Ausarbeitung Übung 10

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg
Datum: 10. Februar 2021

Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Ausarbeitung Übung 10

Studienarbeit von Dominik Schiller, Constanze Kramer, Simon Arnold & Tobias Lingenberg

Datum: 10. Februar 2021

Darmstadt

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit dem Übungsblatt 10 des Faches „Einführung in die numerische Berechnung elektromagnetischer Felder“. Ein Transformator wird zunächst mit linearer Reluktivität und im Anschluss mit nicht-linearer Reluktivität simuliert.

2 Ausarbeitung der Aufgaben

2.1 Impliziter Euler, Octave

Betrachtet wird ein Transformator, bestehend aus zwei Spulen die über einen Eisenkern miteinander verbunden sind. Der anregende Strom $\mathbf{i}(t) = 10[\sin(2\pi f t), 0]^T \mathrm{A}$ hat eine Frequenz von $f = 50\,\mathrm{Hz}$. Das semidiskrete magnetoquasistatische Problem

$$\mathbf{M}_\sigma \dot{\mathbf{a}} + \tilde{\mathbf{C}}\mathbf{M}_\nu \mathbf{C}\mathbf{a} = \mathbf{X}\mathbf{i}(t) \tag{2.1}$$

soll im Zeitbereich $T = [0, 0.02]\mathrm{s}$ gelöst werden. Die Ortsdiskretisierung wird hierbei als schon durchgeführt vorrausgesetzt, deshalb wird im Folgenden nicht darauf eingegangen.

Die Octave-Routine `trafo_linear`(siehe Anhang 2.3) erstellt zu Beginn die Materialmatrix $\mathbf{M}_\sigma$ und entfernt alle überflüssigen Einträge. Mit dem Aufruf der Funktion `fit_operator`(siehe Anhang 2.3) wird die Rotationsmatrix $\mathbf{C}$ konstruiert. Mit Hilfe dieser Matrix kann nun die Rotations-Rotations-Matrix $\mathbf{K} = \tilde{\mathbf{C}}\mathbf{M}_\nu\mathbf{C}$ berechnet werden, wobei $\mathbf{M}_\nu$ die Reluktivitätsmatrix in Diagonalform darstellt. In diesem Fall wird noch ein linearer Verlauf der Reluktivität angenommen, das heißt die Reluktivität ist an jedem Punkt im Eisenkern zu jeder Zeit gleich.

Die Gleichung (2.1) wird anschließend mit dem impliziten Eulerverfahren für jeden Zeitschritt im Zeitbereich gelöst.

Im Postprocessing wird ein Linienplot der Anregung erstellt(siehe Abbildung 2.1). Außerdem wird für jeden Zeitschritt eine VTK-Datei erstellt zur weiteren Visualisierung des entstehenden Magnetfelds in Paraview.

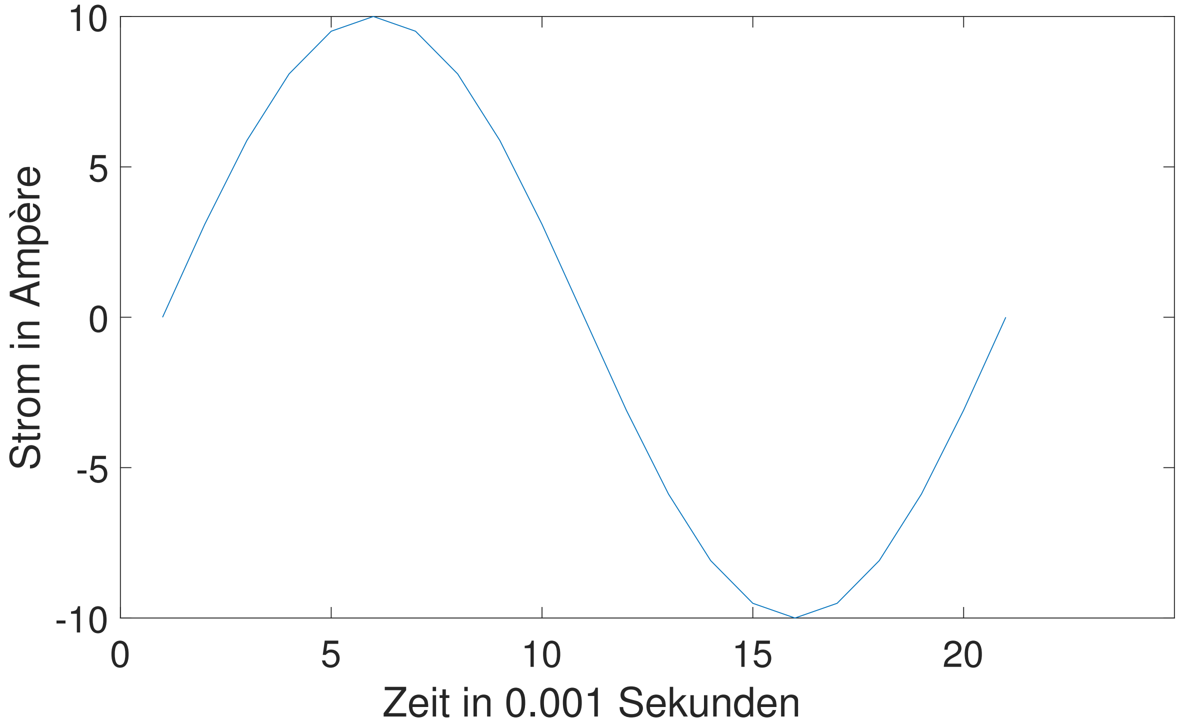

Abbildung 2.1: Anregungsstrom $i(t) = 10\sin(2\pi f t)$

2.2 Nichtlineare Permeabilität

Im vorherigen Abschnitt wurde beschrieben, wie man die magnetische Flussdichte innerhalb eines Transformators mit linearem Material simulieren kann. Nun soll im folgenden der gleiche Prozess durchgeführt werden, dieses mal mit einer nicht linearen Kennlinie der Permeabilität. Die Permeabilität hängt hierbei von der magnetischen Flussdichte B ab, da diese zu Beginn der Berechnung nicht bekannt ist muss ein iteratives Verfahren verwendet werden. Die Kennlinie wird durch

$$\nu(B) = k_1^{k_2|B|^2} + k_3 \text{ mit } |B| = \sqrt{B_x^2 + B_y^2 + B_z^2} \tag{2.2}$$

beschrieben, wobei $k_1 = 0,3374$, $k_2 = 2,970$ und $k_3 = 388.33$ gilt.

Um das entstehende Gleichungssystem der Form $\mathbf{A}(\mathbf{x})\mathbf{x} = \mathbf{b}$ zu lösen wird die Fixpunkt-Iteration verwendet. Man nimmt den Ansatz

$$\mathbf{A}(\mathbf{x}^{(i)})\mathbf{x}^{(i+1)} = \mathbf{b}, \tag{2.3}$$

wobei die $\mathbf{x}^i$ die Lösung im Schritt i und $\mathbf{x}^{i+1}$ die Lösung im Schritt $i+1$ beschreibt. Die neue Lösung wird unter Zuhilfenahme der alten Lösung berechnet. Dieses Verfahren ist in dem Skript `trafo_nichtlinear`, welches im Anhang zu finden ist, mit Hilfe einer while-Schleife implementiert. Die Iteration bricht ab, sobald

$$\frac{||\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}||}{||\mathbf{x}^{(i+1)}||} < 10^{-6} \tag{2.4}$$

erfüllt ist. Zunächst wird $\mathbf{x} = \mathbf{0}$ gewählt, um sicherzustellen, die erste Bedingung der Schleife `i < 2` garantiert, dass mindestens ein „altes“ und ein „neues“ berechnet werden. $\mathbf{A}$ ist in dem zu untersuchenden magnetostatischen Problem durch $\mathbf{K} = \mathbf{C}'\mathbf{M}_\nu\mathbf{C}$ gegeben. Die linke Seite $\mathbf{b}$ des Gleichungssystems ist der Anregungsstromvektor $\mathbf{j}$. Die Unbekannte $\mathbf{x}$ stellt das zu berechnende magnetische Vektorpotenzial $\mathbf{a}$ dar. Zusätzlich dazu sind Randbedingungen festgelegt.
Bei sehr kleinem Anregungsstrom lässt sich kein magnetisches Feld berechnen, je größer der Anregungsstrom wird, desto mehr Iterationsschritte werden benötigt und dementsprechend dauert die Berechnung länger. Exemplarisch ist das magnetische Feld innerhalb eines Transformators mit einem Anregungsstrom der Spulen von $10\,\mathrm{A}$ in der Abbildung 2.2 zu sehen.

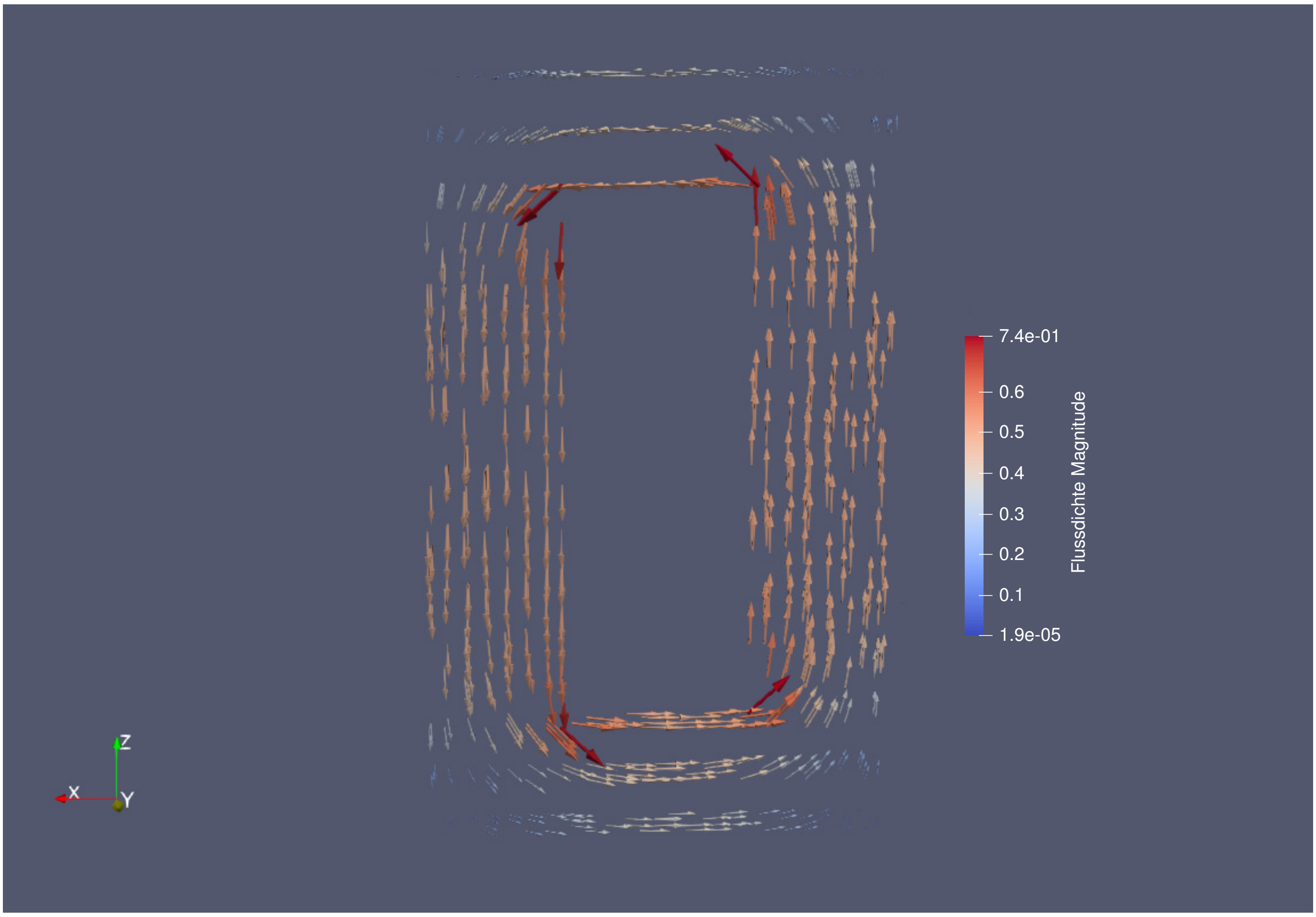

Abbildung 2.2: Das im Transformator entstehende Magnetfeld mit einem Anregungsstrom von $10\,\mathrm{A}$

2.3 Anhang

```
function [C,S,Ss] = fit_operator(Nx,Ny,Nz)
Mx = 1;
My = Nx;
Mz = Ny * Nx;
Np = Nx * Ny * Nz;

Px = sparse(Np,Np);
Py = sparse(Np,Np);
Pz = sparse(Np,Np);

for p = 1 : Np
    for q = 1 : Np
        if (q==p)
            Px(p,q) = -1;
            Py(p,q) = -1;
            Pz(p,q) = -1;
        end
         if (q==p+Mx)
             Px(p,q) = 1;
         end
         if (q==p+My)
             Py(p,q) = 1;
         end
         if (q==p+Mz)
             Pz(p,q) = 1;
         end
    end
end

C = [sparse(Np,Np) -Pz Py; Pz sparse(Np,Np) -Px; -Py Px sparse(Np,Np)];
C = sparse(C);
S = [Px Py Pz];
S = sparse(S);
Ss = [-Px' -Py' -Pz'];
Ss = sparse(Ss);
```

data/fit_operator.m

```
% get data
load trafo_linear.mat;

% get mesh data
nx = length(prb.xmesh);
ny = length(prb.ymesh);
nz = length(prb.zmesh);
np = nx*ny*nz;

% get and shrink conductivity matrix
Msigma = prb.Msigma;
Msigma = Msigma(prb.idxdof,prb.idxdof);
M = Msigma;

% ---- do something here ----
% construct curl-matrix
C = fit_operator(nx,ny,nz);
```

```
% ---- do something here ----

% create and shrink curl-curl matrix
K = C'*spdiags(prb.Nu,0,3*np,3*np)*C;
K = K(prb.idxdof,prb.idxdof);

% get and shrink excitation matrix
X = prb.Qstr;
X = X(prb.idxdof,:);

% time parameters
T0    = 0.00;
Tend  = 0.02;
dt    = 1e-3; % try dt=1e-4 if your computer is fast....

% discrete time vector
T    = T0:dt:Tend;

% initialize solution
f     = 50;
a     = zeros(size(M,1),length(T));
i     = @(t) (10*[sin(2*pi*f*t);0*t]);

% construct preconditioner for pcg
% use "pcg(A,b,1e-6,1000,Z,Z)" to solve
% the linear problem Ax=b for x
Z = sparse(1:size(a,1),1:size(a,1),sqrt(diag(M/dt+K)));

% time loop with equidistant time steps
for jj=2:length(T)

  % ---- do something here ----
  % solve M*a'+K*a=X*i with the implicit Euler method
  h = (1/0.001);
  A = h*M+K;
  b = X*i(T(jj)) + h*M*a(:,jj-1);
  %x = a(:,jj);
  a(:,jj) = pcg(A,b,1e-6,1000,Z,Z);
  %a(:,jj)=a(:,jj);
  % ---- do something here ----

end


% Postprocessing 1: plot the current excitation i(t)
% ---- do something here ----
plot(i(T(:)));
 % ---- do something here ----

 % Postprocessing 2: return the fluxes B!
 A = zeros(length(prb.ds),1);
 B = zeros(length(prb.ds),1);
 for jj=1:length(T)
   A(prb.idxdof) = a(:,jj);

   % ---- do something here ----
   % compute the facet integrated fluxes b
   B = C*A;
```

```
  % ----- do something here -----

  % scaling and write to file
  A(prb.idxs) = A(prb.idxs)./prb.ds(prb.idxs);
  B(prb.idxA) = B(prb.idxA)./prb.dA(prb.idxA);
   fit_write_vtk (prb.xmesh,prb.ymesh,prb.zmesh,['solution_' num2str(jj,'%03d') '.vtr'],...
                 {'MVP',A;'Flux',B},{'Region',prb.Elem(prb.idxV);});
 end
% visualize the result in Paraview, e.g. onprb a cutplane
```

data/trafo_linear.m

```
% lese Eingabedaten
load trafo_nichtlinear.mat;

% Gittertopologie
xmesh = prb.xmesh;
ymesh = prb.ymesh;
zmesh = prb.zmesh;
nx = length(xmesh);
ny = length(ymesh);
nz = length(zmesh);
Np = nx*ny*nz;
material = prb.Elem;

% Gitterabmessungen und Indizes
ddA     = prb.ddA;           % duale Flaecheninhalte
dA      = prb.dA;            % primale Flaecheninhalte
ds      = prb.ds;            % primale Kantenlaengen
idxV    = prb.idxV;          % Indizes der primalen Volumina im Rechengebiet
idxA    = prb.idxA;          % Indizes der primalen Flaechen im Rechengebiet
idxs    = prb.idxs;          % Indizes der primalen Kanten im Rechengebiet
idxdof  = prb.idxdof;        % Indizes der Freiheitsgrade
idxiron = find(material==6); % Indizes fuer Eisen

% Operatoren
C = prb.C;        % Curl-Matrix

% Anregungsvektor mit 10A Strom pro Spule
jsrc  = prb.Qstr*[10;10];
jdof  = jsrc(idxdof);

% Reluktivitaet
mu0 = 4*pi*10^-7;
murE = 1000;
nu  = ones(Np,1)/mu0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Loese lineares Problem
nu(idxiron) = 1/(mu0*murE);
Mnu           = createMny(prb.xmesh',prb.ymesh',prb.zmesh',nu);
K             = C'*Mnu*C;
Kdof          = K(idxdof(:),idxdof(:));
Kdof = sparse(Kdof);
Zdof          = sparse(1:size(Kdof,1),1:size(Kdof,2),sqrt(diag(Kdof)));
```

```
a            = zeros(3*Np,1);
a(idxdof)    = pcg(Kdof,jdof,1e-7,1000,Zdof,Zdof);

% berechne magnetische Flusse im Volumen
b            = C*a;
B            = zeros(3*Np,1);
B(idxA)      = b(idxA)./dA(idxA);
Bc           = fit_pf2pc(xmesh,ymesh,zmesh,B,idxV);

% Ausgabe
fit_write_vtk(xmesh,ymesh,zmesh,['trafo_iter' num2str(0) '.vtr'],{},{'Flussdichte',Bc(idxV,:);
    'Material',material(idxV)})

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Nichtlinear
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Parameter fuer Brauer's Kurve
 k1  = 0.3774;
 k2  = 2.970;
 k3  = 388.33;

 % Startwerte fuer die Fixpunktiteration
 aold = zeros(3*Np,1);
 a    = zeros(3*Np,1);
 i    = 0;

 % Fixpunkt iteration
 while (i<2) || norm(a-aold)/norm(a) > 1e-6

  % update der Variablen
  aold=a;
  i=i+1;
  fprintf('Iteration %d\n',i);

  % berechne magnetische Flusse im Volumen
  b         = C*a;
  B         = zeros(3*Np,1);
  B(idxA)   = b(idxA)./dA(idxA);
  Bc        = fit_pf2pc(xmesh,ymesh,zmesh,B,idxV);

  % Auswertung der Materialkurve
  Biron     = sqrt(sum( Bc(idxiron,:).^2, 2) );
  nu(idxiron) = fit_calc_nu(Biron,k1,k2,k3);

  % Update der Materialmatrix
  Mnu = createMny(xmesh',ymesh',zmesh',nu);

  % create and shrink curl-curl matrix
  K = C'*Mnu*C;
  Kdof = K(idxdof(:),idxdof(:));

  % loese GLS
  Zdof = sparse(1:size(Kdof,1),1:size(Kdof,2),sqrt(diag(Kdof)));
  a(idxdof) = pcg(Kdof,jdof,1e-7,1000,Zdof,Zdof,a(idxdof));

  % Ausgabe
  fit_write_vtk(xmesh,ymesh,zmesh,['trafo_ite2rrrrr' num2str(i) '.vtr'],{},{'Flussdichte',Bc(
```

```
        idxV,:);'Material',material(idxV)})

end
```
data/trafo_nichtlinear.m

Abbildungsverzeichnis