1 Magnetic Field Calculations

The magnetic field offers a variety of possibilities for designing sensor and actuator systems. Two possible sensor principles will be studied within this tutorial:

- Inductivity changes: *Inductivity L = f (Geometry)*

  Geometry caused inductivity changes can be used for positioning sensors.

- Induced voltage: *Voltage U = f (Geometry)*

  Sensors for measuring the rotational speed of a shaft often use the induced voltage in a sensor coil as detection principle.

Electromagnetic actuators use the mechanical force generated by the magnetic field to move an anchor of a magnetic valve or the membrane of a loudspeaker.

1.1 Finite Element Formulation

In the eddy current (= low-frequency) case, the governing equations for the magnetic field are given as

$$\begin{aligned}
\nabla \times \mathbf{H} &= \mathbf{J} \\
\nabla \times \mathbf{E}_{\mathrm{s}} &= -\frac{\partial \mathbf{B}}{\partial t} \\
\nabla \cdot \mathbf{B} &= 0 \\
\mathbf{J} &= \mathbf{J}_{\mathrm{i}} + \gamma(\mathbf{E}_{\mathrm{s}} + \mathbf{v} \times \mathbf{B}) \\
\mathbf{B} &= \mu \mathbf{H} \, .
\end{aligned}$$

The magnetic flux density $\mathbf{B}$ can be expressed using a vector potential $\mathbf{A}$

$$\mathbf{B} = \nabla \times \mathbf{A} \, .$$

By combining these equations, one gets the PDE for the magnetic field

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{A} = \mathbf{J} - \gamma \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t} \, ,$$

where

$$\frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t} = \frac{\partial \mathbf{A}}{\partial t} - (\mathbf{v} \times \nabla \times \mathbf{A})$$

is the total differential of A with respect to t. For the 2D plane case

$$\mathbf{B}(x,y) = \begin{pmatrix} B_x(x,y) \\ B_y(x,y) \\ 0 \end{pmatrix}, \quad \mathbf{A}(x,y) = \begin{pmatrix} 0 \\ 0 \\ A_z(x,y) \end{pmatrix} \tag{1.1}$$

this reduces to the following scalar equation

$$\nabla \times \frac{1}{\mu} \nabla \times A_z = J - \gamma \frac{\mathrm{d}A_z}{\mathrm{d}t} \,. \tag{1.2}$$

The corresponding weak formulation - where $\mathbf{A}'$ is the test function - is given as

$$\int_\Omega \gamma \mathbf{A}' \cdot \frac{\mathrm{d}\mathbf{A}}{\mathrm{d}t} \,\mathrm{d}\Omega + \int_\Omega \nabla \times \mathbf{A}' \cdot \frac{1}{\mu} \nabla \times \mathbf{A} \,\mathrm{d}\Omega = \int_\Omega \mathbf{A}' \cdot \mathbf{J} \,\mathrm{d}\Omega \,. \tag{1.3}$$

The leftmost term models the influence of eddy currents (only in transient / harmonic simulations), which are strongly determined by the electric conductivity γ. The term on the right hand side models the excitation by an electric current density $\mathbf{J}$, which is typically driven by a coil.

The resulting discrete algebraic system reads as

$$\mathbf{M}\underline{\dot{A}} + \mathbf{K}\underline{A} = \underline{f} \,, \tag{1.4}$$

where $\mathbf{M}$ is the magnetic mass matrix, $\mathbf{K}$ the stiffness matrix and $\underline{f}$ the right hand side excitation vector due to an impressed current.

1.2 General Simulation Requirements

To set up a magnetic simulation, you have to select `magnetic` in the `Physic Type Selection` section on the `Model -> Analysis` screen (see Figure 1.1).

The 2D plane simulation corresponds to a setup, where the coil has a much larger extension in the third dimension than compared to its width. Therefore one considers only the field in the x-y-cutplane (see Fig. 1.2).

By default a plane state calculation (if the mesh is 2D) or a standard 3D calculation (with a 3D mesh) is done. If an axisymmetric simulation has to be applied, the option `2D Axi` has to be selected on the `Global -> Mesh` screen.

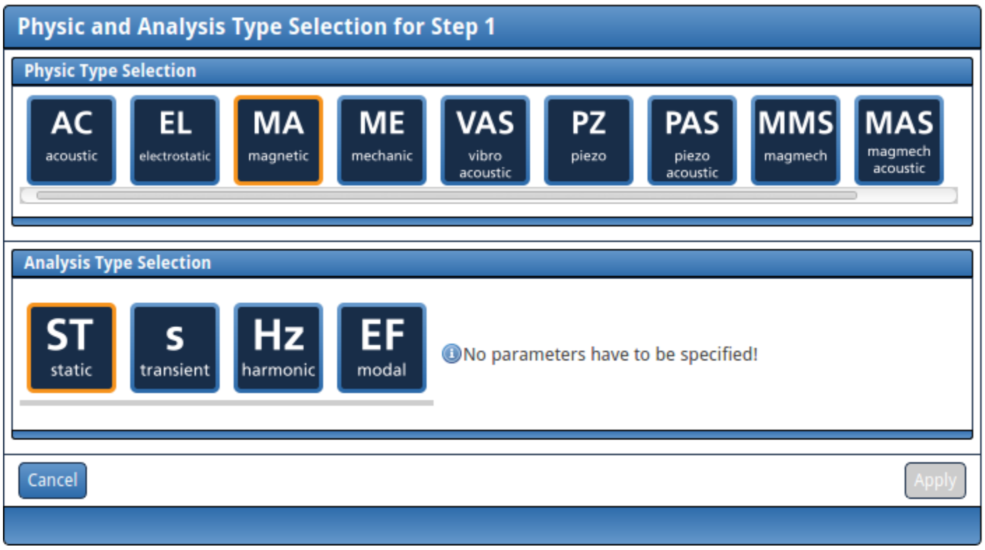

Figure 1.1: Setting up a magnetic analysis

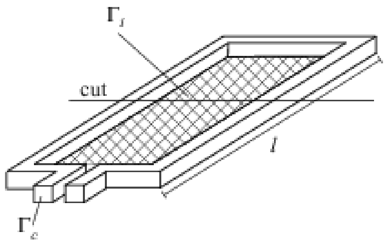

Figure 1.2: *Plane setup of a magnetic coil*

1.3 Degrees of Freedom and derived Quantities

1.3.1 Magnetic Vector Potential

The partial differential equation (PDE) which has to be solved in case of magnetic field calculations is expressed in terms of the magnetic vector potential $\mathbf{A}$. Due to the fact that the magnetic field is solenoidal, the magnetic induction $\mathbf{B}$ is related to the magnetic vector potential via the *curl*-operator

$$\mathbf{B} = \nabla \times \mathbf{A} \,.$$

In 2D- or axisymmetric calculations the magnetic vector potential exhibits only one non-vanishing entry A_z which is oriented perpendicular to the cutting area observed. It can be written out by adding `Mag -> VecPotential` as a result on the `Results -> Set Result` screen (see Figure 1.3).

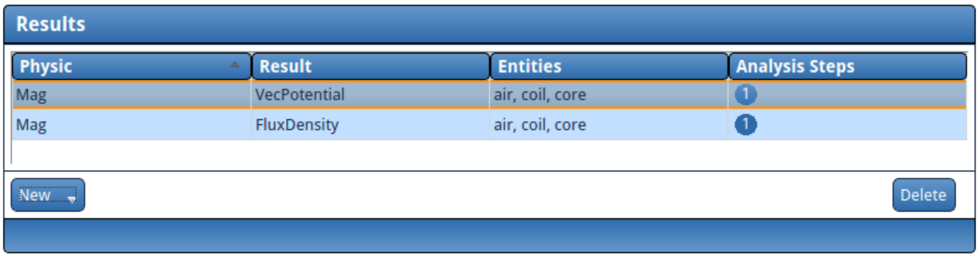

Figure 1.3: Adding the magnetic vector potential as a result.

1.3.2 Element Results

In the case of magnetic field calculation, many element results can be deduced by a postprocessing step:

- Magnetic induction (magnetic flux density) $\mathbf{B}$:
  Derived from the magnetic vector potential $\mathbf{A}$ by applying the curl operator.

  The magnetic flux density can be stored by adding `Mag -> FluxDensity` as a result.

- Eddy current density $\mathbf{J}_{\mathrm{E}}$ (only transient / harmonic case):
  Currents induced by a magnetic field in a conductive material, which can be calculated as

  $$\mathbf{J}_{\mathrm{E}} = -\gamma \frac{\partial \mathbf{A}}{\partial t} . \tag{1.5}$$

  The magnetic field produced by the eddy currents acts against the changes of the applied magnetic field with respect to time. In a 2D setup, the eddy current has just a z-component, like the magnetic vector potential. The eddy currents can be stored by adding `Mag -> EddyCurrentDensity` as a result.

  NOTE:

  As a rule of thumb, the penetration depth

  $$\delta = \frac{1}{\sqrt{\pi \gamma \mu f}} \tag{1.6}$$

  (see book eq. (4.51)) should be resolved by approx. 10 elements!

For the magnetic PDE the only available region result is the magnetic energy, which is a measure for the energy stored in the magnetic assembly, which is defined by

$$W_{mag} = \int_{\Omega} \frac{1}{2} \mathbf{BH} \, \mathrm{d}\Omega \, .$$

This can also be used for deriving the magnetic force F for actuator applications, acting on moveable parts. Note that the energy gets calculated per region, i.e. it is written to the `history` directory. The magnetic energy will be stored if the result `Mag -> Energy` is added (remember to set the checkbox `text` within `Monitor Output`). It can be also used do determine the inductance L of the setup using the relation

$$W_{mag} = \frac{1}{2} L i^2 \ ,$$

where i is the current flowing through the coil. A easier way to obtain the inductance of a setup is to store the inductance of a coil directly (see section **Coil Types**).

1.4 Boundary Conditions

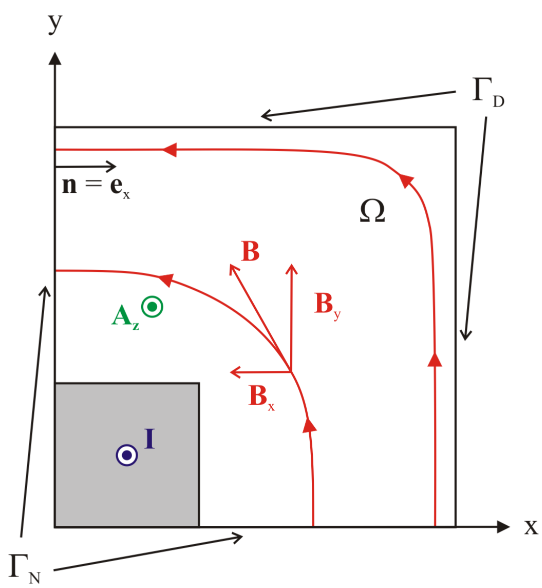

Figure 1.4: *Boundary conditions*

At the boundary of the calculation domain Γ_D, the tangential component of the magnetic vector potential $\mathbf{A}$ is set to zero in order to guarantee a solenoidal field. In the 2D plane as well as axisymmetric case this means to set the nonvanishing component of the magnetic vector potential to zero (Dirichlet boundary condition).

At boundaries where the tangential component of the magnetic induction $\mathbf{B}$ is zero (e.g. at symmetry edges Γ_N), we have to set the normal derivative of the magnetic vector potential to zero (Neumann boundary condition).

- Closed field lines (Γ_D):

$$\mathbf{B} \cdot \mathbf{n} = 0 \rightarrow \mathbf{A} = const.\ (\text{e.g.}\ \mathbf{A} = 0)$$

  Dirichlet boundary condition

- Symmetry boundary condition (Γ_N):

$$\mathbf{B} \times \mathbf{n} = 0 \rightarrow \frac{\partial \mathbf{A}}{\partial \mathbf{n}} = 0$$

  Homogeneous Neumann boundary condition

In general, according to the FE formulation, at all boundaries, where no Dirichlet boundary conditions are specified, the homogeneous Neumann boundary condition is implicitly taken into account. In NACS, the BC `FluxNormal` represents the *homogeneous Neumann BC* whereas `FluxParallel` will apply a *Dirichlet BC*.

1.5 Coil Modeling

For the generation of a magnetic field one uses either permanent magnets or coils. A coil in NACS is defined by a normal region, for which some source term (current, voltage) and additional coil properties can be specified.

Coils have to be defined as boundary conditions on the `Boundary Condition -> Set Condition` screen.

For every coil, a simplified model is used (see Fig. 1.5): Not every wire Γ_w is resolved individually. Instead, a uniform current density distribution is assumed. Therefore, the region filled by the coil is modeled as a single area (region) Γ_s.

The number of windings N results from the cross section of the wire. It is calculated by the overall cross section of the coil $\Gamma_S = b \cdot h$ divided by the cross section of one wire Γ_W

$$N = \frac{\Gamma_S}{\Gamma_W}\ . \tag{1.7}$$

Here we assume, that the windings are very compact and that we can neglect the space between the single wires.

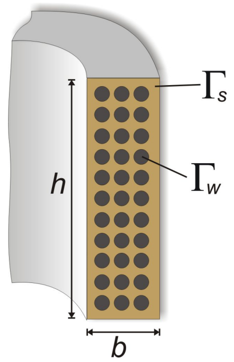

Figure 1.5: *Solid model of coil*

1.5.1 Coil Types

In our simulation, we distinguish different coil types:

- Current / Voltage Driven Coils and
- Measurement Coils.

Each coil has to be given a unique `Name` and an `Orientation` (the direction in which the current density will be treated as positive). It can consist of serveral parts, e.g. the coil in Figure 1.8 would consist of the parts `coil_1a` and `coil_1b`.

Every coil has an individual inductance referring to the setup. This inductance can be stored by adding `Mag -> CoilInductance` as a result for the respective coil on the `Results -> SetResult` screen and activating the checkbox `text` in the `Monitor Output` section. The inductance can be saved for every type of coil, both measurement and current/voltage driven coils.

Depending on the coil type, several additional settings are needed. They are described in the next sections.

1.5.2 Current and Voltage Driven Coils

For this coil type, we either have to apply a current directly or we can apply a voltage and set the overall resistance of the coil. Additionally, either the number of turns has to be set in the `Turns` entry or the cross section of one turn has to be given as `Wire CrossSection`. The `Fill Factor` denotes how dense the coil is filled with single conducting turns, i.e. if the fill factor is 1 then NACS assumes that $100\,\%$ of the coil consists of turns whereas a fill factor of 0.8 means that there is $20\,\%$ empty space present within the coil.

If the analysis is static or transient, the `phase` attribute is neglected. In case of a transient simulation one can also specify time-dependent expressions for the current of the coils (see Figure 1.6).

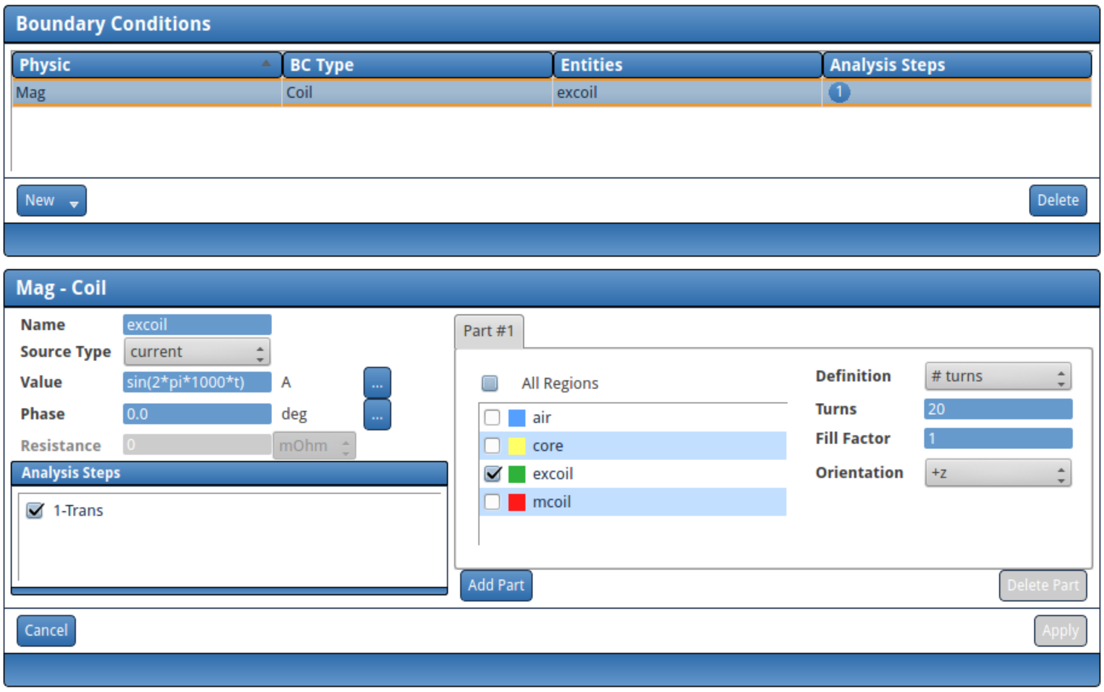

Figure 1.6: Applying a time-dependent sinusoidal current to a coil.

> NOTE:
>
> For the material defining the coil, we can always choose air as it has the same permeability as copper $\mu = \mu_0$ and we do not want to have eddy currents within the conductors, since the current density is fixed to the given value!

1.5.3 Measurement coils

Measurement coils are used a obtain induced currents, voltages, and the inductance of a setup. To use a measurement coil, it has to be defined in the BC section first. Similar to excitation coils, we have to set the orientation of the measurement coil, the number of turns/cross section and it can consist of several parts.

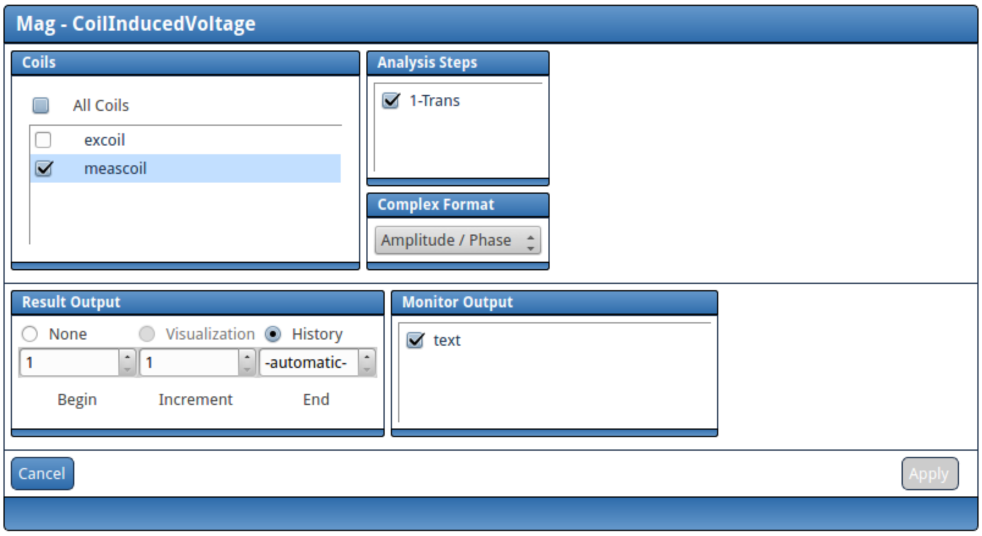

Figure 1.7: Obtaining the induced voltage of a measurement coil.

Since we are interested in the currents/voltages that result from an external magnetic field, we do not apply any excitation (and therefore set the value of the current/voltage for the measuremt coil to 0).

Depending on what quantity we want to measure, we have to set up the measurement coil accordingly:

- **Induced Voltage:** Set the measurement coil to *current driven* and apply 0 A.
- **Current:** Set the measurement coil to *voltage driven*, apply 0 V and provide the overall resistance of the coil in Ω.

Figure 1.7 shows a measurement coil that is used to obtain the induced voltage of a setup. In order to store quantities for measurement coils, they have to be added to the results by navigating to the Results -> Store Result screen and clicking on New -> Mag -> <result> where <result> can be e.g. CoilInducedVoltage , CoilCurrent , etc. .

1.5.4 Coil Definition Examples

The full description of a coil setup as in Fig. 1.8 with with a winding cross section of $1 \times 10^{-3}\,\mathrm{m}^2$, a harmonic excitation of 10 A (frequency 1000 Hz) could be given as seen in Figure 1.9.

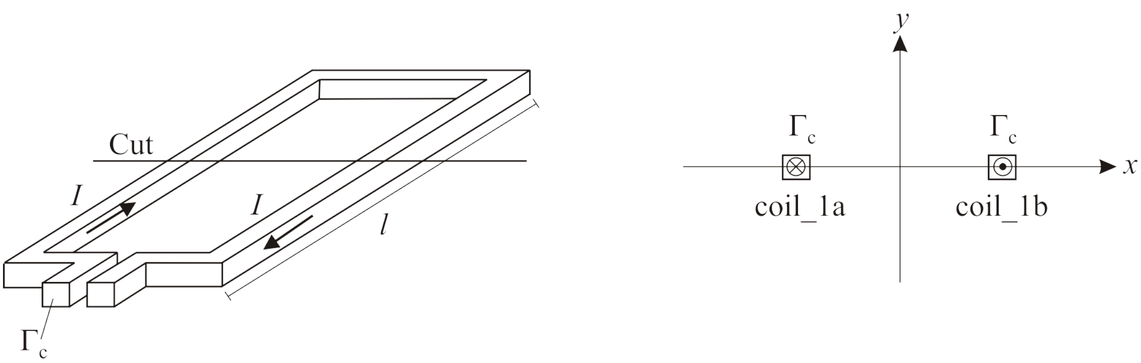

Figure 1.8: *Real coil setup (left) and 2D cross section model (right)*

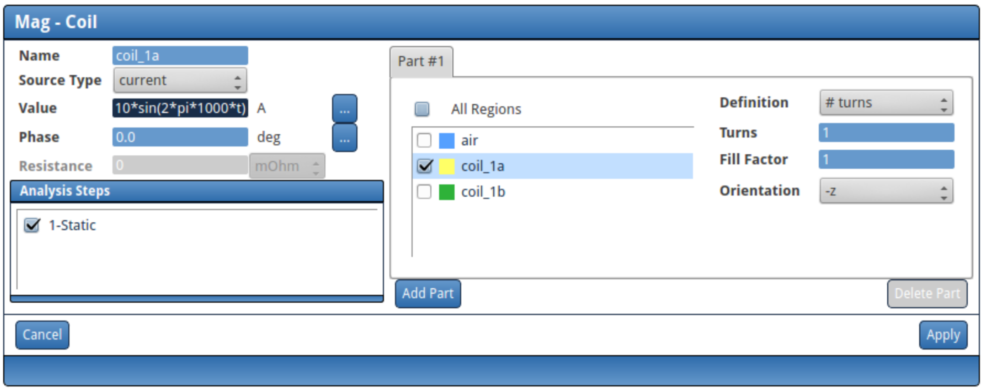

Figure 1.9: Applying a signal to a coil.

1.6 Permanent Magnets

Considering the constitutive equation for the magnetic flux density B,

$$\mathbf{B} = \mu_0 \mathbf{H} + \mathbf{M} \,, \tag{1.8}$$

a permanent magnet is defined by its remanent magnetization M (T).

A permanent magnet can be modeled with a constant magnetic flux density B. This can be done by adding the BC `Mag -> FluxDensity` to the region that represents the permanent magnet. The remanent magnetization is material dependent and usually delivered with the datasheet for the permanent magnet.

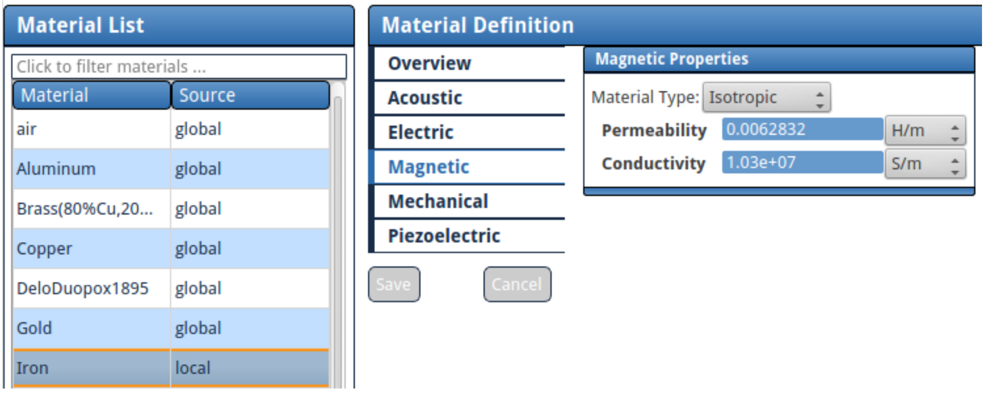

Figure 1.10: Defining the material properties for magnetic simulations.

1.7 Material Parameters

The two major material parameters for magnetic field calculations are the permeability μ and the electric conductance γ. The total permeability μ can be decomposed into μ_0 the total permeability of air and the relative permeability μ_r:

$$\begin{aligned} \mu &= \mu_r \mu_0, \\ \mu_0 &= 4\pi \cdot 10^{-7}\,\mathrm{Vs/Am}. \end{aligned}$$

Within the NACS material editor, we always specify the **total** permeability μ. When performing eddy current computations, we additionally have to specify the electric conductance γ in $(\mathrm{m}^{-1}\Omega^{-1})$ for each material (see Figure 1.10).

1.8 Magnetic Field Example

This example shows the simulation of the electromagnetic field of a cylindrical coil, including an iron core and the surrounding air. The model has the following dimensions:

- 100e-3 m length of coil,
- 15e-3 m inner radius of coil,
- 5e-3 m thickness of coil,
- 100e-3 m distance to boundary.

The coil consists of 20 windings and has a constant impressed current of 1 A.

1.8.1 Magnetic Field Example: Ansys Input File

This Ansys script creates the mesh for the setup.

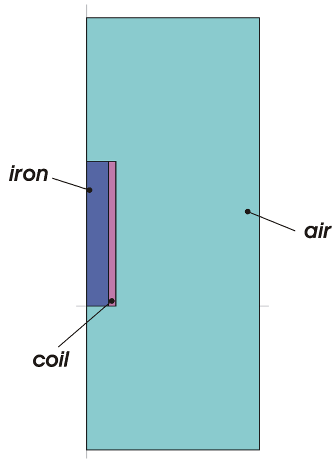

Figure 1.11: Setup of cylindrical coil

```
! ---- Mesh script for exercise manual example 1: magnetic coil ----

! ----------------------- !
! Initialize ANSYS:
! ----------------------- !
FINI
/CLEAR
/PREP7

nacsinit

! ----------------------- !
! User Defined Parameters:
! ----------------------- !
r_inner = 15e-3                         ! radius of the iron core
r_coil = 5e-3                           ! radius of the coil
l_coil = 100e-3                         ! length of the coil
bnd_air = 100e-3                        ! width of the surrounding air

dx = 1e-3                                       ! used mesh size
tol = 1e-6                                      ! selection tolerance

! type of elements ('' for 1st order, 'quad' for 2nd order)
elemMode= ''

! =============================================================== !
! Preprocessing
! =============================================================== !

! create helper variables for easy selection
core_x1 = 0
core_x2 = core_x1 + r_inner
core_y1 = 0
core_y2 = core_y1 + l_coil
coil_x1 = core_x2
coil_x2 = coil_x1 + r_coil
coil_y1 = 0
coil_y2 = coil_y1 + l_coil
! air at the top
air1_x1 = core_x1
air1_x2 = coil_x2 + bnd_air
```

```
air1_y1 = coil_y2
air1_y2 = air1_y1 + bnd_air
! air at the bottom
air2_x1 = air1_x1
air2_x2 = air1_x2
air2_y1 = core_y1 - bnd_air
air2_y2 = core_y1
! air at the right side
air3_x1 = coil_x2
air3_x2 = air3_x1 + bnd_air
air3_y1 = coil_y1
air3_y2 = coil_y2

! ---------------------- !
! Create Geometry:
! ---------------------- !

! core and coil
rectng,core_x1,core_x2,core_y1,core_y2
rectng,coil_x1,coil_x2,coil_y1,coil_y2
! air
rectng,air1_x1,air1_x2,air1_y1,air1_y2
rectng,air2_x1,air2_x2,air2_y1,air2_y2
rectng,air3_x1,air3_x2,air3_y1,air3_y2

! connect areas
asel,all
aglue,all

! core
asel,s,loc,x,core_x1,core_x2
asel,r,loc,y,core_y1,core_y2
cm,core,area

! coil
asel,s,loc,x,coil_x1,coil_x2
asel,r,loc,y,coil_y1,coil_y2
cm,coil,area

! air
asel,all                                        ! select everything
asel,u,,,core                           ! deselect everthing that is not air
asel,u,,,coil
cm,air,area

! define outer boundaries
lsel,s,loc,x,core_x1,core_x1
lsel,r,loc,y,air2_y1,air1_y2
cm,outerbounds_left,line

lsel,s,loc,x,air3_x2,air3_x2
lsel,r,loc,y,air2_y1,air1_y2
cm,outerbounds_right,line

lsel,s,loc,x,core_x1,air3_x2
lsel,r,loc,y,air1_y2,air1_y2
cm,outerbounds_top,line

lsel,s,loc,x,core_x1,air3_x2
lsel,r,loc,y,air2_y1,air2_y1
cm,outerbounds_bot,line

! ---------------------- !
! Create Mesh:
! ---------------------- !
```

```
! assign line size
lsel,all
lesize,all,dx

! area mesh
setelems,'quadr',elemMode
asel,all
amesh,all

! ---------------------- !
! --> Regions
! ---------------------- !
region,'core','core'
region,'coil','coil'
region,'air','air'
group,'outerbounds_left','outerbounds_left'
group,'outerbounds_right','outerbounds_right'
group,'outerbounds_top','outerbounds_top'
group,'outerbounds_bot','outerbounds_bot'

!*set,nacs_dbg__,1
! ---------------------- !
! Write NACS Files:
! ---------------------- !
writemodel,'coil_example'
```

1.8.2 Magnetic Field Example: Python script

After setting up the simulation using the NACS-GUI the Python file that contains the parameters of the magnetical simulation will look like this:

```
# -*- coding: utf-8 -*-
"""
-------------------------------------
 N A C S   P Y T H O N   S C R I P T
-------------------------------------

NACS version: 2.0.2745 - pre3
NACS architecture: CENTOS 5.11 (X86_64)
File generated at Wed Nov 19 14:24:24 2014
On host 'lse83' by 'fschieber'
"""

from __future__ import division
try:
  from nacs.scripting import *
except:
  raise Exception("File is only executable in the NACS python interpreter!")

# =================
#  NACS SIMULATION
# =================
simulation = NacsSimulation()
simulation.setGrid(u'coil_example.nmf', 'axi')

simulation.addOutput(Output.Nacs())
text = Output.Text()
simulation.addOutput(text)
```

```
simulation.addOutput(Output.GiD())


# =====================
#   MATERIAL DEFINITION
# =====================
air = Material('air')
air.density(1.205)
air.permeability.isotropic(1.25664e-06)
air.conductivity.isotropic(0.0)
air.compressionModulus(141650)

iron = Material('Iron')
iron.density(7100.0)
iron.lossTangensDelta([1000],[0.0003])
iron.stiffness.isotropic.byENu(2.06e+11, 0.28)
iron.permeability.isotropic(0.0062832)
iron.conductivity.isotropic(0.0)


simulation.setMat('air', air)
simulation.setMat('coil', air)
simulation.setMat('core', iron)

# ===============
#   ANALYSIS STEP
# ===============
static1 = Analysis.Static()

mag1 = Physic.Magnetic('node')
mag1.addRegions(['coil', 'air', 'core'])
mag1.addBc(mag1.BC.FluxParallel(['outerbounds_bot', 'outerbounds_left', 'outerbounds_right
    ', 'outerbounds_top']))
excitation_coil = mag1.BC.Coil('excitation_coil', "1")
excitation_coil.addPart(excitation_coil.Part('coil', 20))
mag1.addBc(excitation_coil)
mag1.addResult(mag1.Result.FluxDensity(['air', 'coil', 'core']))
mag1.addResult(mag1.Result.Energy(['air', 'coil', 'core'], 'amplPhase', 'history', [text])
    )
mag1.addResult(mag1.Result.VecPotential(['air', 'coil', 'core']))
mag1.addResult(mag1.Result.CoilInductance([excitation_coil], 'amplPhase', 'history', [text
    ]))
static1.addPhysic(mag1)

simulation.addAnalysis(static1)
```