Publishing, Cham, 2021. ISBN 978-3-030-76810-2.

H. H. Rosenbrock. Some general implicit processes for the numerical solution of differential equations. *The Computer Journal*, 5(4):329–330, January 1963. ISSN 0010-4620. doi: 10.1093/comjnl/5.4.329. URL `https://doi.org/10.1093/comjnl/5.4.329`.

Ernst Hairer and Gerhard Wanner. Solving ordinary differential equations I. nonstiff problems. *Mathematics and Computers in Simulation*, 29(5):447, October 1987. ISSN 03784754. doi: 10.1016/0378-4754(87)90083-8. URL `https://linkinghub.elsevier.com/retrieve/pii/0378475487900838`.

J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67, January 1947. ISSN 1469-8064, 0305-0041. doi: 10.1017/S0305004100023197.

Germund G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, March 1963. ISSN 1572-9125. doi: 10.1007/BF01963532. URL `https://doi.org/10.1007/BF01963532`.

Byron L Ehle. *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*. phd, University of Waterloo Waterloo, Ontario, 1969.

Ralf Hiptmair. Numerical Methods for CSE. 2023a.

Bernhard Kähne and Markus Clemens. A GPU-Accelerated Semi-Implicit Method for Large-Scale Nonlinear Eddy-Current Problems Using Adaptive Time Step Control. *IEEE Transactions on Magnetics*, 61(1):1–4, January 2025. ISSN 1941-0069. doi: 10.1109/TMAG.2024.3502313. URL https://ieeexplore.ieee.org/document/10758287/.

Markus Clemens, Jens Lang, Delia Telaga, and Wimmer. Adaptivity in Space and Time for Magnetoquasistatics 1). May 2009.

Raffael Casagrande. Sliding Interfaces for Eddy Current Simulations. Master's thesis, ETH Zurich, 2013.

Ralf Hiptmair. Numerical Methods for Partial Differential Equations. 2023b.

Florian Bachinger, Ulrich Langer, and Joachim Schöberl. Efficient solvers for nonlinear time-periodic eddy current problems. *Computing and Visualization in Science*, 9(4):197–207, December 2006. ISSN 1433-0369. doi: 10.1007/s00791-006-0023-z. URL https://doi.org/10.1007/s00791-006-0023-z.

Gil Strang. Multidimensional-Newton, September 2017. URL https://web.mit.edu/18.06/www/Spring17/Multidimensional-Newton.pdf.

Johan Gyselinck, Patrick Dular, Nelson Sadowski, J.V. Leite, and J.P.A. Bastos. Incorporation of a Jiles-Atherton vector hysteresis model in 2D FE magnetic field computations: Application of the Newton-Raphson method. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, 23:685–693, September 2004. doi: 10.1108/03321640410540601.

J. Rang and L. Angermann. New Rosenbrock W-Methods of Order 3 for Partial Differential Algebraic Equations of Index 1. *BIT Numerical Mathematics*, 45(4):761–787, December 2005. ISSN 1572-9125. doi: 10.1007/s10543-005-0035-y. URL https://doi.org/10.1007/s10543-005-0035-y.

Dominik L. Michels, Vu Thai Luan, and Mayya Tokman. A stiffly accurate integrator for elastodynamic problems. *ACM Trans. Graph.*, 36(4):116:1–116:14, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073706. URL https://dl.acm.org/doi/10.1145/3072959.3073706.

John Henry Poynting and John William Strutt. On the transfer of energy in the electromagnetic field. *Proceedings of the Royal Society of London*, 36(228-231):186–187, January 1997. doi: 10.1098/rspl.1883.0096. URL https://royalsocietypublishing.org/doi/abs/10.1098/rspl.1883.0096. Publisher: Royal Society.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every written paper or thesis authored during the course of studies. **In consultation with the supervisor**, one of the following two options must be selected:

☒ I hereby declare that I authored the work in question independently, i.e. that no one helped me to author it. Suggestions from the supervisor regarding language and content are excepted. I used no generative artificial intelligence technologies[1].

☐ I hereby declare that I authored the work in question independently. In doing so I only used the authorised aids, which included suggestions from the supervisor regarding language and content and generative artificial intelligence technologies. The use of the latter and the respective source declarations proceeded in consultation with the supervisor.

Title of paper or thesis:

Adaptive timestepping for the simulation of electrical machines

Authored by:
If the work was compiled in a group, the names of all authors are required.

Last name(s):	**First name(s):**
Wolfart	Luca

With my signature I confirm the following:
- I have adhered to the rules set out in the Citation Guidelines.
- I have documented all methods, data and processes truthfully and fully.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for originality.

Place, date	**Signature(s)**
Zürich, 26/04/2025	
	If the work was compiled in a group, the names of all authors are required. Through their signatures they vouch jointly for the entire content of the written work.

[1] For further information please consult the ETH Zurich websites, e.g. https://ethz.ch/en/the-eth-zurich/education/ai-in-education.html and https://library.ethz.ch/en/researching-and-publishing/scientific-writing-at-eth-zurich.html (subject to change).