Optimization of Superconducting Magnetic Bearings using Finite Element Modeling

by

Jason Bryslawskyj

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author . . . . . .
Department of Physics
May 8, 2009

Certified by . . . . . .
Ulrich Becker
Professor, Department of Physics
Thesis Supervisor

Accepted by . . . . . .
David E. Pritchard
Senior Thesis Coordinator, Department of Physics

Optimization of Superconducting Magnetic Bearings using Finite Element Modeling

by

Jason Bryslawskyj

Submitted to the Department of Physics
on May 8, 2009, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Physics

Abstract

This project investigated the possibility of using superconducting bearings in large (3 - 100 MW) electric drives. Superconducting bearings are used to levitate the rotors inside electric drives via the Meißner effect, whereby superconductors tend to repel magnetic flux. The Finite Element Method was used to model superconducting bearings and optimize their dimensions. Computer simulations were written to simulate the superconducting state as well as perform the optimization. Not only the effect of changing the dimensions of the bearings was explored, but also how effects specific to type II superconductors -such as the partial penetration of magnetic flux- could be used to improve bearing design were considered. Ultimately, a superconducting magnetic bearing with a carrying force of 3210 N was improved to obtain a carrying force of 5200 N.

Thesis Supervisor: Ulrich Becker
Title: Professor, Department of Physics

Acknowledgments

I would like to acknowledge Dr. Matthias Lang and the development group at Siemens Dynamowerk in Berlin for their assistance and encouragement with this project. I would also like to thank Dr. Ulrich Becker at MIT for his guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As proven by S. Earnshaw in 1839, a mechanically stabile arrangement of magnetic dipoles cannot exist.[4] In 1939, Braunbek proved that this also applied to systems containing paramagnetic materials.[2] He subsequently showed that a stabile arrangement could indeed be found if a perfect diamagnetic material was introduced.[1] As superconducting material, which is diamagnetic, became readily available the real physical instantiation of this arrangement came to being. Braunbek's results have found application in the construction of magnetic bearings. As diamagnets, superconductors are used in magnetic bearings to levitate rotors via their ability to create forces by expelling external magnetic flux. This expulsion of flux creates a levitating force.

Superconducting magnetic bearings have been built for small motors, providing rotor carrying forces of up to 5 kN. During this project, first a computer simulation of the superconducting state as well as computer models of superconductors which have already been built were created. Scripts were subsequently written to optimize the dimensions of the permanent magnets and bulk superconductors. The goal was to achieve a bearing which could support a 3- 100 MW motor within the space of the actual active magnetic bearings used in these types of motors. For this the bearing would need a total carrying force of at least 8.5 kN and a radial stiffness of 4 kN/mm. As will be shown, by creating a novel arrangement of bulk super conductors, a bearing with a carrying force of 5 kN and a stiffness of 4 kN/mm was achieved.

Chapter 2

Basic Superconducting Phenomenology

Superconductors are materials, which exhibit zero electrical resistivity when they are below certain critical Temperatures T_c, Magnetic Fields H_c and Current densities J_c. They also exhibit the Meißner effect, in which the superconductor tends to expel all magnetic flux, even if there was an external field present when it was cooled. This differs significantly from what one would predict from an ideal perfect conductor. As seen on the left side of figure 2-1, an ideal perfect conductor would absorb the external magnetic field if cooled (transformed from normal conducting to perfect conducting) in its presence. The superconductor, however, exhibits the Meißner effect and tends to exclude any magnetic field, even one present during its cooling. The Meißner effect is shown on the left in figure 2-1.

2.1 Properties of Type I Superconductors

Superconductors are split according to their properties into type I and type II. Some typical type I superconductors are listed in table 2.1. They are characterized by their ability to expel all magnetic flux when simultaneously under the critical temperature T_c, magnetic field H_c and current density J_c.

In 1961 London, gave a macroscopic theory of superconductivity based solely

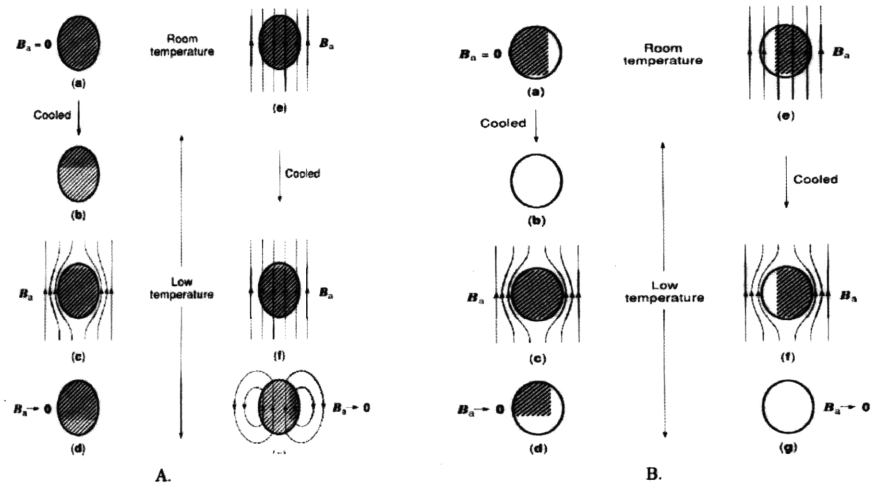

Figure 2-1: A. Hypothetical diagram of an conductor becoming perfectly ideal below a critical temperature. B. Meißner Effect: Flux is expelled from a superconductor when under critcal temperature T_c. Images from [5].

on the fact that type I superconductors expel all magnetic flux. He proposed that the same effect would be observed if the electrons in any conductor were accelerated without damping giving the following relation for the current density of of n electrons, where m is the mass of the electron and e the elementary charge: [6]

$$\frac{\partial \vec{J}}{\partial t} = \frac{ne^2}{m}\vec{E} \tag{2.1}$$

Applying Ampere's law, ignoring $\partial \vec{D}/\partial t$ for slowly varying currents, we find, where μ is the magnetic permeability dependent on material:

$$\frac{\partial}{\partial t}\nabla \times \nabla \times \frac{\vec{B}}{\mu} = \frac{4\pi ne^2}{mc}\nabla \times \vec{E} \tag{2.2}$$

Again applying Ampere's law and integrating with respect to time we find:

$$\nabla \times \nabla \times \frac{\vec{B}}{\mu} = -\frac{4\pi ne^2}{mc^2}(\vec{B} - \vec{B_0}) \tag{2.3}$$

where $\vec{B_0}$ is a constant of integration. Setting $\vec{B_0} = 0$, corresponding to a complete

Table 2.1: Typical type I superconducting materials and their critical temperatures and magnetic fields [5].

Metal:	Nb	Pb	Ta
T_{c0}[K]	9.45	7.2	4.45
B_{c0}[T]	0.198	0.0803	0.083

expulsion of magnetic flux we obtain the London equation:

$$\vec{B} + \lambda_L^2 \nabla \times \nabla \times \vec{B} = 0 \tag{2.4}$$

with a so called London penetration depth λ_L of

$$\lambda_L = \sqrt{\frac{mc^2}{4\pi\mu ne^2}} \tag{2.5}$$

The London penetration depth is dependent on material, but is typically on the order of 500 $\mathring{A}$. [6]

2.2 Properties of Type II Superconductors

Type II superconductors, like type I, have a phase region under a certain T_{c1}, H_{c1}, and J_{c1} in which they exclude all magnetic flux. They also have a second superconducting phase below a second set of critical values T_{c2}, H_{c2}, and J_{c2} and above the first set of critical values. A typical phase plot of a type II superconductor is shown in figure 2-2. In this phase, the material is superconducting, but it cannot expel all of the external magnetic flux. Some flux penetrates the material in the form of flux vortices, so called fluxons, localized areas of normal conducting material surrounded by a circulating surface current which retains the superconducting state of the material surrounding the area. Type II superconductors are usually alloys such as NbTi and Nb_3Sn. The extra phase region, shown in figure 2-2, although allowing some flux penetration, allows type II superconductors to operate at a higher temperature than type I, making them the most commonly used type of superconducting material. Some critical values of typical type II materials are shown in table 2.2.

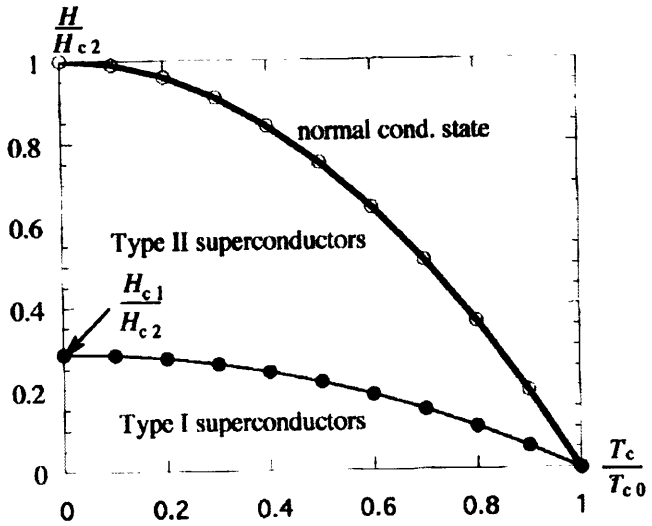

Figure 2-2: Phase diagram for a type II superconductor. Complete expulsion of magnetic flux occurs under the first set of critical values labeled with indices $_{c1}$. Partial expulsion of magnetic flux occurs in the phase in between these values and the second set of critical values labeled with indices $_{c2}$. Image from [5].

Table 2.2: Typical type II superconducting materials and their critical temperatures and magnetic fields [5].

Superconductor:	NbTi	Nb_3Sn
B_{c2}[T]	14-15	24-30
T_{c0}[K]	9.0	18.2

Chapter 3

Modeling Superconductivity

3.1 Finite Element Method

The program FEMM (Finite Element Method Magnetics)[7] was used in this project for the simulation and optimization of superconducting bearings. FEMM provides solutions for magnetostatic problems as well as low frequency time-harmonic magnetic problems. In the magnetostatic case:

$$\nabla \times \vec{H} = \vec{J} \tag{3.1}$$

$$\nabla \cdot \vec{B} = 0 \tag{3.2}$$

where the magnetic field $\vec{B}$ is related to $\vec{H}$ by:

$$\vec{B} = \mu \vec{H} \tag{3.3}$$

Rewriting equation 3.1 in terms of the magnetic vector potential $\vec{A}$, where $\nabla \times \vec{A} = \vec{B}$, we find:

$$\nabla \times \left(\frac{1}{\mu} \nabla \times \vec{A} \right) = \vec{J} \tag{3.4}$$

Taking into account boundary conditions, FEMM uses the finite element method to find solutions to this differential equation 3.4. [7]

As can be seen in figure 3-1 in FEMM objects are drawn, in this case permanent magnets, and then divided into many triangles. Equation 3.4 is then solved numerically to good precision over each triangle. Forces can also be calculated by integrating the Maxwell stress tensor. Properties such as electrical conductivity, B-H curves and magnetic permeability can be applied to each object. Boundary Conditions can also be forced on the boundaries between objects. In all of the simulations run here, a dirichlet boundary condition was set to the edge or frame around the space in which the objects were situated. This forced the magnetic potential $\vec{A} = 0$ at the edges where the space was cut off.

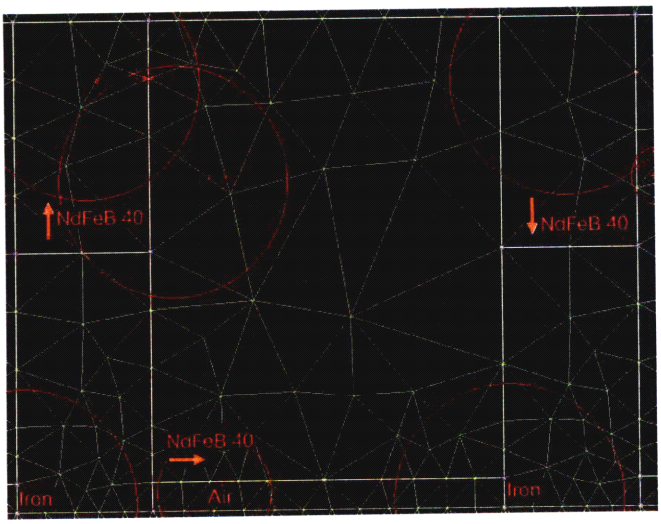

Figure 3-1: Permanent Magnets modeled in FEMM and divided into triangles for numerical solution. Circles indicate the so-called mesh, how many triangles a single block of material is divided into. The larger the circle the lower the mesh density.

Since FEMM solves equation 3.4 for the static case, a script was written in the Lua language to calculate the effect of moving objects. Lua is a scripting language integrated with FEMM. After a solution is found in the first static case, the script continually redraws the objects moving them in small steps, each time repeating the calculation. Unfortunately, this method neglects the effects of electromagnetic induction.

3.2 Kim Approximation

FEMM has a built in material database for modeling materials of different magnetization curves and magnetic permeability, but there is no built in handling of superconducting material. A model of type II superconducting material was accomplished in two steps. The first step, as described in this section, was to determine how much magnetic flux penetrates the material. The second, as described in the next section, was to model flux pinning. Flux pinning is the property of type II superconductors in which the flux that does penetrate the material is pinned, or "frozen" into place.

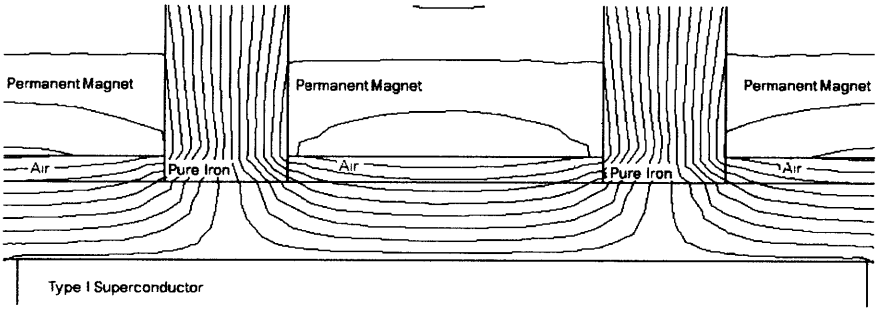

Figure 3-2: Type I superconductor modeled in FEMM, note in the block labeled Type I superconductor the complete expulsion of magnetic flux lines created by the permanent magnets.

As shown in figure 3-2 type I superconducting material can easily be simulated in FEMM by setting property conditions to expel all external magnetic flux. This is achieved by setting the B-H curve to be that of a perfect diamagnet. To determine how much magnetic flux penetrates type II superconducting material, the fact that materials can be described in FEMM by their magnetization curve was exploited. The magnetization curve of the type II superconductor YBCO (Yttrium Barium Copper Oxide) as approximated by the Kim approximation found in reference [3] was inputted in FEMM. This curve is shown in figure 3-3. As can be seen in the output file of FEMM in figure 3-4 this allows partial penetration of magnetic flux.

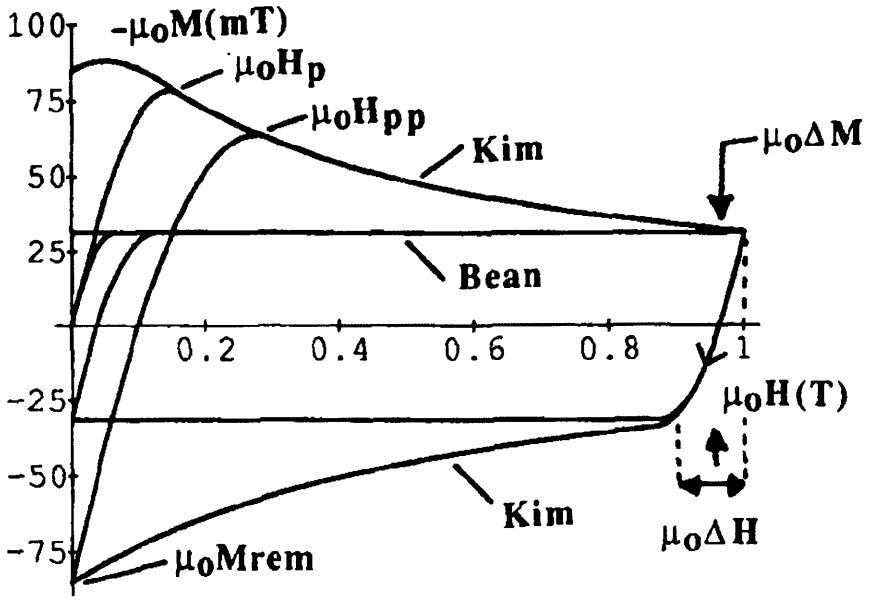

Figure 3-3: Kim approximation to type II superconductor YBCO given as uppermost and lowermost curve [3].

3.3 Fixed Magnetic Vector Potential

The Lua script, as seen in the appendix, in combination with FEMM and the Kim approximate magnetization curve described in the previous section simulates flux-pinning. This is achieved by first running a static calculation of the superconductor in its starting position with the Kim magnetization curve. This determines how much flux penetrates the superconductor. As can be seen in figure 3-4 the boundaries of the superconducting blocks labeled "HTS" are divided into many small segments. The magnetic vector potential is then calculated at the nodes of these segments, denoted as blue squares. Moving the superconductor to a new position in small steps, the Lua script each time applies the calculated magnetic vector potential "frozen" by the superconductor to the segments around the boundaries. The partial penetration of magnetic flux is shown in 3-4. After the described fixed magnetic vector potential process is applied, the magnetic flux is pinned in the superconductor as seen in figure 3-5 with the permanent magnets removed.

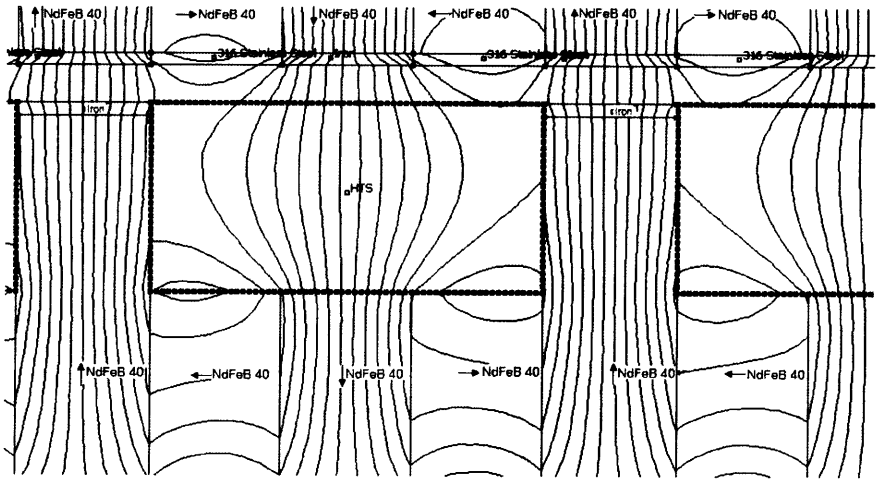

Figure 3-4: FEMM model of YBCO type II superconductor, boundaries are divided into segments where the magnetic vector potential will be calculated and held fast to pin the penetrating flux. In the diagram, permanent magnets are labeled NdFeB and their direction of magnetization is indicated by the arrow next to their label. Bulk Superconductors are labeled HTS. Note the partial penetration of magnetic flux into the HTS blocks.

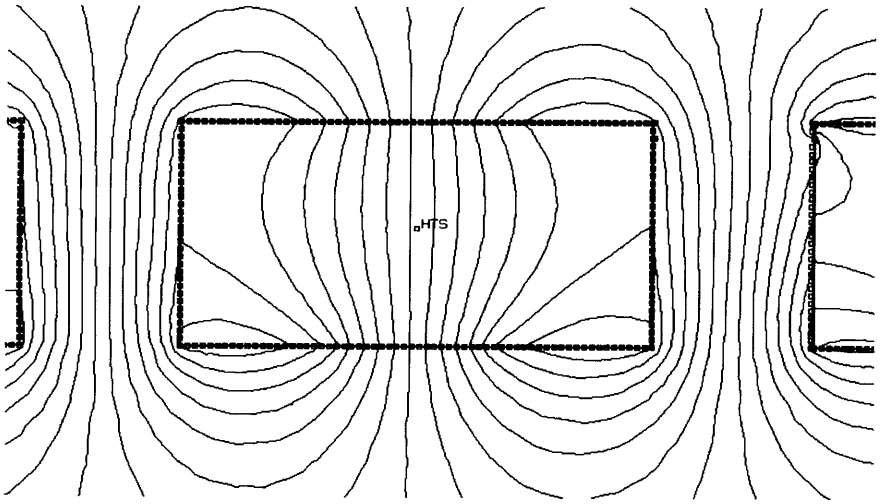

Figure 3-5: FEMM model of YBCO type II superconductor with penetrating flux pinned or "frozen". The model is shown with the permanent magnets removed. Bulk Superconductors are labeled HTS.

Chapter 4

Optimization

4.1 Baseline

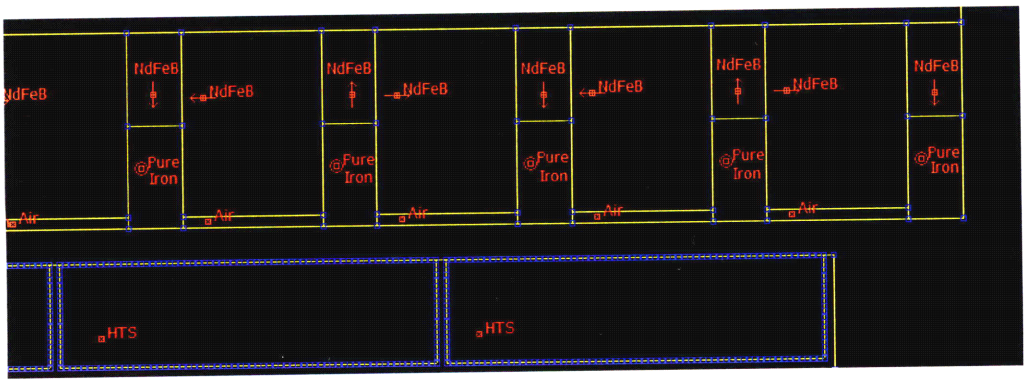

Figure 4-1: FEMM model of an actual superconducting bearing. This is a cross section of the lower half of the bearing. Permanent magnets are labeled NdFeB and their direction of magnetization is indicated by the arrow next to their label. Bulk Superconductors are labeled HTS. The bearing is cylindrically symmetric with the bulk superconductors surrounding the stator side and the assembly of permanent magnets surrounding the rotor.

To test the accuracy of the FEMM model of the superconducting state, a bearing which had already been built and tested was modeled in FEMM. Part of the model is shown in figure 4-1 as a cross section of the lower half of the cylindrically symmetrical bearing. As shown, the bearing consists of NdFeB permanent magnets surrounding the rotor section in a Halbach configuration. Tiles of bulk YBCO type II superconductor surround the stator side of the bearing. The expulsion of magnetic

flux emitted from the permanent magnets by the bulk superconductor creates a force which levitates the rotor shaft. This particular bearing was literally cut into upper and lower halves. The upper half was cooled in its position of operation, where as the lower half was cooled 7 mm below its operating position. The operational position of both halves is 1 mm away from the rotating shaft of permanent magnets which allows sufficient space for thermal insulation. Cooling the lower half of the bearing away from the permanent magnets reduces the amount of flux "frozen" by the superconducting material, thus increasing the amount of magnetic flux expelled and thereby the carrying force when raised to the operating position. It does, however, reduce the axial stiffness of the bearing. This bearing produced a total radial carrying force of 5 kN as well as a radial stiffness of 5 kN/mm.

The bearing was modeled appropriately in FEMM, by first calculating the force and stiffness applied to the stationary upper half of the bearing. Contrastingly, the amount of magnetic flux penetrating the lower half of the bearing was calculated in FEMM at 7 mm below operating position. The bearing was then moved with this flux pinned, to its operating position, using the Lua script as seen in the appendix and described in section 3.1. For both halves, forces were calculated -by integrating the Maxwell stress tensor- with the rotor in different positions by moving it in steps of 0.625 mm in both the axial and radial directions. By finding the radial and axial forces as functions of distance the radial and axial stiffnesses were found. The FEMM model resulted in a calculation of a total radial force of 7.9 kN and a stiffness of 550 N/mm. See files Original.lua in the appendix.

4.2 Hybrid Magnetic Bearing

When the bearing described in the last section is scaled to the dimensions of active magnetic bearings currently used in large electric drives, its carrying force is reduced to 3210 N. From looking at the lines of flux in figure 4-2 it became clear that the flux density and thereby the carrying force would greatly increase if the stator side of the bearing also contained permanent magnets creating a hybrid bearing with both

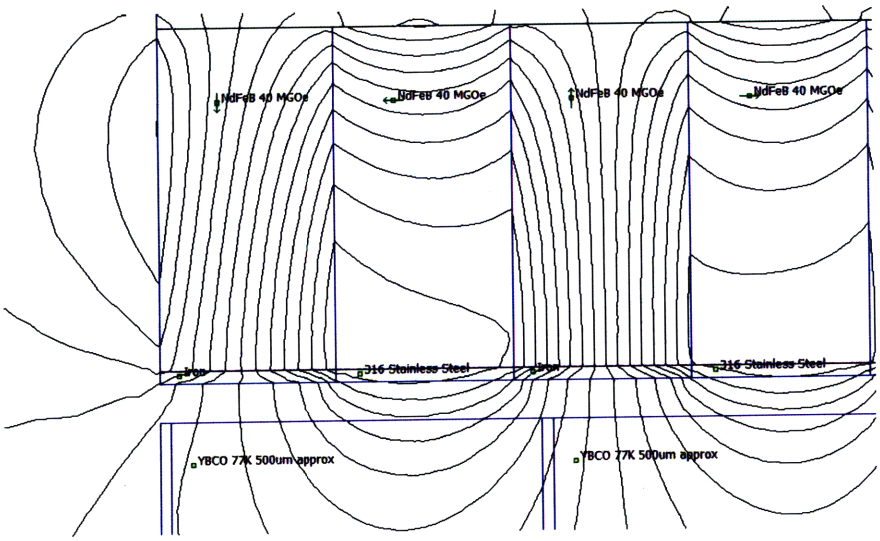

Figure 4-2: Previous bearing scaled to the dimensions of a standard active magnetic bearing.

superconducting and permanent magnets. These were added as shown in figure 4-3. This immediately increased the total carrying force to 25300 N, but significantly reduced the stiffness to a negative value. To remedy this, a program to optimize the dimensions of the bearing as discussed in the next section was written.

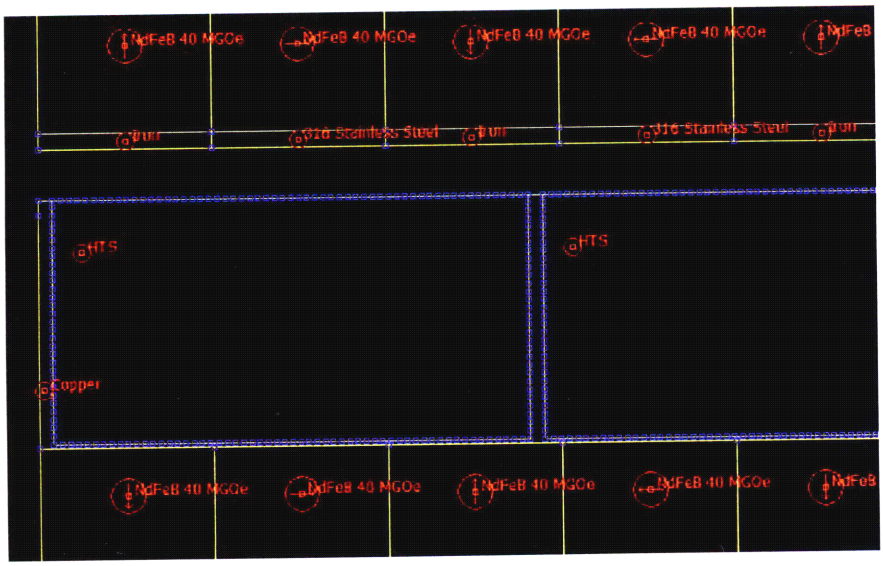

Figure 4-3: Permanent magnetic/superconducting Hybrid Design.

4.2.1 Optimization of Dimensions

Starting with the original hybrid design, a program was written which took the horizontal and vertical dimensions of the bulk superconductors and expanded them in each direction at a time by 1 mm. The forces and stiffnesses were then calculated for each configuration. This procedure was then repeated for the dimensions of the permanent magnets. Since the permanent magnets had to be ordered as a Halbach array, instead of changing the dimensions 1 mm at a time, the optimization code increased the number of permanent magnets each loop by one. After repeating this method a few hundred times a configuration was found with an optimal carrying force of 5200 N and a positive radial stiffness of 3950 N/mm. See files 500AB.lua.

4.3 Maximum Field Cooling

Since bearings with permanent magnets are costly and difficult to assemble, a solution was sought that did not contain permanent magnets. One conceivable way to increase the carrying force of a type II superconducting bearing, is to use active magnetic coils to increase the amount of magnetic flux which is frozen by the bulk superconductors. Such a design is seen in figure 4-4. The coils (labeled Js) are placed below the bulk superconducting tiles. Iron cores guide the magnetic flux to the superconducting tiles. These active coils are turned on when the bottom half of the bearing is in the lowered position. The bulk superconductor is cooled while the active coils are on, allowing some of the magnetic flux from the coils to penetrate. Once cooled, the active coils are turned off and the magnetic flux which penetrated the bulk superconductor remains pinned. The bottom half of the bearing is then raised into its operating position. Charging the bulk superconductors with active coils in this method increases the flux pinned by the superconductor, improving the performance of the bearing over one with bulk superconductors alone. Such a bearing, simulated with the same outer dimensions as the one discussed in the previous section and optimized with a similar method achieved a carrying force of 4970 N and a radial stiffness of 2040 N/mm.

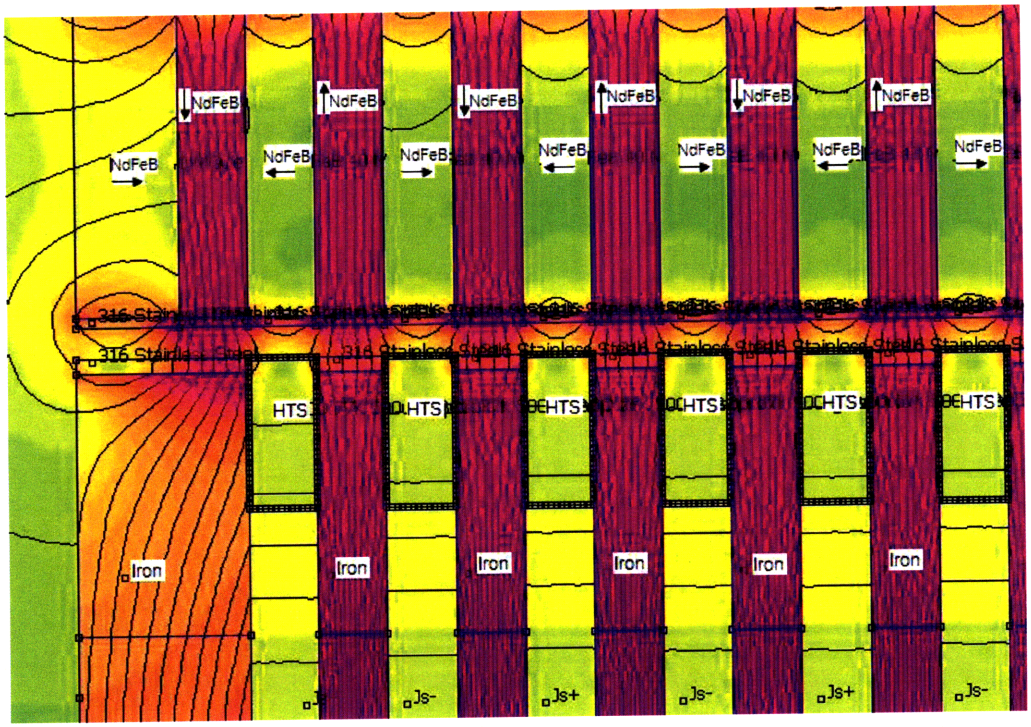

Figure 4-4: A Bearing which employs Maximum Field Cooling. Active Coils are labeled Js.

Chapter 5

Conclusions

In this project, the physics of superconducting material was used to simulate the forces on superconducting magnetic bearings using the finite element method. These simulations were intended to ultimately explore the possibility of using type II superconducting magnetic bearings in large electric drives. A method of simulating superconducting materials within a finite element program was developed by using known properties of type II superconductors, specifically the partial expulsion of magnetic flux and flux pinning. When modeling a real bearing, calculations of carrying force proved to be relatively accurate. The calculations of stiffness, however, were not accurate. There are several possible reasons for this. Since FEMM is a two dimensional Finite Element Simulator, the third dimension had to be approximated. There are also difficulties calculating any time dependence, which comes into play when the components are moved into different positions to calculate stiffness as described in section 4-1. Since FEMM does not have a built in method for handling time-dependence, one was created using frames, like in a movie. Fields, forces, and flux penetration were calculated in one position, the appropriate components were moved in a small increment and the quantities were recalculated and this was repeated until the components were in their end positions. Consequently, this method does not accurately account for induced magnetic fields.

Problems calculating stiffness did not prevent the optimization of carrying force. Creating a hybrid design, by combining the traditional superconducting magnetic

bearing design with a permanent magnetic bearing increased the carrying force almost ten fold, but pushed the radial stiffness below zero. Optimizing the dimensions of the bulk superconductors as well as the permanent magnets found an optimal design with a greater carrying force than without the permanent magnets and with a positive radial stiffness. Although, as mentioned above, the calculation of stiffness was not accurate, the calculated value, however, was always less then the measured value. This indicates that the solution optimized for force and stiffness may indeed have an improved stiffness over the original hybrid design. One possible technical issue with a hybrid design is providing the bulk superconductors with sufficient cooling while the permanent magnets are situation immediately around them. In addition to this hybrid design, a new concept based on the principle of maximum field cooling (MFC) was developed to avoid the use of permanent magnets. A MFC solution was also found with greater carrying force than bulk superconductors alone. In conclusion, the carrying force of superconducting magnetic bearings can be greatly improved by combining them with permanent magnets or active magnets.

Bibliography

[1] Werner Braunbek. Freies Schweben diamagnetischer Körper im Magnetfeld. 112.

[2] Werner Braunbek. Freischwebende Körper im elektrischen und magnetischen Feld. 112(11-12):753–763, March 1939.

[3] J.R. Cave. Calculation of magnetic flux profiles and deduction of critical current densities for type II superconductors. 27(2):1379–1382, March 1991.

[4] S. Earnshaw. On the nature of the molecular forces which regulate the constitution of the luminiferous ether. III(Part I):97–112, March 1842.

[5] Peter J. Lee. *Engineering Superconductivity.* Wiley, 2001.

[6] Michael P. Marder. *Condensed Matter Physics.* Wiley, 2000.

[7] D. C. Meeker. *Finite Element Method Magnetics, Version 4.2.* http://femm.foster-miller.net/Archives/doc/manual42.pdf, 2008.

Appendix A

Simulation Codes

A.1 Original.lua

```
--Measures forces and stiffnesses on original Type II
--Superconducting Bearing

--I. Start HTS from center point (-3mm)
--II. Measure A every 0.5 mm along boundaries of HTS
--III. Calculate A0, A1, A2 for each 0.5 mm segment
--IV. apply Magnetic Vector Potentials from III
--      to HTS surfaces; Step upto +0.4mm by 0.02 mm,
--       measure Forces

NAME = "Original"
FILE = "Originala.fem"
FILE2 = "Original2.fem"
FILE2a = "Original2a.fem"
f1xa =0
f2xa =0
f1ya =0
f2ya =0
f1x0 = 0
f1x1 = 0
f1y0 = 0
```

```
f1y1 = 0
S1x = 0
S1y = 0
f2x0 = 0
f2x1 = 0
f2y0 = 0
f2y1 = 0
S2x = 0
S2y = 0
-- Load Console, Open File, Set Group Mode
showconsole()
clearconsole()
for SCHALE = 1,2 do -- 1 Oberschale 2 Unter Schale

if (SCHALE == 2) then
mydir="./"
open(mydir .. NAME.."_fem")
mi_saveas(mydir .. FILE2)
 mi_clearselected()
 mi_seteditmode("group")
 mi_selectgroup(5)
        mi_movetranslate(0,-4) -- MOVE to ZFC Position
 -- pause()
  mi_analyze()
  mi_loadsolution()

-------------------------------FLUX FREEZING-------------------------------
--Variables
num = 8          -- NUMBER of Blocks -1 (0 in loops)
AB = 33.8   -- Length (x) of each block + space in between
LB = 32.9225   -- Length of block
HO = 8.95        -- Height of block
XSEG = 65           -- Number of x segments
LSEG =  0.5065      -- Length of x segments (total lenght of block (LB)
divided by XSEG)
TX = XSEG -1             -- XSEG -1 (for table indexing)
```

```
YSEG = 20            -- Number of Y segments
HSEG = 0.4475        -- Height of y segments (total Height of block (HO)
 divided by YSEG)
TY = YSEG -1             -- YSEG -1 (for table indexing)

xintial= 12.34 --initial X position                              OF HTS!!
yintial=-166.5 --initial y position, 0 position
xint = xintial
yint = yintial
M= 0
--FIND A Magnetic Vector Potential
  --Write A values of points into a table A
  --Table A has rows corresponding to visual intepretation of block
  --i. e. row[1] -> y = -162.500, row[2] -> y = -161.9475
  --first block starts at (xint,yint) = (12.3400,-165.500)
 --MAke a table of all blocks:
  BA = {}
--FOR Loop selects each block in order
--Size of each block + space =AB
 for B = 0, num do

  A = {}
 --For Loop picks up A on first and last horizontal lines
 for Y = 0, 1 do
 A[(Y*YSEG)]={}
 for X = 0, XSEG do
 A[(Y*YSEG)][X] = mo_getpointvalues(xint+(X*LSEG) + B*AB,yint-(Y*HO))
 end
  end
   --FOR Loop picks up A on first and last Vertical Lines
    for Y = 1, TY do
      A[Y]={}
      for X = 0,1 do
   A[Y][(X*XSEG)] = mo_getpointvalues(xint+(X*LB)+B*AB,yint-(Y*HSEG))
      end
    end
```

```
  --insert into BA
  BA[B]=A
 end --Block Selection For loop

-- -- Calculate A0, A1, A2
--HA(n) - A(n) along horizontal Lines
--VA(n) - A(n) along vertical lines
HA0={}
HA1={}
VA0={}
VA2={}
-- III. FOR Loop selects each block in order
for B = 0, num do
  --create tables
  H={}
  V={}
  HB={}
  VB={}
    --For Loop calculates A(1,2) on first and last horizontal lines
    for Y = 0, 1 do
      H[(Y*YSEG)]={}
      for X = 0, TX do
H[(Y*YSEG)][X] = (( BA[B][(Y*YSEG)][X]-BA[B][(Y*YSEG)][(X+1)]) / (-LSEG) )
      end
    end

    for Y = 0, TY do
      V[Y]={}
      for X = 0,1 do
V[Y][(X*XSEG)] = ( (BA[B][Y][(X*XSEG)]-BA[B][(Y+1)][(X*XSEG)]) / (HSEG) )
      end
    end
    --For Loop calculates A0 on first and last horizontal lines
    for Y = 0, 1 do
      HB[(Y*YSEG)]={}
      for X = 0, TX do
```

```
HB[(Y*YSEG)][X] = BA[B][(Y*YSEG)][X] - ((xint+(X*LSEG)+ B*AB)* H[(Y*YSEG)][X])
      end
    end
    for Y = 0, TY do
      VB[Y]={}
  for X = 0,1 do -- + Adjusts vertical A0 before hts is moved!!
VB[Y][(X*XSEG)] = BA[B][Y][(X*XSEG)] - ((yint-(Y*HSEG)+M)* V[Y][(X*XSEG)])
      end
    end
  HA0[B]=HB
  HA1[B]=H
  VA0[B]=VB
  VA2[B]=V
end -- III. A FIND BLOCK CYCLE
----------Testing only DELETes Everything besides HTS------ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!
--mi_clearselected()
--mi_seteditmode("group")
--mi_selectgroup(2)
--mi_deleteselected()
--mi_clearselected()
-------------------------------------------------------------

--change to line segment mode
mi_clearselected()
mi_seteditmode("segments")
--IV. Apply Found A0, A1, A2 to Segments
--    FOR Loop selects each block in order
 for B = 0, num do
    --For Loop applies A on top segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
mi_addboundprop((B+1)*100000000+X, HA0[B][0][X],HA1[B][0][X], 0,
0, 0, 0, 0, 0, 0)
                        -- Half a segment in
-print("Top Block:",B," X1:",xint+X*LSEG+B*AB," a1",BA[B][0][X]," a2",
BA[B][0][X+1]," A0:", HA0[B][0][X], "A1", HA1[B][0][X])
```

```
  mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB, yint)
   mi_setsegmentprop((B+1)*100000000+X,0,1,0,5)
        mi_clearselected()
      end
   --For Loop applies A on Left segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
mi_addboundprop(((B+1)*100)+XSEG+Y, VA0[B][Y][0], 0, VA2[B][Y][0],
 0, 0, 0, 0, 0, 0)
                                          -- Half a segment in
        mi_selectsegment(xint+B*AB, yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((B+1)*100)+XSEG+Y,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Bottom segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
        mi_addboundprop((1+B)*10000+X, HA0[B][YSEG][X], HA1[B][YSEG][X],
        0, 0, 0, 0, 0, 0, 0)
         -- Half a segment in                        + y block size
        mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB, yint - HO)
        mi_setsegmentprop((1+B)*10000+X,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Right segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
mi_addboundprop(((1+B)*1000000)+XSEG+Y, VA0[B][Y][XSEG], 0, VA2[B][Y][XSEG],
0, 0, 0, 0, 0, 0)
                                  -- + x block size        + Half a segment in
        mi_selectsegment(xint+B*AB+LB, yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((1+B)*1000000)+XSEG+Y,0,1,0,5)
        mi_clearselected()
```

```
      end
   end— IV. block select
--test A values
--print("Block two topX30 A0=",HA0[1][0][30],"Block 4 left 8 A0=",VA0[3][8][0],
"Block 6 topx27 A0=",HA0[5][YSEG][27],"Block 8 left 4 A0=",VA0[7][4][XSEG])
————————END————————END FLUX FREEZING END————————END————————
mo_close() --close output file
-- MOve HTS
--pause()
 -- mi_analyze()
--pause()
 mi_clearselected()
 mi_seteditmode("group")
 mi_selectgroup(5)
        mi_movetranslate(0,4) -- MOVE to Oportational Position
mi_clearselected()
mi_saveas(mydir .. FILE2)
--pause()
end --oberschale if
print("dist     ","Axialforce      ","Radial Force")
for L = 1, 4 do
 sgn = (-1)^L
 M = sgn * 0.125*0.5
if (SCHALE == 1) then
  mydir="./"
  open(mydir .. NAME.."_fem")
  mi_saveas(mydir .. FILE)
else
  mydir="./"
  open(mydir .. FILE2)
  mi_saveas(mydir .. FILE2a)
end
print("dist     ","Axialforce      ","Radial Force")
-- write (handle, "\n", FILE, "\n", "dist          ","Axiale Kraft      ",
"Radiale Kraft      ", "Radiale Steifigkeit      ", "Axiale Steifigkeit
","\n")
```

```
--Variables
num = 8          -- NUMBER of Blocks -1 (0 in loops)
AB = 33.8    -- Length (x) of each block + space in between
LB = 32.9225    -- Length of block
HO = 8.95        -- Height of block
XSEG = 65           -- Number of x segments
LSEG =  0.5065-- Length of x segments (total lenght of block (LB)
divided by XSEG)
TX = XSEG -1             -- XSEG -1 (for table indexing)
YSEG = 20           -- Number of Y segments
HSEG = 0.4475        -- Height of y segments (total Height of block (HO)
divided by YSEG)
TY = YSEG -1             -- YSEG -1 (for table indexing)
xintial= 12.34 --initial X position                          OF HTS!!
yintial=-162.5 --initial y position, 0 position
--For Loop to Move HTS from 0 position to
--Step by M mm
 for k=0, 1 do
  mi_analyze()
  mi_loadsolution()
--change initial positions after block is moved:
    if (L<3) then
        xint=xintial
        yint=yintial + k* M
        else
        xint=xintial + k*M
        yint=yintial
     end
--Calculate Force BEFORE Collecting and setting A
 - -!!!!!!!!!!!!!!!!!!IF BLOCK is moved after finding A, new A(n) values must be
 - -!!!!!!!!!!!!!!!!!!        extrapolated to new position
  mo_groupselectblock(2)
  fx=mo_blockintegral(18)
  fy=mo_blockintegral(19)
--   print("yint", yint, (k*M),"mm from Normal Position",
"Axial Force=",fx," Radial Force=",fy)
```

```
if (L<3) then
      print(3 - k*M," x=0",fx," ",fy)
      dist  = 3 - k*M .. "     x=0    "
      dy = 3 - k*M
            if (dy == (3 - 0.0625) ) then
                  if (SCHALE == 1) then
                        f1y0 = fy
                  else
                        f2y0 = fy
                  end
            else
                  if (dy == (3 + 0.0625) ) then
                        if (SCHALE == 1) then
                              f1y1 = fy
                        else
                              f2y1 = fy
                        end
                  end
                  if (dy == 3) then
                        if (SCHALE == 1) then
                              f1ya = fy
                              f1xa = fx
                        else
                              f2ya = fy
                              f2xa = fx
                        end
                  end
            end
    else
      print(-k*M," y=3",fx," ",fy)
      dist  = -k*M .. "     y=3     "
      dx = -k*M
            if (dx == (-(0.0625 )) ) then
                  if (SCHALE == 1) then
                        f1x0 = fx
                  else
```

```
                                f2x0 = fx
                            end
                    else
                            if (dx == (0.0625)) then
                            if (SCHALE == 1) then
                                f1x1 = fx
                            else
                                f2x1 = fx
                            end
                        end

                    end
        end
print("Schale ", SCHALE)
print("fx = ", fx, "fy =", fy, "f1ya = ", f1ya, "f2ya = ", f2ya)
print("f1y0 = ", f1y0,"f1y1 = ", f1y1,"f1x0 = ", f1x0,"f1x1 = ", f1x1)
print("f2y0 = ", f2y0,"f1y1 = ", f2y1,"f2x0 = ", f2x0,"f2x1 = ", f2x1)
handle = openfile(NAME.."_txt","a")
write (handle, "\n", FILE)
 write (handle, "\n", dist ,"fx = ", fx, "fy =", fy, "f1ya = ", f1ya, "f2ya = ", f2ya)
 write (handle, "\n", dist ,"f1y0 = ", f1y0,"f1y1 = ", f1y1,"f1x0 = ", f1x0,"f1x1 = ",
 write (handle, "\n", dist ,"f2y0 = ", f2y0,"f1y1 = ", f2y1,"f2x0 = ", f2x0,"f2x1 = ",
closefile(handle)
```

---FLUX FREEZING---

```
--FIND A
  --Write A values of points into a table A
  --Table A has rows corresponding to visual intepretation of block
  --i. e. row[1] -> y = -162.500, row[2] -> y = -161.9475
  --first block starts at (xint,yint) = (12.3400,-165.500)
 --MAke a table of all blocks:
  BA = {}
--FOR Loop selects each block in order
--Size of each block + space =AB
 for B = 0, num do
    A = {}
    --For Loop picks up A on first and last horizontal lines
```

```
    for Y = 0, 1 do
      A[(Y*YSEG)]={}
      for X = 0, XSEG do
        A[(Y*YSEG)][X] = mo_getpointvalues(xint+(X*LSEG) + B*AB, yint-(Y*HO))
      end
    end
    --FOR Loop picks up A on first and last Vertical Lines
    for Y = 1, TY do
      A[Y]={}
      for X = 0,1 do
        A[Y][(X*XSEG)] = mo_getpointvalues(xint+(X*LB)+B*AB, yint-(Y*HSEG))
      end
    end
  --insert into BA
  BA[B]=A

 end --Block Selection For loop

-- -- Calculate A0, A1, A2
--HA(n) - A(n) along horizontal Lines
--VA(n) - A(n) along vertical lines
HA0={}
HA1={}
VA0={}
VA2={}
-- III. FOR Loop selects each block in order
for B = 0, num do
  --create tables
  H={}
  V={}
  HB={}
  VB={}
    --For Loop calculates A(1,2) on first and last horizontal lines
    for Y = 0, 1 do
      H[(Y*YSEG)]={}
      for X = 0, TX do
```

```
        H[(Y*YSEG)][X] = (( BA[B][(Y*YSEG)][X]-BA[B][(Y*YSEG)][(X+1)]) / (-LSEG) )
      end
    end
    for Y = 0, TY do
      V[Y]={}
      for X = 0,1 do
        V[Y][(X*XSEG)] = ( (BA[B][Y][(X*XSEG)]-BA[B][(Y+1)][(X*XSEG)]) / (HSEG) )
      end
    end
    --For Loop calculates A0 on first and last horizontal lines
    for Y = 0, 1 do
      HB[(Y*YSEG)]={}
      for X = 0, TX do
        HB[(Y*YSEG)][X] = BA[B][(Y*YSEG)][X] - ((xint+(X*LSEG)+ B*AB)* H[(Y*YSEG)][X])
      end
    end
   for Y = 0, TY do
      VB[Y]={}
      for X = 0,1 do                                    -- + Adjusts vertical A0 before
        VB[Y][(X*XSEG)] = BA[B][Y][(X*XSEG)] - ((yint-(Y*HSEG)+M)* V[Y][(X*XSEG)])
      end
    end
  HA0[B]=HB
  HA1[B]=H
  VA0[B]=VB
  VA2[B]=V
end -- III. A FIND BLOCK CYCLE
----------Testing only DELETes Everything besides HTS------ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
--mi_clearselected()
--mi_seteditmode("group")
--mi_selectgroup(2)
--mi_deleteselected()
--mi_clearselected()
-------------------------------------------------------------------

--change to line segment mode
mi_clearselected()
```

```
mi_seteditmode("segments")
--IV. Apply Found A0, A1, A2 to Segments
--    FOR Loop selects each block in order
 for B = 0, num do
    --For Loop applies A on top segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
        mi_addboundprop((B+1)*100000000+X, HA0[B][0][X],HA1[B][0][X], 0, 0, 0, 0, 0, 0,
                        -- Half a segment in
 --       print("Top Block:",B," X1:",xint+X*LSEG+B*AB,"a1",BA[B][0][X],"a2",BA[B][0][X
        mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB,yint)
        mi_setsegmentprop((B+1)*100000000+X,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Left segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
        mi_addboundprop(((B+1)*100)+XSEG+Y, VA0[B][Y][0], 0, VA2[B][Y][0], 0, 0, 0, 0,
                                                        -- Half a segment in
        mi_selectsegment(xint+B*AB,yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((B+1)*100)+XSEG+Y,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Bottom segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
        mi_addboundprop((1+B)*10000+X, HA0[B][YSEG][X], HA1[B][YSEG][X], 0, 0, 0, 0, 0,
                        -- Half a segment in                    + y block size
        mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB,yint - HO)
        mi_setsegmentprop((1+B)*10000+X,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Right segments
```

```
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
        mi_addboundprop(((1+B)*1000000)+XSEG+Y, VA0[B][Y][XSEG], 0, VA2[B][Y][XSEG], 0,
                                  -- + x block size          + Half a segment in
        mi_selectsegment(xint+B*AB+LB, yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((1+B)*1000000)+XSEG+Y,0,1,0,5)
        mi_clearselected()
      end
   end-- IV. block select
--test A values
--print("Block two topX30 A0=",HA0[1][0][30],"Block 4 left 8 A0=",VA0[3][8][0],"Block 6
----------END----------END FLUX FREEZING END----------END----------
mo_close() --close output file
-- MOve HTS
if (SCHALE == 1) then
  mi_saveas(mydir .. FILE)
else
  mi_saveas(mydir .. FILE2a)
end
 mi_clearselected()
 mi_seteditmode("group")
 mi_selectgroup(5)
  if (L<3) then
        mi_movetranslate(0,M)
     else
        mi_movetranslate(M,0)
  end
end
end
fyatot = f2ya - f1ya
fy0tot = f2y0 - f1y0
fy1tot = f2y1 - f1y1
fxatot = f2xa + f1xa
fx0tot = f2x0 + f1x0
fx1tot = f2x1 + f1x1
```

```
Sy = (fy0tot-fy1tot)/(0.125)
Sx = (fx0tot-fx1tot)/(0.125)
handle = openfile(NAME..".txt","a")
 write (handle, "FX0: ", fxatot, "FY0: ", fyatot)
 write (handle, "RS: ", Sy, "AS: ", Sx)
closefile(handle)
print( "FXa: ", fxatot, "FYa: ", fyatot)
print( "FX0: ", fx0tot, "Fx1: ", fx1tot)
print( "Fy0: ", fy0tot, "Fy1: ", fy1tot)
print("RS: ", Sy, "AS: ", Sx)
end --Schale
```

A.2 500.lua

```
--Finds the optimal dimensions of permanent magnets in the hybrid model
-- Takes 500.fem ADDs different PM Arrays in segments of total length divisible by
a symetric Halbach array /\ -> \/ <- /\
-- And measures Force
-- A. OPEN templet (500.fem)
-- B. Change dimensions
-- D. Move around
-- Load Console, Open File, Set Group Mode
showconsole()
clearconsole()
--Runs multiple lua scripts
for P = 1,100 do
-- Set file name to 500 +P
NAME = (500 + P)
FILE = NAME.."AB.fem"
-- A. OPEN templet (500AB.fem)
mydir="./"
open(mydir .. "500AB.fem")
mi_saveas(mydir .. FILE)
print("dist      ","Axialforce      ","Radial Force")
print(FILE)
handle = openfile("500AB.txt","a")
```

```
 write (handle, "\n", FILE, "\n")
closefile(handle)
fx0 = 0
fx1 = 0
fy0 = 0
fy1 = 0
Sx = 0
Sy = 0
-- SET Dimensions of Templet File :
xintial=-89.1429 --initial X position                              OF HTS!!
yintial=-1.5 --initial y position, 0 position
num = 6          -- NUMBER of Blocks -1 (0 in loops)
AB = 29 + 0.857143   -- Length (x) of each block + space in between
LB = 29     -- Length of block
HO = 15         -- Height of block
XSEG = 58           -- Number of x segments
LSEG =  LB/XSEG      -- Length of x segments (total lenght of block (LB) divided by XSEG
TX = XSEG -1              -- XSEG -1 (for table indexing)
YSEG = 30           -- Number of Y segments
HSEG = HO/YSEG  -- Height of y segments (total Height of block (HO) divided by YSEG)
TY = YSEG -1              -- YSEG -1 (for table indexing)
------------------ B. ADD PM Array ------------------------------------
TL = 180        -- Total Length of bearing to be filled by PM Array
NPM = 4 * P +1   -- Number of permanent magnets (a symetric Halbach array /\ -> \/ <- /\
LPM = TL/NPM  -- Lenght of permanent Magnets
--Top Left corner of Rotor
XR =-90
YR =35.25
HR = 32.75   --Height of Rotor side PM
--Top Left corner of Stator Side Array
XS =-90
YS =-16.75
HS = 18.75   --Height of Stator Side PM
for A = 1, NPM do
  --Rotor Side
  mi_addnode(XR + A*LPM, YR)
```

```
   mi_addnode(XR + A*LPM, YR-HR-1)
   mi_addsegment(XR + A*LPM, YR, XR + A*LPM, YR-HR-1)
--Calculate Magnitization          /\ -> \/ <-
MAGN = A - 4* floor(A/4) --  1   2  3  4  Returns a numer 1-4 according to magnitzation
 if (MAGN == 1) then
  MAG = 90
 else
  if (MAGN == 2) then
   MAG = 0
  else
   if (MAGN == 3) then
    MAG = 270
   else
    MAG = 180
   end
  end
end
```

```
  --add PM Property to center of PM
   mi_addblocklabel(XR + A*LPM - LPM*0.5, YR-HR*0.5)
   mi_clearselected()
   mi_selectlabel(XR + A*LPM- LPM*0.5, YR-HR*0.5)          --Group
   mi_setblockprop("NdFeB 40 MGOe",  0, 0.5, "<none>", MAG,  2,  1)
   mi_clearselected()
  --Add Reinforcment rings
   mi_addblocklabel(XR + A*LPM - LPM*0.5, YR-HR-0.5)
   mi_clearselected()
   mi_selectlabel(XR + A*LPM- LPM*0.5, YR-HR-0.5)          --Group
   if (MAG == 90) then
    mi_setblockprop("Iron",  0, 0.5, "<none>", 0,  2,  1)
    else
      if (MAG == 270) then
         mi_setblockprop("Iron",  0, 0.5, "<none>", 0,  2,  1)
      else
        mi_setblockprop("316 Stainless Steel",  0, 0.5, "<none>", 0,  2,
1)
```

```
    end
   end
   mi_clearselected()
   --Stator Side
   mi_addnode(XS + A*LPM, YS)
   mi_addnode(XS + A*LPM, YS-HS)
   mi_addsegment(XS + A*LPM, YS, XS + A*LPM, YS-HS)
--Calculate Magnitization          /\ -> \/ <-
MAGN = A - 4* floor(A/4) --  1   2  3  4  Returns a numer 1-4 according to magnitzation
 if (MAGN == 3) then
  MAGS = 90
 else
  if (MAGN == 2) then
   MAGS = 0
  else
   if (MAGN == 1) then
    MAGS = 270
   else
    MAGS = 180
   end
  end
end
---------
  --add PM Property to center of PM
   mi_addblocklabel(XS + A*LPM- LPM*0.5, YS-HS*0.5)
   mi_clearselected()
   mi_selectlabel(XS + A*LPM- LPM*0.5, YS-HS*0.5)          --Group
   mi_setblockprop("NdFeB 40 MGOe",  0, 0.5, "<none>", MAGS,  5,  1)
   mi_clearselected()
end
mi_saveas(mydir .. FILE)
---------- D ----------------------------------------------------------------------
print((500+P).."AB.fem")
print("dist     ","Axialforce      ","Radial Force")
handle = openfile("500AB.txt","a")
 write (handle, "\n", (500+P).."AB.fem", "\n", "dist           ","Axiale Kraft
```

```
","Radiale Kraft      ", "Radiale Steifigkeit      ", "Axiale Steifigkeit
","\n")
closefile(handle)
for L = 1, 4 do
mi_close()
 sgn = (-1)^L
 M = sgn * 0.125*0.5
mydir="./"
open(mydir .. FILE)
mi_saveas(mydir .. NAME.."a.fem")
 for k=0, 1 do
  mi_analyze()
  mi_loadsolution()
--change initial positions after block is moved:
    if (L<3) then
        xint=xintial
        yint=yintial + k* M
        else
        xint=xintial + k*M
        yint=yintial
     end
--Calculate Force BEFORE Collecting and setting A
 - -!!!!!!!!!!!!!!!!!!IF BLOCK is moved after finding A, new A(n) values must be
 - -!!!!!!!!!!!!!!!!!!          extrapolated to new position
  mo_groupselectblock(2)
  fx=mo_blockintegral(18)
  fy=mo_blockintegral(19)
--   print("yint", yint, (k*M),"mm from Normal Position", "Axial Force=",fx," Radial For
--   print(yint, " Radial Force= ",fy)
    if (L<3) then
          print(3 - k*M," x=0",fx," ",fy)
          dist  = 3 - k*M .. "     x=0    "
          dy = 3 - k*M
                if (dy == (3 - 0.125*0.5)) then
                   fy0 = fy
                 else
```

```
                        if (dy == (3 + 0.125*0.5)) then
                            fy1 = fy
                end
                 end
        else
          print(-k*M," y=3",fx," ",fy)
          dist  = -k*M .. "     y=3     "
          dx = -k*M
                if (dx == (-0.125*0.5)) then
                   fx0 = fx
                 else
                        if (dx == (0.125*0.5)) then
                            fx1 = fx
                    end
                 end
     end
if (fy0~=0) then
   if (fy1~=0) then
      Sy = (fy0 - fy1)/0.125
    end
end
if (fx0~=0) then
   if (fx1~=0) then
      Sx = (fx0 - fx1)/0.125
    end
end
handle = openfile("500AB.txt","a")
 write (handle, "\n", dist ,fx,"          ",fy,"          ",Sy,"          ",Sx)
closefile(handle)
```

FLUX FREEZING

```
--FIND A
  --Write A values of points into a table A
  --Table A has rows corresponding to visual intepretation of block
  --i. e. row[1] -> y = -162.500, row[2] -> y = -161.9475
  --first block starts at (xint,yint) = (12.3400,-165.500)
 --MAke a table of all blocks:
```

```
  BA = {}
--FOR Loop selects each block in order
--Size of each block + space =AB
 for B = 0, num do
    A = {}
    --For Loop picks up A on first and last horizontal lines
    for Y = 0, 1 do
      A[(Y*YSEG)]={}
      for X = 0, XSEG do
        A[(Y*YSEG)][X] = mo_getpointvalues(xint+(X*LSEG) + B*AB, yint-(Y*HO))
      end
    end
    --FOR Loop picks up A on first and last Vertical Lines
    for Y = 1, TY do
      A[Y]={}
      for X = 0,1 do
        A[Y][(X*XSEG)] = mo_getpointvalues(xint+(X*LB)+B*AB, yint-(Y*HSEG))
      end
    end
  --insert into BA
  BA[B]=A
 end --Block Selection For loop
-- -- Calculate A0, A1, A2
--HA(n) - A(n) along horizontal Lines
--VA(n) - A(n) along vertical lines
HA0={}
HA1={}
VA0={}
VA2={}
-- III. FOR Loop selects each block in order
for B = 0, num do
  --create tables
  H={}
  V={}
  HB={}
  VB={}
```

```
    --For Loop calculates A(1,2) on first and last horizontal lines
    for Y = 0, 1 do
      H[(Y*YSEG)]={}
      for X = 0, TX do
        H[(Y*YSEG)][X] = (( BA[B][(Y*YSEG)][X]-BA[B][(Y*YSEG)][(X+1)]) / (-LSEG) )
      end
    end
   for Y = 0, TY do
      V[Y]={}
      for X = 0,1 do
        V[Y][(X*XSEG)] = ( (BA[B][Y][(X*XSEG)]-BA[B][(Y+1)][(X*XSEG)]) / (HSEG) )
      end
    end
    --For Loop calculates A0 on first and last horizontal lines
    for Y = 0, 1 do
      HB[(Y*YSEG)]={}
      for X = 0, TX do
        HB[(Y*YSEG)][X] = BA[B][(Y*YSEG)][X] - ((xint+(X*LSEG)+ B*AB)* H[(Y*YSEG)][X])
      end
    end
    for Y = 0, TY do
      VB[Y]={}
      for X = 0,1 do                                    -- + Adjusts vertical A0 before
        VB[Y][(X*XSEG)] = BA[B][Y][(X*XSEG)] - ((yint-(Y*HSEG)+M)* V[Y][(X*XSEG)])
      end
    end
  HA0[B]=HB
  HA1[B]=H
  VA0[B]=VB
  VA2[B]=V
end -- III. A FIND BLOCK CYCLE
----------Testing only DELETes Everything besides HTS------ !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
--mi_clearselected()
--mi_seteditmode("group")
--mi_selectgroup(2)
--mi_deleteselected()
```

```
--mi_clearselected()
-------------------------------------------------------------------

--change to line segment mode
mi_clearselected()
mi_seteditmode("segments")
--IV. Apply Found A0, A1, A2 to Segments
--    FOR Loop selects each block in order
 for B = 0, num do
    --For Loop applies A on top segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
        mi_addboundprop((B+1)*100000000+X, HA0[B][0][X],HA1[B][0][X], 0, 0, 0, 0, 0, 0,
                        -- Half a segment in
 --       print("Top Block:",B," X1:",xint+X*LSEG+B*AB," a1",BA[B][0][X]," a2",BA[B][0][X
        mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB, yint)
        mi_setsegmentprop((B+1)*100000000+X,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Left segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
        mi_addboundprop(((B+1)*100)+XSEG+Y, VA0[B][Y][0], 0, VA2[B][Y][0], 0, 0, 0, 0,
                                                    -- Half a segment in
        mi_selectsegment(xint+B*AB, yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((B+1)*100)+XSEG+Y,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Bottom segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for X = 0, TX do
        mi_addboundprop((1+B)*10000+X, HA0[B][YSEG][X], HA1[B][YSEG][X], 0, 0, 0, 0, 0,
                        -- Half a segment in                    + y block size
        mi_selectsegment((LSEG/2)+xint+(X*LSEG) + B*AB, yint - HO)
```

```
        mi_setsegmentprop((1+B)*10000+X,0,1,0,5)
        mi_clearselected()
      end
    --For Loop applies A on Right segments
    --  First load selected segment's A values
    --  to a boundary property, then apply
      for Y = 0, TY do
        mi_addboundprop(((1+B)*1000000)+XSEG+Y, VA0[B][Y][XSEG], 0, VA2[B][Y][XSEG], 0,
                                  -- + x block size          + Half a segment in
        mi_selectsegment(xint+B*AB+LB, yint-(Y*HSEG)-(HSEG/2))
        mi_setsegmentprop(((1+B)*1000000)+XSEG+Y,0,1,0,5)
        mi_clearselected()
      end
   end-- IV. block select
--test A values
--print("Block two topX30 A0=",HA0[1][0][30],"Block 4 left 8 A0=",VA0[3][8][0],"Block 6
----------END----------END FLUX FREEZING END----------END----------
mo_close() --close output file
-- MOve HTS
mi_saveas(mydir .. NAME.."ABa.fem")
 mi_clearselected()
 mi_seteditmode("group")
 mi_selectgroup(5)
  if (L<3) then
        mi_movetranslate(0,M)
     else
        mi_movetranslate(M,0)
  end
end
end
```

```
mi_close()
end -- P loop
```