Coupling of Electromagnetic Fields with Electric Circuits Using Onelab

Kopplung von Elektromagnetischen Feldern mit Elektrischen Schaltungen unter Benutzung von Onelab
Bachelor-Thesis von Alexander Krimm, Studienbereich Computational Engineering
Tag der Einreichung:

1. Gutachten: Prof. Dr. Sebastian Schöps
2. Gutachten: Prof. Christophe Geuzaine

TECHNISCHE UNIVERSITÄT DARMSTADT

Coupling of Electromagnetic Fields with Electric Circuits Using Onelab
Kopplung von Elektromagnetischen Feldern mit Elektrischen Schaltungen unter Benutzung von Onelab

Vorgelegte Bachelor-Thesis von Alexander Krimm, Studienbereich Computational Engineering

1. Gutachten: Prof. Dr. Sebastian Schöps
2. Gutachten: Prof. Christophe Geuzaine

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 12. Juni 2015

(Alexander Krimm)

Abstract

This work deals with the coupling of electromagnetic fields with electric circuits, a problem which is often encountered in engineering applications as the design of electrical machines. First, common computation methods for the distinctive subproblems are examined. For the circuit simulation part the modified nodal analysis is explained. The field problem is treated with the finite element method applied to the magnetoquasistatic field approximation.

Based on these two techniques a coupling scheme to couple magnetostatic field problems to circuit simulations is proposed, after a brief comparison of different general coupling methods. This coupling method is then implemented using the ONELAB framework to exchange the coupling quantities. The circuit part of the coupled simulation is implemented within Octave's OCS package and the standalone circuit simulation software Qucs. The field simulation is performed using the finite element software GetDP, for which a template calculating all the necessary coupling quantities is developed.

Finally the results of the simulation of an academic RL series circuit are examined focusing on the numerical properties of the different iterative methods used in the coupling method. The method converges reliably if the correct Jacobian can be obtained from the field simulation software. Additionally the implementational overhead, which mostly stems from GetDP being started and reading and writing solution files, is quantified to evaluate possible improvements. This can eventually lead to more efficient coupled simulations.

Contents

1 Motivation

For most engineering problems, there is a point at which its behaviour cannot be explicitly calculated because the necessary simplifications introduce modeling errors that are too big to meet the accuracy requirements demanded by the application.

On the other hand, building prototypes is often very expensive and in some cases inconvenient or even impossible.

The methods of computational engineering allow to create a very close representation of the original problem as a computer model with an accuracy only limited by processing power, numerical accuracy and modelling errors. Simulation has many advantages over physical prototypes. They do not have to be run in real-time, can be repeated and run in parallel. Additionally, evaluating the results of an experiment is easier for simulated models because there are no measurements involved, e.g. the spacial distribution of magnetic fields is very difficult to measure but can be easily obtained from a field simulation.

In many applications it is necessary to investigate or optimize devices of a given geometry which are coupled to complex electric networks, for example in the design of drive systems as illustrated in Fig. 1.1. Although field and circuit simulation are based on the same physical laws the level of abstraction differs significantly. Circuit simulation is based on a network of lumped components whose behaviour depends on the adjacent nodal potential, neglecting geometrical arrangement of the components and all stray effects (if they are not explicitly modeled e.g. as parasitical capacitances). Field simulation software can easily account for these effects as it is directly based on the Maxwell equations. This lower level of abstraction comes with a price as it drastically increases the size of the equation system that has to be solved, compared to an equivalent circuit simulation. While for both problems there are very efficient and mature codes, most available programs do not offer coupling of both. The ONELAB project [Geu14] aims to create an abstract framework for coupling many solvers in a very flexible way and therefore field-circuit coupling is here established on top of it.

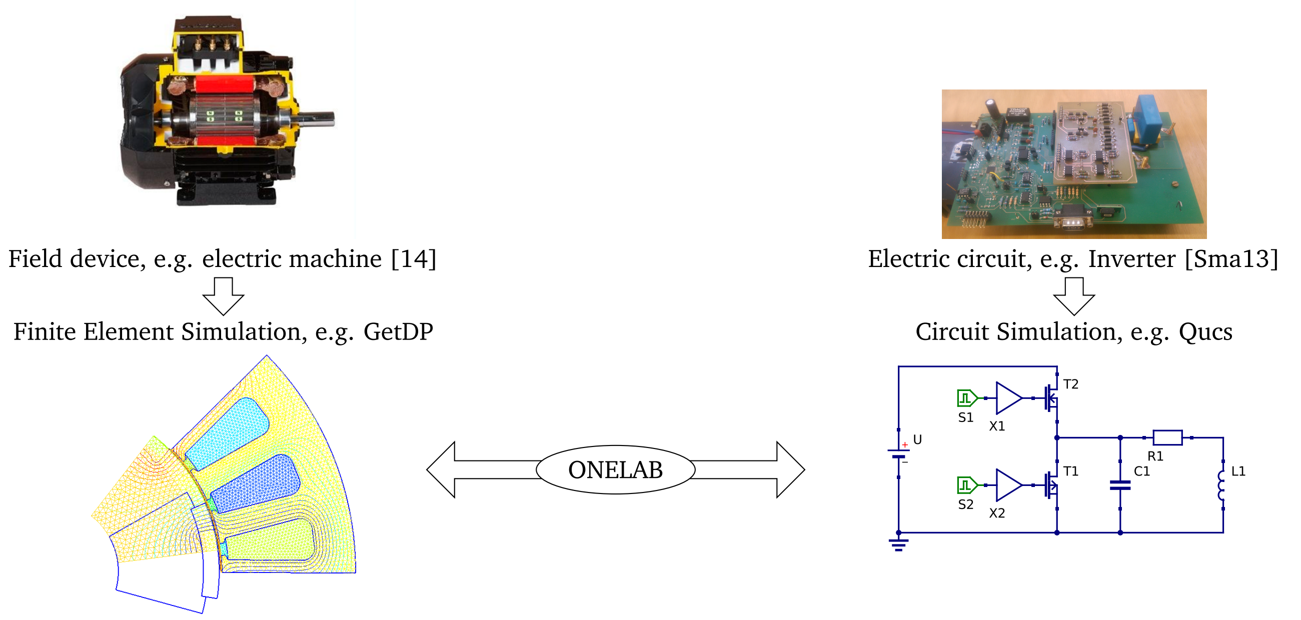

Field device, e.g. electric machine [14]
Finite Element Simulation, e.g. GetDP
Electric circuit, e.g. Inverter [Sma13]
Circuit Simulation, e.g. Qucs
ONELAB
T2
S1
X1
U
R1
C1
L1
T1
S2
X2

Figure 1.1: Example for Coupling the Field Simulation of an Electric Machine with the Circuit Simulation of an Inverter

The aim of this work is to develop a coupling framework that uses ONELAB to couple the finite-element solver GetDP with two different circuit simulation tools and evaluate the possibilities of this setup. The first setup uses the Octave Circuit Simulator, an octave package that provides functions to assemble the circuit matrix equations and solve them in the DC or time domain. Based on the experiences from this simple setup the second more sophisticated circuit simulation tool is Qucs, the Quite Universal Circuit Simulator, which is written in C++ and provides a graphical user interface and a library of components.

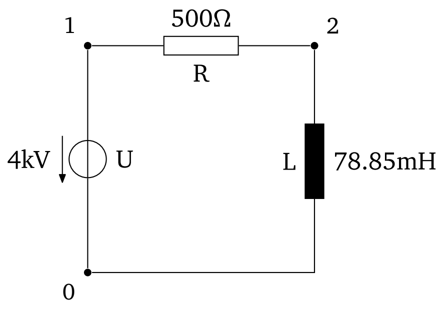
In the first two chapters a brief introduction to circuit and field simulation is given, the third chapter compares several different coupling methods and develops a method suitable for the aim of the work. After this theoretical part, in chapter five the software packages are introduced and the actual implementation is explained and evaluated.

2 Circuit Simulation

Circuit simulators are used to simulate networks of lumped components, whose behavior is described by linear and nonlinear differential equations depending on nodal potentials and branch currents of the network. This allows to express even very complex networks as systems with relatively few degrees of freedom, as the geometric properties are neglected and only the topology of the connection between the components is considered. This chapter will explain the basic concepts that circuit simulators use while focusing on the parts needed for coupling them with field simulation software.

2.1 Electric networks

Electric circuits are described by networks of discrete components that are connected to an electric network. Components can have an arbitrary number of ports. Ports from different components are connected in nodes and share a common electric potential. The behavior of the components depends linearly or nonlinearly on the potentials at the nodes they are connected to and the currents into their ports. Figure 2.1a shows a simple network with three nodes connected by a voltage source, an inductor and a resistor, forming a series RL circuit.



```
IProbe:I _net0 _net1
L:onelab gnd _net2 L="78.85 mH" I="0"
Vdc:V1 _net0 gnd U="4 kV"
VProbe:U_L _net2 gnd
```

(a) Schematic **(b)** Netlist

Figure 2.1: Series RL Circuit

The graphic representation of the network has to be translated to a textual description of the electric network called netlist. There are many different formats for netlists, but their fundamental structure is always a list of components with their properties and the nodes they are connected to. Figure 2.1b shows the netlist generated by qucs for the RL circuit in Fig. 2.1a.

2.2 Kirchhoff's Laws

Kirchhoff's laws describe the behaviour of networks of discrete electrical components. They can be deduced from the Maxwell equations very easily, as shown in [FM11]. Kirchhoff's first law is often called the current law and states that the currents flowing in and out of a node in the electrical network always add up to zero

$$\sum_{k=1}^{n} I_k = 0. \tag{2.1}$$

For a network with k nodes this law can be applied to $k-1$ nodes to obtain $k-1$ linearly independent equations.

Kirchhoff's second law, also known as voltage law states that the sum of all voltages along every loop in the network has to be zero

$$\sum_{k=1}^{n} U_k = 0. \tag{2.2}$$

Applying (2.1) and (2.2) along with Ohm's law or other branch constitutive equations, the most common ones being the ones for resistors, capacitors and inductances,

$$u = Ri, \tag{2.3}$$

$$i = C\frac{du}{dt}, \tag{2.4}$$

$$u = L\frac{di}{dt}, \tag{2.5}$$

to the branches of a spanning tree, it is possible to obtain the equations needed for a solvable equation system. There are many other schemes similar to the one described here, differing in the choice of unknowns for the resulting equation system and the simplifications applied to the network beforehand.

2.3 Modified Nodal Analysis

While the approach in the previous chapter is well suited to compute small networks by hand, implementation on computer systems calls for a more systematic representation, which exists in the form of the Modified Nodal Analysis (MNA) [CRB75]. Another advantage of this method is, that its unknowns automatically contain currents through voltage sources and inductors, so that there is no need for back substitution. The modified nodal analysis describes the construction of matrices that allows to write a network of lumped electric components in the form

$$\underbrace{\begin{bmatrix} A_C C A_C^\mathsf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & L \end{bmatrix}}_{:=M_c} \frac{d}{dt} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{i}_V \\ \boldsymbol{i}_L \end{bmatrix} + \underbrace{\begin{bmatrix} A_G G A_G^\mathsf{T} & A_V & A_L \\ A_V^\mathsf{T} & \mathbf{0} & \mathbf{0} \\ A_L^\mathsf{T} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{:=K_c} \underbrace{\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{i}_V \\ \boldsymbol{i}_L \end{bmatrix}}_{:=\boldsymbol{x}} = \underbrace{\begin{bmatrix} -\boldsymbol{i}_s(t) \\ -\boldsymbol{v}_s(t) \\ \mathbf{0} \end{bmatrix}}_{:=r(t)} \tag{2.6}$$

in which the vector $\boldsymbol{x}^\mathsf{T} := (\boldsymbol{u}^\mathsf{T}, \boldsymbol{i}_V^\mathsf{T}, \boldsymbol{i}_L^\mathsf{T})$ contains the unknown nodal voltages and the currents through the independent voltage sources and inductors. The vector $\boldsymbol{r}^\mathsf{T}(t) := (-\boldsymbol{i}_s^\mathsf{T}(t), -\boldsymbol{v}_s^\mathsf{T}(t), \mathbf{0})$ contains the independent current and voltage sources.

The matrices A_G, A_C, A_V, A_L are incidence matrices, containing only $1, -1$ and 0. They describe the topology of the electrical network. The diagonal matrices $\boldsymbol{G}, \boldsymbol{C}, \boldsymbol{L}$ contain the values of the lumped components (2.3) to (2.5). For simplicity only linear branch constitutive equations are considered here. The matrices are assembled by iterating over the components of the circuit and filling in the respective parts of the system matrices. The entries for nonlinear components have to be updated in every timestep, while the matrix entries for linear components are constant during the whole simulation.

The mass matrix $\boldsymbol{M}_c$ of (2.6) is singular, therefore the resulting system is a "differential algebraic equation". It has to be solved by an appropriate integration method and only implicit integration methods ensure reliable results. In the following the implicit Euler scheme is used, which is obtained by introducing a time discretization

$$t_0 = 0, \qquad t_{i+1} = t_i + h_i \tag{2.7}$$

with timesteps h_i. It is adapted in every time step by estimating the error of the current step and adapting it accordingly. Using the forward difference quotient for the derivative $\dot{x}$ in

$$a\dot{x} + bx = c \tag{2.8}$$

and initializing the unknowns with the starting value

$$x_0 = x(t_0) \tag{2.9}$$

the values of x_i can now be computed using the iterative scheme

$$\left(\frac{1}{h}a + b\right)x_{i+1} = c + \frac{1}{h}ax_i \qquad \text{with } x_i = x(t_i = t_{i-1} + h). \tag{2.10}$$

If the timestep h_i is chosen too big, the resulting possibly nonlinear equation system can become unsolvable. The timestep is then rejected and has to be reiterated with a smaller value for $\frac{1}{h}$.

Applying the implicit Euler to the differential algebraic equation system obtained by the MNA results in the (non)linear system

$$\left(\frac{1}{h}\boldsymbol{M}_\mathrm{c}+\boldsymbol{K}_\mathrm{c}\right)\boldsymbol{x}_{i+1}=\frac{1}{h}\boldsymbol{M}_\mathrm{c}\boldsymbol{x}_i+\boldsymbol{r}(t_{i+1}),\qquad \boldsymbol{x}_0=\boldsymbol{x}(t_0). \tag{2.11}$$

Other methods that have been proven to perform well in circuit simulation are the trapezoidal method and linear multi-step methods due to Adams-Bashford or backward differentiation formulas. They produce similar results and the coupling methodology described in the following chapters can be used independently of the choice of the integration method. For reasons of simplicity, the implicit Euler scheme is used to illustrate the effect of the time discretization in this work, even if the simulation software may use different methods.

Some circuit simulators (e.g. Qucs) apply the integration scheme on the component level, which allows to represent every component as an equvalent voltage or current sources and equvalent resistances which are then assembled into a single nonlinerar equation system. Eventually this approach leads to the same matrix equation system as described above.

2.4 Treatment of Nonlinear Components

If the system contains nonlinear components, the integration scheme produces a nonlinear equation system for every timestep i

$$\boldsymbol{F}_\mathrm{c}(\boldsymbol{x}_i)=\left[\frac{1}{h}\boldsymbol{M}_\mathrm{c}(\boldsymbol{x}_i)+\boldsymbol{K}_\mathrm{c}(\boldsymbol{x}_i)\right]\boldsymbol{x}_i-\boldsymbol{r}-\frac{1}{h}\boldsymbol{M}_\mathrm{c}\boldsymbol{x}_{i-1}=0 \tag{2.12}$$

that can be solved for example by the Newton-Raphson scheme [BSMM12]. At the beginning of each timestep the unknowns are initialized with the solution of the previous timestep, assuming that the resulting waveforms are continous

$$\boldsymbol{x}_{i,j=0}=\boldsymbol{x}_{i-1}. \tag{2.13}$$

$$\tag{2.14}$$

Starting from this value, the Newton scheme

$$\boldsymbol{J}_{\boldsymbol{F}_\mathrm{c}}(\boldsymbol{x}_{i,j})\Delta\boldsymbol{x}_{i,j}=-\boldsymbol{F}_\mathrm{c}(\boldsymbol{x}_{i,j}) \tag{2.15}$$

$$\boldsymbol{x}_{i,j+1}=\boldsymbol{x}_{i,j}+\Delta\boldsymbol{x}_{i,j} \tag{2.16}$$

improves the solution, until convergence is reached, meaning that either the newton step $\Delta\boldsymbol{x}_{i,j}$ or the residuum $\boldsymbol{F}_\mathrm{c}(\boldsymbol{x}_{i,j})$ become smaller than a given tolerance. The scheme requires the Jacobian

$$\boldsymbol{J}_{\boldsymbol{F}_\mathrm{c}}=\frac{\partial\boldsymbol{F}_\mathrm{c}}{\partial\boldsymbol{x}} \tag{2.17}$$

of the system matrix for local quadratic convergence results.

3 Field Simulation

Field simulation software uses specialized solvers to discretize and solve Maxwell's equations or derived formulations on discretisized fields. They can very accurately calculate the effects of changes to the problem's geometry and materials. A commonly used approach is the Finite Element Method [Sal95].

3.1 Maxwell Formulation

The behaviour of electromagnetic fields in a domain Ω of non moving materials is described by Maxwell's Equations and the accompanying material laws

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}, \tag{3.1}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J}_{\text{cond}} + \vec{J}_{\text{src}}, \tag{3.2}$$

$$\nabla \cdot \vec{D} = \rho, \tag{3.3}$$

$$\nabla \cdot \vec{B} = 0, \tag{3.4}$$

$$\vec{D} = \varepsilon \vec{E}, \qquad \vec{J}_{\text{cond}} = \sigma \vec{E}, \qquad \vec{B} = \mu \vec{H} or \vec{H} = \nu \vec{B}. \tag{3.5}$$

The electric field $\vec{E}$, the elctric flux density $\vec{D}$, the magnetic flux density $\vec{B}$, the magnetic field $\vec{H}$ and the electric current density $\vec{J} = \vec{J}_{\text{cond}} + \vec{J}_{\text{src}}$ are time dependent vector fields. The electric conductivity σ, the electric permittivity ε and the magnetic permeability μ and its inverse, the magnetic reluctivity ν, are generally tensor fields, but for isotropic materials they are scalar fields. For nonlinear problems they depend on the respective fields.

For magnetoquasistatic problems, the changes of the electric field responsible for the displacement current $\frac{\partial \vec{D}}{\partial t}$ can be neglected and with the definition of the magnetic vector potential $\vec{A}$ as

$$\vec{B} = \nabla \times \vec{A} \tag{3.6}$$

and the electric potential ϕ as

$$\vec{E} = -\frac{\partial \vec{A}}{\partial t} - \nabla \phi \tag{3.7}$$

the magnetostatic and magnetoquasistatic formulation

$$\nabla \times \nu(\vec{A}) \nabla \times \vec{A} = \vec{J}_{\text{src}} \tag{3.8}$$

and

$$\nabla \times \nu(\vec{A}) \nabla \times \vec{A} + \sigma \frac{\partial \vec{A}}{\partial t} = \vec{J}_{\text{src}} \tag{3.9}$$

can be obtained.

For simplification and without loss of generality for the coupling scheme, (3.9) is simplified to the 2D domain by assuming

$$\vec{A} = \begin{bmatrix} 0 \\ 0 \\ A_z(x,y) \end{bmatrix}, \qquad \vec{J}_{\text{src}} = \begin{bmatrix} 0 \\ 0 \\ J_z(x,y) \end{bmatrix} \tag{3.10}$$

which results in the magnetoquasistatic formulation

$$-\frac{\partial}{\partial x} \nu_y \frac{\partial}{\partial x} A_z - \frac{\partial}{\partial y} \nu_x \frac{\partial}{\partial y} A_z + \sigma \frac{\partial A_z}{\partial t} = J_z. \tag{3.11}$$

The right hand side J_z of (3.11) is called the excitation term. To distribute the discrete currents i_k on the computational region Ω a distribution function

$$\chi(\vec{p}) = \begin{cases} +\frac{1}{A_{\Omega_+}} & \text{for } \vec{p} \in \Omega_+ \\ -\frac{1}{A_{\Omega_+}} & \text{for } \vec{p} \in \Omega_- \\ 0 & \text{else} \end{cases} \tag{3.12}$$

as illustrated in Fig. 3.1 is used, where A_{Ω_+} and A_{Ω_-} are the areas of the respective regions. If there are n_{coils} independent currents, the excitation term consists of a distribution term for each of the currents.

$$J_z = \sum_{k=1}^{n_{\text{coils}}} \chi_k i_k \tag{3.13}$$

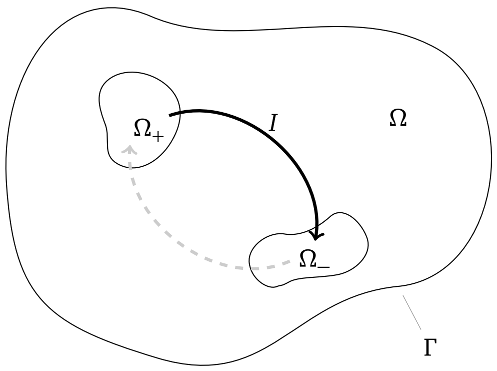


Figure 3.1: Distribution of the Excitation Term for $n_{\text{coils}} = 1$

Without loss of generality, on the boundary $\partial\Omega$ of the domain Ω Dirichlet boundary conditions

$$A_z(\vec{p}) = 0, \quad \forall \vec{p} \in d\Omega \tag{3.14}$$

are assumed.

3.2 Finite Element Method

The finite element method can be used to discretize (3.11) in space [Sal95]. It is based on the weak formulation which is obtained by integrating over the residual of (3.11) multiplied with a weighting function $W(x, y)$. Setting the residual to zero yields

$$-\iint_{\Omega} W(\frac{\partial}{\partial x}\nu_y \frac{\partial}{\partial x}A_z + \frac{\partial}{\partial y}\nu_x \frac{\partial}{\partial y}A_z)\, d\Omega + \iint_{\Omega} \sigma W \frac{\partial A_z}{\partial t}\, d\Omega = \iint_{\Omega} W J_z \, d\Omega, \quad \forall W(x,y). \tag{3.15}$$

Partial integration eliminates the second order derivatives in the first integral

$$\iint_{\Omega} \frac{\partial W}{\partial x}\nu_y \frac{\partial A_z}{\partial x} + \frac{\partial W}{\partial y}\nu_x \frac{\partial A_z}{\partial y}\, d\Omega - \underbrace{\oint_{\Gamma} \nu \frac{\partial A_z}{\partial d\vec{n}}\, d\Gamma}_{=0 \text{ for natural b.c.}} + \sigma \iint_{\Omega} W \frac{\partial A_z}{\partial t}\, d\Omega = \iint_{\Omega} W J_z \, d\Omega, \tag{3.16}$$

resulting in the weak formulation of the magnetoquasistatic problem. In the integral over the boundary Γ of the domain Ω, $\vec{n}$ is the outward normal unit vector on the boundary. Using the Galerkin method on a triangulation of Ω the weighting functions $\boldsymbol{W}$ are chosen to be the same as the linear shape functions $\boldsymbol{N}$ for each triangle with the nodes i, j, k

$$\boldsymbol{W} = \boldsymbol{N} = \begin{bmatrix} \alpha_i + \beta_i x + \gamma_i y \\ \alpha_j + \beta_j x + \gamma_j y \\ \alpha_k + \beta_k x + \gamma_k y \end{bmatrix}, \qquad A_z(x,y) = \boldsymbol{N}(x,y) \cdot \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix}, \quad \forall (x,y)^{\mathsf{T}} \in \Omega^e \tag{3.17}$$

with a_i being the z component of the magnetic vector potential on the node i. The coefficients α, β, γ can be calculated by substituting the coordinates $\boldsymbol{p}_i$ of the nodes into the shape functions. As the potential at a given node only depends on the nodal potential a_i and is independent of the adjacent nodal potentials

$$A_z(\boldsymbol{p}_i) = \boldsymbol{e}_i. \tag{3.18}$$

Doing this for all three nodes of the element, the node coefficients for the element can be calculated. Additionally the shape function is zero on the whole domain except for the triangles adjacent to the node and the sum of all the weighting functions is 1 on the whole domain.

Substituting the shape and weighting functions and their derivatives into (3.16) results in

$$\sum_{e}^{n_{\text{el}}} \iint_{\Omega^e} d\Omega \left(\frac{\partial N^e}{\partial x} \nu_y^e \frac{\partial N^e}{\partial x} + \frac{\partial N^e}{\partial x} \nu_y \frac{\partial N^e}{\partial x} \right) \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix} + \sum_{e}^{n_{\text{el}}} \sigma^e \iint_{\Omega^e} N \cdot N \, d\Omega \frac{\partial}{\partial t} \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix} = \sum_{e}^{n_{\text{el}}} \iint_{\Omega^e} N \, d\Omega \, J^e. \tag{3.19}$$

Assembling the element matrices into global matrices leads to the equation system

$$\boldsymbol{K}_\nu(\boldsymbol{a})\boldsymbol{a} + \boldsymbol{M}_\sigma \dot{\boldsymbol{a}} = \boldsymbol{j}_{\text{s}} = \boldsymbol{X}\boldsymbol{i}_{\text{coil}} \tag{3.20}$$

where $\boldsymbol{X}$ distributes the excitation currents

$$\boldsymbol{i}_{\text{coil}} = [i_1 \dots i_{n_{\text{coils}}}]^{\mathsf{T}} \tag{3.21}$$

on the elements of the domain in the correspondent coils. The dependence of the stiffness matrix $\boldsymbol{K}_\nu$ on the magnetic vector potential stems from the saturation dependent reluctivities $\nu_x(\vec{B})$, $\nu_y(\vec{B})$.

Assuming static fields and thereby neglecting the eddy currents $\boldsymbol{M}_\sigma \dot{\boldsymbol{a}}$ one obtains

$$\boldsymbol{K}_\nu(\boldsymbol{a})\boldsymbol{a} = \boldsymbol{j}_{\text{s}} = \boldsymbol{X}\boldsymbol{i}_{\text{coil}}. \tag{3.22}$$

3.3 Solving the Magnetoquasistatic Formulation for Nonlinear Materials

The nonlinear equation system (3.20) can be time discretized like the circuit system in Section 2.4 using the implicit Euler scheme (2.8)

$$\boldsymbol{F}_{\text{f}}(\boldsymbol{a}) = \left[\frac{1}{h} \boldsymbol{M}_\sigma + \boldsymbol{K}_\nu(\boldsymbol{a}) \right] \boldsymbol{a} - \boldsymbol{X}\boldsymbol{i}_{\text{coil}} - \frac{1}{h} \boldsymbol{K}_\nu \boldsymbol{a}_{i-1} = 0 \tag{3.23}$$

and solved by a Newton-Raphson iteration for the timestep i by initializing the magnetic vector potential with the solution of the previous timestep

$$\boldsymbol{a}_{i,j=0} = \boldsymbol{a}_{i-1} \tag{3.24}$$

and perform the Newton iteration j

$$\boldsymbol{J}_{\boldsymbol{F}_{\text{f}}}(\boldsymbol{a}_{i,j}) \Delta \boldsymbol{a} = -\boldsymbol{F}_{\text{f}}(\boldsymbol{a}_{i,j}) \tag{3.25}$$

$$\boldsymbol{a}_{i,j+1} = \boldsymbol{a}_{i,j} + \Delta \boldsymbol{a} \tag{3.26}$$

until the solution has converged. The Jacobian is given by

$$\begin{aligned} \boldsymbol{J}_{\boldsymbol{F}_{\text{f}}}(\boldsymbol{a}) &:= \frac{\partial \boldsymbol{F}_{\text{f}}(\boldsymbol{a})}{\partial \boldsymbol{a}} \\ &= \frac{\partial \boldsymbol{K}_\nu(\boldsymbol{a}) \cdot \boldsymbol{a}}{\partial \boldsymbol{a}} + \frac{1}{h} \boldsymbol{M}_\sigma. \end{aligned} \tag{3.27}$$

For the rest of this treatise, eddy currents are neglected, so the time derivative of $\boldsymbol{a}$ is disregarded and no time stepping is taking place in the field simulation subproblem. Equations (3.23) to (3.27) still hold, with all the parts containing $\boldsymbol{K}_\nu$ omitted.

3.4 Postprocessing of Field Results

The obtained solution $\boldsymbol{a}^*$ of the magnetostatic problem itself is not necessarily meaningful as a simulation result. In many applications the relevant results of a simulation are not the resulting field distribution but secondary quantities like in the case of a magnetostatic field model containing distinct coils the flux $\boldsymbol{\phi}$, the inductance $\boldsymbol{L}_\text{c}$ or the differential inductance $\boldsymbol{L}_\text{d}$. The process of calculating these quantities from the field results is called postprocessing.

The flux can be directly calculated from the magnetic vector potential

$$\boldsymbol{\phi} = \boldsymbol{X}^\mathsf{T}\boldsymbol{a}, \qquad \boldsymbol{\phi} \in \mathbb{R}^{1\times n_{Coils}} \tag{3.28}$$

As the current is known from the excitation term, the chord inductance can be easily obtained for the trivial case with only one inductance by dividing the flux by the excitation current.

$$\boldsymbol{L}_{\text{c},p,q} = \frac{\boldsymbol{\phi}_p}{i_i} = \frac{\boldsymbol{X}^\mathsf{T}\boldsymbol{K}_\nu^{-1}\boldsymbol{X}\boldsymbol{i}_{\text{coil},p}}{i_q}, \tag{3.29}$$

$$\boldsymbol{L}_\text{c} = \boldsymbol{X}^\mathsf{T}\boldsymbol{K}_\nu^{-1}\boldsymbol{X} \qquad \in \mathbb{R}^{n_\text{coils}\times n_\text{coils}} \tag{3.30}$$

For field problems with multiple independent currents, n_coils linear systems have to be solved

$$\boldsymbol{K}_\nu(\boldsymbol{a}^*)\boldsymbol{a}_q = \boldsymbol{X}_q \tag{3.31}$$

$$\boldsymbol{L}_{\text{c},p,q} = \boldsymbol{X}_p^\mathsf{T}\boldsymbol{a}_q. \tag{3.32}$$

The values on the diagonal of $\boldsymbol{L}_\text{c}$ are the self inductances of the coils while the other values describe the coupling between them. Because of the reciprocal principle the matrix is symmetric.

The differential inductance is defined as

$$\begin{aligned}\boldsymbol{L}_\text{d} = \frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{i}_\text{coil}} &= \begin{bmatrix}\frac{\partial}{\partial i_1}\boldsymbol{\phi} & \frac{\partial}{\partial i_2}\boldsymbol{\phi} & \cdots & \frac{\partial}{\partial i_{n_\text{coils}}}\boldsymbol{\phi}\end{bmatrix}, \qquad \in \mathbb{R}^{n_\text{coils}\times n_\text{coils}} \\ &= \frac{\partial}{\partial\boldsymbol{i}_\text{coil}}\boldsymbol{X}^\mathsf{T}\boldsymbol{K}_\nu^{-1}(a^*)\boldsymbol{X}\boldsymbol{i}_\text{coil} \\ &= \boldsymbol{X}^\mathsf{T}\boldsymbol{K}_{\nu,d}^{-1}(\boldsymbol{a}^*)\boldsymbol{X}\end{aligned} \tag{3.33}$$

where $\boldsymbol{a}^*$ is the solution of the field simulation and $\boldsymbol{K}_\nu^{-1}[d]$ is the inverse of the differential system matrix of the magnetostatic formulation. To programatically obtain the differential inductance, n_coils linear systems have to be solved with the system matrix using the materials for the solution $\boldsymbol{a}^*$.

$$\boldsymbol{K}_{\nu,d}(\boldsymbol{a}^*)\boldsymbol{a}_q = \boldsymbol{X}_q \tag{3.34}$$

$$\boldsymbol{L}_{\text{d},p,q} = \boldsymbol{X}_p^\mathsf{T}\boldsymbol{a}_q \tag{3.35}$$

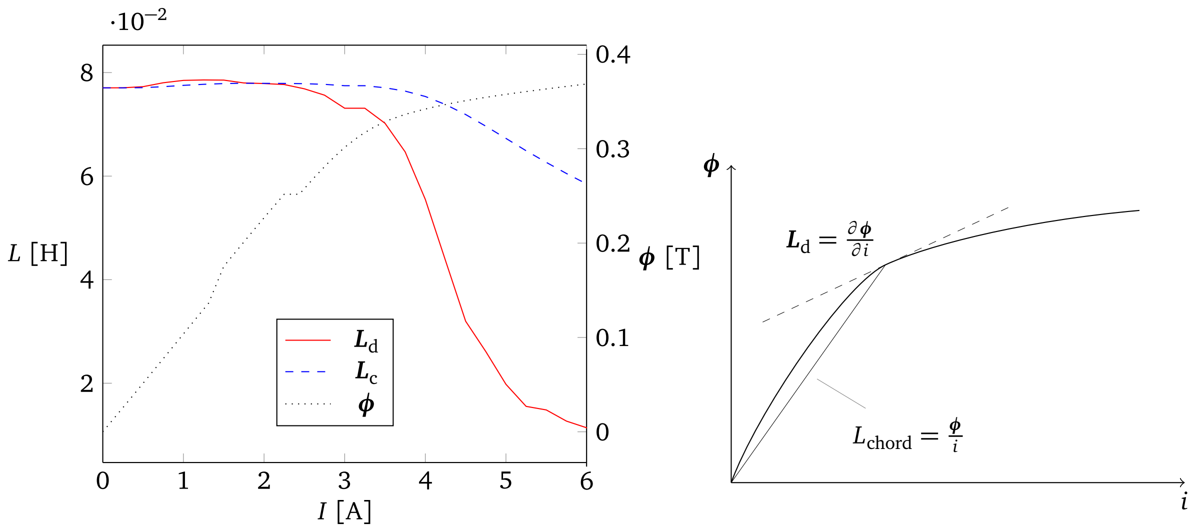
The differential system matrix is obtained by substituting the reluctivity by the differential reluctivity while assembling the system matrix. As an alternative, if the field simulator cannot provide this information, L_d can also be approximated for example with the forward difference quotient

$$\boldsymbol{L}_{\text{d},p,q} = \frac{\boldsymbol{\phi}_p(\boldsymbol{i}_\text{coil}) - \boldsymbol{\phi}_p(\boldsymbol{i}_\text{coil} + \boldsymbol{e}_q\delta)}{\delta}. \tag{3.36}$$

However, this approach involves n_coils additional computations of the full nonlinear field problem and its numerical stability depends on the choice of the perturbation δ.

For a field model with just one excitation current, e.g. a simple model of an inductor, all quantities described in this section become scalars.

The relationship between the chord and the differential inductance and their relation to the flux for the scalar case ($n_\text{coils} = 1$) is illustrated in Fig. 3.2b [BV05]. For idealized, linear inductors, the differential and the chord inductance are equivalent. For low currents where saturation effects have not yet taken into effect $\boldsymbol{L}_\text{c}$ is a good approximation for $\boldsymbol{L}_\text{d}$. Figure 3.2a shows values resulting from the simulation of a simple nonlinear inductor.



(a) Postprocessing Results of a FEM Model of an Inductor

(b) Interpretation of L_c and L_d

Figure 3.2: Comparison of Chord and Differential Inductance

4 Field/Circuit coupling

Many practical applications contain complex electric networks as well as geometry dependent components, e.g. electric machines. This motivates coupling of circuit and field simulation codes and has led to many different approaches, on which the first part of this chapter will provide a comparative overview. The second part develops the coupling method implemented in this work and discuss its properties.

4.1 Classification of coupling Methods

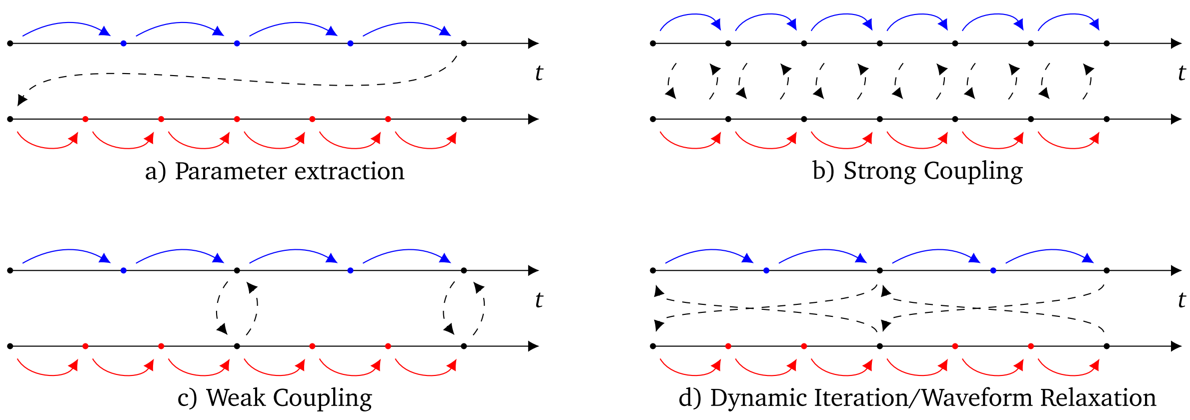


Figure 4.1: Different Classes of Coupling Schemes [CSC+12]

Figure 4.1 shows different approaches for coupling transient problems. For the timesteps marked in red and blue, the respective solver does not have to exchange coupling quantities with the other solver.

Direct Coupling

With "direct" or "strong" coupling, the degrees of freedom are exchanged in every timestep. The equations of the coupled systems can be written into a single matrix equation, like for example with two coupled problems

$$\boldsymbol{A}\boldsymbol{x}_A = \boldsymbol{r}_A \tag{4.1}$$

$$\boldsymbol{B}\boldsymbol{x}_B = \boldsymbol{r}_B \tag{4.2}$$

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C}^\intercal & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_A \\ \boldsymbol{r}_B \end{bmatrix} \tag{4.3}$$

with the coupling matrices $\boldsymbol{C}$. The dimensions of this matrix depend on the number of unknowns in the subproblems and can be highly asymmetric e.g. for field/circuit coupling problems where for most cases the number of degrees of freedom for the spacial discretization is a lot higher than the number of nodes in the circuit part.

Solving systems assembled like this has many drawbacks. The matrices for the different problems often have distinct structures and there are linear solvers which are specialized in solving them. This structure is lost when the systems are coupled.

Direct Coupling with Circuit Elements

To minimize coupling overhead it can be effective not to directly exchange the degrees of freedom, but to calculate an intermediate coupling value. This can be physically motivated as replacing the field model by a simpler model, determining its parameters from the field simulation and mathematically explained as a Schur Complement as shown in [SBD12]. A block Gauss step is performed on (4.3)

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{X} \\ \boldsymbol{X}^\intercal & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_A \\ \boldsymbol{r}_B \end{bmatrix} \tag{4.4}$$

by multiplying the second line with $\boldsymbol{X}\boldsymbol{B}^{-1}$ from the left and subtracting it from the first line

$$\begin{bmatrix} \boldsymbol{A}-\boldsymbol{X}\boldsymbol{B}^{-1}\boldsymbol{X}^{\mathsf{T}} & \boldsymbol{0} \\ \boldsymbol{X}^{\mathsf{T}} & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{x}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_A - \boldsymbol{X}\boldsymbol{B}^{-1}\boldsymbol{r}_B \\ \boldsymbol{r}_B \end{bmatrix}. \tag{4.5}$$

This leads to a smaller subproblem that can be solved independently and contributes with its solution to the remaining problem

$$\boldsymbol{B}\boldsymbol{x}_B = \boldsymbol{r}_B - \boldsymbol{X}^{\mathsf{T}}\boldsymbol{x}_A \tag{4.6}$$

$$\boldsymbol{A} - \boldsymbol{L}\boldsymbol{x}_A = \boldsymbol{r}_A - \boldsymbol{X}\boldsymbol{B}^{-1}\boldsymbol{r}_B. \tag{4.7}$$

The small but typically dense matrix $\boldsymbol{L} = \boldsymbol{X}_{AB}\boldsymbol{B}^{-1}\boldsymbol{X}_{BA}$ is the Schur complement of the coupled system. It is now possible to pass Jacobian information from the subproblem, which significantly reduces computational costs for nonlinear systems and allows the use of different specialized solvers for the different problems, while having minimal overhead due to data exchange. The drawback are the additional steps to compute the Schur complement.

Weak Coupling and Waveform Relaxation

The overhead for exchanging data in between different solvers can become significant as it often includes data conversions and in most cases requires copying significant amounts of data in between the involved solvers.

For some problems where the coupling is not that strong, good results can be obtained by exchanging data only at specific timesteps. This also allows using different time discretization schemes and step sizes, which is especially beneficial if the dominant time constants in the subproblems differ significantly e.g. fast switched circuits with slow changing field devices.

The Waveform Relaxation method performs the simulation by simulating relatively long intervals independently and the solutions are then used for reiteration over the same interval until the solutions are converged. Weak methods can benefit from parallelization and are in general easier to implement for existing solvers. However for strongly coupled systems weak methods may need many iterations to converge and the relaxation method can even diverge.

4.2 Coupling Scheme

In this work, an approach using strong coupling with circuit elements is used. The following chapter reviews its properties.

4.2.1 Circuit Element for Nonlinear Magnetostatic Problems

To couple the field model with the circuit model, a coupling matrix $\boldsymbol{X}$ is defined, which distributes the branch currents $\boldsymbol{i}_{\text{coil}}$ on the coils of the field model, as well as an incidence matrix $\boldsymbol{A}_{\text{f}}$, wich connects these currents to the corresponding nodes of the circuit simulation. For easier illustration, the fluxes $\boldsymbol{\phi}$ and the currents through the field model $\boldsymbol{i}_{\text{coil}}$ are introduced as additional unknowns. The discretized field equation is

$$\boldsymbol{K}_{\nu}(\boldsymbol{a})\boldsymbol{a} - \boldsymbol{X}\boldsymbol{i}_{\text{coil}} = \boldsymbol{0}, \tag{4.8}$$

the flux can be written as

$$\boldsymbol{\phi} - \boldsymbol{X}^{\mathsf{T}}\boldsymbol{a} = \boldsymbol{0} \tag{4.9}$$

and the induction law contributes to the circuit with

$$-\frac{d}{dt}\boldsymbol{\phi}_{\text{old}} - \boldsymbol{A}_{\text{f}}^{\mathsf{T}}\boldsymbol{u} = \boldsymbol{0}. \tag{4.10}$$

Equations (4.8) to (4.10) can be used to extend the MNA System (2.6)

$$\begin{bmatrix} \boldsymbol{A}_{\text{C}}^{\mathsf{T}}\boldsymbol{C}\boldsymbol{A}_{\text{C}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{L} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \frac{d}{dt} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{i}_{\text{V}} \\ \boldsymbol{i}_{\text{L}} \\ \boldsymbol{i}_{\text{coil}} \\ \boldsymbol{\phi} \\ \boldsymbol{a} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{\text{G}}^{\mathsf{T}}\boldsymbol{G}\boldsymbol{A}_{\text{G}} & \boldsymbol{A}_{\text{V}} & \boldsymbol{A}_{\text{L}} & \boldsymbol{A}_{\text{f}} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{A}_{\text{V}}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{A}_{\text{L}}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{A}_{\text{f}}^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{X}^{\mathsf{T}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{X} & \boldsymbol{0} & \boldsymbol{K}_{\nu}(\boldsymbol{a}) \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{i}_{\text{V}} \\ \boldsymbol{i}_{\text{L}} \\ \boldsymbol{i}_{\text{coil}} \\ \boldsymbol{\phi} \\ \boldsymbol{a} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{i}_{\text{s}} \\ -\boldsymbol{v}_{\text{s}} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}. \tag{4.11}$$

Applying the implicit Euler integration scheme yields

$$\begin{bmatrix} \frac{1}{h}A_C^{\top}CA_C + A_G^{\top}GA_G & A_V & A_L & A_f & 0 & 0 \\ -A_V^{\top} & 0 & 0 & 0 & 0 & 0 \\ -A_L^{\top} & 0 & \frac{1}{h}L & 0 & 0 & 0 \\ -A_f^{\top} & 0 & 0 & 0 & -\frac{1}{h}I & 0 \\ 0 & 0 & 0 & 0 & -I & X^{\top} \\ 0 & 0 & 0 & -X & 0 & K_\nu(a) \end{bmatrix} \begin{bmatrix} u_i \\ i_{V,i} \\ i_{L,i} \\ i_{\mathrm{coil},i} \\ \phi_i \\ a_i \end{bmatrix} = \begin{bmatrix} -i_s + \frac{1}{h}A_C^{\top}CA_C u_{i-1} \\ -v_s \\ \frac{1}{h}L i_{L,i-1} \\ \frac{1}{h}\phi_{i-1} \\ 0 \\ 0 \end{bmatrix}. \tag{4.12}$$

Eliminating the last line by a block Gauss step, multiplying it with $-X^{\top}K_\nu^{-1}(a)$ and subtracting it from the fourth line, splits the problem into the circuit simulation and the field simulation.

$$\begin{bmatrix} \frac{1}{h}A_C^{\top}CA_C + A_G^{\top}GA_G & A_V & A_L & A_f & 0 \\ -A_V^{\top} & 0 & 0 & 0 & 0 \\ -A_L^{\top} & 0 & \frac{1}{h}L & 0 & 0 \\ -A_f^{\top} & 0 & 0 & 0 & -\frac{1}{h}I \\ 0 & 0 & 0 & -X^{\top}K_\nu^{-1}(a)X & -I \end{bmatrix} \begin{bmatrix} u_i \\ i_{V,i} \\ i_{L,i} \\ i_{\mathrm{coil},i} \\ \phi_i \end{bmatrix} = \begin{bmatrix} -i_s + \frac{1}{h}A_C^{\top}CA_C u_{i-1} \\ v_s \\ \frac{1}{h}L i_{L,i-1} \\ \frac{1}{h}\phi_{i-1} \\ 0 \end{bmatrix} \tag{4.13}$$

$$K_\nu a_i = X i_{\mathrm{coil}}. \tag{4.14}$$

For circuit simulators that do not have the flux as an unknown, it can also be removed from the circuit system.

$$\begin{bmatrix} \frac{1}{h}A_C^{\top}CA_C + A_G^{\top}GA_G & A_V & A_L & A_f \\ -A_V^{\top} & 0 & 0 & 0 \\ -A_L^{\top} & 0 & \frac{1}{h}L & 0 \\ -A_f^{\top} & 0 & 0 & \frac{1}{h}X^{\top}K_\nu^{-1}(a)^{\top}X \end{bmatrix} \begin{bmatrix} u_i \\ i_{V,i} \\ i_{L,i} \\ i_{\mathrm{coil},i} \end{bmatrix} = \begin{bmatrix} -i_s + \frac{1}{h}A_C^{\top}CA_C u_{i-1} \\ -v_s \\ \frac{1}{h}L i_{L,i-1} \\ \frac{1}{h}X^{\top}K_\nu^{-1}(a)^{\top}X i_{\mathrm{coil},i-1} \end{bmatrix} \tag{4.15}$$

$$\phi_i = X^{\top}K_\nu^{-1}(a)^{\top}X i_{\mathrm{coil},i}. \tag{4.16}$$

It can be seen, that the contribution of the field problem to the system generated by the MNA analysis is similar to the contribution of the inductances

$$L_c = X^{\top}K_\nu(a^*)X \tag{4.17}$$

where a^* is the solution of the field problem for the excitation $i_{\mathrm{coil}} = i_{\mathrm{coil},i}$ in the current timestep i. For the simple case of a single magnetic circuit in the field problem, L_c is a scalar value.

4.2.2 Multilevel Newton

The circuit element method described in the previous chapter leads to two independent nonlinear subproblems described in (4.15) and (4.16) that can be simplified to

$$F_f(a, i_{\mathrm{coil}}) = K_\nu a - X i_{\mathrm{coil}}(x) \qquad = 0 \tag{4.18}$$

$$\widetilde{F}_c(x, i_{\mathrm{coil}}) = \begin{bmatrix} M_{\mathrm{MNA}} & \widetilde{A}_f \\ -\widetilde{A}_f^{\top} & \frac{1}{h}L_c \end{bmatrix} \begin{bmatrix} x_i \\ i_{\mathrm{coil},i} \end{bmatrix} - \begin{bmatrix} r + \frac{1}{h}M_c x_{i-1} \\ \frac{1}{h}L_c i_{\mathrm{coil},i-1} \end{bmatrix} \qquad = 0 \tag{4.19}$$

where $M_{\mathrm{MNA}} = \left[\frac{1}{h}M_c + K_c\right]$ is the system matrix assembled by the MNA analysis and $\widetilde{A}_f^{\top} = \left[A_f^{\top}, 0\right]$ is the incidence matrix assigning the excitation currents of the field problem to the nodes of the MNA Analysis. The connection to the field problem is the inductance matrix L_c which depends on the solution a_i of the field problem for the excitation current. This problem can be solved by using two nested Newton schemes. At the beginning of the outer iteration the unknowns of the circuit simulation are initialized

$$x = x_{i-1}, \qquad i_{\mathrm{coil}} = i_{\mathrm{coil},i-1}, \tag{4.20}$$

then the Newton iteration j is performed by iteratively repeating

$$\widetilde{J}_{F_c,i,j}\Delta\widetilde{x}_{i,j} = -\widetilde{F}_c(\widetilde{x}_{i,j}) \tag{4.21}$$

$$\widetilde{x}_{i,j+1} = \widetilde{x}_{i,j} + \Delta\widetilde{x}_{i,j} \qquad \text{with } \widetilde{x}_{i,j} = \begin{bmatrix} x_{i,j} \\ i_{\mathrm{coil},i,j} \end{bmatrix} \tag{4.22}$$

until a convergence criterion is satisfied. The Newton step $\Delta\widetilde{\boldsymbol{x}}_{i,j}$ is the solution of the linear equation system (4.21) with the Jacobian

$$\widetilde{\boldsymbol{J}}_{F_c} = \begin{bmatrix} \boldsymbol{J}_{F_c} & \widetilde{\boldsymbol{A}}_f \\ -\widetilde{\boldsymbol{A}}_f^\top & \frac{1}{h}\frac{\partial \boldsymbol{X}^\top \boldsymbol{K}_\nu^+ \boldsymbol{X}}{\partial \boldsymbol{i}_{\mathrm{coil}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{F_c} & \widetilde{\boldsymbol{A}}_f \\ -\widetilde{\boldsymbol{A}}_f^\top & \frac{1}{h}\boldsymbol{L}_d \end{bmatrix} \tag{4.23}$$

where $\boldsymbol{J}_{F_c}$ is the Jacobian from the MNA scheme as described in Section 2.4. The Jacobian contains the differential inductance $\boldsymbol{L}_d$ which is therefore needed to solve (4.21) along with $\boldsymbol{L}_c$.

If $\boldsymbol{L}_d$ is not obtained exactly from the field simulation software but approximated, e.g. by finite differences, the convergence may deteriorate from quadratic to superlinear. Finally if the chord inductance $\boldsymbol{L}_c$ is used, this corresponds to a simplified Newton scheme, where the Jacobian is only computed in the beginning of the scheme, as in the beginning of the Newton scheme $\boldsymbol{L}_c$ equals $\boldsymbol{L}_d$ because there are no saturation effects taking part.

For the simple case with a single excitation current the chord inductance $\boldsymbol{L}_c$ only depends on the flux $\boldsymbol{\phi}$ and the excitation current $\boldsymbol{i}_{\mathrm{coil}}$. As they are both known by the circuit simulator, it is equivalent to exchange the flux instead of the inductance and calculate the latter inside the circuit simulator or introduce the flux as an additional unknown to the MNA scheme (4.13) extending the MNA approach to the flux-charge-oriented approach.

As shown in Section 3.4 these quantities can be obtained from the field solution after the inner Newton iteration of the field simulation software has converged. As the field solver is not modified, the inner Newton iteration is described in Section 3.2.

The structure of the two nested Newton iterations is illustrated in Fig. 4.2.

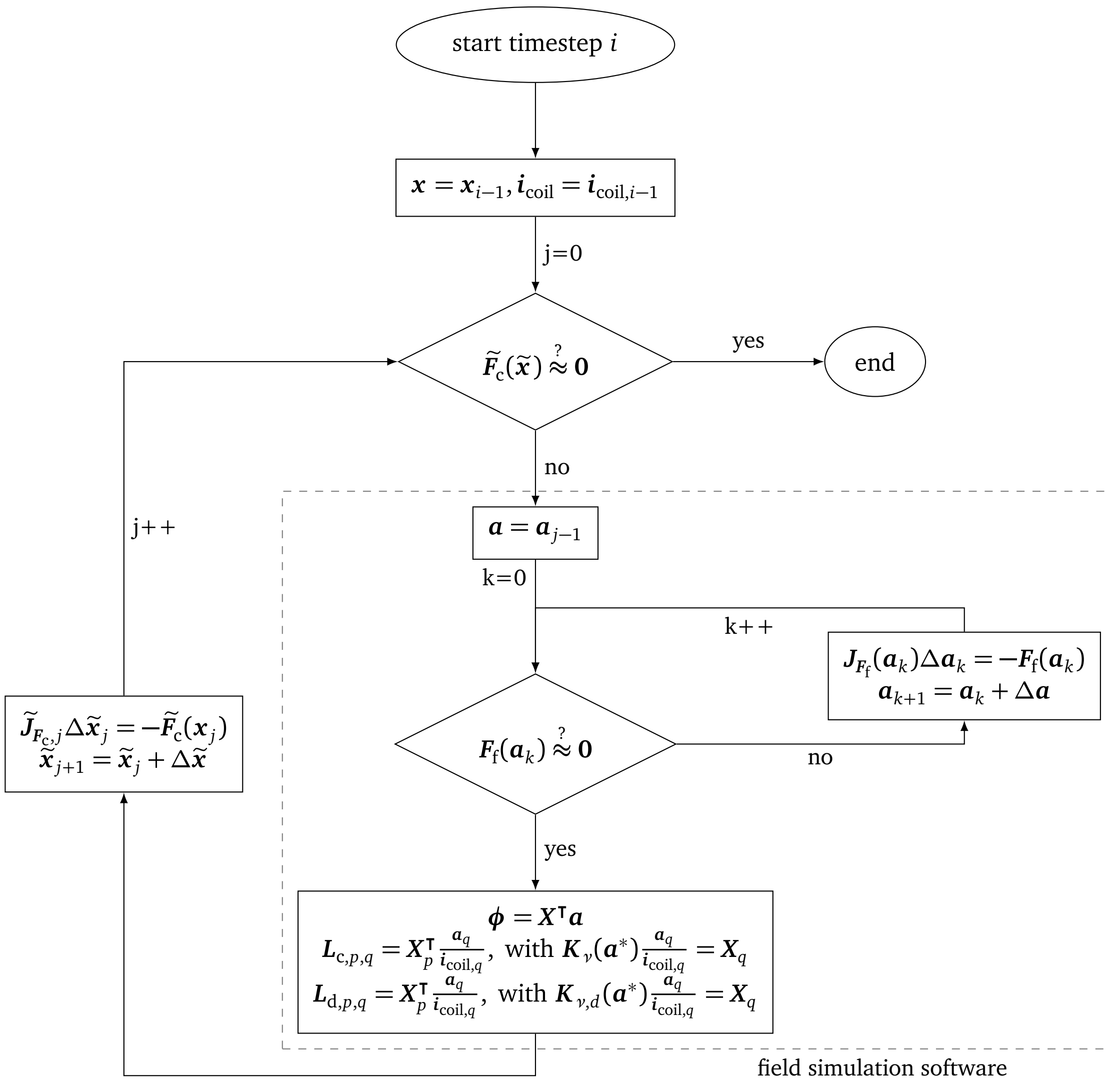

start timestep i
$\boldsymbol{x} = \boldsymbol{x}_{i-1}, \boldsymbol{i}_{\mathrm{coil}} = \boldsymbol{i}_{\mathrm{coil},i-1}$
j=0
$\widetilde{\boldsymbol{F}}_c(\widetilde{\boldsymbol{x}}) \overset{?}{\approx} \boldsymbol{0}$
yes
end
no
j++
$\boldsymbol{a} = \boldsymbol{a}_{j-1}$
k=0
k++
$\boldsymbol{J}_{F_f}(\boldsymbol{a}_k)\Delta\boldsymbol{a}_k = -\boldsymbol{F}_f(\boldsymbol{a}_k)$
$\boldsymbol{a}_{k+1} = \boldsymbol{a}_k + \Delta\boldsymbol{a}$
$\widetilde{\boldsymbol{J}}_{F_c,j}\Delta\widetilde{\boldsymbol{x}}_j = -\widetilde{\boldsymbol{F}}_c(\boldsymbol{x}_j)$
$\widetilde{\boldsymbol{x}}_{j+1} = \widetilde{\boldsymbol{x}}_j + \Delta\widetilde{\boldsymbol{x}}$
$\boldsymbol{F}_f(\boldsymbol{a}_k) \overset{?}{\approx} \boldsymbol{0}$
no
yes
$\boldsymbol{\phi} = \boldsymbol{X}^\top \boldsymbol{a}$
$\boldsymbol{L}_{c,p,q} = \boldsymbol{X}_p^\top \frac{\boldsymbol{a}_q}{i_{\mathrm{coil},q}}$, with $\boldsymbol{K}_\nu(\boldsymbol{a}^*)\frac{\boldsymbol{a}_q}{i_{\mathrm{coil},q}} = \boldsymbol{X}_q$
$\boldsymbol{L}_{d,p,q} = \boldsymbol{X}_p^\top \frac{\boldsymbol{a}_q}{i_{\mathrm{coil},q}}$, with $\boldsymbol{K}_{\nu,d}(\boldsymbol{a}^*)\frac{\boldsymbol{a}_q}{i_{\mathrm{coil},q}} = \boldsymbol{X}_q$
field simulation software

Figure 4.2: Flowchart of the Multilevel Newton Scheme Used to Simulate the Coupled Problem

5 Implementation

In this chapter the implementation of the coupling is described, first for the Octave circuit simulation package OCS and then for the standalone “Quite Universal Circuit Simulator”. Both use GetDP for the field simulation part and ONELAB to exchange simulation data.

5.1 Used Software Packages

This work uses several open source projects, that will be described in this section.

5.1.1 ONELAB

ONELAB provides an interface to share parameters between different solvers [Geu14]. It uses a client-server topology with the default server being integrated into gmsh. The server part is kept relatively simple and just allows posting and querying parameters and starting subclients. To expose solver parameters to the other solvers and the user, an initialization step is performed, in which every solver sends all available parameters with additional properties like range, choices and description text to the server. The parameters are organized in a tree structure on the server to avoid name clashes and provide easy access. Additionally gmsh as the ONELAB server provides (nested) parameter sweeps and plotting facilities, which can also be controlled by the client solvers via parameter properties.

Client implementations exist for various computation programs and client libraries exist in Python and C++. For Octave [EBH09] there is a rudimentary ONELAB package, that exposes the most important parts of the C++ API.

5.1.2 GetDP

GetDP is a FE solver specialized in but not limited to solving electromagnetic problems [Geu07]. As a native ONELAB-client, all computation results can be exported to the ONELAB Server in the post processing step and GetDP’s behaviour can be controlled by ONELAB parameters. GetDP doesn’t provide its own graphical user interface, but is closely associated with gmsh, so most of the features can be accessed through the gmsh GUI.

5.1.3 Octave Circuit Simulator

The Octave Circuit Simulator (OCS) [FMM11] is an Octave package, that provides functions to run DC and transient simulations on circuit networks. It uses the IFF file format specified in [Fal13]. The implementation in Octave is very close to the common matrix representation of the modified nodal analysis, which makes it easy to understand and to extend, but is limited to small circuits of basic elements.

5.1.4 Quite Universal Circuit Simulator

The Quite Universal Circuit Simulator (Qucs) is another circuit simulator written in C++ [BJ09]. It is split into the main Qucs programm, a circuit editor with post processing capabilities and the actual simulator, a command line program called qucsator [Nov13]. The internal design is very modular and allows flexible extension with dynamic modules.

5.2 Coupling OCS with GetDP

To couple OCS with GetDP, an Octave solver was implemented using the Octave ONELAB package. Figure 5.1 shows the interactions between the different processes involved in the coupled simulation. The main file of the solver is `ocs_onelab.m`, which loads the ONELAB and ocs packages and connects to the ONELAB server. According to the value of the “ocs_onelab/Action” field either `ocs_onelab_init.m` or `ocs_onelab_compute.m` is run. The first one exposes the simulation parameters (time, step size) to the ONELAB server and calls GetDP to announce its parameters too. The second one gets executed for the “compute” step and calls the OCS routines to load the circuit from a netlist and

```
% start computation in GetDP
      % set coil excitation
      Iparam = ol_getParameters('Input/4Coil Parameters/0Current (rms) [A]');
      Iparam{1}.value = jgetdp;
      ol_setParameter(Iparam{1});
      % run GetDP
      ol_runBlockingSubClient('incuctance',['getdp ' string '/' string '.pro -pre -cal -pos -msh ' string
      % get inductance
      Lparam = ol_getParameters('Output/50Inductance from Flux [H]');
L = Lparam{1,1}.value;
```

Listing 1: Excerpt from Mgetdp.m

solves the resulting system. `Mgetdp.m` is a device file for OCS, which is evaluated in every iteration of the integration and Newton loops of OCS. It determines the current in the field device and calls GetDP as a ONELAB subclient after setting the excitation, see Listing 1. The differential inductance is obtained by the use of finite differences, which requires another field simulation but may increase the convergence speed from superlinear to quadratic. Afterwards the inductance is read from ONELAB and the resulting MNA matrices are returned.

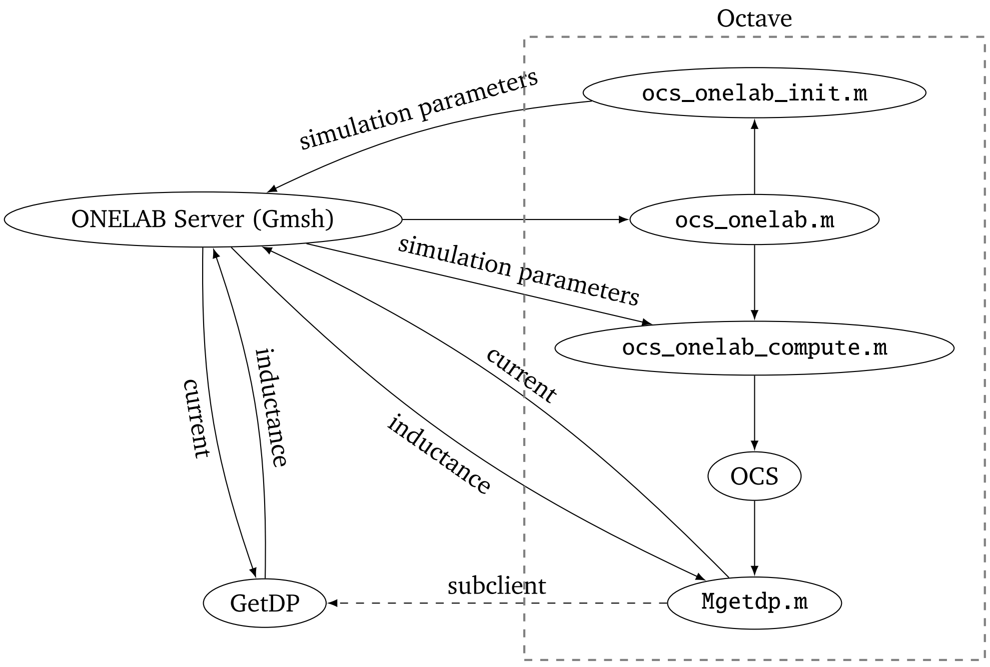


Figure 5.1: Coupling Between GetDP and Octave Circuit Simulator (OCS)

Running Coupled Simulations with OCS and GetDP

As coupling between OCS and GetDP was mainly used to verify the coupling method, the code is hardcoded for the test problem and would have to be changed for different problems.

The simulation of the sample circuit can be started by running

```
gmsh ocs_onelab.m
```

from within the `getdp-ocs` folder and hitting the "run"-Button at the bottom of the parameter tree. Software that needs to be installed is Octave with the OCS and the octaveonelab packages, gmsh and GetDP.

5.3 Coupling Qucs with GetDP

For coupling GetDP field models with Qucs, a dynamically loadable module for Qucs was developed. It contains a custom component called qucsonelab which can communicate with a ONELAB server to start simulations of the subproblem in

GetDP and exchange coupling information. The interactions between Qucs, GetDP, Gmsh as the ONELAB server and the different Python scripts are illustrated in Fig. 5.2. Also a Python ONELAB solver for qucsator — the command line simulation backend of Qucs — was written to be able to call the coupled solver from ONELAB directly. This allows to perform optimizations and parameter sweeps on the coupled problem or even use the coupled simulation as a subproblem in another higher-level simulation. The script takes a Qucs schematic and replaces the inductance named “onelab” with the qucsonelab component, exposes the parameters of the transient solver “solve” as ONELAB parameters and generates a modified netlist. On this modified netlist, qucsator is called via a Python wrapper script. This is necessary as qucsator offers no way to provide command line arguments for specific dynamic modules, so the ONELAB socket and parameter names are read from a temporary file. This non-intrusive solution was preferred over changing the Qucs source. The wrapper also takes care to pass the required command line parameters to qucsator to load the dynamic module containing the qucsonelab component.

When the simulation with qucsator is started, the qucsonelab component is initialized and a connection to the ONELAB server is established with the socket from the temporary file. In each time step and iteration of the Newton scheme, caltTR from the qucsonelab component is called. The current is then set as the excitation for the field model and GetDP is called as a ONELAB subclient. When GetDP has stopped the values for flux and differential inductance are read from ONELAB, according to the defined parameter names. At this point two special cases are considered: if the parameter for the differential inductance is “diffquot” another run of GetDP is started with a perturbation to the current to approximate it with the finite difference. If it is “Lchord” the chord inductance $L_c = \frac{\phi}{i}$ is used. A minimal excitation is ensured to prevent division by zero. In the end the entries in the MNA matrices are set, as described in Section 2.3. As the last step a flag is set to initialize the Newton iteration in GetDP with the solution of the last time step to reduce the number of Newton iterations GetDP needs to converge.

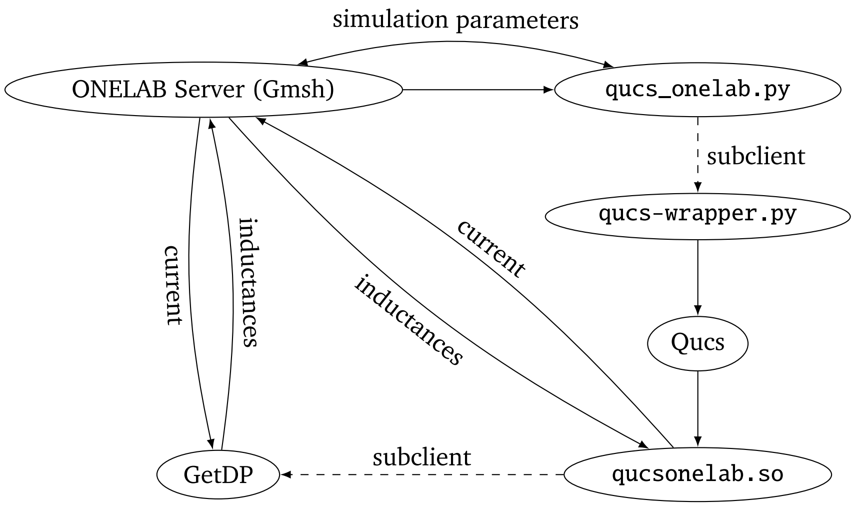


Figure 5.2: Coupling Between GetDP and Quite Universal Circuit Simulator (QUCS)

Compiling the Coupling Component for Qucs

To compile the coupling component ensure that Qucs and Gmsh are installed and their header files are in the systems include path. Then in the `qucs-onelab/qucs-component` folder run

```
make all
```

to compile the component. As Qucs expects the component module in the working directory, the module has to be copied or symlinked to `qucs-onelab` for example with

```
ln -s qucsonelab.so ..
```

Running Coupled Simulations with Qucs and GetDP

To simulate coupled problems with Qucs and GetDP

```
gmsh qucs-onelab.py
```

has to be started from the `getdp-qucs` folder. The field problem can then be selected in gmsh by setting the appropriate check and compute GetDP commands and names of the ONELAB parameters used for the coupling quantities. The circuit is selected by setting a circuit name and the parameters for the transient analysis can be set.

When the simulation is started, the current in the inductor can be plotted while the simulation is running. The current value is saved after every timestep, as well as the simulation time, so that a correct plot can be generated even if the step size is adjusted.

The results of the simulation are written to a solution file by Qucs, which can then be visualized along the original schematic with Qucs postprocessing features.

5.4 Test Problem

To test the coupling code, an academic RL-network with a simple nonlinear field model of an inductor is used. The following section explains the field and circuit models used to verify the coupling method and shows how existing models had to be modified to allow efficient coupling.

5.4.1 Circuit

The circuit of the test problem consists of a DC voltage source, a resistor and the field device. The schematic is shown in Fig. 5.3a. For field problems with linear material laws, this is equivalent to an RL-network and results in an exponential function for the current through the field device.

5.4.2 Field Problem

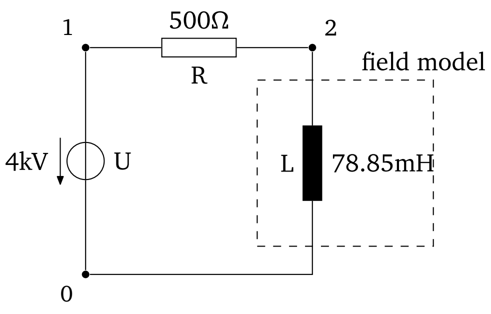
The inductor is based on the example of the ONELAB project. Its geometry is shown in Fig. 5.3b and its saturation characteristics are shown in Fig. 3.2a. It already exposes all necessary data to ONELAB, with the exception of the differential inductance L_d, that was added as an additional postprocessing step to the GetDP problem file `inductor.pro`. In the main postprocessing step the differential reluctivity dhdb is stored in a field

```
PostOperation Get_LocalFields UsingPost MagStaDyn_a_2D {
  Print[ az, OnElementsOf Domain, File StrCat[Dir,StrCat["a",ExtGmsh]], LastTimeStepOnly ];
  Print[ dhdb, OnElementsOf Domain, LastTimeStepOnly, StoreInField 66 ];
}
```

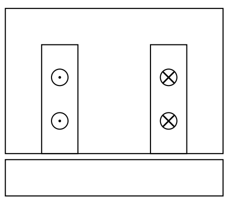
which is then used in a linear system (3.34)

```
{ Name MagStaDyn_a_2D_Ld; Type FemEquation ;
  Quantity {
    { Name a  ; Type Local  ; NameOfSpace Hcurl_a_2D_Ld ; }
  }

  Equation {
    Galerkin { [ Field[XYZ[]]{66} * Dof{d a}  , {d a} ] ;
      In Domain ; Jacobian Vol ; Integration I1 ; }
```



(a) Simple RL-Network



(b) The 2D Geometry of the Inductor

Figure 5.3: Simple Coupled Problem.

```

      Galerkin { [ -NbWires[]/SurfCoil[] * vDir[] , {a} ] ;
        In DomainB ; Jacobian Vol ; Integration I1 ; }
    }
  }
```

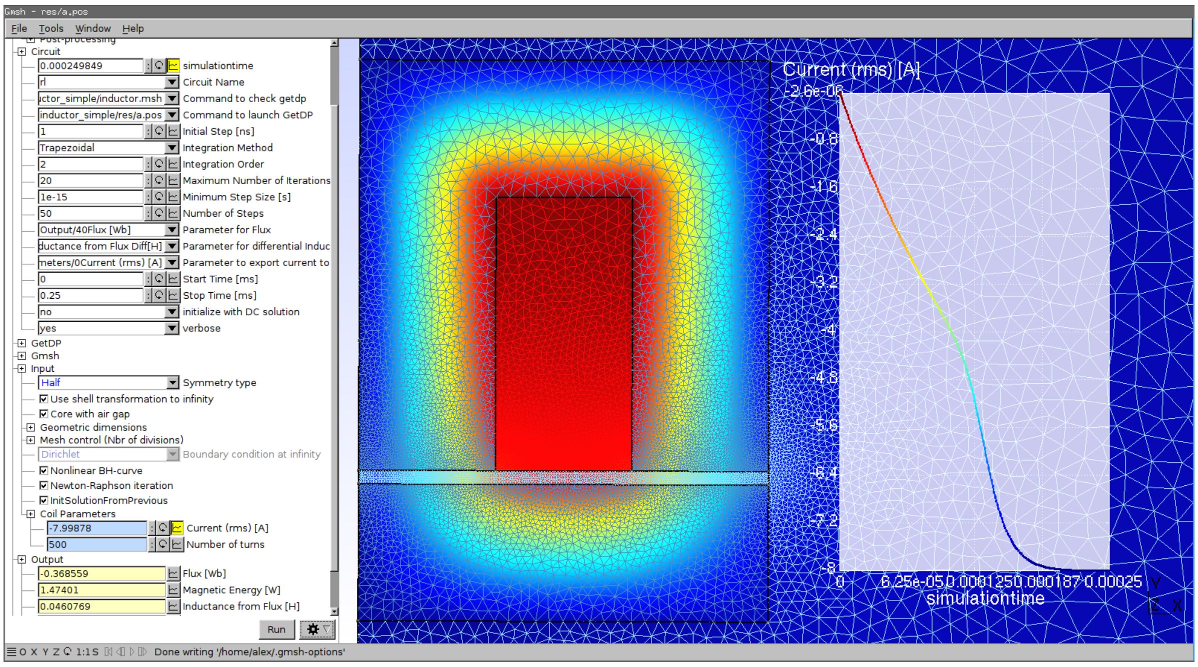
from which L_d can be calculated by integrating over the coils of the inductor.

```
  { Name MagStaDyn_a_2D_Ld ; NameOfFormulation MagStaDyn_a_2D_Ld ;
    PostQuantity {
      { Name Ld; Value {
          Integral { [ SymmetryFactor*Lz*Idir[]*NbWires[]/SurfCoil[]* CompZ[{a}] ] ;
            In Inds  ; Jacobian Vol ; Integration I1 ; }
        }
      }
    }
  } //Postprocessing: MagStaDyn_a_2D_Ld
```

The modified inductor can be optionally initialized with the solution from the previous iteration, which is read from the resolution file. As the changes in the excitation in between the iterations are small, the changes in the magnetic vector potential are also small and initializing the Newton scheme reduces the number of iterations significantly.

The inductor is modeled as a 2D problem, but the coupling approach also works on 3D field problems. The geometry of the inductor is shown in Fig. 5.3b.

5.5 Numerical Results



Circuit	
0.000249849	simulationtime
rl	Circuit Name
ıctor_simple/inductor.msh	Command to check getdp
inductor_simple/res/a.pos	Command to launch GetDP
1	Initial Step [ns]
Trapezoidal	Integration Method
2	Integration Order
20	Maximum Number of Iterations
1e-15	Minimum Step Size [s]
50	Number of Steps
Output/40Flux [Wb]	Parameter for Flux
ductance from Flux Diff[H]	Parameter for differential Induc
meters/0Current (rms) [A]	Parameter to export current to
0	Start Time [ms]
0.25	Stop Time [ms]
no	initialize with DC solution
yes	verbose
GetDP	
Gmsh	
Input	
Half	Symmetry type
☑	Use shell transformation to infinity
☑	Core with air gap
Geometric dimensions	
Mesh control (Nbr of divisions)	
Dirichlet	Boundary condition at infinity
☑	Nonlinear BH-curve
☑	Newton-Raphson iteration
☑	InitSolutionFromPrevious
Coil Parameters	
-7.99878	Current (rms) [A]
500	Number of turns
Output	
-0.368559	Flux [Wb]
1.47401	Magnetic Energy [W]
0.0460769	Inductance from Flux [H]

Done writing '/home/alex/.gmsh-options'

Figure 5.4: The Field Problem in Gmsh with the Plot of the Current in the Coupled Problem.

To evaluate the results of the coupled simulation three simulations were performed. Figure 5.4 shows the results of the coupled simulation in gmsh with the field simulation below. To get a reference behavior, a first simulation is done solely in the circuit simulator with a constant inductance obtained from the field model. Then the problem is simulated with the coupling code, once with linear and once with nonlinear material properties. Figure 5.5 shows that for linear materials the coupling code produces the same results as the uncoupled problem with the extracted inductance. For the nonlinear model, saturation effects come into play, decreasing the inductance.

To compare the behavior of coupling method for the different coupling quantities proposed in Section 4.2.2 the transient simulation of the RL series circuit was repeated using L_d, L_c and an approximation of L_d with a difference quotient

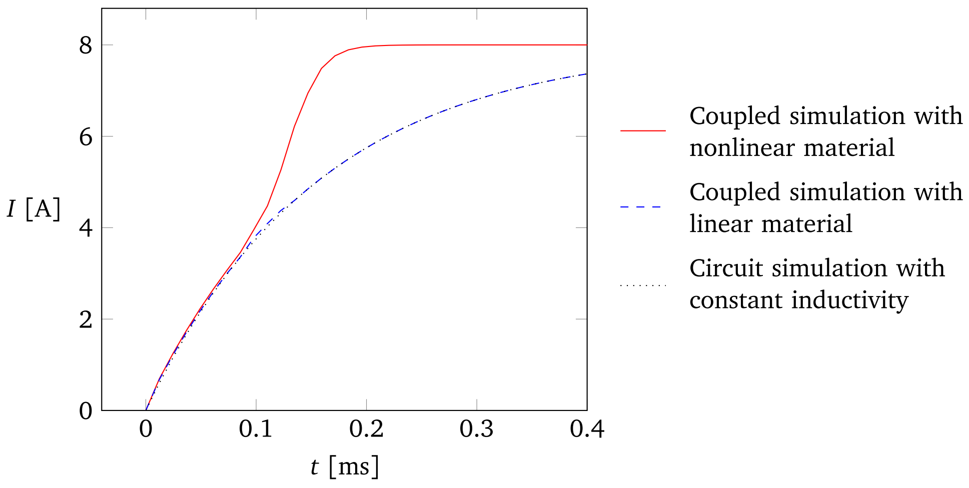


Figure 5.5: Results for Transient Simulations of an RL Network

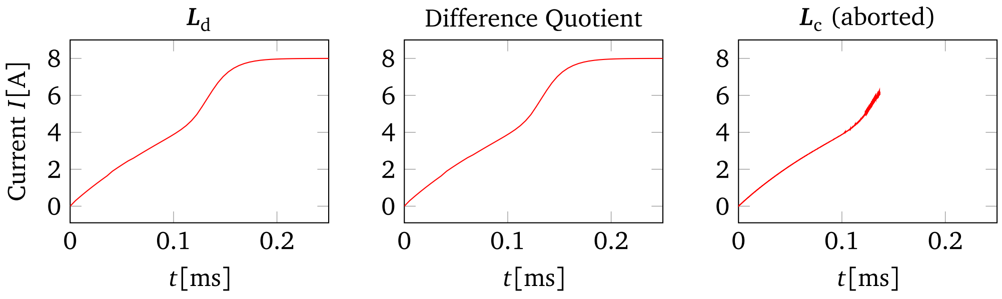


Figure 5.6: Simulation Results for the RL Circuit with different coupling Quantities

in the Jacobian (4.23) of the circuit simulator's Newton scheme. The results of this simulation are shown in Fig. 5.6. It can be seen that the results are the same for all the coupling quantities.

Figure 5.7 shows the number of iterations of the field simulator's Newton scheme, along with the size of the Euler steps and the current through the field problem for a simulation time of 0.25 ms. All simulations are started with the same parameters, except that the Newton scheme for the chord inductance is granted a higher number of iterations per step.

It can be seen that for the first timesteps, many Newton iterations are needed for all implemented coupling quantities. This is caused by the fact, that in the first timestep, there is no previous solution to initialize from. These timesteps are responsible for a significant portion of Newton iterations. Heuristics could be applied to improve this, but they are not implemented within this work.

The number of Newton iterations per timestep shows, that coupling with the differential inductance and the approximated differential inductance from the forward difference quotient offer good performance. It has to be kept in mind, that the differential quotient needs twice the number of field solutions per timestep compared to the use of $\boldsymbol{L}_d$.

As expected, the chord inductance works well when saturation of the field model is low, but as soon as saturation effects become significant, the convergence of the circuit solver's Newton scheme deteriorates and the timestep is reduced to keep the number of iterations low. This simulation was eventually aborted, because there was no progress for a reasonable amount of time, as can be seen on the small stepsize.

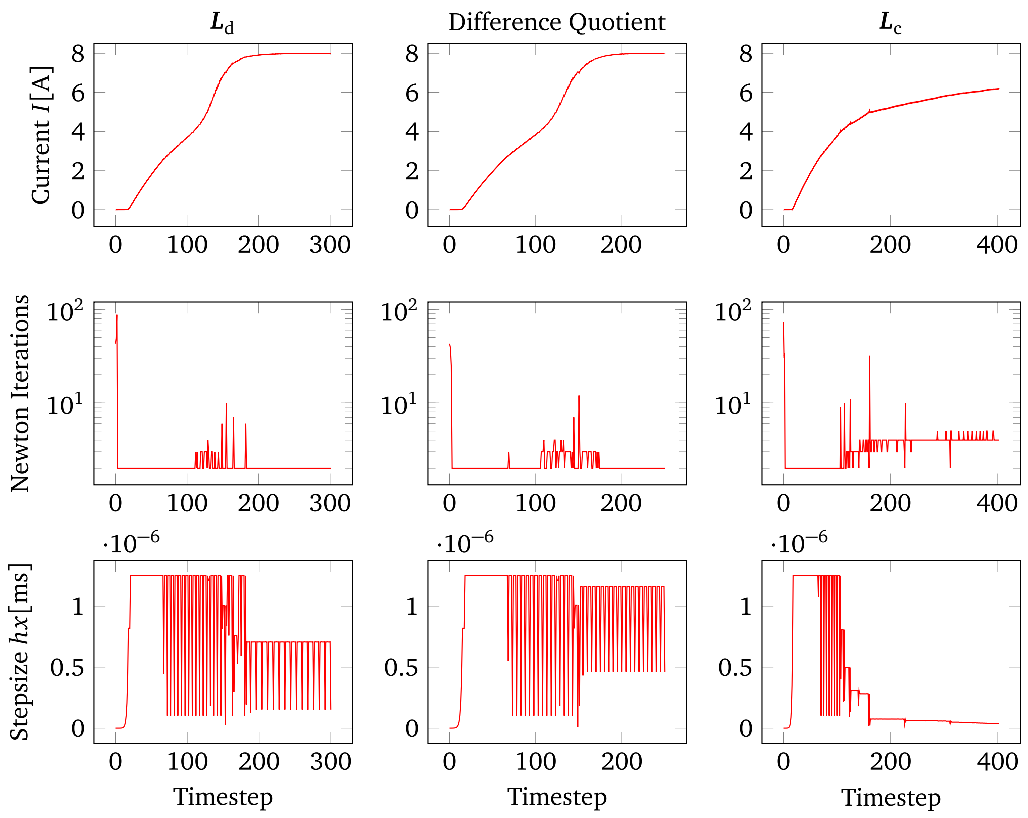


Figure 5.7: Convergence of the Timesteps for the Circuit Simulation for Different Coupling Quantities

5.6 Computational Effort and Coupling Overhead

Simulating the approximately 300 Timesteps of the example takes almost 90 min on a quad core CPU with 8 GB of RAM. Therefore an analysis is necessary to determine where this computation time is used and if there is potential for improvement.

Fig. 5.8 shows which components of the coupling account for the simulation time. The time spent in the circuit simulator was measured with the cputime command, neglecting wait instructions and assuming that it runs as a single threaded process. The time spent in getdp was evaluated by summation of the cputime that it logs into the ONELAB log window as well as the timestamps from the log to estimate the time used for file access and other asynchronous events. The time used for solving the field problem is estimated by the cputime of the solve steps from the log. This duration is subtracted from the total real time from the start and stop timestamp from the log to obtain the overhead in getdp. The difference to the total simulation time, again extracted from the timestamps in the ONELAB log is accounted for parameter storage and processing in ONELAB. While this approach is far from accurate, it is sufficient for an overview over the time consumption of the different parts of the simulation.

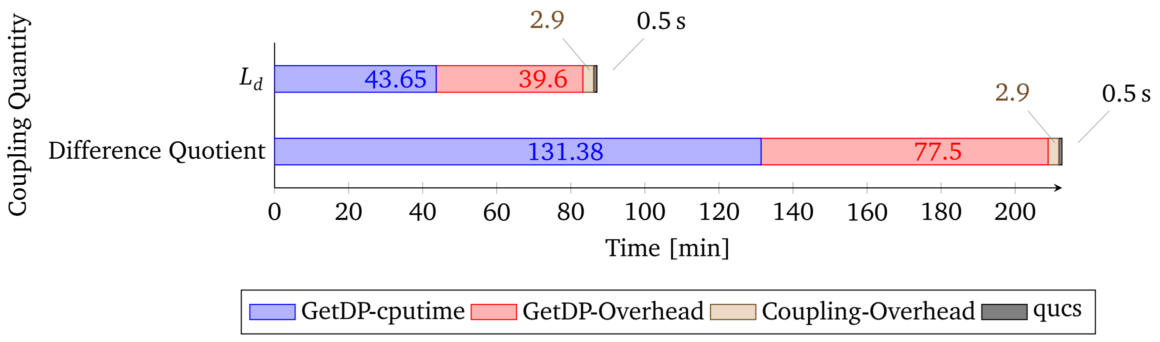


Figure 5.8: Distribution of Computation Time Used by the Software Components of the Setup

The circuit simulator takes almost no computational time due to the very simple electric network with only three nodes. Its time in Fig. 5.8 was slightly increased to make it visible. Nevertheless, it verifies that the overhead introduced by the coupling module is negligible for the circuit simulation part.

The durations for the field simulation show, that approximating the differential inductance increases simulation time significantly, even for simple cases with $n_{\text{coils}} = 1$. For field devices with multiple independent currents, this effect will be even more significant, because for each independent current the perturbed problem has to be solved. However the time needed to compute the difference quotients could be reduced by parallelization, which is currently not implemented.

The biggest improvement could be made by eliminating the overhead GetDP produces in every timestep. For every invocation the problem file, the geometry, the mesh and the last solution have to be read from disc and parsed. Then the structures for the matrices and field vectors have to be allocated in memory (GetDP has a memory usage of approximately 150 MB for the test problem) and finally the resulting field has to be written to the resolution file.

6 Conclusion and Outlook

6.1 Outlook

While the software presented in the previous chapter can be used as it is, there are several possible enhancements that could be added to improve performance and add features. As shown in Section 5.6 simulating even simple problems consumes a lot of computation time. The biggest optimization potential is obviously in reducing the effort to solve the field problem, as well as the number of invocations of the field problem software. A significant amount of time is used because the finite element solver has to be started at every iteration of the Newton scheme in every timestep, which involves accessing the file system, allocating memory and generating matrices. Currently with ONELAB and GetDP it is not easily possible to keep GetDP running in the background and request new solutions with ONELAB commands. The number of needed field simulations depends on the convergence rate of the circuit simulator's Newton scheme, but as the average Newton iteration takes approximately two iterations (Fig. 5.7), the only way to significantly reduce the number of field simulations would be to move on to a weak coupling scheme by bypassing.

The concept of bypassing is based on the observation, that for many applications even for very fast changing excitations, the field distribution changes only very slowly. This motivates the introduction of a criterion — e.g. the change in energy in a field device — which has to reach a tolerance value before the field simulation is done again. In the meantime, it is assumed, that the field has not changed significantly and the coupling quantities remain constant. This approach could be implemented, for example based on the criterion used in [SBD12]. This would also significantly lower the impact of the slow convergence of the Newton scheme in the first timestep, as all but the first field simulation in this timestep would be bypassed because the current is not changing significantly.

The code written in this work only supports field devices with a single current but the coupling method is described for an arbitrary number of independent currents $\boldsymbol{i}_{\text{coil}}$. This allows simulation of complex devices like transformers, inductive charging application and electrical machines.

The implementation in Qucs can be done according to [JMHJ07, Chapter 6.3.3]. The development branch of qucs even supports mutual inductances with the `MUTX` device which allows a variable number of coils. This would even allow to create the circuits directly in qucs.

The effort for obtaining the differential inductances increases as n_{coils} linear systems have to be solved or difference quotients calculated to obtain them.

The coupling in this work always assumes magnetostatic field problems. This prevents the simulation of eddy currents, which are important in many field devices like transformers and electric machines. For the magnetoquasistatic field formulation a similar coupling stamp can be deduced. This leads to a complex impedance instead of the inductance found for the magnetostatic stamp. As the eddy currents depend on the derivative of the field quantities, the field simulation software has to be integrated into the timestepping scheme.

At the moment the coupling component is implemented as a dynamic module loaded by qucsator. To be able to edit circuits with Qucs, the coupling component is represented by an inductance with the specific name "onelab", which is replaced by the qucs-onelab script. If Qucs provided way to make dynamic components available in the user interface, the component could be provided there along with a ONELAB solver component to start ONELAB simulations directly from Qucs. Alternatively, this could be done by adding this functionality to the qucs source code.

6.2 Conclusion

The coupling method proposed in this work allows to do coupled simulations with very few changes to the existing simulation software and the simulation models. The only necessary steps to couple two existing models are to calculate the needed coupling quantities in the postprocessing of the field model and to rename the component in the circuit which should be replaced by the field model. Especially the choice of the field simulation software is not limited to GetDP, in general every software that can compute the quantities described in Section 3.4 can be employed.

While the method is working reliably, the inefficient invocation of the field simulation software evaluated in Section 5.6 make the simulation very time consuming.

List of Figures

Bibliography

[14] *Hyosung Product Catalog SPRiPM Super Premium Efficiency Motor*, 2014. [Online]. Available: `www.hyosungpni.com`.

[BJ09] M. E. Brinson and S. Jahn, "Qucs: A GPL software package for circuit simulation, compact device modelling and circuit macromodelling from DC to RF and beyond", *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 22, no. 4, pp. 297–319, Jul. 2009, ISSN: 08943370. DOI: `10.1002/jnm.702`.

[BSMM12] I. N. Bronštejn, K. A. Semendjaev, G. Musiol, and H. Mühlig, Eds., *Taschenbuch der Mathematik*, 8., vollst. überarb. Aufl. Frankfurt am Main: Deutsch, 2012, ISBN: 9783817120086.

[BV05] A. v. d. Bossche and V. Valchev, *Inductors and tranformers for power electronics*. Boca Raton: Taylor & Francis, 2005, ISBN: 978-1574446791.

[CRB75] Chung-Wen Ho, A. Ruehli, and P. Brennan, "The modified nodal approach to network analysis", *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, 1975, ISSN: 0098-4094. DOI: `10.1109/TCS.1975.1084079`.

[CSC+12] M. Clemens, S. Schöps, C. Cimala, N. Gödel, S. Runke, and D. Schmidthäusler, "Aspects of Coupled Problems in Computational Electromagnetics Formulations", *ICS Newsletter (International Compumag Society)*, vol. 3, 2012.

[EBH09] J. W. Eaton, D. Bateman, and S. Hauberg, *GNU Octave version 3.0.1 manual: A high-level interactive language for numerical computations*. CreateSpace Independent Publishing Platform, 2009, ISBN: 1441413006. [Online]. Available: `http://www.gnu.org/software/octave/doc/interpreter`.

[Fal13] C. d. Falco, *IFF version 0.1b1 File Format Specification*, 2013.

[FM11] H. Frohne and F. Moeller, Eds., *Moeller Grundlagen der Elektrotechnik: Mit 36 Tabellen und 182 Beispielen*, ger, 22., verb. Aufl, ser. Studium. Wiesbaden: Vieweg + Teubner, 2011, ISBN: 9783834808981 3834808989 9783834808981.

[FMM11] C. d. Falco, C. Massimiliano, and M. Merlin, *The 'ocs' Package*, 2011. [Online]. Available: `http://octave.sourceforge.net/ocs/`.

[Geu07] C. Geuzaine, "Getdp: A general finite-element solver for the de Rham complex", *Proceedings in Applied Mathematics and Mechanics*, vol. 7, no. 1, 2007, ISSN: 16177061. DOI: `10.1002/pamm.200700750`.

[Geu14] ——, *Open Numerical Engineering LABoratory ONELAB Rapport technique N° 3*, 2014. [Online]. Available: `http://onelab.info/onelab_wiki/images/a/a5/RapportTechnique3.pdf` (visited on 06/02/2015).

[JMHJ07] S. Jahn, M. Margraf, V. Habchi, and R. Jacob, *Qucs Technical Papers*, 2007. [Online]. Available: `http://qucs.sourceforge.net/docs/technical/technical.pdf` (visited on 04/08/2015).

[Nov13] C. Novak, *Using qucs in textmode*, 2013. [Online]. Available: `http://qucs.sourceforge.net/docs/tutorial/textmode.pdf`.

[Sal95] S. J. Salon, *Finite element analysis of electrical machines*, ser. The Kluver International Series in Engineering and Computer Science. Troy, New York: Kluwer Academic Publishers, 1995, ISBN: 0898382653.

[SBD12] S. Schöps, A. Bartel, and H. De Gersem, "Multirate Time Integration of Field/Circuit Coupled Problems by Schur Complements", in *Scientific Computing in Electrical Engineering SCEE 2010*, ser. Mathematics in Industry, Springer, 2012, pp. 243–251, ISBN: 978-3-642-22452-2.

[Sma13] E. Smailus, *Entwicklung einer eigensicheren Halbbrücke für Schutzkleinspannung zum modularen Aufbau von Laborversuchen*, 2013.