Development of Magnetic Induction Machines for Micro Turbo Machinery

by

Hür Köşer

M.Eng., Massachusetts Institute of Technology (1999)
B.S., Massachusetts Institute of Technology (1998)

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2002

© 2002 Massachusetts Institute of Technology
All rights reserved

Signature of Author.................................................................
Department of Electrical Engineering and Computer Science

23 May 2002

Certified by........................................
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by........................................
Barton L. Weller Professor of Electrical Engineering
Thesis Co-supervisor

Accepted by........................................
Chairperson, Department Committee on Graduate Students

© 2002 Massachusetts Institute of Technology
All rights reserved
Development of Magnetic Induction Machines for Micro Turbo Machinery

by

Hür Köşer

Submitted to the Department of Electrical Engineering and Computer Science on 23 May 2002, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

This thesis presents the nonlinear analysis, design, fabrication, and testing of an axial-gap magnetic induction micro machine, which is a two-phase planar motor in which the rotor is suspended above the stator via mechanical springs, or tethers. The micro motor is fabricated from thick layers of electroplated NiFe and copper, by our collaborators at Georgia Institute of Technology. The rotor and the stator cores are 4 mm in diameter each, and the entire motor is about 2 mm thick. During fabrication, SU-8 epoxy is used as a structural mold material for the electroplated cores. The tethers are designed to be compliant in the azimuthal direction, while preventing axial deflections and maintaining a constant air gap. This enables accurate measurements of deflections within the rotor plane via a computer microvision system.

The small scale of the magnetic induction micro machine, in conjunction with the good thermal contact between its electroplated stator layers, ensures an isothermal device which can be cooled very effectively. Current densities over $10^9$ A/m$^2$ simultaneously through each phase is repeatedly achieved during experiments; this density is over two orders of magnitude larger than what can be achieved in conventional macro-scale machines. More than 5 $\mu$Nm of torque is obtained for an air gap of about 5 $\mu$m, making this micro motor the highest torque density micro-scale magnetic machine to date. About 0.3 $\mu$Nm for the large air gap of 70 $\mu$m is also achieved in systematic tests that reveal the influence of strong eddy-currents and associated nonlinear saturation within the micro motor.

Eddy-current effects are modeled using a finite-difference vector potential formulation. Its results demonstrate the presence of flux crowding on the stator surface, which leads to heavy saturation. To capture saturation effects, a fully nonlinear finite-difference time-domain simulation is developed to solve Maxwell's Equations within the computational space of the micro machine. To mitigate the inherent stiffness in the partial differential equations, the speed of light is artificially reduced by five orders of magnitude, taking special care that assumptions of magnetoquasistatic behavior are still met. The results from this model are in very good agreement with experimental data from the tethered magnetic induction micro motor.

Thesis Supervisor: Jeffrey H. Lang
Title: Professor of Electrical Engineering and Computer Science

Thesis Co-supervisor: Stephen D. Senturia
Title: Barton L. Weller Professor of Electrical Engineering
# Contents

1 **Introduction** ................................................................. 17
   1.1 Micro Gas Turbine Generator ........................................ 19
   1.2 Magnetic Induction Machines ........................................ 20
   1.3 Magnetic Induction Micro Machine ................................. 21
   1.4 Thesis Scope .......................................................... 23
   1.5 Thesis Overview and Contributions ............................... 25
   1.6 Collaboration ......................................................... 27

2 **Analysis** ................................................................. 30
   2.1 The Big Picture ....................................................... 30
   2.2 The Rotor Section .................................................... 32
      2.2.1 Vector Potential Formulation ................................. 34
      2.2.2 Diffusion Transfer Relations for a Planar Conductor Layer in Translation 36
      2.2.3 Rotor Model ..................................................... 37
      2.2.4 Time-average Force ........................................... 39
      2.2.5 Stator Model .................................................... 39
   2.3 Implementation ....................................................... 46
      2.3.1 Finite Element Analysis Studies .............................. 47
      2.3.2 Initial Designs .................................................. 50
   2.4 Summary .............................................................. 53

3 **Fabrication** ............................................................. 55
   3.1 Tethered Motor Concept ............................................ 56
6.3 Suggestions for Future Work .................................................. 155

A Using the Schwarz-Christoffel Transformation for Magnetic Circuits 158

B Material Properties ................................................................. 163
  B.1 Electroplated NiFe ........................................................... 163
  B.2 Copper ........................................................................... 166

C Lamination Thickness Estimation ........................................... 168

D Measured Data ................................................................. 173

E Source Codes ................................................................. 183
List of Figures

1-1 The exponential rise in microprocessor transistor density has continued for the last three decades. (Source: Intel Corp.) .................................................. 18
1-2 Power consumption of some common microprocessors over the last thirty years. (Source: Intel Corp.) ................................................................. 18
1-3 A schematic cross-section of the micro engine. ........................................ 19
1-4 A macro-scale magnetic induction motor. .................................................. 21
1-5 The topology of the magnetic induction micro machine consists of planar surfaces extruded into the third dimension. The rotor spins on top of the stator, supported by air bearings. (Depiction courtesy of Florent Cros) ................................. 22
1-6 Variable reluctance micro motors, fabricated by H. Guckel’s research group at the University of Wisconsin [7]. ....................................................... 23

2-1 The rotor and the stator of a typical magnetic induction micro machine that will be built. The dimensions given are representative for the designed induction motor. ................................................................. 31
2-2 Concept of the analysis presented in this chapter. ..................................... 32
2-3 Magnetic circuit analysis for the stator is coupled to a multi-modal magnetic diffusion solution for the air gap and the rotor sections. The coupling is achieved through the flux values $\phi_i$ entering the stator, and the tangential magnetic field just on the stator surface. ......................................................... 33
2-4 A slab of rotor conductor $\Delta$ thick, moving with constant velocity $V$ in the $\hat{x}$ direction. The magnetic permeability and electrical conductivity are assumed to be uniform across the conductor. .................................................... 35
2-5 Setup of the multilayered boundary problem for the rotor. The top air layer is taken thick enough to represent an infinite boundary.

2-6 Setup for the reluctance model of the stator. The current, $i$, passing through the wire gap is the sum of the currents in both windings.

2-7 Tangential magnetic field just over the stator surface. The magnitude of each contribution has been taken as unity for illustration purposes. The ripples in the waveforms are due to truncation of the Fourier series.

2-8 Depiction of tangential magnetic field just over the stator along one wire slot.

2-9 $H_{ST}$ over one period in a two pole machine. Each neighboring section is $90$ degrees out of phase. Contributions to $H_{ST}$ have been taken to have unity magnitude for simplicity.

2-10 Graphical user interface for the design program based on Model I. The numbers within the figure correspond to those in Figure 2-12.

2-11 Even for unrealistically low magnetic permeabilities such as $50\mu_0$ as used here, the model does an excellent job in performance estimation. The solid lines correspond to the model predictions, whereas the points are FEA results for the same geometry.

2-12 Typical micromotor dimensions possible to achieve today, and the corresponding numbers for certain performance criteria. The radius of the machine is 2 mm. Input current is 17 A, corresponding to a current density of $6.9 \times 10^8$ A/m$^2$ (over $1 \times 10^9$ A/m$^2$ has been achieved later in tests). Magnetic permeability of electroplated NiFe is taken to be an average value of $3000\mu_0$.

3-1 A computer rendered image of the tethered micro motor stator die (SU-8 epoxy not shown). The pin holes at the corners are for alignment with the rotor die. All electroplated NiFe structures are at the same height.

3-2 Rendered image of the tethered rotor die aligned on top of the stator die (the rotor at the center not shown). Hexagonal holes are for the sacrificial layer etch that releases the rotor die. Three large holes on the top left allow current probes to make contact with the stator windings.
3-3  Rendered image of a side view of the tethered micro motor. The rotor-stator gap, and the vertical features of the stator are exaggerated for clarity; otherwise, the relative dimensions are accurate. ................................................................. 58

3-4  A computer rendered image of the rotated rotor core and the twisted tethers (the rest of the die is not shown). Bending in the tethers is exaggerated for clarity. ................................................................. 59

3-5  The boundary conditions that the tethers are subject to are essentially the same as those of half a clamped-clamped beam deflected by a point load at the center. ................................................................. 60

3-6  Thickness variation of along the length of the test beam and the tip deflection with attached masses. The two plots are used to deduce the Young's modulus of the SU-8 epoxy. ................................................................. 61

3-7  Tip deflection of a Kapton cantilever beam as a function of attached weight at the tip. The extracted Young's modulus of Kapton is about 4.54 GPa. ................................................................. 63

3-8  Cross-section of a Kapton tether shows a linear taper (Figure rotated counter-clockwise by quarter of a turn). The thin tip corresponds to the surface side on which laser drilling started. ................................................................. 64

3-9  The thickness of each tether at the top and the bottom surface of the Kapton rotor die. The overall average tether thickness is 57 μm. ................................................................. 65

3-10 Fabrication flow of the rotor die. ................................................................. 66

3-11 Pictures of the fabricated SU-8 rotor: (a) The fabricated SU-8 rotor die, with the rotor core glued at the center; (b) a close-up view of the sacrificial etch holes; (c) the center rotor housing ring and tethers – the bottom surface is a conductive tape used within the SEM; (d) one of the tethers; (e) in a close-up view, surface cracks at the base of the tether are clearly visible; (f) tether surface from the top, revealing the side-wall roughness which is due to the photoreduction masks. ................................................................. 68

3-12 Unit fabrication process for the stator (Figure courtesy of Florent Cros). ................................................................. 69

3-13 Fabrication flow for the stator die (Figure courtesy of Florent Cros). ................................................................. 69
3-14 Stator pictures, courtesy of Florent Cros: (a) Copper windings without the electroplated NiFe poles; (b) Complete stator (early generation); (c) SEM picture of the stator, with the top insulating SU-8 layer removed, revealing details of the teeth and the top copper winding; (d) A close-up SEM view of the stator around the inner end-turns, with the teeth removed to reveal detail.

4-1 The packaging assembly for the magnetic induction micro machine. Once the micro machine is aligned, spring loaded screws are tightened, pressing the top plate (the rotor die) gently onto the bottom aluminum plate (the stator die).

4-2 Typical dimensions of the “4-pass” HiContact™ liquid-cooled heat sink. The length “x” is six inches for the model used in our experiments. (Source: AAVID Thermalloy specification sheets, also at http://www.aavid.com/datashts/ThermoModules/index.html)

4-3 Cooling performance and the pressure drop in the plumbing vs. flow rate. The recommended maximum flow rate is 1.5 GPM. (Source: AAVID Thermalloy specification sheets, also at http://www.aavid.com/datashts/ThermoModules/index.html)

4-4 The test setup for the tethered magnetic induction micro machine.

4-5 The microvision system and the test electronics.

4-6 The hardware for the tethered magnetic micro machine test setup.

4-7 A simplified diagram of the power electronics concept.

4-8 Wiring schematic of the power circuitry. Only the circuit for one phase is shown; the electronics for the other phase is identical.

4-9 Torque reversal switch circuitry. Only the circuit for one of the phases is shown here; the other is identical.

4-10 Pictures from an iron powder test on an earlier generation stator. The windings of this stator have the wrong spatial phase (they have the same orientation, which cannot produce a traveling wave). When both phases are excited by the same input, the current in the windings flows in the same direction in every wire slot, resulting in a strong accumulation of iron powder over every tooth gap. With out of phase inputs, the accumulation is much weaker.
4-11 The conceptual schematic of the overall test setup. ......................... 85
4-12 The timing diagram for representative signals from certain locations in the test
setup of Figure 4-11. ....................................................... 86
4-13 The azimuthal tether deflection and deflection phase as a function of torque
reversal frequency. In the top plot, stars indicate measured values, whereas the
solid curve corresponds to a fit from a second-order system response. ............. 91
4-14 Torque as a function of various input electrical frequencies at 5 A of peak current
input. Data represents the results from an SU-8 tethered magnetic induction
micro motor. The estimated air gap is about 5 μm. ........................................ 92
4-15 Measured torque from the tethered magnetic induction micro-motor for several
current levels. The device consists of a six pole stator and a rotor suspended
above it via Kapton tethers. The error bars correspond to the maximum fit
uncertainty. ................................................................. 93
4-16 Predictions of torque vs. frequency from Model I for the same tethered magnetic
micro motor tested. Comparison with Figure 4-15 makes it clear that the torque
values predicted by Model I are about an order of magnitude higher than the
measured results; the peak frequencies are also higher than the measurements.
The source of the discrepancy is the effect of eddy-currents within the stator,
which are not captured by Model I. ........................................... 94

5-1 A given radial cross-section of the micro motor is analyzed by mapping it onto a
two-dimensional Cartesian plane, and assuming matching boundary conditions
on either side. ............................................................... 98
5-2 Meshing along the x-direction. ............................................. 100
5-3 Meshing scheme for the micro motor geometry. Each unit cell and its neighbor
along one direction share a common Δh along the orthogonal direction. ............. 102
5-4 Torque predictions by the linear model for the parameters of the motor tested.
The result from the linear eddy-current model for 6A is superimposed for com-
parison. ................................................................. 103
5-5 Magnitude of B field (first harmonic) within the magnetic induction micro machine, as predicted by the linear eddy-current model. The scale units on the right are in Teslas.

5-6 One of the simplest two-dimensional magnetic diffusion examples involves two magnetic cores sandwiching a wire carrying a traveling current.

5-7 The magnitude of the magnetic flux density vector (|\( \mathbf{B} \)|) along the \( \hat{x} \) direction. The snapshot is taken when the magnitude at the surface is at its maximum.

5-8 Magnetic flux density magnitude just inside the NiFe layer. As skin depth within the material becomes shorter than its thickness, magnetic flux at the boundary begins to "crowd".

5-9 Comparison of ANSOFT’s simulated torque predictions at 6A with measured torque values from the tethered magnetic induction motor. ANSOFT overpredicts the torque by more than 300%.

5-10 The smallest computation space needed for the FDTD scheme. For clarity, a diffuse mesh is shown.

5-11 The general meshing scheme used in an FDTD algorithm.

5-12 Surface for magnetic field update.

5-13 The measured magnetization curve of the NiFe around 7.5 kHz (stars) and the corresponding fit to data (thin solid curve). The slope of the curve at the origin and at high H values is \( \mu_0 \). Given a particular \( |\mathbf{B}| \), the corresponding magnetic field magnitude is found using the fit (as illustrated by the thick solid line).

5-14 The magnetic diffusion test case revisited. This time, the central wire carries either a standing-wave, or a traveling wave current excitation.

5-15 Snapshots of field variables \( (E_z, H_y, \text{and } B_y \text{ respectively}) \) within linear materials inside the computation space of Figure 5-14. The magnetic permeability of NiFe is taken as \( 3000\mu_0 \). The snapshots are taken at different phases of the input excitation: (a) phase 0; (b) phase \( \frac{\pi}{4} \); (c) phase \( \frac{\pi}{2} \); (d) phase \( \frac{3\pi}{4} \).
Snapshots of field variables \((E_z, H_y, \text{ and } B_y)\) respectively) within nonlinear materials inside the computation space of Figure 5-14. The snapshots are taken at different phases of the input excitation: (a) phase 0; (b) phase \(\frac{\pi}{4}\); (c) phase \(\frac{\pi}{2}\); (d) phase \(\frac{3\pi}{4}\).

Magnitude of magnetic flux density, \(|\mathbf{B}|\), magnetic field magnitude, \(|\mathbf{H}|\), and local magnetic permeability within the computational space of Figure 5-14, in the case when the excitation is a traveling wave, and the materials are assumed linear. The magnetic permeability of NiFe is taken to be \(3000\mu_0\).

Magnitude of magnetic flux density, \(|\mathbf{B}|\), magnetic field magnitude, \(|\mathbf{H}|\), and local magnetic permeability within the computational space of Figure 5-14, in the case when the excitation is a traveling wave, and the materials are assumed nonlinear. The magnetic permeability of NiFe is based on the measured magnetization curve in Figure 5-13.

Some keyframes of the magnitude of magnetic flux density within the tethered magnetic micro motor. Saturation is clearly evident in these simulation results.

Time evolution of torque for a sample current input (2 Amps, 30 kHz). The dashed line corresponds to the steady-state torque value extracted.

A radial cross-section schematic of the magnetic induction micro-motor, and the relevant dimensions used to simulate it.

Torque predictions from simulations for various tooth gaps. Measured values are overlayed for reference. The rotor conductor thickness is taken to be 8 \(\mu m\), and the temperature within the micro-motor is constant at 250°C above room temperature.

Torque predictions from simulations for various rotor conductor thicknesses. Measured values are overlayed for reference.

The rotor is recessed within the Kapton housing at a slight angle (exaggerated in this figure). The average recess is about 5 \(\mu m\). Tethers are not shown for clarity.
5-27 The time-evolution of simulated torque output from the magnetic micro motor, with two different B-H curves for the electroplated NiFe. The slightly lower steady-state value of torque corresponds to the B-H curve which is shifted towards lower H by a factor of two from the original curve.

5-28 The evolution of torque with two different B-H curves for the NiFe wafer. The two transients are essentially identical, except that the original B-H curve eventually results in numerical instabilities.

5-29 Simulated torque values for various temperature rises (from 300 K) within the electroplated NiFe of the stator and the rotor. Rotor conductor temperature taken constant at 250°C.

5-30 Torque predictions from simulations for various rotor Cu temperatures (and hence, rotor conductor resistivity). For purposes of comparison, the temperature has not been factored in for NiFe conductivity.

5-31 Simulated torque values at different overall temperatures within the micro-motor.

5-32 Simulated torque values at temperatures between 200°C and 300°C above room temperature make good fits to the measured data at 6 A.

5-33 Comparison of simulated power loss mechanisms within the test device. Power dissipation due to stator winding and contact resistance dominates the overall heat generation process.

5-34 Simulated torque values for 2A input current amplitude.

5-35 Simulated torque values for 3A input current amplitude.

5-36 Simulated torque values for 4A input current amplitude.

5-37 Simulated torque values for 5A input current amplitude.

5-38 A conceptual depiction of the range of validity of the three magnetodynamic models presented so far.

5-39 Potential lamination schemes for the stator. (Figure courtesy of Florent Cros of Georgia Institute of Technology)

A-1 A rectangle within the W-plane, with horizontal equipotential lines, is mapped into an L-shaped polygon in the z-plane.
A-2 The four dimensions that determine the size of the L-shaped region, and hence, the reluctance value $R_{te}$. ................................. 161

B-1 Electrical resistivity of NiFe as a function of the percentage of Ni in the alloy for various temperatures. ................................. 164
B-2 Electroplated NiFe conductivity as a function of temperature. ................................. 165
B-3 Copper conductivity as a function of temperature. ................................. 166

C-1 The stator will be laminated in an onion peel fashion, with laminations running perpendicular to the radial direction. With thin laminations, the magnetic field density on either side of a lamination block will be the same. ................................. 169
C-2 Magnetic field amplitude within the NiFe lamination block (top plot), and its square (bottom block) as a function of position in the $z$ direction of Figure C-1. 171
List of Tables

3.1 Dimensions of the fabricated tethers. ............................................ 63
3.2 Contents of the NiFe bath used for electroplated stator and rotor cores. Saccharin
is added to reduce the residual stress of the films, and hence to allow the plating
of thicker films. ................................................................. 65
Acknowledgments

I have the most gratitude for my advisor Jeffrey Lang, whose never ending patience, kindness, constant support, enthusiasm, and willingness to help have been strong sources of inspiration and motivation. I appreciate his advice on every aspect of research and life in the academic world in general.

I also thank Stephen Senturia for the many discussion sessions that helped me make sense of everything. Steve is an excellent mentor; he is caring, honest, and direct.

I feel extremely lucky to have collaborated with Florent Cros at the Georgia Institute of Technology side. Florent is a master in disguise when it comes to microfabricating; he is also a dear friend.

The experiments conducted as part of this thesis would not be possible without the generous support of equipment from Steven Leeb and Denny Freeman. Many thanks also to Jacob White for finding the time to have numerical algorithm discussions, which greatly helped me implement the finite-difference simulations described in this thesis. My thanks also go to Mark Allen, for making the fabrication of devices possible, and for keeping us honest.

I would like to thank all of the folks on the MIT Micro Gas Turbine Engine Project and in MTL, especially Carol Livermore, Xin Zhang, and the entire sixth floor, for providing a friendly, supportive environment to work in. I am also grateful to Kiyome Boyd Karin Janson-Strasswimmer for lending me their supportive ears when I needed it the most.

Big thanks to Dave, Annie, Jay, Xin, Xinwang, Zhu, Luca, Jung Hoon and Ole for support, and for allowing me to run large simulations at their computers. I am also indebted to Mr. Fred Donovan for letting me take over entire computer clusters at the Aero/Astro Department. Without the trust and patience of these kind people, this thesis would not be possible.

I wish to thank my Mom and Dad for their constant support and love during my entire time at MIT. A huge thank you to my sister, for allowing me to remember the light side of life whenever I needed it. Finally, I want to share my deep appreciation for Cansu Tunca. She has always believed in me, and I am grateful for her presence in my life.

This work was supported by the Army Research Office (Research Grant: DAAG55-98-1-0292) and DARPA (Research Grant: DABT63-98-C-0004).
Chapter 1

Introduction

The trend in the electronics industry has been faster, smaller, and cheaper since the advent of the first solid-state transistor more than half a century ago. Computer memory and processing speed per chip has been doubling every eighteen to twenty-four months since the late 1960’s (Figure 1-1), thanks mostly to innovations in integrated circuits (IC) fabrication. One of the direct consequences of this trend is an ever-increasing demand for systems offering mobile computing and telecommunication. Smaller and portable electronic systems are growing ever so hungry for lighter weight, denser power supplies. Moreover, power consumption of microprocessors is also on the rise, as Figure 1-2 illustrates.

For the last two decades, miniaturization and integration trends in IC technology have also caught up with the mechanical world. Fabrication techniques established initially for ICs are now being applied toward the realization of micron-scale mechanical systems. Such small mechanical devices are now commonly referred to as microelectromechanical systems, or MEMS for short. Initially interesting fabrication curiosities, MEMS devices are now being designed for virtually every possible actuation and sensing scheme imaginable, from micro rockets to high resolution displays. In pressure sensors, gyroscopes and accelerometers, the market size for MEMS is already in the multi-billion dollars. As stand-alone or integrated MEMS solutions continue to hit the market, the demand for miniaturized power sources is also on the rise.

Unfortunately, innovations in lightweight batteries have not kept pace with this demand for power. In most portable electronic devices, battery power and life cycle, not the electronics themselves, are the main bottlenecks for system performance. Another practical problem is that
Figure 1-1: The exponential rise in microprocessor transistor density has continued for the last three decades. (Source: Intel Corp.)

Figure 1-2: Power consumption of some common microprocessors over the last thirty years. (Source: Intel Corp.)
drained batteries need careful and special handling, due to the high toxicity of their ingredient chemicals. In an attempt to address these short-comings, a new branch of MEMS called power microelectromechanical systems has recently been defined. The idea underlying Power MEMS is to push the fabrication technologies inherited from the IC industry to new levels that will enable the production of micro-scale power generators. Researchers have taken different approaches to address this challenge, such as vibration-to-electric converters [1], or micro fuel cells [2]. These MEMS devices are engineering triumphs; nevertheless, they lack the power density required by state-of-the-art mobile electronics.

1.1 Micro Gas Turbine Generator

An ambitious research effort recently began at MIT to prototype a micro-scale gas turbine generator, capable of producing tens of watts of electrical power [3] (see Figure 1-3). The project to develop this engine envisions a high-speed rotating turbine that converts the chemical energy of its fuel into thrust and electricity. This is a very complex engineering undertaking requiring multi-disciplinary collaborations across the Institute. The development has been divided into
separate tasks, each of which aims to demonstrate a particular functionality on a test device. One of these development devices is the microcombuster and its associated test rig [12],[13], which endeavors to prove the feasibility and sustainability of micro scale combustion, and to extract the relevant parameters for continued operation. Another test device is the micro bearing rig [14], [15], [16], [17], which has tested microfabrication challenges and helped verify the turbine design. Yet another device is the micromotor-compressor, which is a test bed for the compressor functionality of the micro gas turbine generator.

Naturally, a critical component of the micro generator is the electromagnetic machine that both brings the turbine up to speed, and generates electricity once the gas turbine is operating. Given the high speeds of rotation and air bearings that rule out contact brushes, induction machines and variable capacitance/reluctance motors are the only viable electromagnetic components for the micro gas turbine generator. Compatibility with standard silicon fabrication schemes has been the main motivation that led to the fabrication and testing of an electric induction machine [5], [9] as part of the micro engine project. However, it has been quickly realized that the electric induction machine requires a very small air gap between its rotor and stator, resulting in large viscous losses at the air bearings. This means reduced speed of operation, lower power and lower efficiency. In order to provide a much larger operating air gap (hence, negligible drag losses) for the micro gas turbine generator, a magnetic counterpart to the electric induction machine has been proposed. The motivation behind the first magnetic induction machine has been to address these performance issues.

1.2 Magnetic Induction Machines

From refrigerator pumps to four-story tall turbine generators, magnetic induction motors and generators are the most pervasively used electrical machines on the macro scale. Figure 1-4 shows a typical magnetic induction motor that weighs over a metric ton. The standard design involves a rotor on a shaft that is concentrically inside a stator. Heating is a major concern in large induction machines; generally, part of the output mechanical power is used to cool the stator windings via forced convection of air. As a rule of thumb, power efficiency goes up with machine size. Depending on the frequency of operation, eddy currents within the magnetic
Toshiba EQP III-XS Severe

Max. power out: 200 HP
Weight: 2314 lbs.
Overall efficiency: 96.2%

\[
\begin{align*}
\text{Power/mass} &= 141.8 \text{ W/kg} \\
\text{Torque/mass} &= 0.76 \text{ N.m/kg}
\end{align*}
\]

Figure 1-4: A macro-scale magnetic induction motor.

materials may necessitate the use of laminated cores to reduce eddy-current losses, and hence, achieve greater efficiencies. We are interested in miniaturizing these machines into a practical and convenient micro machine to be used within the micro gas turbine engine.

1.3 Magnetic Induction Micro Machine

At the current size scales of the micro engine, magnetic induction machines offer power densities significantly greater than those promised by the electric induction counterparts. Also, a magnetic induction machine can achieve a given power density with a much lower pole count, which translates into a much larger rotor-stator gap compared to that inside an electric induction machine. This results in negligible viscous losses and higher efficiency.

The classical architecture of a rotor on a shaft inside a concentric stator is not practical for MEMS purposes. This is partly because it is much easier to define planar patterns using MEMS fabrication techniques than to create truly three-dimensional structures. Another reason is the fabrication constraints of the micro gas turbine itself. The magnetic induction micro machine chosen for implementation still involves an axially symmetric, cylindrical geometry; however,
Figure 1-5: The topology of the magnetic induction micro machine consists of planar surfaces extruded into the third dimension. The rotor spins on top of the stator, supported by air bearings. (Depiction courtesy of Florent Cros)

the motoring action takes place on the circular surfaces of the rotor and the stator (Figure 1-5), as opposed to the sidewalls as in a macro-scale induction machine. This topology enables the device to be built using current MEMS fabrication techniques, mainly planar photolithography and electroplating.

In fact, other magnetic micro machine types and geometries have been explored in the literature. The only other magnetic machine type compatible with the micro engine is the variable reluctance machine [6], [7], since it also requires no contact for rotor excitation (Figure 1-6). In theory, one could imagine different versions of the motors in Figure 1-6, adapted for the topology of Figure 1-5. However, the torque output from such devices are still orders of magnitude less than what is achievable with the magnetic induction micro machine. Indeed, no currently available device has the potential to offer the required energy density within the Micro Engine topology. Hence, magnetic induction micro machines appear to be natural fits for the purposes of the Micro Engine Project.
1.4 Thesis Scope

This thesis focuses on the study, characterization, and quantification of the performance of the first magnetic induction micro machine. In this thesis, this micro machine takes on the form of a micro motor whose rotor is suspended above the stator via mechanical springs, or tethers. The work includes a thorough theoretical study and modeling to predict the performance of the tethered micro motor, as well as the design, and testing of this machine. The purpose of the tethered micro motor is to isolate the electromechanical operation from the bearings and viscous losses, thereby providing a test bed for our models. The tethered micro motor also facilitates the testing of our fabrication techniques. The author undertook the fabrication of the rotor die, whereas the stators were fabricated by our collaborators at the Georgia Institute of Technology (GIT), Atlanta, Georgia.

Some of the basic research tasks undertaken as part of this thesis work are listed below.

1) Analysis: An electromagnetic study of a micro-scale magnetic induction machine was completed. The results of this analysis led to a design tool that enabled an understanding of the capabilities of this device, and predicted its performance. The design tool's predictions were cross-checked and validated via a finite element analysis (FEA) software.
2) **Design and Optimization:** Fabrication limitations, in conjunction with the demands of the gas turbine engine, define the available design space. Using some specific design objectives, such as maximizing system efficiency and power density at a given mechanical rotation speed, a set of criteria on the required geometrical dimensions of the machine were deduced. The final design is an optimization based on many iterations and trade-offs between high performance and fabrication limitations. A tethered magnetic micro motor was later built as a metrology device to test and validate both the design tool and fabrication processes.

3) **Process Sequence Development:** Once a candidate design was chosen, a detailed process sequence was developed in order to fabricate the tethered magnetic micro motor. This task naturally involved several cycles of design and process sequence development iterations until a plausible fabrication scheme was created for an acceptable design that incorporated fabrication limitations.

4) **Unit Process Development:** Given a fabrication process sequence, individual processing steps were detailed and studied for achieving the ultimate goal of a fabricated device. The unit process that is of utmost significance is the micromolding and electroplating (MIME) sequence \[6\], which enables fabrication of tall, high-aspect ratio metal structures. A special form of photosensitive epoxy (SU-8) was used both as a mold and as a structural material.

5) **Fabrication:** The tethered magnetic micro motor was fabricated using the unit processes mentioned above. The stator and the rotor dies were fabricated separately. For ease of fabrication, we chose to fabricate the tethered rotor die out of SU-8 epoxy, and integrate the NiFe rotor core afterwards; this scheme was undertaken by the author. The same approach was later adopted for the next generation rotor die whose outside structure was laser etched from Kapton films. It was observed that, due to its complexity, the stator fabrication process was the determining factor in the fabrication timeline. The stator fabrication and Kapton rotor development were performed by our collaborators.

6) **Testing:** This broad objective involved constructing the drive and sense electronics, heat sinking, packaging and interfacing all subsystems with a microvision setup that allows automated measurements of tether deflection as a function of electrical input excitation and slip frequencies. The resulting resonance curve data for each tethered rotor were converted into a torque and pull-in characterization. Comparisons with the predictions from the design tool
described above were made in order to validate the models.

7) Application: The tethered magnetic induction micro motor serves two main purposes: to validate the model and the design program predictions about the motor performance, and to serve as a test-bed for the magnetic starter/generator that will eventually be integrated into the micro engine. It is for the latter purpose that the specific lateral dimensions of the tethered motor have been chosen to fit inside the compressor section of the micro turbine engine. Hence, a major objective of this thesis was to provide a framework through which lessons learned from the tethered motor could be applied to later generation magnetic induction micro machines that get incorporated as an electromagnetic subsystem of the micro engine.

1.5 Thesis Overview and Contributions

The chapters that follow detail the scope of this thesis as outlined above. Chapter 2 introduces the linear, non-eddy current model (Model I) used to design the magnetic induction micro machine. A basic design for a particular machine geometry that can be fabricated within current MEMS fabrication limitations is given.

Chapter 3 details tether design considerations and summarizes the fabrication of the tethered magnetic induction micro motor. Here, the rotor fabrication that the author has engaged in is presented in some detail. Unit fabrication steps are outlined. Stator fabrication, undertaken at GIT, is explained briefly for completeness; the interested reader is referred to a more detailed account [11].

Chapter 4 describes the drive electronics, and the experimental setup for the tethered magnetic induction micro motor. The testing procedure using a microvision system [10] is outlined. The results of torque measurements from a micro motor with a six pole stator are presented. The discrepancy between Model I and the experimental results points out to the need for further, more sophisticated modeling, which is undertaken in the next chapter.

Chapter 5 is a detailed account of our modeling efforts that progressively include more physics. First, a linear, eddy current model based on finite differences (Model II) is described. The results from Model II indicate the need to model nonlinear magnetic saturation. The chapter then proceeds to develop a nonlinear, two-dimensional, finite-difference time-domain
solution (Model III) of magnetic diffusion, in particular within the tethered magnetic induction micro motor geometry. The results from Model III are then discussed within the context of the experimental conditions, and good agreement is shown with the test data.

Finally, Chapter 6 summarizes the activities, conclusions and contributions of this thesis. Recommendations for future work are also made.

Among the scientific contributions of this thesis, in collaboration with [11], to the MEMS field is the development of a fabrication scheme that enables a multistack of high aspect ratio, electroplated metals (core and windings) that are insulated from each other. This thesis focuses on the application of the unit process steps developed during the course of this work to the fabrication of a micro-scale magnetic induction machine. Moreover, the tethered magnetic micro motor is the first platform in which SU-8 epoxy is used as a mold for electroplating, as an insulating layer and as a structural layer (in the case of the rotor die) all at once. Therefore, part of the development efforts has been devoted to the analysis, design and fabrication of tether structures out of SU-8 epoxy.

The most fundamental engineering contribution of this thesis work to the field is the design and demonstration of the first MEMS scale magnetic induction machine. Another major contribution is the demonstration of the highest torque density MEMS magnetic machine to date.

One important factor that limits power output from macro scale magnetic motors and generators is cooling constraints. Even liquid cooling does not allow a macro-scale current density much higher than $10^7$ Amps/m$^2$. However, in the MEMS scale, cooling is much more effective. This is because cooling is mainly a surface phenomenon, whereas heating is a volume effect; as things get smaller, the ratio of the cooling surface area to the heating volume increases inversely with the typical dimension. In the case of the magnetic induction micro machine, electroplating ensures all materials are in good thermal contact with each other. The small size of the die, combined with high thermal conductivity of the electroplated metals, assures an isothermal structure that can be heat sunk very effectively. Current densities in excess of $10^9$ Amps/m$^2$ have already been achieved in the micro motor. Hence, effective heat sinking in the micro-scale allows a high current density that results in a high power density machine. This concept is bound to change the way people think about power in the MEMS scale.
1.6 Collaboration

This thesis work has been performed with Florent Cros and Prof. Mark G. Allen from Georgia Institute of Technology. In particular, process development and fabrication has taken place at the cleanroom facilities at GIT [11]. Also, Prof. Dennis Freeman of MIT has kindly agreed to supply a metrology equipment and its associated software: the MIT Computer Microvision System [10].
Bibliography


Chapter 2

Analysis

This chapter presents a compilation of our initial modeling efforts for magnetic induction micro machines. One of the most important outcomes of these efforts is a powerful software tool for analyzing the electromagnetic aspects of magnetic induction micro machines quickly and efficiently. This program enables us to make appropriate design decisions without the need for extensive iterations on a finite element analysis (FEA) package. Excellent agreement with FEA results are obtained nevertheless, which further justifies the value of this software.

We will begin by introducing a typical magnetic induction micro machine. What follows next is a compendium of the physics behind the particular model being used to design the machines. We will then present a typical design chosen for the first generation of induction micro machines.

2.1 The Big Picture

Induction machines on the macro scale generally involve a cylindrically symmetric geometry, with the rotor located concentrically inside the stator. While this configuration is ideal for producing maximum torque for a given volume, it is not possible to adopt in the case of a MEMS induction machine. MEMS fabrication constraints permit planar structures almost exclusively. Therefore, the magnetic induction micro machine is necessarily composed of planar structures, extruded in the axial dimension. Figure 2-1 below depicts a typical magnetic induction micro machine that could be built with current MEMS technology.
Figure 2-1: The rotor and the stator of a typical magnetic induction micro machine that will be built. The dimensions given are representative for the designed induction motor.

Instead of inside the stator, the rotor now stands just above it, supported by air bearings. The stator carries two windings in each wire slot, which drive a traveling magnetic field above its surface. Currents induced in the rotor conductor interact with the traveling field, which, in the right frequency range, yields net torque on the rotor. In the following sections, we discuss this phenomenon in some detail, deriving the relevant relationships for obtaining the output torque of the machine. Our approach in this chapter involves mapping the circular geometry of the micro machine into the Cartesian plane at different radii (Figure 2-2) and integrating our results over the entire active region to obtain numerical predictions about machine performance. The periodicity of the winding pattern is used to enforce periodic boundary conditions on the computation space, thereby reducing the problem size. The inherent assumption in this two-dimensional method is that the problem extends infinitely into the page. Analysis proceeds via a multi-modal approach that incorporates magnetic diffusion in two-dimensions for the uniform spatial layers just above the stator, and coupling this physics to a magnetic circuit analysis for the stator (Figure 2-3). Solutions to magnetic diffusion equations yield transfer relations at the interface of each layer of material above the stator. Given the tangential field just on
the stator surface (boundary 1 in Figure 2-3), the continuum rotor model outputs the normal flux at each boundary. In turn, the magnetic circuit stator model takes the normal flux at the stator surface as input, and gives the tangential field on that surface as its output, and as the input to the upper continuum model. This is how the two “half models” are combined to get the full magnetic induction micro machine model. The resulting equations are solved for each Fourier component (up to first 20 modes), and the output torque is computed based on all the Fourier modes considered.

2.2 The Rotor Section

Consider a conductor moving with constant velocity \( V \). The electric field and the current density inside the conductor are related by

\[
J = \sigma (E + V \times B) \tag{2.1}
\]
Figure 2-3: Magnetic circuit analysis for the stator is coupled to a multi-modal magnetic diffusion solution for the air gap and the rotor sections. The coupling is achieved through the flux values $\phi_i$ entering the stator, and the tangential magnetic field just on the stator surface.
Taking the curl of both sides in Equation 2.1 above, we get

\[ \nabla \times \left( \frac{J}{\sigma} \right) = \nabla \times E + \nabla \times (\nabla \times B) \]
\[ = -\frac{\partial B}{\partial t} + \nabla \times (\nabla \times B) \tag{2.2} \]

where we have used Faraday's law to substitute in for the curl of the electric field. Also, inside the conductor, \( \nabla \times H = J \), so

\[ \nabla \times \left( \frac{1}{\sigma} \nabla \times H \right) = -\frac{\partial B}{\partial t} + \nabla \times (\nabla \times B) \tag{2.3} \]

Equation 2.3 is used in the next section to obtain multilayer transfer relations that apply to the rotor section. It is interesting to note here, though, that the form of Equation 2.3 is that of a diffusion equation. Indeed, using \( B = \mu H \), and some vector identities\(^1\), Equation 2.3 becomes

\[
\frac{1}{\mu \sigma} \nabla^2 B = \frac{\partial B}{\partial t} + (\nabla \cdot \nabla) B
= \left( \frac{\partial}{\partial t} + (\nabla \cdot \nabla) \right) B \tag{2.4}
\]

which is the governing magnetic diffusion equation for linearly conducting and permeable materials [1].

### 2.2.1 Vector Potential Formulation

In order to derive the relations we will use to calculate the forces acting on the rotor, it is more useful to express Equation 2.3 in terms of the vector potential, and match the boundary conditions across different materials. Using \( \nabla \times A = B \) and choosing \( \nabla \cdot A = 0 \), Equation 2.3 within a given material becomes

\[
\nabla \times \left[ \frac{1}{\mu \sigma} \nabla \times (\nabla \times A) \right] = -\frac{\partial}{\partial t} (\nabla \times A) + \nabla \times (\nabla \times (\nabla \times A)) \tag{2.5}
\]

---

\(^1\nabla \times (\nabla \times B) \equiv \nabla (\nabla \cdot B) - B (\nabla \cdot \nabla) + B \nabla \nabla - V \cdot \nabla B = -V \cdot \nabla B \) where we have used \( \nabla \cdot B = 0 \) and assumed constant velocity (hence, \( \nabla \cdot \nabla = \nabla \nabla = 0 \)). Also, \( \nabla \times (\nabla \times B) \equiv \nabla (\nabla \cdot B) - \nabla^2 B = -\nabla^2 B \)
Figure 2-4: A slab of rotor conductor $\Delta$ thick, moving with constant velocity $V$ in the $x$ direction. The magnetic permeability and electrical conductivity are assumed to be uniform across the conductor.

\[ \nabla \times \left[ \frac{1}{\mu \sigma} \nabla \times (\nabla \times A) + \frac{\partial A}{\partial t} - (V \times (\nabla \times A)) \right] = 0 \quad (2.6) \]

Since $\nabla \times \nabla (-\phi) = 0$ for any scalar $-\phi$, we get

\[ \frac{1}{\mu \sigma} \nabla \times (\nabla \times A) + \frac{\partial A}{\partial t} - (V \times (\nabla \times A)) = -\nabla \phi \quad (2.7) \]

Equation 2.7 is linear in $A$, so for a given material, solutions for different Fourier modes can be superimposed. Using $\nabla \times (\nabla \times A) = -\nabla^2 A$ (which follows from the definition $\nabla \cdot A = 0$), the homogeneous solution is found to be the solution to

\[ \frac{1}{\mu \sigma} \nabla^2 A = \frac{\partial A}{\partial t} - V \times (\nabla \times A) \quad (2.8) \]

This is the vector potential equation that will be used to determine the magnetic field that diffuses inside the rotor conductor. In solving this equation, the cylindrical symmetry of the rotor is helpful. In particular, Equation 2.8 is solved for each incremental circular strip at a given radius from the rotor center, and the corresponding torque is then integrated along the radius. In this fashion, the problem essentially becomes two-dimensional and Cartesian. Figure 2-4 below shows a sketch of such a conductor strip, stretched out and moving at its corresponding tangential velocity, $V$. Notice that due to planar symmetry, the variation in $H$.
will be in the $x$-$y$ plane, hence $\mathbf{A}$ will be along the $\hat{z}$ direction: $\mathbf{A} = A(x,y)\hat{z}$. With $\mathbf{V} = V\hat{x}$, Equation 2.8 becomes

$$\frac{1}{\mu_0} \nabla^2 A = \frac{\partial A}{\partial t} - V \frac{\partial A}{\partial x}$$  \hspace{1cm} (2.9)$$

In anticipation of the next section in which a traveling sinusoidal drive is applied to the stator of an induction motor, let us take $A(x,y,t) = \text{Re}\left\{\hat{A}(y)e^{i(\omega t-kz)}\right\}$. Equation 2.9 then becomes

$$\frac{d^2 \hat{A}(y)}{dy^2} = \hat{A}(y) \left[ k^2 + \mu_0 j \omega - k j V \mu_0 \right]$$  \hspace{1cm} (2.10)$$

$$\Rightarrow \frac{d^2 \hat{A}(y)}{dy^2} - \gamma^2 \hat{A}(y) = 0, \text{ where } \gamma = \sqrt{k^2 + \mu_0 j \omega - k j V \mu_0}$$  \hspace{1cm} (2.11)$$

### 2.2.2 Diffusion Transfer Relations for a Planar Conductor Layer in Translation

For the conductor in Figure 2-4, the solutions to Equation 2.11 are of the form

$$\hat{A}(y) = \hat{A}_2 \frac{\sinh(\gamma y)}{\sinh(\gamma \Delta)} - \hat{A}_1 \frac{\sinh(\gamma (y - \Delta))}{\sinh(\gamma \Delta)}$$  \hspace{1cm} (2.12)$$

Using Equation 2.12 above and solving for $H_x = \frac{1}{\mu} \frac{d\hat{A}(y)}{dy}$ at $y = 0$ and $y = \Delta$, one gets the desired transfer relations:

$$\begin{pmatrix} \hat{H}_{1,x} \\ \hat{H}_{2,x} \end{pmatrix} = \gamma \begin{pmatrix} -\coth(\gamma \Delta) & 1 \\ -\frac{1}{\sinh(\gamma \Delta)} & \coth(\gamma \Delta) \end{pmatrix} \begin{pmatrix} \hat{A}_1 \\ \hat{A}_2 \end{pmatrix}$$  \hspace{1cm} (2.13)$$

or inverting this set of equations and using $\nabla \times \mathbf{A} = \mathbf{B}$, we have

$$\begin{pmatrix} \hat{A}_1 \\ \hat{A}_2 \end{pmatrix} = -\frac{j}{k} \begin{pmatrix} \hat{B}_{1,y} \\ \hat{B}_{2,y} \end{pmatrix} = \mu \gamma \begin{pmatrix} -\coth(\gamma \Delta) & 1 \\ -\frac{1}{\sinh(\gamma \Delta)} & \coth(\gamma \Delta) \end{pmatrix} \begin{pmatrix} \hat{H}_{1,x} \\ \hat{H}_{2,x} \end{pmatrix}$$  \hspace{1cm} (2.14)$$

36
Figure 2-5: Setup of the multilayered boundary problem for the rotor. The top air layer is taken thick enough to represent an infinite boundary.

Here, $\gamma = \sqrt{k^2 + j\mu\sigma(\omega - kV)}$ [1], and the $H_x$ coefficients are the complex amplitudes of the $x$ directed magnetic field at the surface indicated by the numeric subscript.

### 2.2.3 Rotor Model

Figure 2-5 shows a section of an induction machine air gap and rotor above the stator. For this geometry, the boundary conditions (for a given wavenumber $k$) at surfaces 1 through 8 are:

$$
H_{1,x} = \text{Re}\left\{\hat{H}_{ST}(k) e^{j(\omega t - kx)}\right\}
$$

$$
H_{2,x} = H_{3,x} \quad \& \quad B_{2,y} = B_{3,y} \quad \Rightarrow \quad A_2 = A_3
$$

$$
H_{4,x} = H_{5,x} \quad \& \quad B_{4,y} = B_{5,y} \quad \Rightarrow \quad A_4 = A_5
$$

$$
H_{6,x} = H_{7,x} \quad \& \quad B_{6,y} = B_{7,y} \quad \Rightarrow \quad A_6 = A_7
$$

$$
B_{8,y} = 0
$$

Here, $\hat{H}_{ST}(k)$ represents the Fourier amplitude (for the given wavenumber) of the tangential field at the stator surface. Applying the transfer relations we derived in the previous section to
each layer in Figure 2-5, we get

\[
\begin{align*}
\hat{A}_1 &= -\frac{j}{k} \begin{pmatrix} \hat{B}_{1,y} \\ \hat{B}_{2,y} \end{pmatrix} = \mu_0 \frac{1}{\gamma_0} \begin{pmatrix} -\coth(\gamma_0 d) & \frac{1}{\sinh(\gamma_0 d)} \\ -\frac{1}{\sinh(\gamma_0 d)} & \coth(\gamma_0 d) \end{pmatrix} \begin{pmatrix} \hat{H}_{ST} \\ \hat{H}_{2,x} \end{pmatrix} \\
\hat{A}_2 &= -\frac{j}{k} \begin{pmatrix} \hat{B}_{2,y} \\ \hat{B}_{4,y} \end{pmatrix} = \mu_0 \frac{1}{\gamma_c} \begin{pmatrix} -\coth(\gamma_c a) & \frac{1}{\sinh(\gamma_c a)} \\ -\frac{1}{\sinh(\gamma_c a)} & \coth(\gamma_c a) \end{pmatrix} \begin{pmatrix} \hat{H}_{2,x} \\ \hat{H}_{4,x} \end{pmatrix} \\
\hat{A}_3 &= -\frac{j}{k} \begin{pmatrix} \hat{B}_{4,y} \\ \hat{B}_{6,y} \end{pmatrix} = \mu_0 \frac{1}{\gamma_s} \begin{pmatrix} -\coth(\gamma_s b) & \frac{1}{\sinh(\gamma_s b)} \\ -\frac{1}{\sinh(\gamma_s b)} & \coth(\gamma_s b) \end{pmatrix} \begin{pmatrix} \hat{H}_{4,x} \\ \hat{H}_{6,x} \end{pmatrix} \\
\hat{A}_4 &= -\frac{j}{k} \begin{pmatrix} \hat{B}_{6,y} \\ 0 \end{pmatrix} = \mu_0 \frac{1}{\gamma_0} \begin{pmatrix} -\coth(\gamma_0 g) & \frac{1}{\sinh(\gamma_0 g)} \\ -\frac{1}{\sinh(\gamma_0 g)} & \coth(\gamma_0 g) \end{pmatrix} \begin{pmatrix} \hat{H}_{6,x} \\ \hat{H}_{8,x} \end{pmatrix}
\end{align*}
\]

Here, \(\gamma_i = \sqrt{k^2 + j\mu_i\sigma_i(\omega - kV)}\), where \(i\) is the layer index. Note that all field variables above are functions of a given wavenumber \(k\).

Once Equations 2.16-2.19 are solved, we can find the complex field amplitude at each boundary surface as a function of the amplitude, \(\hat{H}_{ST}\), of the traveling wave at the stator surface. In the most general formulation, the field at the stator surface is not a simple sinusoid in space. The argument so far is still applicable in the case, since one can express \(H_{ST}\) as the Fourier series

\[
H_{ST}(x,t) = \sum_{m} \hat{H}_{STm} e^{j(\omega t - kmx)}
\]

The Fourier components of the tangential magnetic field at the stator surface are orthogonal, and the multilayer structure discussed in the rotor model preserves that orthogonality. Hence, given each \(\hat{H}_{STm}\), Equations 2.16-2.19 can still be solved to yield the Fourier amplitude of each waveform at the corresponding boundary surface. The Fourier components of each waveform may then be summed in the same manner as in Equation 2.20.
2.2.4 Time-average Force

With a pure, temporally sinusoidal excitation, the time-average force per unit \( x-z \) area, \( \langle f \rangle_t \), is independent of \( x \). We find \( \langle f \rangle_t \) by integrating the Maxwell Stress Tensor \([1]\) over the surface \( M_1 \) through \( M_4 \) of Figure 2-5. The location of \( M_1..M_4 \) over the \( x \) direction is arbitrary. The time-average force over \( M_1 \) cancels that over \( M_3 \). The average shear force on surfaces parallel to the page is zero, since, by symmetry, field lines lie entirely within the \( x-y \) plane. The force on \( M_4 \) is also zero if the height of the top air boundary (\( g \) in Figure 2-5) is taken to be large enough. In general, however, the relevant shear and pull-in forces acting on the rotor are then found using the field values on the \( M_2 \) and \( M_4 \) surfaces\(^2\)

\[
\langle \tau_{xy} \rangle_t = \sum_m \frac{\mu_0}{2} \left[ \text{Re} \left\{ \hat{H}_{2,xm} \hat{H}^*_{2,ym} \right\} - \text{Re} \left\{ \hat{H}_{7,xm} \hat{H}^*_{7,ym} \right\} \right] \tag{2.21}
\]

\[
\langle \tau_{yy} \rangle_t = \sum_m \frac{\mu_0}{4} \left[ \text{Re} \left\{ \hat{H}_{2,ym} \hat{H}^*_{2,ym} - \hat{H}_{2,xm} \hat{H}^*_{2,xm} \right\} - \text{Re} \left\{ \hat{H}_{7,ym} \hat{H}^*_{7,ym} - \hat{H}_{7,xm} \hat{H}^*_{7,xm} \right\} \right] \tag{2.22}
\]

Equations 2.16 through 2.19 are solved for \( \hat{H}_{2,xm}, \hat{H}_{2,ym}, \hat{H}_{7,xm}, \) and \( \hat{H}_{7,ym} \) in terms of \( \hat{H}_{STm} \). Once an expression for each \( \hat{H}_{STm} \) is obtained from an analysis of the stator geometry, these values are substituted into Equations 2.21-2.22 to find the time-average shear and normal stresses on the rotor. Notice that, due to orthogonality, the contributions to the shear and normal stresses on the rotor from each Fourier wave number (as denoted by the index \( m \)) are independent of the rest. This fact allows the simple summation of each component in Equations 2.21 and 2.22 above.

2.2.5 Stator Model

The stator geometry is spatially non-uniform, with structures such as pole teeth, wire slots, etc. Hence, the modeling of the stator is more easily carried out using magnetic circuit models based on reluctance paths for the magnetic flux. Figure 2-6 below represents once such approach.

It can be seen from the model in Figure 2-6 that one can express the flux contributions

\(^2\)Here we use the identity \( \langle \text{Re} \{ A e^{j\omega t} \} \text{Re} \{ B e^{j\omega t} \} \rangle_t = \frac{1}{2} \text{Re} \{ \hat{A} \hat{B}^* \} = \frac{1}{2} \text{Re} \{ \hat{A}^* \hat{B} \} \)
Figure 2-6: Setup for the reluctance model of the stator. The current, $i$, passing through the wire gap is the sum of the currents in both windings.

\[ \pi/2 \text{ symmetry} \]

\[ x = 0 \quad x = T_p \pi/kL \quad x = \pi/4k \quad x = \pi/2k - T_p \pi/kL \quad x = \pi/2k \]

\[ L/4 \]

\[ T_p/2 \]

\[ L_p \]

\[ T_{bi} \]
entering the stator as

\[ \phi_{1m} = \int \frac{\pi}{4k} - \frac{\pi}{4k} \frac{1}{\mu_0} e^{j(\omega t - kmx)} \frac{jW}{km} \left( e^{-j \frac{\pi m}{4}} - e^{-j \frac{\pi T_p}{L}} \right) \]  

(2.23)

\[ \phi_{2m} = \int \frac{\pi}{4k} - \frac{\pi}{4k} \frac{1}{\mu_0} e^{j(\omega t - kmx)} \frac{jW}{km} \left( e^{-j \frac{\pi m}{2}} - e^{-j \frac{\pi T_p}{L}} \right) \]  

(2.24)

\[ \phi_{3m} = \int \frac{\pi}{4k} - \frac{\pi}{4k} \frac{1}{\mu_0} e^{j(\omega t - kmx)} \frac{jW}{km} \left( e^{-j \frac{\pi m}{4}} - e^{-j \frac{\pi T_p}{L}} \right) \]  

(2.25)

In this model, we consider the stator in subsections, each with a certain magnetic reluctance. By analogy with the electrical resistor, those stator regions that correspond to rectangular patches in which flux lines travel straight correspond to a magnetic reluctance that is directly proportional to length and inversely to cross-sectional area. Those reluctance values in Figure 2-6 corresponding to sections where flux travels straight are given by

\[ R_s = \frac{L_s}{\mu_0 T_l W} \]

(2.26)

\[ R_{t1} = \frac{L_t}{2\mu_s T_l W} \]

\[ R_{p} = \frac{(L_p/2)}{\mu_s T_p W} \]

\[ R_a = \frac{T_l + (L_p/2)}{\mu_s T_p W} \]

(2.27)

Those regions where flux lines bend are modeled using a conformal mapping technique known as the Schwarz-Christoffel transformation [3]. In Figure 2-6, the magnetic reluctance values of these L-shaped regions where flux lines bend are given by \( R_{dc} \) and \( R_{da} \), respectively. The details of how the conformal mapping method is used to extract the magnetic reluctances are given in Appendix A.

Solving for the resulting magnetic circuit proceeds just as solving KVL and KCL equations in an electrical circuit. With the definitions

\[ RR_s = 4R_{t1} + R_s + 2R_a + 2R_b + 2R_p \]  

(2.28)

\[ RR_1 = R_{t1} + R_a + \left( 1 - e^{-\frac{j\pi}{2}} \right) (R_b + R_p) \]  

(2.29)

\[ RR_2 = \left( e^{\frac{j\pi}{2}} - 1 \right) (R_b + R_p) - (R_{t1} + R_a) \]  

(2.30)

\[ RR_3 = \left( 1 - e^{-\frac{j3\pi}{2}} \right) (R_b + R_p) \]  

(2.31)
Figure 2-7: Tangential magnetic field just over the stator surface. The magnitude of each contribution has been taken as unity for illustration purposes. The ripples in the waveforms are due to truncation of the Fourier series.

we can solve for $\phi_s$ to obtain

$$\phi_{sm} = \frac{RR_1\phi_{1m} + RR_2\phi_{2m} + RR_3\phi_{3m} - i}{RR_s} \quad (2.32)$$

Ideally, $\phi_s$, as given by Equation 2.32, is the dominant factor that determines $H_{ST}$. This is because air will have a much lower magnetic permeability than the stator steel; hence, the gap between the teeth will have the highest reluctance, resulting in the highest “potential drop” anywhere in the magnetic circuit. However, for configurations in which the stator steel has a relatively low magnetic permeability, “potential drops” along the teeth contribute to $H_{ST}$ as well. Therefore, in the most general case, we must consider the $x$-directed flux along each tooth surface as well as $H_{ST}$.

Figure 2-7 presents an example where the relative extent of each such contribution is shown along the $x$-axis over one wire slot. In general, each waveform in Figure 2-7 has a different height (see Figure 2-8). The approximation that the $x$-component of the flux along the corner reluctance $R_{dc}$ decays linearly (as shown on either edge of the plot in Figure 2-7) is a good one,
Figure 2-8: Depiction of tangential magnetic field just over the stator along one wire slot.

as justified later with comparisons to FEA results.

Notice that Equations 2.16 through 2.19 are in terms of complex amplitudes of travelling waves of a given frequency and wavenumber. However, as Figure 2-7 shows, $H_{ST}$ is composed of periodic rectangular and triangular waveforms. This means that the tangential field at the stator surface must be written as a weighed sum of Fourier components, $H_{ST_m}$. This fact explains the ripples in the plot; they are due to truncation of the Fourier series after the twentieth term.

The height of each rectangle in Figure 2-8 is proportional to the tangential (x-directed) magnetic field through the corresponding section of the stator tooth. Hence, we have

$$H_{Sc} = \frac{\phi_s}{\mu_0 T_s W}; \ H_{Sr1} = H_{Sl1} = \frac{\phi_s}{\mu_s T_s W}; \ H_{Sr2} = \frac{\phi_s + \phi_2}{\mu_s T_s W}; \ H_{Sl2} = \frac{\phi_s - \phi_1}{\mu_s T_s W} \tag{2.33}$$

where $\mu_s$ is the magnetic permeability of the electroplated NiFe of the stator. Let us briefly consider the contributions to each $H_{ST}$ amplitude for a two-pole machine (the simplest stator geometry), focusing on the rectangular and triangular waveforms separately. In the following discussion, we will make use of the dimensions and quantities shown in Figure 2-8.
Rectangular Contributions

First, consider the contribution of the tangential magnetic field just over the tooth gap (through $R_s$) to each Fourier amplitude $\hat{H}_{ST,m}$. Recall that we are studying a two-pole machine, which consists of four wire slots, hence just four tooth gaps. Given a mode number $m$, the corresponding Fourier coefficient of each $H_{Sc}$ over these gaps is scaled with the appropriate phase, in accordance with the coordinate system of Figure 2-8, and added to find $\hat{H}_{STc,m}$. In other words, we have

$$
\hat{H}_{STc,m} = \frac{k}{2\pi} \left( \int_{\frac{\pi}{4k} - \frac{\nu}{2}}^{\frac{\pi}{4k} + \frac{\nu}{2}} H_{Sc} e^{jkmx} dx + \int_{\frac{3\pi}{4k} - \frac{\nu}{2}}^{\frac{3\pi}{4k} + \frac{\nu}{2}} H_{Sc} e^{jkmx} dx \right)
$$

$$
= \frac{kH_{Sc}}{2\pi jkm} 4e^{\frac{\pi m}{4}} \left( e^{\frac{jkmL_s}{2}} - e^{-\frac{jkmL_s}{2}} \right)
$$

$$
= \frac{4e^{\frac{\pi m}{4}}}{\pi m} \sin \left( \frac{kmL_s}{2} \right)
$$

where we have made use of Equations 2.23 through 2.33. Following a similar algebra, we find the expressions for the remaining rectangular contributions in Figure 2-8.

$$
\hat{H}_{STR1,m} = \frac{H_{SR1}}{2\pi jm} 4e^{\frac{\pi m}{4}} \left( e^{\frac{jkm(L_s+L_o)}{2}} - e^{-\frac{jkmL_s}{2}} \right)
$$

$$
\hat{H}_{STR2,m} = \frac{H_{SR2}}{2\pi jm} 4e^{\frac{\pi m}{4}} \left( e^{-\frac{jkm(L_s+L_o)}{2}} - e^{\frac{jkmL_s}{2}} \right)
$$

$$
\hat{H}_{STL1,m} = \frac{H_{SL1}}{2\pi jm} 4e^{\frac{\pi m}{4}} \left( e^{-\frac{jkmL_s}{2}} - e^{\frac{jkm(L_s+L_o)}{2}} \right)
$$

$$
\hat{H}_{STL2,m} = \frac{H_{SL2}}{2\pi jm} 4e^{\frac{\pi m}{4}} \left( e^{\frac{jkm(L_s+L_o)}{2}} - e^{-\frac{jkmL_s}{2}} \right)
$$

Triangular Contributions

In a similar fashion as above, we find the contribution to $\hat{H}_{ST,m}$ of the triangular shaped regions of tangential magnetic field in Figure 2-8. The triangular sections on the right of each wire gap yield the following Fourier coefficients

44
\[
\hat{H}_{ST_{riR,m}} = \frac{k}{2\pi \left( \frac{T_{wg}}{2} - \frac{\pi}{4k} \right) km^2} \left( \begin{array}{c}
\int_{\pi/4k + \frac{T_{wg}}{2}}^{\frac{\pi}{2k}} H_{R2} (x - \frac{\pi}{2k}) e^{jkmx} dx \\
+ \int_{\frac{\pi}{2k} + \frac{T_{wg}}{2}}^{\frac{3\pi}{2k}} H_{R2} (x - \frac{\pi}{k}) e^{jkmx} dx \\
+ \int_{\frac{3\pi}{2k} + \frac{T_{wg}}{2}}^{\frac{5\pi}{2k}} H_{R2} (x - \frac{3\pi}{2k}) e^{jkmx} dx \\
+ \int_{\frac{5\pi}{2k} + \frac{T_{wg}}{2}}^{\frac{3\pi}{4k}} H_{R2} (x - \frac{2\pi}{k}) e^{jkmx} dx 
\end{array} \right)
\]

\[
= \frac{H_{R2}}{8\pi \left( \frac{T_{wg}}{2} - \frac{\pi}{4k} \right) km^2} \left( \begin{array}{c}
4e^{\frac{1}{4} \pi jm} - \exp \left( \frac{1}{4} (\pi + 2T_{wg}k) jm \right) jm\pi \\
+2 \exp \left( \frac{1}{4} (\pi + 2T_{wg}k) jm \right) jmT_{wg}k \\
-4 \exp \left( \frac{1}{4} (\pi + 2T_{wg}k) jm \right) + 4e^{\frac{1}{4} \pi jm} \\
- \exp \left( \frac{1}{4} (3\pi + 2T_{wg}k) jm \right) jm\pi \\
+2 \exp \left( \frac{1}{4} (3\pi + 2T_{wg}k) jm \right) jmT_{wg}k \\
-4 \exp \left( \frac{1}{4} (3\pi + 2T_{wg}k) jm \right) + 4e^{\frac{3}{4} \pi jm} \\
- \exp \left( \frac{1}{4} (5\pi + 2T_{wg}k) jm \right) jm\pi \\
+2 \exp \left( \frac{1}{4} (5\pi + 2T_{wg}k) jm \right) jmT_{wg}k \\
-4 \exp \left( \frac{1}{4} (5\pi + 2T_{wg}k) jm \right) + 4e^{2\pi jm} \\
- \exp \left( \frac{1}{4} (7\pi + 2T_{wg}k) jm \right) jm\pi \\
+2 \exp \left( \frac{1}{4} (7\pi + 2T_{wg}k) jm \right) jmT_{wg}k \\
-4 \exp \left( \frac{1}{4} (7\pi + 2T_{wg}k) jm \right)
\end{array} \right)
\]

whereas the ones on the left yield
\[
\hat{H}_{ST}_{tri,m} = \frac{k}{2\pi \left(\frac{Twg}{2} - \frac{\pi}{4k}\right)} \left( \int_{\frac{\pi}{4k}}^{0} H_{S_{L2}}(x - 0) e^{jkmx} dx + \int_{\frac{3\pi}{4k}}^{\frac{\pi}{2}} H_{S_{L2}}(x - 0) e^{jkmx} dx + \int_{\frac{\pi}{2}}^{\frac{5\pi}{4k}} H_{S_{L2}}(x - \frac{\pi}{k}) e^{jkmx} dx + \int_{\frac{5\pi}{4k}}^{\frac{3\pi}{2}} H_{S_{L2}}(x - \frac{3\pi}{2k}) e^{jkmx} dx \right)
\]

Finally, we sum up the individual contributions to find the amplitude for each Fourier component.

\[
\hat{H}_{ST_m} = \hat{H}_{STc,m} + \hat{H}_{STR1,m} + \hat{H}_{STR2,m} + \hat{H}_{STtr1R,m} + \hat{H}_{STL1,m} + \hat{H}_{STL2,m} + \hat{H}_{STtriL,m} \quad (2.35)
\]

For completeness, \(H_{ST}\) over one period is shown in Figure 2-9, where the magnitude of each contributing section is chosen as unity for illustration purposes.

2.3 Implementation

The equations derived above, as well as other performance criteria such as heat dissipation in the windings, estimated hysteresis losses in the NiFe given the input frequency and amplitude,
Figure 2-9: $H_{ST}$ over one period in a two pole machine. Each neighboring section is 90 degrees out of phase. Contributions to $H_{ST}$ have been taken to have unity magnitude for simplicity.

windage losses, safety margin for possible magnetic saturation, etc. have been placed in the context of a software design engine. The backbone of this tool has been programmed in C for speed, whereas the user interface and data analysis is inside the MATLAB environment, which brings in flexibility and post-processing power. The associated software code for this design tool is included in Appendix E.

The graphical user interface allows the designer to select various geometries for the entire magnetic machine, as well as material properties, operating temperature, and number of machine poles, among others. Figure 2-10 shows the interface window of this program.

2.3.1 Finite Element Analysis Studies

A commercially available FEA package, ANSOFT, has been used to study various micro motor geometries. This software provides the capability to examine a great variety of electromagnetic phenomena, including eddy current effects and magnetic diffusion. Therefore, it is an ideal tool for investigating magnetic induction machines. Nevertheless, it is not practical to design the
Figure 2-10: Graphical user interface for the design program based on Model I. The numbers within the figure correspond to those in Figure 2-12.
Figure 2-11: Even for unrealistically low magnetic permeabilities such as $50\mu_0$ as used here, the model does an excellent job in performance estimation. The solid lines correspond to the model predictions, whereas the points are FEA results for the same geometry.

entire micro motor using ANSOFT; as with many other powerful FEA packages, simulations simply take too long. This is especially the case for the micro motor geometry, where thin but long structures necessitate very fine meshing for accuracy.

We have used this FEA software exclusively for validating our model in every stage of its development. For the case of a very large NiFe magnetic permeability, the model and the FEA results are indistinguishable. Initial measurements indicate that about the frequency ranges of interest, the relative permeability of NiFe is in excess of several thousand. Still, comparisons between results of the two tools have been made for relative permeabilities down to 50, and the worst case discrepancy has been found to be less than 5%, as Figure 2-11 shows.

It is important to note that the frequency of the peak is calculated accurately in the model. Notice from Figure 2-11 that the torque output of the rotor is underestimated, whereas the pull-in force is overestimated. This approach guarantees, among other things, that the air bearings that are designed accordingly will be strong enough for supporting the rotor.

In short, the design tool built based on our model is not only very fast and practical, but
also is robust to permeability and geometry variations. The agreement with the FEA and the design tool results is excellent. The MATLAB user interface and analysis functions allow designs to be completed in a matter of a few days, as opposed to up to many months on the ANSOFT package.

2.3.2 Initial Designs

Using the design tool described above, a first generation of micro motors, with 4, 6 and 8 poles, was designed. For these initial designs, care was taken to make sure the NiFe cores are safely far away from saturation\(^3\), which leads to somewhat pessimistic performance estimates. Figure 2-12 presents some performance assessments based on our model, and what is achievable by using currently tested fabrication techniques. It is important to emphasize that there are no eddy-currents modeled in the stator of Figure 2-12. The performance criteria listed in Figure 2-12 are for a six-pole machine that can be constructed using established fabrication capabilities with the Micro-Molding and Electroplating (MIME) technique. The thicker the electroplated structures (especially within the stator), the more difficult it is to fabricate a functional device. However, thicker structures (especially thicker teeth) enable more flux to pass through the stator before the onset of saturation, hence increasing maximum output torque that can be achieved. The conduction losses reported in Figure 2-12 are the \(I^2R\) losses within the copper in the stator and in the rotor (assuming 10\% slip). Performance can also be improved by increasing the wire heights (which will support a larger current with reduced stator heat losses, thereby increasing the efficiency).

The magnetic permeability of electroplated NiFe here is taken to be 3000\(\mu_0\), and that of the NiFe wafer is taken as 10^5\(\mu_0\), which are average values from the magnetization curves of these materials (see Appendix B). In Figure 2-12, the peak input current per phase is 17 A, which, given the wire cross-sections, corresponds to a current density of 6.9 \times 10^8 A/m^2. To put this number into perspective, current densities over 1 \times 10^9 A/m^2 simultaneously through both phases have been achieved in later tests with stators.

The windage loss and the output power level reported in Figure 2-12 are valid for the

\(^3\)The maximum allowed flux density within the micro machine is 0.86 T, which is \(|B_{sat}|\). Magnetization curve of electroplated NiFe is given in Appendix B.
Estimated Core losses at 50 kHz estimated
To be less than 0.5 W

Electronics Loss To be less than 0.5 W

Stator Cond. Loss 5.6 W

Rotor Cond. Loss 1.1 W

Windage Loss 1.3 W

48% Overall Efficiency
234 MW/m³

Net Mechanical Power Out 11.5 W

Electrical Power In 23.9 W

6 pole motor

Figure 2-12: Typical micromotor dimensions possible to achieve today, and the corresponding numbers for certain performance criteria. The radius of the machine is 2 mm. Input current is 17 A, corresponding to a current density of $6.9 \times 10^8$ A/m² (over $1 \times 10^9$ A/m² has been achieved later in tests). Magnetic permeability of electroplated NiFe is taken to be an average value of $3000\mu_0$. 

51
rotational speed of 2.4 Mrpm. The windage loss is given by [4]

\[
\text{windage loss} \approx \frac{\nu \pi \Omega^2 (R_{\text{out}}^4 - R_{\text{in}}^4)}{2d}
\]  

(2.36)

where \( \nu \) is the viscosity of air, \( \Omega \) is the angular speed of rotation \((2\pi \times 2.4 \text{ Mrpm}/60 \text{ rad/s})\), and \( d \) is the rotor-stator air gap. Here, the inner radius is 1 mm, and outer radius is 2 mm. Since the eventual micro machine will have thrust bearings at its center, the rotor-stator gap for \( r < R_{\text{in}} \) will be much larger than 25 \( \mu \text{m} \), hence, windage losses need to be considered only for \( R_{\text{in}} < r < R_{\text{out}} \). Notice that an air gap of 25 \( \mu \text{m} \) for the magnetic micro machine is an order of magnitude larger than the design gap for the electric micro machine [5], which means the windage losses in our case are an order of magnitude less.

The output mechanical power is calculated simply by multiplying the maximum torque from the machine with the angular speed of machine operation. For the machine parameters of Figure 2-12, the maximum torque is 45.8 \( \mu \text{N.m} \). The output power is then found as

\[
P_{\text{out}} = \frac{2\pi \times 2.4 \text{ Mrpm}}{60} \times 45.8 \mu \text{N.m} = 11.5 \text{ W}
\]  

(2.37)

The conductivity of the rotor NiFe in Figure 2-12 is taken as \( 5 \times 10^6 \text{ S/m} \), which is the measured value for our electroplated NiFe. Conductivity of the stator NiFe cannot be modeled due to the nature of the magnetic circuit model. This means that potential eddy-current effects within the rotor core are included in our model, but eddy-currents within the stator are ignored. Neglecting NiFe conductivity within the rotor in this model, for instance, results in a maximum power output of 14.5 \( \text{W} \), which is a 20% difference from the results in Figure 2-12. There is no way of knowing exactly how much the high conductivity of NiFe in the stator will impact the performance of the micro machine without either modeling eddy-currents explicitly, and/or characterizing the performance of a test device and comparing it with the model of this chapter. The design we are considering here is optimal given fabrication constraints and for a machine in which either laminations or magnetic materials with reduced conductivity deems eddy-currents negligible. Since the eventual magnetic induction micro machine will be laminated, this is a good starting point for our design. Moreover, if eddy-currents eventually prove very significant, our design is still “optimal”, in the sense that it will still yield the highest torque given eddy-
currents. In that case, however, the stator teeth or the backirons for both the stator and the rotor need not be as thick.

2.4 Summary

This chapter has developed and described a linear model used for analysis and design of the magnetic induction micro machine. This model will henceforth be referred to as Model I. First, the particular approach of mapping the circular geometry of the micro machine into the Cartesian plane was introduced. The magnetic micro machine was divided into two sections modeled separately: the stator, and the uniform layers above the stator. The stator was modeled using a magnetic circuit analysis, whereas the rotor and air gap portions were studied using transfer relations based on magnetic diffusion. A Schwarz-Christoffel transformation was used for those sections of the stator where flux lines bend. The two sections are coupled to each other through perpendicular magnetic flux entering the stator surface, and tangential magnetic field entering the air gap on the stator surface. High accuracy (with respect to FEA studies of the same geometry) is obtained by including many Fourier modes of the fields. The resulting model is implemented as a design tool via a combination of C programming language for speed and MATLAB for ease and post-processing power. This design tool is used to study various machine geometries and material properties, and to define the parameters for a magnetic induction micro machine that is “optimal” within prespecified fabrication constraints. An example of such an optimal configuration is presented.

The design tool described in this chapter is extremely fast and practical; it is also directly applicable to macro-scale magnetic machine design as well as micro. However, in the context of the magnetic materials available for microfabrication, the model developed in this chapter has a particular shortcoming: it does not model the potential effects of eddy-currents within the stator. Though eddy-currents within the rotor core are modeled and their effects are found small, eddy-currents within the stator are likely to be much more significant, as we shall indeed see in Chapter 5. In the end, we need a physical device to test our model in the context of the magnetic induction micro machine. In the next chapter, we discuss the fabrication of one such test device: the tethered magnetic induction micro motor.
Bibliography


Chapter 3

Fabrication

The electromagnetic model and the associated design tool of the preceding chapter must be tested with a device that captures the underlying electromagnetic phenomena. This device should allow the separation of the electromagnetic actuation from the complicating effects of high speed rotary machine dynamics, such as viscous losses. The construction of this test device should also serve to further reveal the fabrication challenges and constraints associated with thick electroplated MEMS structures. An ideal candidate for this test bed is a motor in which the rotor is suspended on tethers, or springs, above the stator. Torque can then be measured through the bending of the tethers. This chapter focuses on the basic tether design and the fabrication of such a tethered micro motor.

The fabrication processes outlined in this chapter have taken place at Georgia Institute of Technology (GIT) Microelectronics Research Laboratories. The author has engaged in tether design and rotor mask preparation, and has even contributed to the fabrication of the rotor dies. In retrospect, this chapter outlines tether design and rotor fabrication. As most of the process development and the entire stator fabrication has been performed by our collaborators at GIT, we will not dwell on the intricate details of the process flow for the stator; these details can be found in [5]. However, the fabrication steps will be summarized for completeness.
3.1 Tethered Motor Concept

Figures 3-1 through 3-3 depict three-dimensional computer drawings of parts of the tethered micro motor. These drawings are based on scaled versions of the actual mask set, hence are useful depictions for visualizing the tethered micro motor. The rotor die is constructed either out of developed SU-8 epoxy, or laser-etched Kapton film. The stator die is composed of electroplated NiFe and Cu using SU-8 epoxy molds on top of a commercially available NiFe disc that functions as the stator backiron. The rotor die is designed to be placed on top of the stator die (Figure 3-2) and aligned using ceramic alignment posts (see Chapter 4 for the discussion on packaging).

3.2 Tether Design

The tethers hold the rotor core in place and provide a restoring force when torque is applied to the rotor. They are extremely stiff in the out-of-plane direction, and much more compliant for rotational motions in the plane of the die. The number of tethers is chosen with some
Figure 3-2: Rendered image of the tethered rotor die aligned on top of the stator die (the rotor at the center not shown). Hexagonal holes are for the sacrificial layer etch that releases the rotor die. Three large holes on the top left allow current probes to make contact with the stator windings.
Figure 3-3: Rendered image of a side view of the tethered micro motor. The rotor-stator gap, and the vertical features of the stator are exaggerated for clarity; otherwise, the relative dimensions are accurate.
Figure 3-4: A computer rendered image of the rotated rotor core and the twisted tethers (the rest of the die is not shown). Bending in the tethers is exaggerated for clarity.

discretion; six seems to be the convenient number that prevents out-of-plane rotations, while providing enough compliance for observable in-plane deflections. At the end where a tether attaches to the substrate, it is subject to fixed position and zero slope boundary conditions; at the other end, the tether can deflect with a zero slope boundary condition with respect to the rotated coordinate axes of the rotor. Figure 3-4 illustrates the twisting rotor with Kapton tethers. For easier future analysis, linear mechanical behavior is desired, which means the tethers should be designed to deflect by no more than a fraction of their width.

Let us adopt the convention that the thickness of a tether is defined as the cross-sectional dimension in the direction of motion. For a well-guided design, it is desirable to know how much the tethers will deflect under the influence of a given amount of torque. Luckily, calculating the exact spring constant of the six-tether system is unnecessary for our design purposes. All we need is a first-order estimate. A very good estimate can indeed be obtained by realizing that tether tip rotation at the rotor base can be neglected for small rotations and long enough tethers. In that case, the boundary conditions for the deflection of each tether are the same as those of a beam clamped at either end with a point load at the center – just consider half that beam, from one clamped end to the center (Figure 3-5). In fact, the spring constant of
Figure 3-5: The boundary conditions that the tethers are subject to are essentially the same as those of half a clamped-clamped beam deflected by a point load at the center.

our tethers can be approximated well by half that of the center loaded doubly clamped beam.

The spring constant of the doubly clamped beam with a point load at its center can be found using a trial solution to the beam equations that capture the energies due to bending and stretching [1]. Consider a doubly-clamped beam of length $L$, width $W$, and thickness $H$, with a Young’s modulus of $E$, and an internal stress of $\sigma_\circ$. Substituting a cosine-shaped trial function into the beam equations, we find that the force, $F$, applied at the center is related to the center deflection, $c$, as

$$F = \left\{ \left( \frac{\pi^2}{2} \right) \left( \frac{\sigma_\circ WH}{L} \right) + \left( \frac{\pi^4}{6} \right) \left( \frac{EH^3W}{L^3} \right) \right\} c + \left( \frac{\pi^4}{8} \right) \left( \frac{EHW}{L^3} \right) c^3$$

(3.1)

For small deflections (where $c \ll H$), the $c^3$ term is negligible. In that case, half the coefficient of the term linear in the center deflection gives the spring constant of a single tether, i.e.

$$k_z = \frac{N}{2} \left[ \left( \frac{\pi^2}{2} \right) \left( \frac{\sigma_\circ WH}{L} \right) + \left( \frac{\pi^4}{6} \right) \left( \frac{EH^3W}{L^3} \right) \right]$$

(3.2)

where $N$ is the number of tethers. Keep in mind that the length $L$ in Equation 3.2 is twice the length of each tether.

A critical number in Equation 3.2 is the Young’s modulus of the tether material. During the tether design phase, the Young’s modulus of SU-8 epoxy was somewhat of a mystery, as the very limited amount of relevant literature is not in consensus. Part of the reason for the
Figure 3-6: Thickness variation of along the length of the test beam and the tip deflection with attached masses. The two plots are used to deduce the Young’s modulus of the SU-8 epoxy.

lack of reliable data for the mechanical properties of SU-8 epoxy is that this material is not typically used as a mechanical MEMS component. Hence, we have opted to create our own test structures and experiments to determine the Young’s modulus of thick SU-8. The test structures consist of cantilever beams, 1.5 mm wide and 15 mm long, with a thickness tapering as shown in the top plot of Figure 3-6. By attaching small, calibrated lead weights on the tip of the cantilever, the effective spring constant of the beam can be obtained (inverse slope of the bottom fit curve in Figure 3-6).

The ordinary differential equation that describes the linear deflection of a cantilever beam under a point load \( F \) applied at its tip is given by [1]

\[
\frac{d^2w}{dx^2} = \frac{12F(L-x)}{EWH^3}
\]  

(3.3)

where \( L, W, \) and \( H \) depict the length, the width, and the height of the beam, respectively, and
$x$ is the distance from the fixed end of the cantilever. Here, $H$ is a function of position $x$. If the taper is gradual, we may simply integrate Equation 3.3 to find a very good approximation for the resulting tip deflection. The Young’s modulus can hence be found by iterating Equation 3.3 for various values of $E$ until the measured deflection is matched. The Young’s modulus of SU-8 epoxy, as measured by this technique, is about 3.5 GPa.

Given the Young’s modulus of SU-8, Equation 3.2 predicts that six tethers, each of which is 700 $\mu$m long, 500 $\mu$m wide and 60 $\mu$m thick, result in about 1.5 $\mu$m of deflection under the influence of 20 $\mu$Nm of torque. Here, we assume a rotor radius of 2.0 mm, and an enclosing ring that is about 0.1 mm thick. Of course, the deflection at resonance will be much larger; however, for typical quality factors of a few tens, the deflections will still be much smaller than the tether thickness until we get to very high currents. The maximum vertical deflection due to a pull-in force as large as 1 N can be found by simply interchanging $W$ and $H$ in Equation 3.2; it evaluates to 2.1 $\mu$m. Hence, if we assume a nominal stator-rotor air gap of 25 $\mu$m, we find that the worst change in the stator-rotor air gap is less than 10%. For comparison, the rotational deflection and the vertical pull-in have been calculated using an alternative, perhaps a more accurate formulation that captures the correct tether behavior [2]. Using this alternative approach for the same tether dimensions as above, the in-plane deflection and the out-of-plane pull-in turn out to be 1.1 $\mu$m and 1.6 $\mu$m, respectively. It is important to point out that these numbers are worst case scenarios, assuming very large torque levels can be achieved experimentally. Linearity of the deflections can be confirmed by inspection once the mechanical resonance curves of the tethered micro motor are obtained.

Tethers have also been constructed out of Kapton, which offers much better thermal tolerance. In fact, Kapton withstands temperatures much higher than the glass transition temperature of cured SU-8 epoxy, which is around 200 °C. This is important at high stator current levels which may cause the tethered motor to heat up substantially. Cantilever test structures die sawed out of Kapton have been used to measure its Young’s modulus, which is about 4.54 GPa (see Figure 3-7). The Kapton tethers used in our experiments are laser etched at GIT facilities. A rotor die based on Kapton has been fabricated after initial tests with SU-8 tethers. Tether dimensions that are chosen for either material are summarized in Table 3.1.

Figure 3-8 illustrates the cross-section of one of the Kapton tethers used in the tethered
Figure 3-7: Tip deflection of a Kapton cantilever beam as a function of attached weight at the tip. The extracted Young’s modulus of Kapton is about 4.54 GPa.

<table>
<thead>
<tr>
<th>Tether material</th>
<th>Y.M. (Gpa)</th>
<th>L (μm)</th>
<th>W (μm)</th>
<th>T (μm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SU-8 epoxy</td>
<td>3.5</td>
<td>700</td>
<td>500</td>
<td>60</td>
</tr>
<tr>
<td>Kapton</td>
<td>4.5</td>
<td>1230</td>
<td>764</td>
<td>57 (ave)</td>
</tr>
</tbody>
</table>

Table 3.1: Dimensions of the fabricated tethers.
Figure 3-8: Cross-section of a Kapton tether shows a linear taper (Figure rotated counterclockwise by quarter of a turn). The thin tip corresponds to the surface side on which laser drilling started.

micro motor experiments. The thickness value for the Kapton tethers listed in Table 3.1 is an overall average of the linear taper in thickness, which is a result of the laser etching procedure. As the laser drill moves deeper into the substrate, it melts more of the surrounding Kapton. The thin top surface of the Kapton tethers is where laser drilling started, and therefore where the laser beam spent the longest melting sidewalls. Figure 3-9 depicts measured tether thickness values at the top and the bottom of the Kapton rotor die.

The taper in the tether thickness is bad news, as coming up with an analytical formula for the spring constant is complicated by the fact that lateral force acting on a tether will not be uniformly distributed along the width of the tether. Also, tether stiffness is extremely sensitive to slight deviations from linear in the taper. A good approximate answer may be obtained through the use of a mechanical finite element analysis (FEA) software. However, we simply need an approximate value; a 50-100% error is acceptable. For that, we can imagine dividing the tether into parallel strips of varying width along its thickness, and sum the resulting spring constants obtained using Equation 3.2 on each thin strip. That method predicts a lateral tether deflection of about 1.5 μm under the influence of 20 μN.m.

3.3 Fabrication Flows

In this section, we briefly discuss the main steps of rotor and stator fabrication. The main process that is common to both the rotor and the stator fabrication is NiFe electroplating. The interested reader is referred to some of the references ([3], [4]) which present a detailed account of the electroplating conditions and fabrication flows for structures made using electroplated
Table 3.2: Contents of the NiFe bath used for electroplated stator and rotor cores. Saccharin is added to reduce the residual stress of the films, and hence to allow the plating of thicker films.

<table>
<thead>
<tr>
<th>Chemical</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>NiSO₄·7H₂O</td>
<td>200 g/L</td>
</tr>
<tr>
<td>NiCl₂·6H₂O</td>
<td>5 g/L</td>
</tr>
<tr>
<td>FeSO₄·6H₂O</td>
<td>8 g/L</td>
</tr>
<tr>
<td>H₃BO₃</td>
<td>25 g/L</td>
</tr>
<tr>
<td>Saccharin</td>
<td>3 g/L</td>
</tr>
</tbody>
</table>

3.3.1 Rotor Fabrication

The fabrication of the rotor is considerably simpler than that of the stator. However, the unit processes are essentially the same. Rotor fabrication depends heavily on SU-8 epoxy.

All processes involving SU-8 epoxy have the same progression of fabrication steps: spin coat, soft bake, expose, post bake, and develop. First, a thin (5-10 μm) layer of sacrificial aluminum is deposited on the silicon substrate using a DC sputtering tool (Figure 3-10, step 1). Later, SU-8 epoxy is spun over the wafer to yield a 500 μm thick layer over the aluminum (step 2). The epoxy is then soft baked on a leveled hot plate at 100 °C for several hours. This is necessary
1) DC Sputter 5-10 μm of Al

2) Spin and cure 500-600 μm of SU-8 Epoxy

3) Shoot, cure and develop the SU-8

Two ways to integrate the rotor disk with the tether structure:

4) DC Sputter Cu and electroplate the NiFe

4) Place and glue the NiFe disk externally

5) Release tethered rotor in 10 % HF sol’n

5) Release tethered rotor in 10 % HF sol’n

---

Figure 3-10: Fabrication flow of the rotor die.
to evaporate the solvent and to densify the film. Following contact-mask exposure under a near UV (350-400 nm) light source, the wafer is post baked for about half an hour for selective cross-linking of the exposed portions of the SU-8 film. The cured epoxy is developed under an ultrasonic bath, and rinsed with acetone before drying under a gentle stream of nitrogen (step 3). At this stage, copper and NiFe can either be directly electroplated within the rotor housings to form the rotor core, or the rotor disk could be manufactured separately and put in place later. We have experimented with both approaches, and have observed that the latter approach allows us the flexibility to optimize the fabrication of the SU-8 mechanical structure and the rotor disk separately. Therefore, the separate fabrication method is the method of choice here. Finally, the sacrificial aluminum layer is etched away in a 10% HF solution, releasing the rotor dies. Figure 3-11 shows some microscope and SEM pictures of the finished rotor die. The honeycomb structures seen in Figure 3-11 (a) and (b) come quite handy during the final sacrificial layer etch.

3.3.2 Stator Fabrication

The primary challenge in fabricating the magnetic induction micro machine is interspersing and stacking thick multi-layer conductors and large volumes of magnetic material to form the stator. We have met this challenge by using a Micro-Molding and Electroplating (MIME) process, in which SU-8 epoxy is used as the mold material. Figure 3-12 illustrates the unit process for the stator. Once copper or NiFe is electroplated within the molds of a thick SU-8 film, the top surface is lapped and polished. A new layer of SU-8 starts the unit process for the subsequent levels. Figure 3-13 summarizes the fabrication flow for the stator. Details about the fabrication of the stator, as well as the process flow development, can be found in [6] and [5]. Figure 3-14 show some microscope and SEM pictures of the stator.

\[^1\text{It is worth noting, however, that fully functional magnetic induction micro machines may eventually require the integration of magnetic MEMS fabrication technologies with those of single crystal silicon micro-machining.}\]
Figure 3-11: Pictures of the fabricated SU-8 rotor: (a) The fabricated SU-8 rotor die, with the rotor core glued at the center; (b) a close-up view of the sacrificial etch holes; (c) the center rotor housing ring and tethers – the bottom surface is a conductive tape used within the SEM; (d) one of the tethers; (e) in a close-up view, surface cracks at the base of the tether are clearly visible; (f) tether surface from the top, revealing the side-wall roughness which is due to the photoreduction masks.
Figure 3-12: Unit fabrication process for the stator (Figure courtesy of Florent Cros).

Figure 3-13: Fabrication flow for the stator die (Figure courtesy of Florent Cros).
Figure 3-14: Stator pictures, courtesy of Florent Cros: (a) Copper windings without the electroplated NiFe poles; (b) Complete stator (early generation); (c) SEM picture of the stator, with the top insulating SU-8 layer removed, revealing details of the teeth and the top copper winding; (d) A close-up SEM view of the stator around the inner end-turns, with the teeth removed to reveal detail.
Bibliography


Chapter 4

Experiments

The tethered magnetic induction motor is intended to be an accurate torque-metrology device. In this context, it can be considered as a magnetic transducer, with the stator and the rotor acting together as the actuator, and the tethers as the spring support structure for the moving rotor. In order to measure the torque output from this transducer, the tethered motor stator is excited to produce a square-wave torque acting on the rotor. In response, the tethered rotor twists back and forth periodically, in fashion similar to the agitators of top-loading washing machines during a wash cycle. A machine-vision system [1] is used to measure the rotation amplitude of the rotor. The rotor mass, the attached tethers and the air damping associated with the motion constitute a linear, second-order mechanical system. By varying the torque reversal frequency, the frequency response of this system can be measured, and by fitting a second-order linear resonance curve to this response, the magnitude of the applied torque can be extracted.

In this chapter, we first introduce the experimental setup developed to test the tethered magnetic induction micro machine. Later, we discuss the relevant experimental results obtained. For consistency, we will henceforth refer to the frequency at which the input torque is reversed as the “torque reversal frequency”. The excitation frequency of the stators will remain as the “stator excitation frequency”.
Figure 4-1: The packaging assembly for the magnetic induction micro machine. Once the micro machine is aligned, spring loaded screws are tightened, pressing the top plate (the rotor die) gently onto the bottom aluminum plate (the stator die).

4.1 Test Setup

4.1.1 General Assembly

The magnetic induction micro machine contains alignment holes at the corners of both the stator and the rotor dies. These holes enable the fast alignment of the rotor die on top of the stator die through the use of machined ceramic pins. A special packaging unit houses the micro machine (see Figure 4-1). The top plastic plate is transparent, which is convenient when aligning the two dies. The bottom plate is machined out of thin aluminum to provide a good thermal contact between the stator and the heat sink underneath.

The packaging unit sits on top of a HiContact™ liquid-cooled cold plate from AAVID Thermalloy, LLC (typical dimensions shown in Figure 4-2). The specification provided by the
manufacturing company recommends a maximum flow rate of 1.5 gallons per minute (GPM), which is the flow rate used during the tethered motor tests (Figure 4-3). The heat sink cools both the micro machine and the power transistors that provide the current inputs for the stator. Thermal grease is applied at the bottom surfaces of both the packaging unit and the transistors for good thermal contact. The transistors additionally require electrical isolation from each other\(^1\); a commercially available thin thermal tape that is both electrically insulating and thermally conducting is used for this purpose.

The packaging unit and the heat sink sit on top of a small platform that carries the power electronics, associated high-current connections, decoupling capacitors and the micro-manipulators used to position current leads in and out of the micro machine (Figure 4-4). The platform is, in turn, on top of a Zeiss microscope stage, which can be removed from the microscope for easy access and modification (Figure 4-5). The microscope is part of the Microvision system [1] that allows precise measurements of periodic deflections within MEMS devices. When the stage is attached to the microscope, the center of the test die is directly underneath the objective. The stage allows the platform to be maneuvered in plane over a limited range. Once

---

\(^1\)This is because the thermal contact surface of the TO-220 package that the transistors come in is also their drain terminal.
Figure 4-3: Cooling performance and the pressure drop in the plumbing vs. flow rate. The recommended maximum flow rate is 1.5 GPM. (Source: AAVID Thermalloy specification sheets, also at http://www.aavid.com/datasheets/ThermoModules/index.html)
Figure 4-4: The test setup for the tethered magnetic induction micro machine.
Figure 4-5: The microvision system and the test electronics.
the region of interest is within the field of view, focusing can be achieved either manually, or through software control using the piezoelectric objective lens, which offers 100 μm of focusing range. A charge-coupled device (CCD) camera picks up the image in real time and sends it to the Microvision system controller computer, where further processing on the image can be performed. Figure 4-6 shows the rest of the hardware involved in the experiment setup.

4.1.2 Electronics

Each phase winding in the stator is driven by a power amplifier that acts as a voltage-controlled current source. These amplifiers deliver a balanced excitation (up to 20A and 120 kHz) in quadrature through the two windings in the stator. The input waveforms are provided by the two waveform generators (Agilent 33120A) shown in Figure 4-6. A special switch circuitry is implemented to alternate the inputs to the stator phases, thereby changing the direction of the applied torque. Figure 4-7 illustrates a basic diagram of the electronic setup.

Power Electronics

The high frequency and the high current requirements of the micro machine offer a challenge in the design of the power electronics. Since no commercially available product that could satisfactorily meet these requirements could be identified, the drive circuitry has been designed and built from scratch. Figure 4-8 depicts the wiring schematic of the power circuitry that drives one of the stator phases. The underlying idea is the same as that used in stereo power amplifiers for speakers. A “push-pull” stage, formed by the two CMOS power transistors, is wrapped within a feedback loop that tries to get the current output be proportional to the voltage input. The feedback amplifiers used must have a very high gain-bandwidth product in order for the feedback to act quickly enough on an already 120 kHz fast signal.

The other challenge for the amplifiers is a very high slew rate. The PMOS and the NMOS power transistors have gate-to-source threshold voltages around 3V and -3V, respectively. Therefore, in the push-pull configuration, there exists a gate input voltage zone, between -3V to 3V, in which both transistors are effectively off. This zone is referred to as the “dead zone”. For robust operation, the feedback amplifier outputs have to swing as fast as possible out of this dead zone, a feat that requires a high slew rate. The burden on the feedback amplifiers
Figure 4-6: The hardware for the tethered magnetic micro machine test setup.
can be eased by the addition of four diodes in series at each gate, as seen in Figure 4-8. At a voltage drop of just over 0.6V across each diode, the overall effect is the reduction of the size of the dead zone, from over 6V to about 1V.

**Torque Reversal Circuitry**

A torque-reversal switch which swaps the driver inputs between the phases is implemented to switch the direction of the applied torque. Figure 4-9 depicts the wiring diagram of this circuitry that corresponds to one of the phases. This switch operates at frequencies far lower than the input electrical frequency. In this fashion, the torque delivered by the stator approximates a square-wave pattern with a period determined by the operating frequency of the reversal switch. By periodically reversing the torque, the mechanical resonance of the rotor and tethers can be excited. The response of this resonance as a function of torque reversal frequency is then used to determine the torque on the rotor. During testing, it has been observed that the switched driver signals exhibit a slight overshoot which settles within a few microseconds, two orders of magnitude faster than the highest frequency torque reversal signal, which is 5 kHz.

The 10 kΩ potentiometers are used to manually adjust the gain of each signal path. Impedance...
Figure 4.8: Wiring schematic of the power circuitry. Only the circuit for one phase is shown; the electronics for the other phase is identical.
Figure 4.2: Torque reversal switch circuitry. Only the circuit for one of the phases is shown here; the other is identical.

Waveform Generator I Signal

Square-wave input from the signal generator controlled by the Microvision Controller PC

Feed into Phase I input of the driver

Waveform Generator II Signal
mismatches through the entire driver circuitry may alter the amplitudes of the input signals, especially at high frequencies; these variable resistors allow for ensuring that both the switched and unswitched phase inputs have matching amplitudes at a given frequency.

4.1.3 Experiment Concept

The very first experiment performed on the micro motors is an iron powder test on the stator. In this simple procedure, the functionality of the newly fabricated stators is tested by spilling a tiny amount of iron powder on the stator. With the two stator windings excited in phase, the dispersal pattern of the iron powder on the stator surface should fill every other tooth slot, as the currents in the two windings will act to cancel each other out in one wire slot and reinforce each other in the neighboring slot. If the windings are excited out of phase, the dispersal pattern should be the same, rotated by one wire slot. In fact, this procedure has enabled us to determine a mask set flaw in the first generation of fabricated stators (see Figure 4-10). Once a stator passes this test, it is excited with the phases in quadrature (90° out of phase) at around 1 Hz, which creates a slowly traveling wave on the surface. The stator is deemed a potential candidate for the Microvision experiment if the iron particles on its surface can continuously travel around that surface. The “good” stator is then packaged with the rotor die for the tethered micro motor experiments. Figure 4-11 illustrates how the test setup for the tethered magnetic induction micro machine functions. Figure 4-12 depicts some of the relevant signals at selected locations in Figure 4-11. The Microvision system signal generator outputs a TTL square wave that controls the switching circuitry described above (location (2) in Figure 4-11). The signal generator also produces short, synchronized TTL pulses that drive the light emitting diode (LED) of the microscope and illuminate the field of view briefly (location (1) in Figure 4-11). The CCD camera grabs the image during these illumination periods and sends it to the controller computer through a video digitizer. For a given focal plane, images at different, but equally spaced phases (eight, in our case) with respect to the square wave pattern are acquired. Deflections out of the plane can be measured by repeating this process for different focal depths, which the software controls via the piezoelectric objective lens. Through these images taken at different phases during the periodic deflections of the tethered rotor, the fundamentals of the oscillation over three orthogonal directions are extracted. By plotting the amplitudes of these
Figure 4-10: Pictures from an iron powder test on an earlier generation stator. The windings of this stator have the wrong spatial phase (they have the same orientation, which cannot produce a traveling wave). When both phases are excited by the same input, the current in the windings flows in the same direction in every wire slot, resulting in a strong accumulation of iron powder over every tooth gap. With out of phase inputs, the accumulation is much weaker.
Figure 4-11: The conceptual schematic of the overall test setup.
Figure 4-12: The timing diagram for representative signals from certain locations in the test setup of Figure 4-11.
fundamentals at various frequencies of the square wave signal, the mechanical resonance curve of the rotor-tether system is obtained. The next section discusses how torque is extracted from these mechanical resonance curves.

4.2 Analysis

The tethers are designed to deflect by no more than a few microns at low frequencies and maximum current levels, and less than the tether width at full resonance. This design criterion ensures that the differential equation that describes the deflection dynamics is linear. The mechanical dynamics of the rotor-tether structure is described by

\[ J\ddot{\theta} + b\dot{\theta} + k\theta = \tau(t) \]  

(4.1)

which is a simple, second-order linear ordinary differential equation (ODE). Here, \( \theta \) is the angle of rotation at the measurement radius \( r_m \), which is the distance from the center to the base of the tether connecting to the rotor disk. Hence, \( \theta \) is given by \( x(t)/r_m \), where \( x(t) \) is the tangential deflection at the measurement radius as extracted by the Microvision system. In Equation 4.1, \( b \) is the viscous damping due to tethers moving in air, \( k \) is the effective spring constant of the six tethers, and \( \tau \) is the applied torque function. The inertial moment of the rotor, \( J \), is given by \( J = \frac{1}{2}M_r r_m^2 \), where \( M_r \) is the mass of the rotor and the surrounding enclosure ring of the rotor die. The contribution to the inertial moment from the tiny mass of the tethers themselves is inconsequential and is therefore ignored.

The torque function is a square wave that can be expressed in its Fourier components as

\[ \tau(t) = \tau_o(f_e, I) \sum_{n=1, n \text{ odd}}^{\infty} \frac{4}{n\pi} \sin(n\omega t) \]  

(4.2a)

\[ = \sum_{n=1, n \text{ odd}}^{\infty} \text{Im} \left\{ \tau_o(f_e, I) \frac{4}{n\pi} e^{jn\omega t} \right\} \]  

(4.2b)

where \( \omega = 2\pi f \) and \( f \) is the torque reversal frequency. Here, \( \tau_o \) is the amplitude of the square wave deflection, and is a function of the input current amplitude and frequency, given a particular micro machine geometry. Notice that the linearity of Equation 4.1 implies that the
measured deflection is given by

$$\theta(t) = \sum_{n=1, n \text{ odd}}^{\infty} |\Theta_n| \sin(\omega nt + \phi_n)$$  \hspace{1cm} (4.3)$$

$$= \sum_{n=1, n \text{ odd}}^{\infty} \text{Im}\{\Theta_n e^{j\omega t}\}$$  \hspace{1cm} (4.4)$$

where $\Theta_n$ are the complex Fourier amplitudes for the angular deflection and $\phi_n$ are their associated phases. Substituting Equations 4.4 and 4.2b into Equation 4.1 and matching terms for every Fourier mode, we obtain

$$(-Jn^2\omega^2 + k + bjn\omega) \Theta_n = \tau_o (f_e, I) \frac{4}{n\pi}$$  \hspace{1cm} (4.5)$$

which yields

$$\Theta_n = \frac{\tau_o (f_e, I) \frac{4}{n\pi}}{(-Jn^2\omega^2 + k + bjn\omega)}$$  \hspace{1cm} (4.6)$$

Taking the magnitude of Equation 4.6 yields

$$|\Theta_n| = \frac{\tau_o (f_e, I) \frac{4}{n\pi}}{\sqrt{(k - Jn^2\omega^2)^2 + (bn\omega)^2}}$$  \hspace{1cm} (4.7)$$

Dividing the numerator and the denominator by $J$, recognizing the resonance frequency $\omega_o = \sqrt{\frac{k}{J}}$, and the quality factor $Q = \frac{J\omega_o}{b}$, we get

$$|\Theta_n| = \frac{\tau_o (f_e, I) \frac{4}{n\pi J}}{\sqrt{(\omega_o^2 - n^2\omega^2)^2 + \left(\frac{\omega_o n\omega}{Q}\right)^2}}$$  \hspace{1cm} (4.8)$$
With $|\Theta| = \frac{|X|}{r_m}$, Equation 4.8 becomes

$$|X_n| = \frac{\tau_o (f_e, I) \frac{4r_m}{n\pi J}}{\sqrt{(\omega_0^2 - n^2\omega^2)^2 + \left(\frac{\omega_0 n\omega}{Q}\right)^2}}$$

(4.9)

Finally, defining a gain factor as

$$G = \frac{\tau_o (f_e, I) \frac{r_m}{J \omega_0^2}}$$

(4.10)

we find

$$|X_n| = \frac{G\omega_0^2 \frac{4}{n\pi}}{\sqrt{(\omega_0^2 - n^2\omega^2)^2 + \left(\frac{\omega_0 n \omega}{Q}\right)^2}}$$

(4.11)

The phase of the deflection is given by the phase of $\Theta_n$, which is

$$\angle \Theta_n = -\arctan \left(\frac{\omega_0 n \omega / Q}{\omega_0^2 - n^2\omega^2}\right)$$

(4.12)

Equations 4.11 and 4.12 are rather convenient, since they are expressed in terms of the natural frequency and the quality factor of the mechanical system, both of which are rather easy to extract from a given resonance curve. It is important to point out at this point that the Microvision system reports the first time harmonic, i.e., $n = 1$, for the deflection and the phase in Equations 4.11 and 4.12. The applied torque is extracted using Equation 4.10 by making a second-order system fit to Equation 4.11, with $G$, $\omega_0$, and $Q$ as parameters. For a system with a moderate to high $Q$ (larger than 20), the resonance peak is very sharp and the resonance frequency $\omega_0$ can be read off quite accurately from the curve. In that case, the exact value of $Q$ is not that significant in obtaining a torque value. The only truly unknown parameter in the fit to the resonance curve is the gain factor, $G$. Notice that such a fit would extract $\frac{4}{\pi} \tau_o$ as the amplitude of the torque. Hence, the extracted values are scaled by $\frac{\pi}{4}$ to obtain the amplitude of the square wave torque function, namely, $\tau_o$. 89
Figure 4-13 illustrates the amplitude and the phase of the fundamental \((n = 1)\) of the azimuthal tether deflection, as well as the second-order system fit obtained as described above. The data correspond to a rotor within a Kapton die. The fitted curve in Figure 4-13 is based on an inertia calculated from the measured rotor mass (47 mg). The extracted torque is about 0.29 μNm.

4.3 Results and Discussion

The process of torque extraction is repeated for a set of experiments, over which both the amplitude of the stator currents, and the electrical frequency, are varied. The first set of experiments using SU-8 tethers have revealed that micro motor heating at high currents raises the temperature of the device up to and beyond the glass transition temperature of SU-8 (about 200 °C). The accompanying deterioration in the mechanical properties of SU-8 causes the tethers to warp in both the azimuthal and the axial directions. As a result, the air gap is reduced substantially, if not collapsed all together. In one of the devices with a non-zero air gap, the torque results of Figure 4-14 have been obtained. Here, the amplitude of the stator currents is 5 A. It is estimated that the air gap corresponding to the data in Figure 4-14 is about 5 μm. However, it is very likely that the air gap is changing as the input excitation frequency is varied. Hence, Figure 4-14 does not provide a reliable source of data against which our model can be tested. Using Kapton tethers instead, we have managed to obtain much more reliable and systematic data with a controlled air gap. This is mainly because Kapton is quite resistant to high temperatures. The extracted torque from the Kapton tethered micro motor for various input current amplitudes is displayed as a function of the stator excitation frequency in Figure 4-15. All the raw data and the accompanying fits are included in Appendix D.

The measured torque values shown in Figure 4-15 are almost an order of magnitude less than the predictions of Model I in Chapter 2 for the parameters of the micro machine that was tested (see Figure 4-16). This discrepancy indicates that substantial eddy-currents are present within the micro machine stator, and that they have a crippling effect on the torque output of the machine. In order to understand the real mechanisms responsible for the reduction in torque, and to find solutions to alleviate the problem effectively, we must study in more detail
Figure 4-13: The azimuthal tether deflection and deflection phase as a function of torque reversal frequency. In the top plot, stars indicate measured values, whereas the solid curve corresponds to a fit from a second-order system response.
Figure 4-14: Torque as a function of various input electrical frequencies at 5 A of peak current input. Data represents the results from an SU-8 tethered magnetic induction micro motor. The estimated air gap is about 5 μm.
Figure 4-15: Measured torque from the tethered magnetic induction micro-motor for several current levels. The device consists of a six pole stator and a rotor suspended above it via Kapton tethers. The error bars correspond to the maximum fit uncertainty.
Figure 4-16: Predictions of torque vs. frequency from Model I for the same tethered magnetic micro motor tested. Comparison with Figure 4-15 makes it clear that the torque values predicted by Model I are about an order of magnitude higher than the measured results; the peak frequencies are also higher than the measurements. The source of the discrepancy is the effect of eddy-currents within the stator, which are not captured by Model I.

the effects of eddy currents in the magnetic induction micro machine. This is done in the next chapter.
Bibliography

Chapter 5

Magnetic Diffusion Revisited

In Chapter 2, we presented models for the magnetic induction micro machine. These models were then used to create a design tool to optimize the performance of the magnetic induction micro machine within prespecified fabrication limits. The modeling and the subsequent design optimization was performed under the assumption of a nonconductive NiFe. This assumption is a valid starting point, given that the eventual micro machines will be densely laminated. In reality, though, NiFe has a relatively high conductance (about $5 \times 10^6$ S/m), and the frequencies of operation for the tethered micro motor yield skin depths that are smaller than the stator dimensions. This means that eddy-current effects within the micro motor are likely to be very significant and mostly responsible for the disparity between the predicted design torque and the lower measured torque (Chapter 4).

In this chapter, we revisit magnetic diffusion. We first explore the effects of linear eddy-currents with a vector potential formulation based on a numerical finite-difference analysis. This model and the insights obtained from it are then used to motivate the formulation of a fully nonlinear magnetic diffusion model, which will be utilized to simulate experimental conditions. The chapter concludes with a detailed discussion of the simulation results and comparisons with the experiments of Chapter 4.
5.1 Linear Eddy-Current Model

Let us start by considering Maxwell’s Equations that apply to a magnetoquasistatic (MQS) and stationary system. They are

\[
\begin{align*}
\nabla \times \mathbf{H} &= \mathbf{J}_{\text{tot}} = \sigma \mathbf{E} + \mathbf{J}_{\text{ext}} \\
\nabla \cdot \mathbf{B} &= 0 \\
\mathbf{B} &= \mu \mathbf{H} \\
\nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t}
\end{align*}
\]

(5.1) \hspace{1cm} (5.2) \hspace{1cm} (5.3) \hspace{1cm} (5.4)

In general, the material properties are assumed to be a function of position alone; the field quantities vary both spatially and temporally. Combining Equations 5.1 and 5.4, we have

\[
-\mu \frac{\partial \mathbf{H}}{\partial t} = \nabla \times \left( \frac{1}{\sigma} \right) (\nabla \times \mathbf{H} - \mathbf{J}_{\text{ext}})
\]

(5.5)

Let us rewrite Equation 5.5 in terms of the vector potential, defined as \( \nabla \times \mathbf{A} = \mathbf{B} \); replacing \( \mathbf{H} \) with \( \frac{1}{\mu} \left( \nabla \times \mathbf{A} \right) \), we get

\[
-\frac{\partial (\nabla \times \mathbf{A})}{\partial t} = \nabla \times \left( \frac{1}{\sigma} \right) \left( \nabla \times \left( \frac{1}{\mu} \right) (\nabla \times \mathbf{A}) - \mathbf{J}_{\text{ext}} \right)
\]

(5.6)

which can be rewritten as

\[
-\nabla \times \frac{\partial \mathbf{A}}{\partial t} = \nabla \times \left( \frac{1}{\sigma} \right) \left( \nabla \times \left( \frac{1}{\mu} \right) (\nabla \times \mathbf{A}) - \mathbf{J}_{\text{ext}} \right)
\]

to yield

\[
-\frac{\partial \mathbf{A}}{\partial t} = \left( \frac{1}{\sigma} \right) \left( \nabla \times \left( \frac{1}{\mu} \right) (\nabla \times \mathbf{A}) - \mathbf{J}_{\text{ext}} \right)
\]

(5.7)

It is interesting to note that Equation 5.7 above has the same form as the generalized diffusion equation \[1\]. Any linear MQS system can be characterized by the corresponding solutions to Equation 5.7. However, solving this diffusion equation analytically is a daunting
task even for very simple geometries. In the case of the magnetic micro machine, an analytical solution is simply impractical; the stator is described by material property functions that vary discontinuously in space, which leads to singularities in Equation 5.7. Trying to integrate out the singularities only causes all the modes of the system to mix, further aggravating the problem. Therefore, we have preferred a numerical approach to solve Equation 5.7, as it is relatively simpler and more practical computationally, compared to an analytical solution.

In what follows, we introduce some simplifying assumptions that are valid for the particular geometry of the micro motor. Later, we present a finite-difference solution to Equation 5.7.

### 5.1.1 Finite-Difference Numerical Solution (Model II)

Once again, we will exploit the axial symmetry of the micro motor, and consider radial cross-sections of it as mapped onto a two-dimensional Cartesian plane (Figure 5-1). Let us consider
a sinusoidally varying current input to the wire slots within the stator, of the form

\[ J_{\text{ext}} = J_0 (x, y) e^{j\omega t} \tag{5.8} \]

where \( J_0 (x, y) \) is non-zero only inside the wire slots. In steady-state, this current results in a periodic vector potential whose fundamental harmonic is given by

\[ A (x, y, t) = A (x, y) e^{j\omega t} \tag{5.9} \]

Consider a uniform material, with a particular magnetic permeability \( \mu \) and electrical conductivity \( \sigma \), in which Equation 5.7 will be solved. In this case, the magnetic diffusion equation simplifies to

\[-\mu \sigma \frac{\partial A}{\partial t} = (\nabla \times (\nabla \times A)) - J_{\text{ext}} \]
\[ = \nabla (\nabla \cdot A) - \nabla^2 A - J_{\text{ext}} \tag{5.10} \]

Here, the divergence of the vector potential can be any constant since the definition \( \nabla \times A = B \) specifies only the curl of \( A \). Left with the second term on the right, we can substitute Equation 5.9 into Equation 5.10, to get

\[ \mu \sigma j \omega A - \left( \frac{\partial^2 A}{\partial x^2} + \frac{\partial^2 A}{\partial y^2} \right) = J_{\text{ext}} \tag{5.11} \]

For now, let us focus on variations along one dimension alone, and imagine meshing our material into \( n \) elements along the \( x \)-direction. In general, the mesh spacing may be non-uniform. We will assume that within each unit cell at a given time, the vector potential and the material properties are uniform, and they are given by their respective values evaluated at the center of the unit cell. Studying Figure 5-2, it can be seen that the spatial derivative in Equation 5.11 can be approximated as

\[ \left. \frac{\partial^2 A}{\partial x^2} \right|_{x_i} = A'' (x_i) \approx \frac{A' ((x_i + x_{i+1})/2) - A' ((x_i + x_{i-1})/2)}{(x_i + x_{i+1})/2 - (x_i + x_{i-1})/2} \tag{5.12} \]
Figure 5-2: Meshing along the x-direction.

where

\[ A' \left( \frac{x_i + x_{i+1}}{2} \right) \approx \frac{A(x_{i+1}) - A(x_i)}{x_{i+1} - x_i}; \]  \hspace{1cm} (5.13)

\[ A' \left( \frac{x_i + x_{i-1}}{2} \right) \approx \frac{A(x_i) - A(x_{i-1})}{x_i - x_{i-1}} \]  \hspace{1cm} (5.14)

which yields

\[ \frac{\partial^2 A}{\partial x^2} \bigg|_{x_i} \approx \frac{\frac{A(x_{i+1}) - A(x_i)}{(x_{i+1} - x_i)} - \frac{A(x_i) - A(x_{i-1})}{(x_i - x_{i-1})}}{(x_{i+1} - x_{i-1})/2} \]

\[ = \frac{\frac{A(x_{i+1}) - A(x_i)}{(\Delta x_{i+1} + \Delta x_i)/2} - \frac{A(x_i) - A(x_{i-1})}{(\Delta x_i + \Delta x_{i-1})/2}}{(\Delta x_{i+1} + 2\Delta x_i + \Delta x_{i-1})/2} \]  \hspace{1cm} (5.15)

We can simplify our notation by representing the potential in each unit cell as an entry in a vector which corresponds to the entire collection of \( A \) values everywhere at a given time. In this convention, each \( A(x_i) \) is represented by a vector entry \( A_i \), and Equation 5.11 for \( i^{th} \) unit cell along the \( \hat{x} \)-direction becomes

\[ \mu_0 j \omega A_i - \frac{\frac{A_{i+1} - A_i}{(\Delta x_{i+1} + \Delta x_i)/2} + \frac{A_{i-1} - A_i}{(\Delta x_i + \Delta x_{i-1})/2}}{(\Delta x_{i+1} + 2\Delta x_i + \Delta x_{i-1})/2} = J_{\text{ext},i} \]  \hspace{1cm} (5.16)

Here, \( J_{\text{ext},i} \) represents the current that travels into the page within the \( i^{th} \) unit cell. It is straightforward to extrapolate Equation 5.16 into two-dimensions: simply insert another term,
like the one inside brackets in Equation 5.16, that incorporates nearest neighbor interactions along the orthogonal direction. The form of Equation 5.16 suggests that the magnetic diffusion equation can be solved using linear algebra. In fact, with a suitable definition of a “conductivity matrix”, \( G \), Equation 5.16 may be rewritten in the form

\[
G^{(n \times n)} A^{(n \times 1)} = j^{(n \times 1)} \tag{5.17}
\]

Inspection of 5.16 reveals that \( G \) is a positive semi-definite matrix; it is tri-diagonal (two neighbors for each element) in the one dimensional case, and penta-diagonal (four neighbors for each mesh unit) in two dimensions. The entries of the conductivity matrix look like

\[
G_{kl} = \begin{cases} 
\mu \sigma j \omega + \sum_{m=\text{nearest to } k} g_{km}, & \text{for } k = l \\
-g_{kl}, & \text{otherwise}
\end{cases} \tag{5.18}
\]

where

\[
g_{kl} = \begin{cases} 
\frac{1}{((\Delta h_l + \Delta h_k) / 2)^2}, & \text{for } k \text{ nearest to } l \\
0, & \text{otherwise}
\end{cases} \tag{5.19}
\]

Here, \( \Delta h \) represents either \( \Delta x \) or \( \Delta y \), corresponding to the relative position of the nearest neighbor mesh cell.

Figure 5-3 below illustrates the particular meshing scheme employed in modeling the micro machine geometry. Recall that the micro machine stator is designed such that the wavelength of the traveling excitation spans four wire slots, which means that the right and left sides of Figure 5-3 are a quarter of a wavelength out of phase. This phenomenon is captured by forcing the appropriate boundary conditions on either side of the mesh geometry, which introduces complex off-diagonal entries to the \( G \) matrix.

Numerical experiments have established that flux does not exit the micro motor from either the top or the bottom, because both the rotor and the stator backirons are too thick to allow any flux leakage – even in the absence of eddy currents. This fact is implicitly included in the computation simply by not connecting the corresponding unit cells at the top and bottom to anything outside the mesh geometry.
Figure 5-3: Meshing scheme for the micro motor geometry. Each unit cell and its neighbor along one direction share a common $\Delta h$ along the orthogonal direction.

5.1.2 Modeling Results

The finite-difference modeling outlined above has been implemented for the particular geometry of the magnetic induction micro machine. The magnetic permeability of electroplated NiFe has been taken constant around $3000\mu_0$, a medium-range value that is inferred from the experimentally measured B-H curves of NiFe rings (see Appendix B). The electrical conductivity of NiFe has been taken to be $5.0 \times 10^6$ S/m, and that of copper around $5.8 \times 10^7$ S/m.

Once the conductivity matrix, $G$, is computed from the meshed geometry and material properties, Equation 5.17 is solved by Gaussian elimination. The solution desired is then given by the real part of the resulting vector potential. Since $G$ is sparse, the solution process is vastly accelerated by use of sparse matrix techniques. One advantage of this method is linearity: the resulting vector potential solution can be multiplied by any phase to yield the corresponding

---

1 In reality, the magnetic permeability of NiFe is larger than $10,000\mu_0$ for low field and frequency values; the higher the permeability in this range of the B-H curve, the sooner the material exhibits nonlinear behavior as the field strength is raised.
Figure 5-4: Torque predictions by the linear model for the parameters of the motor tested. The result from the linear eddy-current model for 6A is superimposed for comparison.

solution for an excitation with that particular, instantenous phase.

Figure 5-4 shows torque predictions by Model I, and presents the result by the eddy current model (Model II) of a 6A input for comparison. In the light of the experimental results discussed in Chapter 4, the eddy current model appears to move us in the right direction, though the results are still way off. Apparently, this model has captured only part of the underlying physics. Further investigation indeed reveals the shortcoming of this model. Consider Figure 5-5, which illustrates the magnetic field density within the micro motor, meshed as shown in Figure 5-3. Notice that the field density gets quite high around the wire slot (wire not shown in figure). In fact, it has been observed that flux density inside the stator just around the wire slot increases with frequency well beyond the point where material non-linearities begin to take effect. This phenomenon (which we will henceforth refer to as “flux crowding”) appears to be a direct consequence of the traveling nature of the input excitation; it does not occur if the excitory waveform is simply a uniform current varying sinusoidally in time.
Figure 5-5: Magnitude of B field (first harmonic) within the magnetic induction micro machine, as predicted by the linear eddy-current model. The scale units on the right are in Teslas.
Figure 5-6: One of the simplest two-dimensional magnetic diffusion examples involves two magnetic cores sandwiching a wire carrying a traveling current.

Before we continue, let us attempt to obtain more physical insight into flux crowding by exploring a simplified setup that better illustrates the phenomenon.

5.1.3 A Simpler Example

Consider the simpler setup of Figure 5-6. Here, the geometry extends infinitely to either side. A current is applied to the center conductor in the \( \hat{z} \) direction; its magnitude pattern is a sinusoid traveling along the \( \hat{y} \) direction, at a phase velocity of \( \frac{\omega}{k} \). The amplitude of the current is chosen such that with a standing wave excitation, the magnitude of the B field just inside the NiFe surface would be about 0.5 T at low frequencies. The material properties of the magnetic slabs are chosen to match those of NiFe used in the micro machine, but with linear material assumptions. Using the eddy current model described in the previous section, it is straightforward to find the magnitude of the flux density within the core along the material thickness (\( \hat{z} \) direction) at a given time and \( y \) location (Figure 5-7). As expected, the magnetic diffusion wave decays exponentially within the NiFe. However, contrary to our initial intuition, \( |B| \) just on the NiFe surface turns out to be much larger than 0.5 T. Somehow, if the input excitation is a traveling wave, it causes an amplification of the flux density at the material surface.
Figure 5-7: The magnitude of the magnetic flux density vector (|B|) along the x direction. The snapshot is taken when the magnitude at the surface is at its maximum.

For this simpler setup, one can determine the surface magnetic flux density analytically. In Figure 5-8, this is exactly what we have done, and compared the results with those from an FEA software.

The results presented in Figure 5-8 represent the fundamental amplitude of the magnetic flux density on the NiFe surface as a function of $\omega$ for constant $k$. Notice that the flux density amplitude remains constant with increasing frequency until about 700 Hz or so, which is when skin depth within NiFe drops below the thickness of the magnetic slab. At higher frequencies, the B field at the surface rises by about 10 dB per decade in frequency\(^2\). It is important to stress that this type of “flux crowding” occurs only in the presence of a traveling wave excitation. In fact, when the current excitation in Figure 5-6 is replaced with a spatially uniform, temporally sinusoidal waveform of the same amplitude, B field amplitude stays constant (at the low frequency value in Figure 5-8) over all frequencies.

\(^2\)The trend eventually does level off, though certainly not within the frequencies of interest for the micro machine.
Figure 5-8: Magnetic flux density magnitude just inside the NiFe layer. As skin depth within the material becomes shorter than its thickness, magnetic flux at the boundary begins to “crowd”.

5.1.4 Need for a new model

The model presented above suggests an explanation for the low values of measured torque (Chapter 4). The high frequency nature of the traveling excitation input leads to eddy currents within the highly conductive magnetic material, where the skin depth ends up smaller than the geometrical dimensions of the motor. This, in turn, leads to flux crowding within the stator and the rotor, squeezing the field lines towards the inner NiFe surfaces. As the field lines are squeezed, the magnetic field density, $B$, rises dramatically above the saturation level of the electroplated NiFe. Of course, in reality, the inner surfaces will saturate, and the “saturation wave” will penetrate deeper, until the MMF driven by the windings is cancelled by the combination of induced eddy currents and flux carried within the NiFe. Inside the tethered motor, the saturation wave travels with the excitation around the stator. As a result of eddy currents and saturation, a dramatic reduction in output torque is observed.

It is clear that any model that has a hope of explaining the measured torque data (see end of Chapter 4) needs to capture not only the presence of eddy currents, but also the nonlinearity in
Figure 5-9: Comparison of ANSOFT's simulated torque predictions at 6A with measured torque values from the tethered magnetic induction motor. ANSOFT overpredicts the torque by more than 300%.

the magnetization of NiFe. Such a model should also be able to include the effects of multiply excited modes, since the level of nonlinearity is likely to be intense enough to cause significant mixing of modes.

The commercial magnetics simulation package ANSOFT attempts to solve a nonlinear eddy current problem by assuming that only the fundamental component for $A$ contributes to the solution. It essentially solves Equation 5.11 with repeated Newton's iterations over a user-defined B-H curve. One problem with this approach is that it does not capture the effects of higher harmonics, which are significant in the nonlinear system of the micro machine. Also, mathematically speaking, use of Equation 5.11 in a nonlinear case is unwarranteed, since the equation is derived with linear assumptions in the first place. Therefore, it is not surprising that ANSOFT's torque prediction for the same motor geometry and inputs as the tested tethered motor is far off, e.g. more than 300% higher than the measured value for the peak at 6A (see Figure 5-9).

Our search for a more comprehensive, nonlinear, commercially available magnetics simu-
lution package yielded no better candidates. It was decided that creating our own magnetics simulation software was the best solution.

5.2 A Finite-Difference Time-Domain Approach

A fully nonlinear magnetics approach is a demanding task computationally. A common way to tackle a nonlinear steady-state problem is to use the so-called Shooting Method [6]. The problem with this matrix approach is that all field variables and the material properties that change at every spatial point at every phase within a period must somehow be stored in computer memory. Even if memory is abundant, the method quickly gets overwhelmingly expensive in terms of the CPU time needed to solve the resulting system of equations. Improved algorithms, such as the Generalized Minimum Residuals approach (GMRES) based on matrix-free Krylov-subspace methods, have recently been employed [2]; unfortunately, such simulations still take on the order of days to complete. In order to be practical, a nonlinear simulation tool for magnetic diffusion needs to be both accurate and very fast – on the order of less than an hour per simulation.

One useful way to solve for magnetic diffusion in the micro scale involves a direct time integration of the relevant partial differential equations over a finite-difference mesh. In this manner, matrix inversion and storage issues are all but eliminated. Given a clever setup of computation parameters, a finite-difference time-domain (FDTD) approach has the potential to provide the practicality that we seek in a nonlinear simulation method.

The general FDTD method for electromagnetics was first outlined by Yee in his original 1966 paper [3]. The method solves the full set of Maxwell’s equations; hence, the approach could be used to study and explain virtually all electromagnetic phenomena [4]. It is especially popular within the optics community, as it enables the systematic study of short duration (pico or femtosecond) pulses of light and their interactions with any dielectric material (possibly nonlinear) or crystal structure. Designs of efficient photonic bandgap structures, refractive arrays and optical fibers have greatly accelerated since the method’s repopularization in the 1990’s. The FDTD method is also commonly used in the design of microwave circuits and antennae, and recently, biologists have begun employing the algorithm to estimate the effects
of cell phone radiation on the human brain.

Applying FDTD algorithms to magnetic diffusion is not a straightforward task. In what follows, we first introduce our implementation of a FDTD magnetic diffusion algorithm. Later, we discuss some of the issues associated with FDTD in general and our implementation in particular. The test case of Section 5.1.3 is revisited for more physical insight. Finally, we present micro motor simulation results and discuss how they compare to experiments.

5.2.1 Discretization Setup

As usual, we will exploit the axial symmetry of the micro motor; the model will solve for electromagnetic fields in radial cross-sections as mapped onto a two-dimensional Cartesian plane in Figure 5-1.

Consider the full set of Maxwell's Equations

\[
\begin{align*}
\epsilon \frac{\partial E}{\partial t} + \sigma E + J_{\text{ext}} &= \nabla \times H \quad \text{(Ampere's Law)} \quad (5.20) \\
\frac{\partial B}{\partial t} &= -\nabla \times E \quad \text{(Faraday's Law)} \quad (5.21) \\
H &= \frac{1}{\mu} B \quad \text{(Material Property)} \quad (5.22)
\end{align*}
\]

Here we choose a convention in which the total current density is represented as the sum of externally applied current density and the induced current. Keep in mind that in those locations where a source sets the total current, the \( \sigma E \) term is not present. We shall not consider the divergence of the electric field since the presence of free charges is not a concern in the magnetic machine. Also, though the divergence-free nature of the \( B \) field is not explicitly stated in the equations above, it will be included in the manner the discretized mesh is setup. Due to the two dimensional nature of our setup, we need only to consider the magnetic field components along the plane of our setup (\( \hat{x} \) and \( \hat{y} \) directions), and the electric field component perpendicular to that plane.

Since this will be a time domain integration, field variables will take on real values. Recall that the traveling excitation goes through one wavelength over four stator wire slots. If one considers only half the wavelength, the rest of the motor can be simulated with odd symmetric
boundary conditions on either side. Hence, it suffices to simulate over two wire slots of the stator. Figure 5-10 below illustrates the computation space and the discretization scheme.

Figure 5-10: The smallest computation space needed for the FDTD scheme. For clarity, a diffuse mesh is shown.

Electric field update

Next, let us integrate Equation 5.20 over the surface $S_{ij}$ of a given unit cell on the $x$-$y$ plane at location $(x_i, y_j)$. This yields

$$
\int_{S_{ij}} \left( \epsilon \frac{\partial \mathbf{E}}{\partial t} + \sigma \mathbf{E} + \mathbf{J}_\text{ext} \right) \cdot d\mathbf{S} = \int_{S_{ij}} (\nabla \times \mathbf{H}) \cdot d\mathbf{S} = \int_{C_{ij}} \mathbf{H} \cdot d\mathbf{l}
$$

where the RHS is converted into a line integral along $C_{ij}$ around the unit cell. We now employ our finite difference scheme and evaluate the variables in Equation 5.23 at discrete time steps and spatial locations. Here, Yee’s algorithm calls for a discretization scheme in which magnetic and electric field points are separated both in space and time by half a discretization (Fig. 5-11). We shall evaluate electric field values at integer multiples of the time step, and find
Figure 5-11: The general meshing scheme used in an FDTD algorithm.

magnetic field values half a time step later, in a leap-frogging manner. Interestingly, it turns out, this method ensures that first order finite differences for derivatives are second order accurate. Moreover, the finite difference grid, with each electric field point encircled by magnetic field lines, is divergence-free (both in E and B) by construction. The proof of either of these statements is omitted here for brevity; the interested reader is referred to [4].

Let us continue by evaluating Equation 5.23 at time step \( n - \frac{1}{2} \). This yields

\[
\int_{S_{ij}} \left( \epsilon \frac{\partial E}{\partial t} \right)^{n-\frac{1}{2}} + \sigma E |^{n-\frac{1}{2}} + J_{\text{ext}} |^{n-\frac{1}{2}} \right) \cdot dS = \int_{C_{ij}} H |^{n-\frac{1}{2}} \cdot dl \quad (5.24)
\]

Here, we shall represent \( \epsilon \frac{\partial E}{\partial t} |^{n-\frac{1}{2}} \) as \( \epsilon \left( \frac{E^n - E^{n-1}}{\Delta t} \right) \), and use a semi-implicit approximation to write \( \sigma E |^{n-\frac{1}{2}} \) and \( J_{\text{ext}} |^{n-\frac{1}{2}} \) as \( \sigma \left( \frac{E^n + E^{n-1}}{2} \right) \) and \( \left( J_{\text{ext}}^n + J_{\text{ext}}^{n-1} \right) / 2 \), respectively. Then,
applying Equation 5.24 to the two-dimensional discretization in Figure 5-11, we have

\[
\left[ \varepsilon_{ij} \left( \frac{E_{zij}^n - E_{zij}^{n-1}}{\Delta t} \right) + \sigma_{ij} \left( \frac{E_{zij}^n + E_{zij}^{n-1}}{2} \right) + \left( \frac{J_{\text{ext}ij}^n + J_{\text{ext}ij}^{n-1}}{2} \right) \right] \Delta x_{ij} \Delta y_{ij} =
\]

\[
\left( H_{y_{i+1,j}}^{n-\frac{1}{2}} - H_{y_{ij}}^{n-\frac{1}{2}} \right) \Delta y_{ij} - \left( H_{x_{i,j+1}}^{n-\frac{1}{2}} - H_{x_{ij}}^{n-\frac{1}{2}} \right) \Delta x_{ij}
\]

(5.25)

Collecting terms and solving for \( E_{zij}^n \), the electric field value at the new time step, we find

\[
E_{zij}^n = \left[ 1 - \beta_{ij} \left( 2/\sigma_{ij} \right) \right] E_{zij}^{n-1} + \beta_{ij} \Delta x_{ij} \left( H_{y_{i+1,j}}^{n-\frac{1}{2}} - H_{y_{ij}}^{n-\frac{1}{2}} \right) - \beta_{ij} (\Delta y_{ij}) \left( H_{x_{i,j+1}}^{n-\frac{1}{2}} - H_{x_{ij}}^{n-\frac{1}{2}} \right) - \beta_{ij} (2) \left( J_{\text{ext}ij}^n + J_{\text{ext}ij}^{n-1} \right)
\]

(5.26)

where the function

\[
\beta_{ij} (a) = \frac{\Delta t}{a c_{bij}} \frac{\Delta t \sigma_{ij}}{2 \varepsilon_{ij}}
\]

(5.27)

has been defined for notational convenience.

Notice that, assuming magnetic field values at time step \( n - \frac{1}{2} \) have already been computed, and that input \( J_{\text{ext}} \) is specified for all time, Equation 5.26 is explicit; it can be solved by direct substitution.

**Magnetic field update**

We need another equation to find the magnetic field values at each spatial location; for that, we shall employ Faraday’s law and follow the same approach as above. This time, however, in order to convert the curl of the electric field into a line integral, the contours must be around a surface \( S_{ij} \) pointing along the \( \hat{y} \) direction (Figure 5-12). We begin by integrating both sides of Equation 5.21 over the surface \( S'_{ij} \) of Figure 5-12 at time step \( n \) to obtain

\[
\int_{S'_{ij}} \frac{\partial \mathbf{B}}{\partial t} |^n \cdot dS' = - \int_{S'_{ij}} (\nabla \times \mathbf{E}) |^n \cdot dS' = - \int_{C'_{ij}} \mathbf{E} |^n \cdot dl'
\]

(5.28)
Once again, the RHS is converted into a line integral over the contour $C_{ij}'$ that bounds the surface $S'_{ij}$. Assuming that the surface of integration extends $\Delta z$ in the third dimension, the RHS simply evaluates to $(E_{zi}^n - E_{zi-1,j}^n) \Delta z$.

One must pay special care in evaluating the LHS of Equation 5.28, since both the material properties and the flux density field values are not uniform over the surface of integration\(^3\). Evaluation then yields

$$
\int_{S'_{ij}} \frac{\partial B}{\partial t} \cdot dS' = \frac{\partial B_{y_{i-1,j}}}{\partial t} \Delta x_{i-1,j} \Delta z + \frac{\partial B_{y_{ij}}}{\partial t} \Delta x_{i,j} \Delta z
$$

$$
= \frac{B_{y_{i-1,j}}^{n+1/2} - B_{y_{i-1,j}}^{n-1/2}}{\Delta t} \Delta x_{i-1,j} \Delta z + \frac{B_{y_{ij}}^{n+1/2} - B_{y_{ij}}^{n-1/2}}{\Delta t} \Delta x_{i,j} \Delta z
$$

Equation 5.29

We can solve for $B_{y_{ij}}^{n+1/2}$ by using the continuity of $H_y$ across tangential material interfaces, which implies $\mu_{i-1,j}^{n+1/2} B_{y_{i-1,j}}^{n+1/2} = \mu_{ij}^{n+1/2} B_{y_{ij}}^{n+1/2}$. Notice that we have allowed the magnetic permeability of each unit cell to vary with time. Equation 5.28 finally reduces to

$$
\left[ \frac{B_{y_{i-1,j}}^{n+1/2}}{\Delta t} \right] \left( \frac{\mu_{i-1,j}^{n+1/2}}{\mu_{i-1,j}^{n+1/2}} \right) - \frac{B_{y_{ij}}^{n-1/2}}{\Delta t} \left( \frac{\mu_{i}^{n-1/2}}{\mu_{i-1,j}^{n-1/2}} \right) \Delta x_{i-1,j} \Delta z
$$

$$
+ \frac{B_{y_{ij}}^{n+1/2} - B_{y_{ij}}^{n-1/2}}{\Delta t} \Delta x_{ij} \Delta z = \left( E_{zi}^n - E_{zi-1,j}^n \right) \Delta z
$$

Equation 5.30

\(^3\)Recall that material properties and field values are uniform over the unit cells on the $x$-$y$ plane.
The problem is that we do not yet know what $\mu_{ij}$ is at time step $n + \frac{1}{2}$, as we would first need $B_{yi,j}^{n+1/2}$ to find it. Equation 5.30 is not fully explicit as desired. Short of solving a matrix inversion problem, this implicitness may be mitigated by using $\mu_{ij}^{n-1/2}$ as the initial guess for $\mu_{ij}^{n+1/2}$ and iterating Equation 5.30 between the flux density and the permeability until $\mu_{ij}^{n+1/2}$ eventually converges. Depending on the size of the time step, a few such iterations generally suffice. Nevertheless, the time step, $\Delta t$, is often so small that approximating $\mu_{ij}^{n+1/2} \approx \mu_{ij}^{n-1/2}$ for the purposes of the above equation works quite well. In fact, with that approach, Equation 5.30 simplifies to

$$B_{yi,j}^{n+1/2} = B_{yi,j}^{n-1/2} + \gamma_{ij}^{n-1/2} (\Delta x_{ij}) \Delta t \left( E_{zi,j}^{n_n} - E_{zi,j-1}^{n_n} \right)$$

(5.31)

where

$$\gamma_{ij}^{n-1/2} (\Delta x_{ij}) = \frac{1}{\left[ \left[ \frac{\mu_{i-1,j}^{n-1/2}}{\mu_{ij}^{n-1/2}} \left( \frac{\Delta x_{i-1,j}}{2} + \frac{\Delta x_{ij}}{2} \right) \right] \right]}$$

(5.32)

is defined for notational simplicity. Notice that if the unit cell $(i, j)$ and its neighbor to the left represent the same material, the ratio of the magnetic permeabilities in Equation 5.32 is unity. In that case, $\gamma_{ij} (\Delta x_{ij})$ corresponds simply to the inverse of the distance between the centers of the two neighboring cells.

Magnetic flux density along the $\hat{x}$ direction at unit cell $(i, j)$ can be found in exactly the same manner as above. The update equation for $B_x$ is

$$B_{z_{x,ij}}^{n+1/2} = B_{z_{x,ij}}^{n-1/2} - \gamma_{ij}^{n-1/2} (\Delta y_{ij}) \Delta t \left( E_{z_{x,ij}}^{n_n} - E_{z_{x,ij-1}}^{n_n} \right)$$

(5.33)

### Magnetic permeability update

Now that the B field is known everywhere at time $t = (n + \frac{1}{2}) \Delta t$, the magnetic permeability of NiFe across the computational grid can be updated using the B-H curve of the magnetic material. For this purpose, a look-up table has been created by fitting a function to the measured magnetization curve (Figure 5-13). The particular fitting function used in Figure
Figure 5-13: The measured magnetization curve of the NiFe around 7.5 kHz (stars) and the corresponding fit to data (thin solid curve). The slope of the curve at the origin and at high H values is $\mu_0$. Given a particular $|\mathbf{B}|$, the corresponding magnetic field magnitude is found using the fit (as illustrated by the thick solid line).
5-13 is of the form

$$|\text{B}| = \left( \frac{a|\text{H}|}{b + c|\text{H}|^d} + \mu_0 \right) |\text{H}|$$

(5.34)

where the parameters $a$ through $d$ are varied to obtain the best fit to the data. The variable $d$ is very close to 2.0. Notice that on the far right of Figure 5-13 where full saturation is evident, Equation 5.34 settles to a relative magnetic permeability of unity. The slope at the origin is also $\mu_0$, as desired. Given $|\text{B}|$, $|\text{H}|$ and hence $\mu(|\text{B}|)$ is found by direct interpolation. Care is taken to ensure that the magnetic field and the magnetic flux density vectors are collinear, i.e.,

$$H_{x_{ij}}^{n+1/2} = \frac{1}{\mu_{ij} (|\text{B}|^{n+1/2})} B_{x_{ij}}^{n+1/2}$$

and

$$H_{y_{ij}}^{n+1/2} = \frac{1}{\mu_{ij} (|\text{B}|^{n+1/2})} B_{y_{ij}}^{n+1/2}$$

**Implementation issues**

Maxwell’s equations are among the set of partial differential equations (PDE’s) that define the behavior of a wave (in this case, electromagnetic in nature) with a characteristic speed of travel (speed of light in our case). Literature on explicit numerical solutions of such PDE’s has deemed it convenient to define a number that is a measure of how the physical speed of propagation in the problem ($U$) compares to the numerical resolution of that speed. That number, known as the Courant number, is defined as

$$C = U \frac{\Delta t}{\Delta x}$$

(5.35)

A finite-difference scheme that is based on an explicit discretization of both space and time is subject to what is known as Courant stability condition [4]. In the case of linear materials, a stability analysis of the FDTD equations reveals that

$$C \leq 1, \text{ i.e., } \Delta t \leq \frac{\Delta x}{U}$$

(5.36)
is a necessary (but not sufficient) condition to guarantee stability of our numerical scheme. What is more, $\Delta x$ must be much shorter than the shortest wavelength or characteristic distance present in the problem. In the case of the micro motor, $\Delta x$ has to be chosen smaller than the shortest skin depth. This presents a problem. Assuming a skin depth of about 20 $\mu$m in NiFe around 50 kHz, $\Delta x$ must be on the order of at most 10 $\mu$m, resulting, according to Equation 5.36, in a time step on the order of a few tens of femtoseconds! Now, the excitation at 50 kHz has a period of 20 $\mu$s, which is the minimum time span we need to integrate before transients could begin to disappear and a steady-state solution could be obtained. This corresponds to a mind boggling $10^9$ time steps; with a unit cell count on the order of 10000 and each unit cell requiring $O(10)$ operations for field computations, the number of total double precision operations needed is larger than $10^{14}$. If we were to use a state-of-the-art Cray supercomputer with over 500 parallel processors, it would take on the order of an hour for the computation to finish. The fastest available PC, on the other hand, would still be running the same simulation years later. Obviously, this brute force approach is not the most practical one.

The tight constraint on stability arises from the inherent stiffness in the PDE's: electromagnetic waves travel essentially at the speed of light ($1/\sqrt{\mu_0\varepsilon_0}$) in air, yet the magnetic diffusion waves travel much slower ($\sqrt{\omega \mu_s \sigma_s}$) inside NiFe. For instance, even at full saturation that drives $\mu_s \sim \mu_0$, the diffusion wave speed is over two hundred thousand times slower than the speed of light. Moreover, the input excitation has a phase speed of $\omega / k$, which is also much slower than $c$. Indeed, given the stator winding pattern, the input excitation completes a period over four wire slots, which, at a mid-radius of 1.5 mm at 50 kHz, corresponds to a phase speed of just under 1000 m/s (three hundred thousand times slower than light). If this inherent disparity between the different characteristic times could be mitigated, the Courant constraint could be relaxed to give a larger integration time step.

In this modeling work, we adopted an implementation in which the speed of light has been artificially reduced many orders of magnitude, in order to reduce the stiffness of the PDE's. Since the emphasis is on magnetic diffusion phenomena, the reduction of $c$ involves increasing the electric permittivity of all materials in our computation space, without altering their magnetic permeabilities. Care is taken to make sure that $c$ is still many times faster than both the input traveling wave and the magnetic diffusion waves within all materials. This approach has
enabled a stable time step as large as 2 ns – more than a sixty thousand times improvement. With this method, a simulation takes less than an hour to conclude on an average PC. To give a perspective, the same simulation would have taken about seven years without the modification.

Another numerical trick that helps in achieving stability is to avoid sharp gradients by turning on the input excitation slowly and letting the system settle into its steady-state behavior gradually. In our particular implementation, we modulated the input current by a factor of $(1 - e^{-at})$, with $a$ chosen typically around a few microsecond$^{-1}$. This allows enough time for initial magnetic fields to diffuse in and start to saturate the material, before the input reaches its maximum amplitude. Depending on the input frequency, this numerical trick speeds up the integration by about order of magnitude.

### 5.2.2 Back to test cases

Let us consider the test case introduced in 5.1.3 to see the effects of material nonlinearities (together with eddy currents) using our modified FDTD approach. Figure 5-14 is essentially the same as Figure 5-6, except that the current excitation into the page can either be of the form $J_z \cos(\omega t)$ (spatially uniform, temporally sinusoidal wave, or SUTS wave for short), or of the form $J_z \cos(ky - \omega t)$ (traveling wave). These excitation forms have been simulated both
for linear magnetic materials (with $\mu$ set to $3000\mu_0$), as well as for NiFe with the nonlinear B-H curve depicted in Figure 5-13. Below are brief discussions of the results from each of these simulations.

**SUTS wave, linear materials**

In the spatially uniform, temporally sinusoidal (SUTS) wave case, the symmetry of the problem reduces it to a one-dimensional geometry. In the series of plots in Figure 5-15, we have depicted electromagnetic field snapshots (again, with $\mu$ set to $3000\mu_0$) at different times across a line along the $x$ direction. The snapshots span half a period of the input waveform.

Each plot in Figure 5-15 showcases, from top to bottom, the electric field ($E_x$), the magnetic field ($H_y$), and the flux density ($B_y$), respectively. The flat region in the center where the electric field is uniform defines the conductor and the air gaps around it, whereas the rest on either side corresponds to NiFe. Notice that the diffusion wave inside the NiFe can be described by a sinusoid traveling into the material on either side, modulated by an exponential decay, as expected. Indeed, the analytical solution to this test case matches the results of the FDTD simulation exactly. It has also been observed that the linear eddy current result discussed in 5.1.3 gives the fundamental mode of the diffusion wave observed in Figure 5-15. Therefore, the FDTD algorithm in the linear, one-dimensional case is believed to work well.

**SUTS wave, nonlinear NiFe**

Let us see what happens when we allow NiFe to exhibit nonlinear magnetization. Figure 5-16 depict the diffusion waves in the presence of saturation. This time, following the B-H characteristics of Figure 5-13, the $B_y$ field inside NiFe saturates near the value of 0.86 T. It appears that the nonlinear diffusion wave penetrates slightly further into the material than does the linear diffusion wave.

**Traveling wave, linear materials**

In the case of a traveling wave, our problem becomes two dimensional. For the series of plots in Figures 5-17 and 5-18, we have used symmetric boundary conditions on either side of the computation space, truncating it along half a wavelength.
Figure 5-15: Snapshots of field variables \((E_z, H_y, \text{ and } B_y \text{ respectively})\) within linear materials inside the computation space of Figure 5-14. The magnetic permeability of NiFe is taken as \(3000\mu_0\). The snapshots are taken at different phases of the input excitation: (a) phase 0; (b) phase \(\frac{\pi}{4}\); (c) phase \(\frac{\pi}{2}\); (d) phase \(\frac{3\pi}{4}\).
Figure 5-16: Snapshots of field variables ($E_z$, $H_y$, and $B_y$ respectively) within nonlinear materials inside the computation space of Figure 5-14. The snapshots are taken at different phases of the input excitation: (a) phase 0; (b) phase $\frac{\pi}{4}$; (c) phase $\frac{\pi}{2}$; (d) phase $\frac{3\pi}{4}$. 
Figure 5-17: Magnitude of magnetic flux density, $|\mathbf{B}|$, magnetic field magnitude, $|\mathbf{H}|$, and local magnetic permeability within the computational space of Figure 5-14, in the case when the excitation is a traveling wave, and the materials are assumed linear. The magnetic permeability of NiFe is taken to be $3000\mu_0$. 123
Figure 5-18: Continuation of Figure 5-17.
In Figure 5-17, the first two columns of figures depict the magnitudes of the \( \mathbf{B} \) and \( \mathbf{H} \) fields inside the computation space, respectively. The last column shows the magnetic permeability, which is constant at 3000\( \mu_0 \) within the NiFe and \( \mu_0 \) elsewhere. The first three rows of snapshots are chosen to illustrate the transient behavior, which includes the modulated section of the input. The remaining rows of pictures depict the field distribution pattern traveling with the excitation current over half a period. The slight asymmetry in the field plots arises from the asymmetry of the particular meshing scheme involved. As depicted in Figure 5-11, a particular unit cell belongs in a single material, and material properties change at the location of the \( \mathbf{H} \) fields. By convention, \( \mathbf{H} \) field nodes on the top and left are associated with a given unit cell. For instance, the unit cell on top of the bottom NiFe in Figure 5-18 possesses NiFe properties, and the \( \mathbf{B} \) field value right on the surface represents the magnetic flux density within NiFe on that surface. However, the \( \mathbf{B} \) field node on the top NiFe surface represents the flux density within air. Hence, the first \( \mathbf{B} \) field within the top NiFe is actually \( \Delta x \) over the geometric boundary. This is simply an artifact of the particular bookkeeping method employed. As long as the physical location of the field nodes are kept consistent, the results are accurate.

In Figures 5-17 and 5-18, the excitation frequency is 50 kHz, resulting in a skin depth of about 40 \( \mu \text{m} \) within NiFe. In fact, all the material properties, dimensions and excitation parameters (except for amplitude) are chosen to be identical to those studied in Section 5.1.3. Therefore, as Figure 5-8 illustrates, we are operating in the “flux crowding” regime. The fundamental of the \( \mathbf{B} \) field at the NiFe surface is amplified by a factor of about 8, compared to its value in the standing-wave case. The amplitude of the input current excitation in Figure 5-17 has been deliberately chosen to correspond to a case which yields \( \mathbf{B} \) magnitudes just over \( B_{\text{sat}} \) at 50 kHz, and much lower than \( B_{\text{sat}} \) at low frequencies. In the next section, saturation that occurs in the nonlinear case due to this flux crowding will be illustrated.

**Traveling wave, nonlinear materials**

This time, the setup of the problem is the same as in the linear case above, except here, the magnetization curve of Figure 5-13 determines the instantenous magnetic permeability used for NiFe. The series of snapshots depicted in Figures 5-19 and 5-20 demonstrate how the magnetic flux density within NiFe rises, and how the magnetic permeability responds locally to saturation.
Several observations are in order. First, in the nonlinear case, the transient behavior takes longer to settle (Figure 5-19) for the same input, compared to the linear case discussed earlier. Also, notice that the initial value of the magnetic permeability is $\mu_0$ everywhere (Figure 5-13), as enforced by the region of the magnetization curve near the origin. As the saturation pattern penetrates deeper and settles in, NiFe beyond the saturation region continues to exhibit only a slight increase in local $\mu$. This indicates that the H field deep inside NiFe is non-zero but much below $H_{sat}$. However, this area beyond the saturation region carries negligible flux. Indeed, the total net flux within the material is lower compared to the linear case, which explains the torque reduction in the presence of saturation.

The FDTD simulation tool created is also capable of computing eddy current and resistive losses within materials; this will be discussed in more detail in Section 5.4.1. It is interesting to note in this context, however, that eddy current losses are lower in the presence of material nonlinearities, thanks again to reduced flux levels within the NiFe. Hysteresis losses, on the other hand, do not come into play until many tens of MHz, and are ignored in the loss computations.

### 5.3 Modeling the micro motor

The nonlinear FDTD algorithm has been extended to the computation space of Figure 5-10. The stator and rotor geometry and material properties have been mostly selected to match those of the actual tethered magnetic micro motor tested; see Figure 5-23 and Section 5.4 below. The snapshots in Figure 5-21 illustrate the evolution of magnetic flux density within the micro motor, over a time span of about 1.5 periods at 50 kHz. Though not separately labeled, the wire slots and the stator-rotor air gap are clearly distinguishable due to high B field boundaries within the surrounding NiFe surfaces. Recall that there exist two winding phases within each wire slot, carrying currents in quadrature. Since the skin depth within copper at the operating frequencies of the micro motor is larger than the smallest dimension of the wires, an effective uniform current density within the wire slot is assumed. In Figure 5-21, the only difference is that the teeth separation in these simulations is larger – 20 $\mu$m, instead of about 15 $\mu$m.
Figure 5-19: Magnitude of magnetic flux density, $|\mathbf{B}|$, magnetic field magnitude, $|\mathbf{H}|$, and local magnetic permeability within the computational space of Figure 5-14, in the case when the excitation is a traveling wave, and the materials are assumed nonlinear. The magnetic permeability of NiFe is based on the measured magnetization curve in Figure 5-13.
Figure 5-20: Continuation of Figure 5-19.

128
Figure 5-21: Some keyframes of the magnitude of magnetic flux density within the tethered magnetic micro motor. Saturation is clearly evident in these simulation results.
the excitation amplitude is 6 A within each winding, corresponding to one of the experimental conditions.

Again, several observations are in order. Notice the heavy saturation around the wire gaps, and especially the bottleneck for flux at the stator teeth. Most of the stator pole and backiron volumes are useless, in that they do not contribute to flux linkage at all. The same can be said for most of the rotor backiron, as well. Already, one can see that intense saturation must be the main reason for low values of torque measured with the tethered micro motor.

The second and the third columns of pictures in Figure 5-21 are organized such that they are almost half a period apart in time within each row. By the time one full period has completed (third column), we can observe an interesting phenomenon: a new saturation pattern around a wire slot begins to ripple out before an earlier saturation region has time to completely dissipate.

### 5.3.1 Computing torque

Torque output from the micro motor is calculated in the same manner as outlined in Chapter 2. Basically, the Maxwell Stress Tensor is integrated around the rotor section (from \( x_l \) on the left to \( x_r \) on the right) in the computation space of Figure 5-10 to find the shear force acting on the rotor. At a given radius, the computed shear force is multiplied by the number of poles to obtain the torque per unit depth at that radius. The final expression for torque in a continuous system would be

\[
\tau = p \int_{R_{inner}}^{R_{outer}} \left( \int_{x_l(r)}^{x_r(r)} \mu_0 H_x(x, y_o) H_y(x, y_o) \, dx \right) r \, dr
\]

where \( p \) is the number of stator poles (six in our case), and \( y_o \) is taken to be the ordinate value just on the rotor conductor. Note that the extent of the computation space depends on the radius. When applied to the FDTD discretization, the shear force at a given radial cross-section can be approximated by

\[
\tau_{xy}^{n+1/2} = p \sum_{i,j=y_o} \mu_0 H_{x_{i,j,o}}^{n+1/2} H_{y_{i,j,o}}^{n+1/2} \Delta x_{i,j,o}
\]

In our implementation, several simulations at different radii are performed, and the total
Figure 5-22: Time evolution of torque for a sample current input (2 Amps, 30 kHz). The dashed line corresponds to the steady-state torque value extracted.

5.4 Results and Discussion

5.4.1 Certain Uncertainties

Recreating the exact experimental conditions in numerical simulations is not possible. For one thing, our model is only a two-dimensional approximation of the three-dimensional micro machine. Certain three-dimensional effects such as eddy currents turning corners at the inner and outer radii are not captured in our model. Another issue is that local nonuniformities in each material’s set of properties—such small voids within the electroplated NiFe—are not considered in these simulations. Moreover, parameters such as dimensions and operating temperature vary across the micro motor, and using their average values is yet another approximation that introduces potential errors. In this section, we discuss those parameters whose variation or uncertainty has the most prominent effect on simulated results.
Figure 5-23: A radial cross-section schematic of the magnetic induction micro-motor, and the relevant dimensions used to simulate it.

**Geometric dimensions**

In order to accurately represent and simulate the micro motor that has been tested, we need to know the relevant dimensions. Figure 5-23 depicts these dimensions. Notice that some of the dimensions span a range of values, either due to experimental uncertainty in their measurement or to spatial variation in value. The mask set was designed for 10 μm of teeth gap, for instance, but home-made photo-reduction chromium masks end up having a substantial edge roughness, and aspect ratio limitations of SU-8 end up increasing the gap. Consequently, the tooth gap, on the average, is much closer to 15 μm than 10 μm. The rotor conductor, on the other hand, was fabricated to be around 8 μm, and the real average thickness is probably very close to 8 μm (DC sputtering of metal films can be controlled quite accurately at this thickness). The 10 μm upper bound on the film thickness in Figure 5-23 arises mainly from thickness variations due to edge effects, and measurement uncertainties on the SEM that has been used for confirmation. The rotor conductor thickness and the tooth gap have been systematically varied over their respective ranges during simulation runs, along with other variables with potentially significant
impacts on torque so as to obtain a best fit to the experimental data. A tooth gap of 15 \( \mu m \), in combination with a rotor conductor thickness of 8 \( \mu m \), does indeed yield the best agreement with the measured torque.

Because the most prominent saturation effects take place at the maximum current, the measured torque at 6 A provide the ideal set of data with which to study the effects of geometric variations. Figures 5-24 and 5-25 illustrate the simulated effects of variation in the tooth gap and rotor conductor thickness, respectively, while all other parameters are kept constant. A smaller tooth gap results in increased saturation just around the stator teeth and yields less torque. Also, with saturation effects ever more prominent at lower frequencies, the peaks of the torque curves shift down in frequency. Reducing the simulated rotor conductor thickness has the effect of shifting the torque curves up in frequency. One way to reason this that it takes a higher input frequency to achieve the same magnitude of penetration within copper. In other words, field magnitudes inside the thinner rotor copper need to be at a higher frequency to decay to the same levels at the Cu-NiFe interface. As the peak is shifted higher in frequency, the peak amplitude is reduced slightly due to larger eddy-current effects. In summary, a careful study of Figures 5-24 and 5-25 reveals that 15 \( \mu m \) of tooth separation with 8 \( \mu m \) of rotor copper thickness is the only combination (within the allotted range) of these geometric dimensions that agrees well both with the magnitude and the peak frequency of the measured torque at 6 A.

It is interesting to note here that, in the absence of eddy-currents and saturation, the uncertainty in the tooth gap does not change the solution. However, in the nonlinear case, output torque is quite sensitive to stator tooth gap. Simulating the micro motor with an average tooth gap is an optimistic approximation; it ignores localized changes in saturation and flux density and flux re-routing, all of which eventually effect the output torque. Hence, simulation results should be evaluated with this inevitable uncertainty in mind.

Finally, consider the rotor-stator air gap. Figure 5-23 depicts this air gap to be about 70 \( \mu m \). In fact, the Kapton die that houses the rotor core sits on top of a spacer layer that is

\[ \text{Here is another way to explain the same phenomenon: There is an effective } \frac{L}{R} \text{ time constant associated with the rotor of induction machine. Torque output of the machine peaks when } \omega_{\text{slip}} = \frac{R_t}{L_t}. \text{ In the case of the tethered magnetic induction micro machine, slip frequency is the same as the input electrical frequency. As the rotor conductor gets thinner, } R_t \text{ increases, and so does the frequency at which the torque peak occurs.} \]
Figure 5-24: Torque predictions from simulations for various tooth gaps. Measured values are overlayed for reference. The rotor conductor thickness is taken to be $8 \mu m$, and the temperature within the micro-motor is constant at $250^\circ C$ above room temperature.

Figure 5-25: Torque predictions from simulations for various rotor conductor thicknesses. Measured values are overlayed for reference.
The rotor is recessed within the Kapton housing at a slight angle (exaggerated in this figure). The average recess is about 5 μm. Tethers are not shown for clarity.

65 μm thick, but the rotor core itself is slightly recessed within its housing. This is probably because the rotor was manually dropped into its housing over a glass slide and glued from top; as the glue dried, it shrank and pulled the rotor up with it. Unfortunately, this recess is not uniform around the circumference of the rotor (Figure 5-26). The average recess is about 5 μm, resulting in an average air gap of about 70 μm, which is the dimension used in simulations.

**Magnetization curve characteristics**

Besides some of the geometric dimensions, another variable that introduces uncertainty is the magnetization curve. Specifically, the B-H characteristic of electroplated NiFe has been measured using thin rings of this material wound in a transformer configuration (Appendix B), which is an AC measurement. Due to voltage resolution limitations of the experimental apparatus, the lowest measured frequency is around a few kHz. Ideally, however, a B-H curve at DC is used to determine saturation effects; this way, eddy-current effects are excluded from the measurement. The magnetization curve in Figure 5-13, however, was measured at 7.5 kHz, a frequency at which the skin depth within NiFe is smaller than the width and the thickness of the rings. How, then, are we justified in using that curve as our input B-H relationship for the FDTD simulations? The answer is that the micro motor operates so deep inside the
Figure 5-27: The time-evolution of simulated torque output from the magnetic micro motor, with two different B-H curves for the electroplated NiFe. The slightly lower steady-state value of torque corresponds to the B-H curve which is shifted towards lower H by a factor of two from the original curve.

saturation region that it does not matter too much whether we use the magnetization curve at a mid-frequency or at DC. Of course, it matters somewhat, and to see just by how much, we simulated the micro motor both with the magnetization curve of Figure 5-13, as well as the same curve shifted lower in H by a factor of two. In the worst case (see Figure 5-27), the difference in simulated steady-state torque has been found to be 7%, with the typical discrepancy being around 4%. In fact, simulated steady-state torque results are quite insensitive to shifts in the B-H curve of electroplated NiFe, as long as $B_{sat}$ is known accurately and $H_{sat}$ is much smaller than the typical H field magnitudes within NiFe.

The same trick of shifting the B-H characteristics of a material can be applied to the magnetization curve of the NiFe wafer, which saturates at least two orders of magnitude earlier in $|H|$. Recall that one of our assumptions in arriving from the $\mathbf{B}$ field update Equation 5.30 to the final, explicit Equations 5.31 and 5.33 is that the magnetic permeability does not change
Comparison of simulated torque evolutions with different B-H curves

Original and modified wafer B-H curves

Figure 5-28: The evolution of torque with two different B-H curves for the NiFe wafer. The two transients are essentially identical, except that the original B-H curve eventually results in numerical instabilities.

by much over a time step. With that assumption, we have

\[ \mu_{ij}^{n+1/2} \approx \mu_{ij}^{n-1/2} \]  

for any \( i \) and \( j \). However, the approximation in Equation 5.39 becomes invalid for the B-H characteristics of the NiFe wafer, because it saturates too fast, and the magnetic permeability actually changes appreciably during a unit time step. The result is numerical instability in the FDTD algorithm. This problem could be addressed by reducing the time step substantially to make Equation 5.39 valid. However, shifting the B-H curve instead to relieve the stiffness in the problem works like a charm. Figure 5-28 illustrates this method. Notice that the output torque result based on the shifted B-H curve yields identical results as that based on the original B-H curve, without the numerical instability. Once again, we can get away with such a lateral shift in the B-H curve, because the input current levels for the micro motor are large enough to
guarantee operation well over the $H_{sat}$ of the resulting curves.

Temperature

Since the rotor and the stator surfaces face each other across a very small air gap, incorporating thermocouples on those surfaces during torque measurements is not possible. Even adding a thermocouple on the rotor backiron is not practical, since we must know the moving rotor mass accurately to determine torque. Using infrared cameras to deduce operating temperatures is also not possible because the rotor core blocks the region of interest from view. Hence, temperature remains a variational parameter in our simulation studies.

The relative insensitivity of the results to B-H curves shifted along the H axis comes in handy in the consideration of uncertain temperatures. To first order, the effect of temperature on the NiFe magnetization characteristics is to shift $H_{sat}$ [5]. As long as the operating temperature is not very near the Curie temperature (450 °C for 80-20% NiFe), $B_{sat}$ remains mostly constant. As will be discussed below, the maximum temperature at the stator and rotor surfaces does not exceed 300 °C, and the resulting shift in the B-H curve [5] does not significantly alter the simulation results. The effect of temperature on the B-H curves is therefore ignored (but not neglected).

A change in temperature, however, changes the electrical conductivity of both NiFe and copper⁶, and this is where temperature effects become critical. An increase in temperature reduces the NiFe conductivity and increases the skin depth within the material, which, in turn, reduces saturation and increases the torque (Figure 5-29). Such a temperature increase also reduces the rotor copper conductivity, which has the effect of shifting the peak of the torque towards higher frequencies (Figure 5-30). This is partly because field magnitudes inside the more resistive rotor copper must be at a higher frequency to decay to the same levels at the Cu-NiFe interface⁷. Also, the peak torque values are lowered as the magnitude of the induced currents inside the rotor copper drops off slightly with increased resistivity.

Figures 5-29 and 5-30 illustrate the effect of temperature on micro motor performance from

---

⁶See Appendix B for a discussion of the temperature dependence of NiFe and Cu electrical conductivity.

⁷Again, the alternative explanation is that an increase in the rotor conductor resistance reduces the $\frac{L}{R}$ time constant of the rotor, hence shifting the peak frequency towards higher values.
Simulated Torques at Various Electroplated NiFe Temperatures

- Elect. NiFe temp = 0°C
- Elect. NiFe temp = 50°C
- Elect. NiFe temp = 150°C
- Elect. NiFe temp = 250°C
- Elect. NiFe temp = 400°C

Simulated for teeth gap: 15 μm,
Rotor cond. thickness: 8 μm,
Rotor cond temp = 250°C,
Input Current: 6 A

Electrical Excitation Frequency (Hz)

Figure 5-29: Simulated torque values for various temperature rises (from 300 K) within the electroplated NiFe of the stator and the rotor. Rotor conductor temperature taken constant at 250°C.

Simulated Torques at Various Operating Temperatures

- Temp rise = 0°C
- Temp rise = 50°C
- Temp rise = 150°C
- Temp rise = 250°C
- Temp rise = 300°C
- Temp rise = 400°C

Simulated for teeth gap: 15 μm,
Rotor cond. thickness: 8 μm,
Elect. NiFe cond: 22μS/m
(taken constant at 250°C),
Input Current: 6 A

Electrical Excitation Frequency (Hz)

Figure 5-30: Torque predictions from simulations for various rotor Cu temperatures (and hence, rotor conductor resistivity). For purposes of comparison, the temperature has not been factored in for NiFe conductivity.
Simulated Torques at Various Micro-Motor Temperatures

Overall term = 0°C
Overall temp = 50°C
Overall temp = 150°C
Overall temp = 250°C
Overall temp = 400°C

Simulated for teeth gap: 15 μm,
Rotor cond. thickness: 8 μm,
Input Current: 6 A

Figure 5-31: Simulated torque values at different overall temperatures within the micro-motor.

...the individual point of view of electroplated NiFe and rotor copper, respectively. Consistency requires, of course, that the temperature of electroplated NiFe and the rotor Cu is taken to be the same for those simulations intended to represent operating conditions within the magnetic micro motor. Figure 5-31 explores the effect of overall micro motor temperature on output torque. Notice that an overall temperature of 250°C appears to provide the best match to the measured data; the corresponding simulation data also predicts the peak frequency correctly. An exhaustive study of temperature around this “nominal” value has indicated that a temperature variation of ±50°C about 250°C result in slightly worse, yet plausible fits to measurements (Figure 5-32).

Temperature at lower current levels

We could go ahead and make similar temperature fits to the measurements at lower current levels. However, consistency once again requires that we have a temperature model that applies to all current inputs. The factors that have an impact on heating within the micro machine are resistive ($I^2R$) losses within the stator windings, resistive losses within the rotor copper, and eddy-current losses within the NiFe core (hysteresis losses are inconsequential in this frequency...
Simulated Torques at Various Micro-Motor Temperatures

Simulated for teeth gap: 15 μm,
Rotor cond. thickness: 8 μm,
Input Current: 6 A

Figure 5-32: Simulated torque values at temperatures between 200°C and 300°C above room temperature make good fits to the measured data at 6 A.

range). The latter phenomenon depends both on frequency and input current amplitude. Resistive losses in the stator include the resistance of the copper windings at either phase, as well as the contact resistance of the current probes. The total resistance of the stator current path has been measured to be around 0.5 Ω; given the conductivity of copper (5.8 × 10⁷ S/m) and the cross-sectional area of the windings (200 μm×65 μm inside the wire slots and inner end-turns, wider at the outside), less than 0.1 Ω (a good estimate is about 0.05 Ω) of that is winding resistance. Hence, stator resistive losses are dominated by the contact resistance.

Interestingly, it turns out that the overall dissipation within the tethered micro machine test device is also dominated by contact resistance. Since FDTD simulations compute, among other variables, the electric field everywhere inside the computation space, it is straightforward to extract instantaneous eddy-current losses within the NiFe core (both electroplated and the stator wafer) and the rotor conductor by summing up resistive loss densities in these volumes.
Figure 5-33: Comparison of simulated power loss mechanisms within the test device. Power dissipation due to stator winding and contact resistance dominates the overall heat generation process.

This yields

$$P_{\text{diss}}(t) \approx W \sum_{i,j} \sigma_{ij} E_{zij}^2(t) \Delta x_{ij} \Delta y_{ij}$$  \hspace{1cm} (5.40)$$

where $W$ is the depth of the computation space (1 mm), and the extent of the summation is chosen to include the region of interest. Incorporating Equation 5.40 into the FDTD simulation program, we have computed estimated power losses within the test device. Figure 5-33 illustrates how the power dissipation due to stator winding and contact resistance compares to the rest of the dissipation processes. We have chosen to depict the comparison in the worst-case eddy-current loss – which occurs at the highest applied frequency given a current amplitude.
(winding losses are independent of frequency). Notice that even in the worst-case scenario, the winding losses are still an order of magnitude larger than the rest of the dissipated power. In a way, this is welcome news, as it implies that in an eventually integrated magnetic micro machine where contact resistance will not be major issue, effective heat sinking techniques could achieve a very small temperature rise within the device.

Sticking with our assumption of an isothermal micro machine, Figure 5-33 suggests that the operating heat loss of the test device depends mainly on the square of the input current amplitude. Assuming thermal coefficients independent of temperature, we find that the micro machine temperature rise is proportional to input current amplitude squared. In other words, in order to estimate the operating temperatures for current levels lower than 6 A, all we need to do is to scale the temperature values in Figure 5-32 with the ratio of the current amplitudes squared. In Figures 5-34 through 5-37, this is exactly what we have done. The worst-case discrepancy between the measurements and simulation results is about 30%, in the case of 3A current amplitude, Figure 5-35. This is a huge improvement over the results from ANSOFT.

Interestingly, Figure 5-33 also shows that, unlike winding losses, eddy current losses do not scale as \( I^2 \) in the presence of saturation.

---

Figure 5-34: Simulated torque values for 2A input current amplitude.
Figure 5-35: Simulated torque values for 3A input current amplitude.

Figure 5-36: Simulated torque values for 4A input current amplitude.
Simulated Torques at Various Micro-Motor Temperatures

Overall temp = 174 °C
Overall temp = 208 °C
Overall temp = 243 °C

Simulated for teeth gap: 15 μm,
Rotor cond. thickness: 8 μm,
Input Current: 5 A

Electrical Excitation Frequency (Hz)

Figure 5-37: Simulated torque values for 5A input current amplitude.

We believe part of the discrepancy at 3A may be due to poorly balanced stator phases during the experiment at this current amplitude, which would reduce the measured torque. Also, the simulations are two-dimensional and cannot model such effects as eddy-currents turning at the inner and outer radii of the micro machine. Such three-dimensional effects act to reduce the measured torque, and they are more prominent at lower current levels where saturation is less severe. These effects could account for part of the discrepancy, especially at lower current amplitudes.

5.4.2 Afterthoughts

Compared with the results from commercial finite element analysis (FEA) software packages such as ANSOFT, the FDTD results and their agreement with the measured data in a consistent manner provide evidence for the value and the validity of the general approach. Not only does the modified FDTD method improve simulation accuracies by an order of magnitude, it is also quite fast. The modified FDTD approach is a promising candidate for a micro-scale nonlinear electromagnetic simulation package; as implemented, the method can be used to study any magnetic actuator and microsystem. It can also be modified easily for nonlinear microsystems in
the electrical domain, such as electrostatically actuated micro-mirrors for optical applications. The extension of the FDTD algorithm into three dimensions is also straightforward, though perhaps unnecessary at this stage.

The power of the FDTD approach is not limitless, however. In linear cases, established matrix methods can usually provide results much faster. Also, being an explicit and only second-order accurate algorithm, the FDTD method does not generally yield results that are more than 10-20% accurate. The approach is better suited to study transient effects and nonlinear dynamics associated with stiff partial differential equations that may otherwise be impractical to solve using other methods.

As Figure 5-38 illustrates, our modeling efforts have taken us through successively larger regimes of applicability, from a linear non-eddy-current forming model to a full magnetic diffusion solution in the presence of eddy currents and saturation.

At each step, the validity of the new model has been confirmed with the results from previous models and commercial FEA software, as well as analytical expressions where possible. With this methodology, the numerical accuracy of the predictions from our models (in light of the tethered magnetic induction micro machine torque measurements) has been improved from an
order of magnitude off (Model I) to less than 30% off in the worst case (Model III). The author believes that this modeling methodology has resulted in efficient computation tools that can be applied to the design and study of virtually any magnetic sensor or actuator, both in the micro, as well as the macro scale.

For instance, the designer would first determine the relative effects of eddy currents and whether flux density levels with the magnetic materials are high enough to warrant a nonlinear model. This determination could be done speedily and efficiently using Model II. Subsequent design optimizations could be performed using the appropriate model. Model II, especially if implemented in three dimensions, is a convenient means to determine the required lamination thickness if undesirable eddy current effects are bound to be present at the operating requirements of the magnetic machine.

In the case of the magnetic induction micro machine, the NiFe material properties, together with the operating frequencies of the device, determine the lamination thickness necessary to achieve close to full torque. If the micro gas turbine rotates at 2.4 Mrpm, the corresponding synchronous electrical frequency is 120 kHz for a six pole stator. Assuming a 10% slip in the generator mode, the operating frequency at the peak torque will be 108 kHz, resulting in a necessary lamination thickness of approximately 17 μm to recover 90% of the torque in the absence of eddy-currents. This estimation does not take the packing factor of laminations into account; when factored in, a practically achievable packing factor of 50% will reduce the output torque accordingly. A more detailed discussion of the lamination thickness estimation can be found in Appendix C.

The conclusion is that a good portion of the torque predicted by the design tool based on Model I can be recovered through the lamination of the stator. Figure 5-39 shows a potential lamination scheme for the electroplated NiFe of the stator. Early versions of the laminated devices are intended to test fabrication challenges and limitations; for simplicity, they will not incorporate laminations at either the stator or rotor backiron cores. Depending on the lamination thickness that can be achieved, we expect that most of the design torque will be recovered. In order to achieve full torque, however, the entire machine must be laminated.

It is the author’s conviction that research for better magnetic materials must be conducted in parallel with lamination efforts. The eddy currents within the NiFe can be brought down
Second Cu winding

Second level of Laminated SU-8 and NiFe poles

Complete stator with laminated NiFe poles and ferrite substrate

Figure 5-39: Potential lamination schemes for the stator. (Figure courtesy of Florent Cros of Georgia Institute of Technology)
by reducing the electrical conductivity of the electroplated NiFe in the first place. In the macro-scale manufacturing, a small percentage of silicon compounds is introduced into the iron to reduce its conductivity. In the micro-scale fabrication, deliberate introduction of low density nonmagnetic and nonmetal ionic contaminants that dissolve in water, such as calcium ions, into the electroplating bath may help reduce the conductivity of the NiFe substantially. Such impurities will no doubt degrade the magnetic properties (such as decrease the magnetic permeability) of the NiFe slightly, but the accompanying reduction in electrical conductivity is well worth it. Besides, a lower magnetic permeability actually helps increase the skin depth within the magnetic material. As long as the saturation flux density, $B_{\text{sat}}$, is not substantially lowered, the resulting NiFe will significantly improve the performance of the magnetic induction micro machine, making mildly laminated stators successful.
Bibliography


Chapter 6

Concluding Remarks

6.1 Summary

In this thesis, we presented the analysis, design, fabrication, testing, and nonlinear re-analysis of an axial gap magnetic induction micro machine. As part of this thesis, a tethered magnetic induction micro motor was created as a test-bed for our models and the specific fabrication techniques. Chapter 2 presented a two-dimensional, multi-modal, linear model (Model I) that is valid for a machine incorporating a linear, non-conducting (or laminated) NiFe core. This model coupled a stator section that was analyzed via a magnetic circuit approach to a rotor section that was modeled using continuous boundary layer equations derived from magnetic diffusion in moving media [1]. Many Fourier modes (up to 20) were included in the analysis to guarantee high accuracy, and the results were cross-checked with a commercially available FEA program (ANSOFT) for confirmation. Based on this model, a software design tool was created. This design tool is orders of magnitude faster than the FEA program, yet it is also reliable and accurate. Magnetic induction micro machine designs performed using this software tool revealed the potential for greatly improved torque and power levels with respect to the electric induction micro machine. Moreover, the low pole count of the stator of the magnetic machine made large air gaps possible, a fact that rendered viscous fluidic losses negligible. This meant higher efficiency and simpler drive electronics requirements (due to the low voltage requirements, and the much lower excitation frequencies needed) in comparison to the electric machine. In short, it was concluded that magnetic induction micro machines are viable candidates for the
micro gas turbine engine.

Chapter 3 summarized the microfabrication efforts needed to develop and demonstrate a test-bed for both our models, as well as the required fabrication processes. This test-bed device is the tethered magnetic induction micro motor [2]. The fabrication of this device was undertaken by our collaborators at Georgia Institute of Technology, Atlanta, GA. As a visiting student at GIT, the author was involved in the rotor die design and its fabrication using SU-8 epoxy. The NiFe rotor was suspended within the rotor die via six tethers, which were designed to be rather stiff for deflections out of the rotor plane, yet providing a compliant spring structure for rotations in the plane of the die. The stator fabrication was briefly summarized in Chapter 3 for completeness. The fabrication work summarized in Chapter 3 demonstrated that it is indeed possible to fabricate thick and stacked electroplated structures of the magnetic induction micro machine. It was observed that most fabrication limitations depended on either the skill and the patience of the fabricator, or the highest aspect ratio structures achievable in SU-8 epoxy.

In Chapter 4, we introduced the experimental setup used to test the tethered micro motor, including the specialized computer vision system [3]. Drive electronics were contracted to provide up to 20 A peak of AC stator current (through each phase) at frequencies up to 120 kHz. These drive electronics were integrated with the computer vision system, resulting in a test setup capable of generating and measuring oscillatory motions in the tethered motor. Torque was extracted from the functional dependence of the oscillation amplitudes to the mechanical oscillation frequency. In a series of experiments, both input stator excitation frequency and the current amplitude was varied systematically. In initial results using SU-8 tethers, over 5 μNm of torque was obtained for a small air gap estimated to be around 5 μm. Melting problems in the SU-8 led to more systematic tests using Kapton tethers. Up to 0.3 μNm for an air gap of 70 μm was measured. Though orders of magnitude larger than what has been achieved in magnetic micro motors, these torque results are much lower than expected by the corresponding predictions of Model I, indicating that eddy-currents within the stator indeed result in significantly reduced torque levels. Hence, the need for further modeling was established.

Chapter 5 undertook further modeling in two steps. First, a linear, eddy-current model (Model II) based on a vector potential formulation using finite-differences was introduced.
This model was used to study the effects of eddy-currents within the electroplated NiFe. It showed that the traveling-wave nature of the stator excitation, combined with the small skin depth within NiFe at the frequencies of operation, was responsible for flux crowding within the magnetic material. This phenomenon was further studied within the context of a gedanken experiment, which involved the simplified setup of a copper wire sandwiched between two NiFe slabs. It was observed that flux crowding quickly raised the magnetic field density within the material surface to levels where deep saturation would be prominent. Hence, the reasons for the inadequacy of Model I – namely, eddy-currents and the saturation they cause – in explaining the experimental results were determined, and the need for a new, nonlinear model was established. Chapter 5 then went on to develop a fully nonlinear, finite-difference time-domain method (Model III) to solve Maxwell’s equations. Model III took advantage of the fact that magnetic diffusion waves are much slower than the speed of light, which was artificially reduced (through increasing the dielectric permittivity everywhere) to a point where stiffness inherent in the equations was mitigated. Using this method, time-evolution of magnetic fields and field densities within the micromotor was computed, and steady-state torque values were extracted. It was observed that, given a simple yet plausible temperature model, the results from Model III explained the data well.

6.2 Conclusions

The primary goal in the design and development of the tethered magnetic induction micro motor was to both push and understand the fabrication challenges involved and to provide a test-bed for the validation of our models. In that respect, the immediate goal of this thesis has been successfully met. As is, the performance of the tethered micro motor does not meet the eventual power requirements of the micro gas turbine generator; however, the physical understanding based on the experimental and modeling results discussed in this thesis provides the basis to achieve the eventual power goal. In particular, the results of this thesis indicates that a magnetic induction micro machine powerful enough for the purposes of the Micro Engine Project can be built if the eddy-currents within the stator can be prevented or reduced. Assuming a perfect packing factor, it is estimated that a lamination thickness of about 17 µm within the six-pole
stator can capture 90% of the torque levels\(^1\) needed to achieve the performance predictions discussed at the end of Chapter 2. However, such thin laminations with high packing factors will very likely present new fabrication challenges. It is the author’s conviction that fabrication of stator laminations must proceed in parallel with research for higher resistivity magnetic materials for MEMS.

The foremost achievement of this thesis is the design and demonstration of the first magnetic induction micro machine. Among the contributions of this thesis to the field of MEMS is the creation of design and analysis tools for any magnetic micro machine operating at any frequency and input level. The analysis tool based on Model II, for instance, is a good starting point to test the operation regime of a magnetic micro machine, and to see whether eddy-currents and/or saturation effects are significant in a chosen device configuration. Then, depending on the regime of operation, the software based on either Model I or Model III could be used to optimize the design and predict performance. Models I and II could be used for macro scale magnetic induction machine design, as well. Model III is more general in its application scope within MEMS; it can be used to study virtually any micro scale magnetoquasistatic system, such as microinductors or magnetic actuators.

Another major achievement is the demonstration of current density levels in excess of \(1 \times 10^9\) A/m\(^2\), simultaneously through both phases within the tiny volume of the micro motor stator. Cooling limitations within conventional macro scale magnetic machines generally restrict the input current densities to less than \(1 \times 10^7\) A/m\(^2\). However, in the MEMS scale, cooling is much more effective. This is because cooling is mainly a surface phenomenon, whereas heating is a volume effect; as things get smaller, the ratio of the cooling surface area to the heating volume increases inversely with the typical dimension. In the case of the magnetic induction micro machine, electroplating ensures all materials are in good thermal contact with each other. The small size of the die, combined with high thermal conductivity of the electroplated metals, assures an isothermal structure that can be heat sunk very effectively. Hence, effective heat sinking in the micro-scale allows a high current density that results in a high power density machine. This concept is bound to change the way people think about power in the MEMS scale.

\(^1\)See Appendix C for details.
SU-8 epoxy has been utilized for some time as a mold and structure material within MEMS devices. This thesis, in conjunction with [4], has pioneered the use of SU-8 epoxy also as a mechanical transducer component in MEMS. As part of the development for the tethered magnetic induction micro motor, we have characterized the mechanical properties of SU-8. We have found that its relatively small Young’s modulus, in conjunction with the lack of plastic deformations in this brittle material deems SU-8 to be a good candidate for large deflection transducers. We predict that in low temperature (less than 100 °C) MEMS applications such as optical components and especially in bioMEMS, SU-8 may have a great impact as the material of choice.

6.3 Suggestions for Future Work

In order to guarantee proper operation, care must be taken ensure that the magnetic induction micro machine is actively cooled inside the hot micro gas turbine generator. This is partly because the wall temperature of the micro engine during operation – about 900 K – is higher than the Curie temperature of NiFe, which is around 450 °C, or about 723 K. Also, SU-8 epoxy starts to melt around 200 °C. The temperature of the magnetic machine may be lowered by using the cold inlet fuel of the micro engine as a coolant. This will probably involve a partial redesign of the micro engine to accommodate the required diversions in the fluidic paths. Note that not only the stator but also the rotor must be cooled, as the rotor conductor losses cannot be eliminated. Whether the air flow passing through the compressor will prove sufficient to keep the rotor temperature below the Curie point is a question that should be answered via careful thermal studies.

Another materials issue is that the mechanical strength of the electroplated metals at elevated temperatures and rotor integrity at high rotation speeds have yet to be tested. More research on high temperature magnetic materials, and on how they would be integrated with the silicon micro engine, is needed. Such an integration is critical to producing higher strength rotors that can achieve high spin rates.

More materials research is also needed in decreasing the electrical conductivity of electroplated magnetic materials, in order to reduce the crippling effects of eddy-currents and the
associated flux crowding which leads to saturation. The technique used in the manufacture of large scale magnetic machines is the addition of small amounts of a nonmagnetic and non-conducting agent, such as silicon, into the molten ferromagnetic material. In the case of the magnetic micro machine, one suggestion is to try contaminating the electroplating bath with ions from a nonmetal material, such as calcium.

In the interim, though, currently available NiFe can be used with appropriate laminations\(^2\) of the stator. Since, for micro turbomachinery, magnetic materials eventually must be integrated with silicon, we suggest laminations are eventually micromachined out of silicon directly. One fabrication approach could involve using deep reactive ion etching (DRIE) to create the molds and the laminations needed for stator electroplating directly on silicon wafers. In this fashion, the low temperature SU-8 epoxy in the stator could be replaced with silicon. The new fabrication scheme for the stator would proceed in a manner similar to that outlined in Chapter 3, with each stator level being defined by a different, pre-processed silicon wafer.

In the short term, integration of magnetic materials with silicon may not result in functional generators for the Micro Engine Project, as the issues of high temperature magnetic materials still need to be resolved. However, magnetic induction micro machines integrated with silicon may still have a great impact, as electrically driven micro turbopumps used in microcooling systems for anything from computer processors to personal AC units.

\(^2\)See Appendix C.
Bibliography


Appendix A

Using the Schwarz-Christoffel Transformation for Magnetic Circuits

According to the Schwarz-Christoffel theorem, any closed polygon in the complex plane \( z = x + jy \), can be transformed into the real axis of another complex plane \( z_1 = x_1 + jy_1 \) [2]. This transformation is achieved by the equation

\[
\frac{dz}{dz_1} = A (z_1 - u_1)^{(\alpha_1/\pi)-1} (z_1 - u_2)^{(\alpha_2/\pi)-1} \ldots (z_1 - u_n)^{(\alpha_n/\pi)-1} \tag{A.1}
\]

where \( \alpha_i \)'s are the internal angles of the closed polygon in the \( z \)-plane, and \( u_i \)'s are the points on the \( x_1 \)-line corresponding to the corners of the polygon in the \( z \)-plane. In Equation A.1, \( A \) is a constant.

Consider the potential function \( \phi(x, y) \) and the stream lines function \( \psi(x, y) \), whose constant stream lines are perpendicular to the equipotential lines of \( \phi \). Let \( W(z) = \psi + j\phi \) stand for an analytical function of \( z \). If \( \phi \) and \( \psi \) both satisfy Laplace’s equation, as well as the Cauchy-Riemann conditions [3] given by

\[
\frac{\partial \phi}{\partial x} = \frac{\partial \psi}{\partial y}, \quad \frac{\partial \phi}{\partial y} = -\frac{\partial \psi}{\partial x} \tag{A.2}
\]
Figure A-1: A rectangle within the W-plane, with horizontal equipotential lines, is mapped into an L-shaped polygon in the z-plane.

then these functions in one plane can be mapped into a closed polygon in another plane, using the z1-plane as intermediary [1]. In this manner, horizontal equipotential lines within the W-plane become the equipotential lines within the polygon in the z-plane. The complex algebra associated with the particular transformations of different shapes is beyond the scope of this thesis. However, conceptually speaking, Figure A-1 illustrates the point. In Figure A-1, imagine that the rectangle on the left corresponds to a conducting material. It is assumed that the electric field is confined within this material. On the top and at the bottom edges, we apply $\phi = \phi_0$ V and $\phi = 0$ V, respectively. Horizontal lines correspond to equipotentials (same $\phi$ values for points on the lines), whereas the vertical lines correspond to the electric field between the top and the bottom. Using the Schwarz-Christoffel transformation of Equation A.1 twice, the configuration on the left of Figure A-1 is mapped onto the L-shaped polygon on the right. The new polygon is still the same material, with the top edge at $\phi_0$ V and the right edge at 0 V. The corresponding equipotential field lines are also shown.

The power of this technique is that any physical phenomenon subject to Laplace’s equation (such as heat flow, electrical fields within a strip resistor on a chip, etc.) inside a polygon composed of an arbitrary number of straight segments can be solved in a similar manner. This means high computational efficiencies can be achieved using this method, since it allows us to
avoid lengthy finite element analysis studies on these shapes. A simple computer program could numerically integrate the ordinary differential equation resulting from the transformations. In certain cases, analytical solutions are even possible.

Application of this technique to the calculation of magnetic reluctance in magnetic circuits is straightforward. In direct analogy to the electrical resistance example of Figure A-1, the magnetic reluctance of the L-shaped polygon, from the top to the right edge, is given by

\[ R = \frac{l_m}{\mu_s WH} \]

where \( l_m \) is the average magnetic path length, \( W \) is the width, and \( H \) is the depth of the magnetic slab. The average magnetic path is simply given by the height of the rectangular slab on the left of Figure A-1. Hence, calculating the magnetic reluctance of the L-shaped region on the right reduces to computing the height of the rectangle on the left.

Within the context of the model described in Chapter 2, the magnetic reluctance of the corner pieces (i.e., \( R_{tc} \) and \( R_{bi} \)) has been calculated in exactly the manner outlined here. Since the design program based on the model of Chapter 2 is intended to be applicable to any stator geometry that the designer can think of, a look-up table has been created to store the magnetic reluctance of L-shaped polygons of different relative sizes. The four dimensions that determine the size of the L-shaped region (see Figure A-2) are normalized such that one of them is always set to unity. In this fashion, the look-up table becomes three dimensional. Reluctance values for dimensions that fall between points on the look-up table are interpolated. This approach provides an accurate and extremely fast way to extract magnetic reluctance from a given stator geometry.
Figure A-2: The four dimensions that determine the size of the L-shaped region, and hence, the reluctance value $R_{tc}$. 
Bibliography


Appendix B

Material Properties

In this appendix, we briefly present the electrical conductivity of NiFe and Cu as a function of temperature, as indicated from available literature on these materials.

B.1 Electroplated NiFe

Figure B-1, taken from [1], illustrates the temperature dependence of resistivity of NiFe as a function of composition. The resistivity values in Figure B-1 are for a NiFe alloy, not necessarily for a specimen prepared using electroplating. Depending on the particular fabrication method, tiny voids of various sizes that form within the NiFe during electroplating tend to increase the resistivity of NiFe slightly. Microfabrication literature on electroplated NiFe (for instance, [3], [4], and [5]) agrees on a median resistivity value of 20 $\mu\Omega$.cm at room temperature. We have used a scaled version of the data in Figure B-1, as well as that from [2], to estimate the temperature dependence of resistivity of electroplated NiFe through a linear fit. Resistivity points have been scaled to make sure that the corresponding room temperature resistivity is 20 $\mu\Omega$.cm. Figure B-2 shows the corresponding functional dependence on temperature that is used in simulations.
Figure B-1: Electrical resistivity of NiFe as a function of the percentage of Ni in the alloy for various temperatures.
Figure B-2: Electroplated NiFe conductivity as a function of temperature.
Figure B-3: Copper conductivity as a function of temperature.

B.2 Copper

The room temperature conductivity of both the rotor copper (DC sputtered) and that of the stator windings (electroplated) is $5.8 \times 10^7$ S/m, which is the resistivity value for bulk copper. The temperature dependence of resistivity has a strong linear coefficient; therefore, a linear fit to the resistivity data as a function of temperature is also employed here. Figure B-3 illustrates the resulting functional form for electrical conductivity of copper used in simulations.
Bibliography


Appendix C

Lamination Thickness Estimation

Eddy-currents within the stator flow so as to block the flux carried inside the NiFe core. The laminations must run perpendicular to the flow direction of the eddy-currents. Since almost all magnetic flux is perpendicular to the radial direction, the stator must be laminated in an onion peel fashion (see Figure C-1). If the laminations are thin enough, the magnetic field on either side of a lamination block will essentially be equal. In that case, we can get an estimate of the extent of magnetic diffusion within each NiFe slab by approximating it as having an infinite extent along the \( \hat{y} \) direction. In this fashion, our problem becomes one-dimensional. Let the lamination thickness be \( \Delta \), with its magnetic permeability \( \mu \) and electrical conductivity \( \sigma \). The magnetic diffusion equation within the NiFe is given by

\[
\frac{\partial H}{\partial t} = \frac{1}{\mu \sigma} \frac{\partial^2 H}{\partial z^2} \tag{C.1}
\]

where \( H \) is the magnetic field in the \( \hat{y} \) direction, of the form

\[
H = \tilde{H} e^{j(\omega t - k z)} \tag{C.2}
\]

Substituting Equation C.1 into C.2, we obtain

\[
k = \pm j (1 + j) \frac{1}{\delta} \tag{C.3}
\]
Figure C-1: The stator will be laminated in an onion peel fashion, with laminations running perpendicular to the radial direction. With thin laminations, the magnetic field density on either side of a lamination block will be the same.
where $\delta$ is the skin depth expressed as

$$\delta = \sqrt{\frac{2}{\omega \mu \sigma}} \quad (C.4)$$

Equation C.3 indicates that the magnetic field, in general, is composed of a forward and a backward traveling wave, expressed as

$$H(z, t) = H_+ e^{j(\omega t - \eta z)} + H_- e^{j(\omega t + \eta z)} \quad (C.5)$$

where

$$\eta = \frac{1 - j}{\delta} \quad (C.6)$$

Taking our origin as the middle of the lamination block, the boundary condition on either side of the NiFe slab is

$$H(z = \pm \frac{\Delta}{2}) = H_0 e^{j\omega t} \quad (C.7)$$

The boundary condition of Equation C.7, together with Equation C.5, yields

$$H_0 = H_+ e^{-j\frac{\delta}{2} \eta} + H_- e^{j\frac{\delta}{2} \eta} \quad (C.8)$$
$$H_0 = H_+ e^{j\frac{\delta}{2} \eta} + H_- e^{-j\frac{\delta}{2} \eta} \quad (C.9)$$

which can be solved for $H_+$ and $H_-$ to obtain

$$H_+ = H_- = \frac{H_0}{2 \cosh \left( \frac{\eta \Delta}{2} \right)} \quad (C.10)$$

Hence, the general solution becomes

$$H(z, t) = H_0 F(\eta, \Delta) e^{j\omega t} \quad (C.11)$$
The amount of magnetic field penetration within the lamination is given by the quantity $F$, which depends on the skin depth (through $\eta$) and the lamination thickness, $\Delta$. In the limit as the lamination thickness goes to zero, the magnetic field penetration within the lamination grows to 100%. The top plot of Figure C-2 depicts $F$, which represents the normalized magnitude of the magnetic field inside the lamination as a function of position $z$, for a lamination thickness that is about 34% larger than the skin depth. Here, the skin depth is 12.5 $\mu$m, and is calculated for an average magnetic permeability of 3000$\mu_0$, conductivity of $5 \times 10^6$ S/m, and a frequency of 108 kHz, which corresponds to a 10% slip in a 6-pole generator. The area under the curve in the top plot of Figure C-2 is proportional to the total flux carried within the lamination. The bottom plot shows $F^2$ as a function of position. Assuming the lamination is thin enough

$$F(\eta, \Delta) = \frac{\cosh(\eta z)}{\cosh\left(\frac{\eta \Delta}{2}\right)}$$ (C.12)
to prevent material nonlinearities, the area under that curve is essentially proportional to the eventual output torque.

In Figure C-2, a $\Delta$ of 17 $\mu$m results in 90% of the torque being recovered by the lamination scheme. This estimation assumes a perfect packing factor, where the insulation thickness is taken infinitesimally small. In reality, the maximum aspect ratio that can be achieved within the stator insulator material will determine the packing factor of the laminations. For instance, assuming the insulation layers between the laminations have the same thickness as $\Delta$, we obtain a packing factor of 50%, which, for the case given here, results in 45% recovery of the total torque in the absence of eddy-currents. Incidentally, an insulation layer of about 17 $\mu$m in the stator, where the maximum electroplated NiFe layer depth is about 100 $\mu$m (for the teeth section), is very near the maximum aspect ratios (1:8 to 1:10) achieved in SU-8 so far. This indicates that using the fabrication processes and materials of today, somewhat less than half the power density predicted in Chapter 2 is possible to achieve through laminations.
Appendix D

Measured Data

This appendix is a compilation of the deflection measurements conducted on Kapton tethered micro motors using the Microvision system described in Chapter 4. The error bars associated with the torque values extracted from these measurements are given by the maximum uncertainty in the fits. Each set of data below is listed for the input current amplitude at each stator phase. The figures show the measurement points as stars, and the nominal fits to those measurements as solid curves. The quality factor in all the fits is about 80. The input stator frequency is represented by \( f_e \), the mechanical resonance frequency is given by \( f_o \), and the gain (as described in Chapter 3) is \( G \).

**Stator Current: 2A Peak**

\[
\begin{align*}
\text{Stator Current: 2A Peak} & \\
\text{Stator} & = 20 \text{ kHz}, f_o = 2800 \text{ Hz}, G = 1.3 \times 10^{-9}. f_e = 30 \text{ kHz}, f_o = 2800 \text{ Hz}, G = 2.1 \times 10^{-9}. 
\end{align*}
\]
$f_e = 40 \text{ kHz}, f_o = 2800 \text{ Hz}, G = 2.6 \times 10^{-9}$. $f_e = 50 \text{ kHz}, f_o = 2800 \text{ Hz}, G = 2.6 \times 10^{-9}$.

$fe = 60 \text{ kHz}, f_o = 2807 \text{ Hz}, G = 2.3 \times 10^{-9}$. $fe = 70 \text{ kHz}, f_o = 2800 \text{ Hz}, G = 2.1 \times 10^{-9}$.

$fe = 80 \text{ kHz}, f_o = 2822 \text{ Hz}, G = 2.1 \times 10^{-9}$. $fe = 90 \text{ kHz}, f_o = 2827 \text{ Hz}, G = 1.9 \times 10^{-9}$.

Stator Current: 3A Peak
$f_e = 20 \text{ kHz}, f_o = 2838 \text{ Hz}, G = 2.57 \times 10^{-9}$.  
$f_e = 30 \text{ kHz}, f_o = 2830 \text{ Hz}, G = 2.8 \times 10^{-9}$.

$f_e = 40 \text{ kHz}, f_o = 2829 \text{ Hz}, G = 3.77 \times 10^{-9}$.  
$f_e = 50 \text{ kHz}, f_o = 2790 \text{ Hz}, G = 4.5 \times 10^{-9}$.

$f_e = 60 \text{ kHz}, f_o = 2796 \text{ Hz}, G = 4.83 \times 10^{-9}$.  
$f_e = 70 \text{ kHz}, f_o = 2790 \text{ Hz}, G = 5.3 \times 10^{-9}$.
$f_e = 80 \text{ kHz}, f_o = 2786 \text{ Hz}, G = 5.13 \times 10^{-9}$.

$fe = 90 \text{ kHz}, fo = 2783 \text{ Hz}, G = 4.62 \times 10^{-9}$.

$fe = 100 \text{ kHz}, fo = 2739 \text{ Hz}, G = 4.4 \times 10^{-9}$.

$fe = 110 \text{ kHz}, fo = 2758 \text{ Hz}, G = 4.2 \times 10^{-9}$.

$fe = 120 \text{ kHz}, fo = 2762 \text{ Hz}, G = 3.9 \times 10^{-9}$.

Stator Current: 4A Peak
\[ f_e = 20 \text{ kHz}, \; f_o = 2785 \text{ Hz}, \; G = 4.8 \times 10^{-9}. \]

\[ f_e = 30 \text{ kHz}, \; f_o = 2773 \text{ Hz}, \; G = 5.6 \times 10^{-9}. \]

\[ f_e = 40 \text{ kHz}, \; f_o = 2753 \text{ Hz}, \; G = 7.1 \times 10^{-9}. \]

\[ f_e = 50 \text{ kHz}, \; f_o = 2751 \text{ Hz}, \; G = 8.1 \times 10^{-9}. \]

\[ f_e = 60 \text{ kHz}, \; f_o = 2758 \text{ Hz}, \; G = 9.39 \times 10^{-9}. \]

\[ f_e = 70 \text{ kHz}, \; f_o = 2764 \text{ Hz}, \; G = 9.82 \times 10^{-9}. \]
$f_e = 80 \text{ kHz}, f_o = 2745 \text{ Hz}, G = 1.08 \times 10^{-8}$. $f_e = 90 \text{ kHz}, f_o = 2753 \text{ Hz}, G = 1.17 \times 10^{-8}$.

$fe = 100 \text{ kHz}, f_o = 2752 \text{ Hz}, G = 1.11 \times 10^{-8}$. $f_e = 110 \text{ kHz}, f_o = 2690 \text{ Hz}, G = 1.13 \times 10^{-8}$.

$fe = 120 \text{ kHz}, f_o = 2692 \text{ Hz}, G = 1.07 \times 10^{-8}$.

Stator Current: 5A Peak
$f_e = 20 \text{ kHz}, f_o = 2761 \text{ Hz}, G = 6.4 \times 10^{-9}$. $f_e = 30 \text{ kHz}, f_o = 2749 \text{ Hz}, G = 8.7 \times 10^{-9}$.

$fe = 40 \text{ kHz}, f_o = 2741 \text{ Hz}, G = 1.05 \times 10^{-8}$. $f_e = 50 \text{ kHz}, f_o = 2726 \text{ Hz}, G = 1.14 \times 10^{-8}$.

$fe = 60 \text{ kHz}, f_o = 2714 \text{ Hz}, G = 1.25 \times 10^{-8}$. $fe = 70 \text{ kHz}, f_o = 2670 \text{ Hz}, G = 1.5 \times 10^{-8}$.
$f_e = 90 \text{ kHz}, f_o = 2665 \text{ Hz}, G = 1.64 \times 10^{-8}$. $f_e = 100 \text{ kHz}, f_o = 2651 \text{ Hz}, G = 1.75 \times 10^{-8}$.

$fe = 110 \text{ kHz}, f_o = 2642 \text{ Hz}, G = 1.61 \times 10^{-8}$. $f_e = 120 \text{ kHz}, f_o = 2639 \text{ Hz}, G = 1.69 \times 10^{-8}$.

**Stator Current: 6A Peak**

$fe = 20 \text{ kHz}, f_o = 2753 \text{ Hz}, G = 9.80 \times 10^{-9}$. $f_e = 30 \text{ kHz}, f_o = 2734 \text{ Hz}, G = 1.35 \times 10^{-8}$. 
$f_e = 40 \text{ kHz}, f_o = 2680 \text{ Hz}, G = 1.68 \times 10^{-8}$. 

$fe = 50 \text{ kHz}, f_o = 2638 \text{ Hz}, G = 2.0 \times 10^{-8}$.

$fe = 60 \text{ kHz}, f_o = 2635 \text{ Hz}, G = 2.03 \times 10^{-8}$. 

$fe = 70 \text{ kHz}, f_o = 2629 \text{ Hz}, G = 2.3 \times 10^{-8}$.

$fe = 80 \text{ kHz}, f_o = 2618 \text{ Hz}, G = 2.35 \times 10^{-8}$. 

$fe = 90 \text{ kHz}, f_o = 2607 \text{ Hz}, G = 2.48 \times 10^{-8}$.
$f_e = 100 \text{ kHz}, f_o = 2590 \text{ Hz}, G = 2.28 \times 10^{-8}.$ $f_e = 110 \text{ kHz}, f_o = 2584 \text{ Hz}, G = 2.24 \times 10^{-8}.$

$f_e = 120 \text{ kHz}, f_o = 2558 \text{ Hz}, G = 2.15 \times 10^{-8}.$
Appendix E

Source Codes

In this Appendix, we provide a listing of the relevant computer software code. The first software is the designer program based on Model I, described in Chapter 2. The second code belongs to the simulation based on Model II, explained in detail in Chapter 4. The last software is the implementation of Model III, which is also discussed in detail in Chapter 4.
THIS IS THE C-CODE FOR THE DESIGNER PROGRAM BASED ON MODEL I
written by
HUR KOSER

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <math.h>
#include "F:/MSDEV/include/nrutil.h"
#include "F:/MSDEV/include/nrutil.c"
#include "G:\Matlab\extern\include\mex.h"
#include "F:/Matlab\extern\include\matrix.h"
#include <cdmath.h>
#define pi 3.14159265358979
#define float double
#define points 99

double g, b, a, d, mu rs, mu ssg, mu cond, mu air;
double sigma cond, sigma rs, sigma air, sigma ssg;
double minReff, maxReff, IA, wireh, wirew, Rpoints, poles;
double mu = 4 * pi * 1e-7;
double Twg, Ts, Tp, Lp, Lo, Ls, Lsmid;
double R3, Rb, Ra;
double *freq, *rmatrixpc, *lossdata;
double *B RtlIm, *BcornerIm, *B attopIm;
double lossat80, lossat8;
double row rmatrixpc, col rmatrixpc, Tbi, mu bi;
double row lossdata, col lossdata;
double B Rtlmagsum, B Rt2magsum, B cornermagsum, B attopmagsum;
dComplex Rtl, Rs, Rp, Rc, Rbi;
void mydesign(double *X pt, double *Y pt, double *B Rt1mag, double *B Rt2mag, double
*Bcornermag, double *B attopmag, double *mycorelosssta, double *mycorelossrot, double
*B_bimag) {
    long counter = 0, count = 0, myfreqcount;
double mReff, widthinc = 1.0, mm = 0.0, mmcount;
dComplex mygamma air, mygamma_cond, mygamma rs;
dComplex AA, BB, CC, DD, EE;
dComplex mymu air, mymu cond, mymu rs, f1, f2, f3, ftop, f4;
double k, L, w = 10000.0, factor, areasum = 0.0;
double Lnew;
dComplex temp1, temp2, temp3, temp4, imag j, one;
dComplex mynewtemp1, mynewtemp2, mynewtemp3, mynewtemp4, mynewYYY1, mynewYYY2;
dComplex Mc, Mr1, Mr2, Ml1, Ml2, mpTr1, mpTr2, mpTr3, mpTr4, mpTr5, mpTr21;
dComplex mpTl3, mpTl4, MTr, MTL, YYYY1, YYYY2, Hst, H6x, H6y, H10x, H11y;
dComplex RR1, RR2, RR3, negi1;
dComplex B5, B6x, phi_1phase, phi_s, phi_2, phi_3phase, B_Rtl, B_Rt2, Bcorner, B_attop, B bi;
int ii;
char mybuffer1[200], mybuffer2[15];
double B Rtlmagnew = 0.0, B Rt2magnew = 0.0, B cornermagnew = 0.0, B attopmagnew = 0.0,
B_bimagnew = 0.0;
    Double B_Rtlmagd = 0.0, B_Rt2magd = 0.0, B_cornermagd = 0.0, B_attopmagd = 0.0,
B_bimagd = 0.0;
double corelossrot, corelosssta;
double cornerres(double, double, double, double, double, double);
double getRentry(long, long);
double getlossentry(long, long);
double determiner(double, double, double, double, double);
double determineloss(double, double);
// define some initial parameters to use in the calculations
imag j.Re=0;
imag j.Im=1;
one.Re=1;
one.Im=0;
mymu air.Re=muair;
mymu air.Im=0;
mymu cond.Re=mucond;
mymu cond.Im=0;
mymu rs.Re=umrs;
mymu rs.Im=0;
widthinc=(maxReff-minReff)/Rpoints;

// initialize B fields inside the stator and the rotor
B Rtlmagd=0.0;
B Rt2magd=0.0;
Bcornermagd=0.0;
B attnmagd=0.0;
B bimagd=0.0;
corelossrot=0.0;
corelosssta=0.0;

/*
B RtlRe=dvector(1,100);
B RtlIm=dvector(1,100);
B Rt2Re=dvector(1,100);
B Rt2Im=dvector(1,100);
BcornerRe=dvector(1,100);
BcornerIm=dvector(1,100);
B attnRe=dvector(1,100);
B attnIm=dvector(1,100);
*/
B RtlRe=(double *) calloc(100, sizeof(double));
B RtlIm=(double *) calloc(100, sizeof(double));
B Rt2Re=(double *) calloc(100, sizeof(double));
B Rt2Im=(double *) calloc(100, sizeof(double));
BcornerRe=(double *) calloc(100, sizeof(double));
BcornerIm=(double *) calloc(100, sizeof(double));
B attnRe=(double *) calloc(100, sizeof(double));
B attnIm=(double *) calloc(100, sizeof(double));
B bIm=(double *) calloc(100, sizeof(double));
B bIm=(double *) calloc(100, sizeof(double));

//printf("1) widthinc = %f\n", widthinc);

negI=cd_mulRe(cd_exp(cd_mulRe(imagj,-pi/4)), -sqrt(2)*IA*wirew*wireh);
areausum=0.0;
//printf("2) widthinc = %f\n", widthinc);

for (ii=0;ii<points;ii++){
    //printf("ii = %d\n", ii);
    *(Xpt+ii)=0.0;
    *(Ypt+ii)=0.0;
}

for (ii=0;ii<points;ii++){
    *(BcornerRe+ii)=0.0;
    *(BcornerIm+ii)=0.0;
    // *(B attnRe+ii)=0.0;
    // *(B attnIm+ii)=0.0;
}

for (mReff=minReff+widthinc; mReff<=maxReff; mReff=mReff+widthinc){
motordesign.c

gcvt( mReff, 10, mybuffer2 );
strcpy(mybuffer1,"set(findobj(DesignHandle,'Tag','StaticText26'),'String',");
strcat(mybuffer1,mybuffer2);
strcat(mybuffer1,"; drawnow;");  
mexEvalString(mybuffer1);

factor=mReff/((maxReff+minReff)/2.0);
L=4.0*pi*mReff/100;
k=2.0*pi/L;
Ls=1.0*Lsmid;

// the physical dimensions in the stator
Twg=(1.0*(wirew))e100e-6; //assuming 50 um insulator on each side
Tp=(L/4)-Twg;
Lo=1.0*((Twg-Ls)/2.0);
Lpnew=(2.0*wireh)+110e-6; // assuming 30, 30, and 50 microns of insulator

// CAUTION!!! CAUTION!!! CAUTION!!!
// the magnetic reluctances corresponding to the physical dimensions
// THIS IS WHERE I WILL BE INSERTING THE NEW, EDDY-CURRENT DEFINITIONS
// BEWARE!!! BEWARE!!! BEWARE!!!

/*
 * Rs=Lo/mu_air/Tp/widthinc; // this stays the same, since air is not a conductor
 * Rlt=Lo/mu_ssg/Tp/widthinc;  
 * Rp=(Ts+((Lpnew/2.0)/mu_ssg/Tp/widthinc;
 * Rc=determineR(Tp,Ts,Lo2,Lpnew/2.0)/mu_ssg/widthinc; //cornerres(Tp/2.0, Ts,
 * if (abs((mu bi-mu ssg)/mu ssg)) > 0.05){
 * Rbi=((Lpnew/2.0)/Tp/mu ssg/widthinc)+(determineR(Tp,Tbi,Twg/2.0,Lpnew/2.0
 * /100.0)/mu bi/widthinc);  
 * }else{
 Rbi=determineR(Tp,Tbi,Twg/2.0,Lp/2)/mu_ssg/widthinc;
 }

RRs.Re=2*Rt1+Rs+2*(Rc+Rbi);
RRs.Im=0;
RR1.Re=Rc+Rbi;
RR1.Im=Rbi;
RR2.Re=-Rc-Rbi;
RR2.Im=Rbi;
RR3.Re=Rbi;
RR3.Im=Rbi;
*/

B Rtlmagnew=0.0;
B Rt2magnew=0.0;
Bcornermagnew=0.0;
B attopmagnew=0.0;
B bimagnew=0.0;
for (ii=0;ii<points;ii++){
 *(B RtlRe+ii)=0.0;
 *(B RtlIm+ii)=0.0;
 *(B Rt2Re+ii)=0.0;
 *(B Rt2Im+ii)=0.0;
 // *(BcornerRe+ii)=0.0;
 // *(BcornerIm+ii)=0.0;
 *(B attopRe+ii)=0.0;
 *(B attopIm+ii)=0.0;
 *(B biRe+ii)=0.0;
 *(B biIm+ii)=0.0;
}

areasum=areasum*(widthinc*Tp);
for (mmcount=1.0; mmcount<20.1; mmcount=mmcount+1.0) {
    mm=(2*mmcount-1.0)*pow(-1.0,mmcount+1.0);
    f1=cd mulRe(cd mulRe(imag j,-widthinc/k/mm), cd_sub(cd_exp(cd_mulRe( imag j, -mm*pi/4)), cd_exp(cd_mulRe( imag j, -mm*pi/2)), cd_mulRe( imag j, mm*Tp*pi/L))), cd_exp(cd_mulRe( imag j, -mm*pi/4)));
    f2=cd mulRe(cd mulRe(imag j,-widthinc/k/mm), cd_sub(cd_exp(cd_mulRe( imag j, -mm*pi/4)), cd exp(cd_mulRe( imag j, -mm*pi/2), cd_mulRe( imag j, mm*Tp*pi/L))));
    f3=cd mulRe(cd mulRe(imag j,-widthinc/k/mm), cd sub(cd_exp(cd_mulRe( imag j, -mm*pi/4)), cd exp(cd_mulRe( imag j, mm*pi/4))));
    f4=cd mulRe(cd mulRe(imag j,-widthinc/k/mm), cd_sub(cd_exp(cd_mulRe( imag j, 0.0)), cd exp(cd_mulRe( imag j, mm*Tp*pi/L))));
    M0=cd mulRe(cd exp(cd mulRe(imag j,pi*mm/4)),4*sin(k*mm*Ls/2)/pi/mm);
    Mr=cd mul cd div(cd mulRe(cd exp(cd mulRe(imag j,pi*mm/4)),2/pi/mm), imag j), cd sub(cd_exp(cd_mulRe( imag j,k*mm*(Ls+Lo)/2) ), cd_exp(cd_mulRe( imag j,k*mm*Twg/2)));
    M01=cd mul( cd div(cd mulRe(cd exp(cd_mulRe(imag j,pi*mm/4)),2/pi/mm), imag j), cd sub(cd_exp(cd_mulRe( imag j,-k*mm*Ls/2) ), cd_exp(cd-mulRe( imag j,-k*mm*(Ls+Lo)/2)));
    M02=cd mul( cd div(cd mulRe(cd exp(cd_mulRe(imag j,pi*mm/4)),2/pi/mm), imag j), cd sub(cd_exp(cd_mulRe( imag j,-k*mm*Twg/2) ), cd_exp(cd-mulRe( imag j,-k*mm*(Ls+Lo)/2)));
    tc=cd mulRe(cd exp(cd mulRe(imag j,3*pi*mm/2-(3*pi/2))),4);
    temp2=cd mulRe(cd mulRe(imag j,(pi/2)-(pi/2)),4);
motordesign.c

2)), -pi*mm), imag j);
    temp3 = cd_mul(cd_mulRe(cd_exp(cd_mulRe(imag_j, ((7*pi+2*Twg*k)*mm/4)-(3*pi/2))), 2*Twg*k*mm), imag j);
    temp4 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((7*pi+2*Twg*k)*mm/4)-(3*pi/2))), -4);

    mpTr4 = cd_divRe(cd_add(cd_add(temp1, temp2), cd_add(temp3, temp4)), k*k*m*mm);

    templ = cd_mulRe(cd_exp(cd_mulRe(imag_j, 0)), -4);
    temp2 = cd_mul(cd_mulRe(cd_exp(cd_mulRe(imag_j, (-pi+2*Twg*k)*mm/4)-pi*mm)), imag j);
    temp3 = cd_mulRe(cd_mulRe(cd_exp(cd_mulRe(imag_j, (-pi+2*Twg*k)*mm/4)-pi*mm)), 2*Twg*k*mm);
    temp4 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-5*pi+2*Twg*k)*mm/4)-(pi/2))), 4);
    mpTl1 = cd_divRe(cd_add(cd_add(temp1, temp2), cd_add(temp3, temp4)), k*k*m*mm);

    temp1 = cd_mulRe(cd_exp(cd_mulRe(imag_j, (pi*mm/2)-(pi/2))), -4);
    temp2 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-3*pi+2*Twg*k)*mm/4)-(pi/2))), -pi*mm), imag j);
    temp3 = cd_mulRe(cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-3*pi+2*Twg*k)*mm/4)-(pi/2))), 2*Twg*k*mm), imag j);
    temp4 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-5*pi+2*Twg*k)*mm/4)-(pi/2))), 4);
    mpTl2 = cd_divRe(cd_add(cd_add(temp1, temp2), cd_add(temp3, temp4)), k*k*m*mm);

    temp1 = cd_mulRe(cd_exp(cd_mulRe(imag_j, (3*pi*mm/2)-(3*pi/2))), -4);
    temp2 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-7*pi+2*Twg*k)*mm/4)-(pi/2))), -pi*mm), imag j);
    temp3 = cd_mulRe(cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-7*pi+2*Twg*k)*mm/4)-(pi/2))), 2*Twg*k*mm), imag j);
    temp4 = cd_mulRe(cd_exp(cd_mulRe(imag_j, ((-7*pi+2*Twg*k)*mm/4)-(3*pi/2))), 4);
    mpTl3 = cd_divRe(cd_add(cd_add(temp1, temp2), cd_add(temp3, temp4)), k*k*m*mm);

    temp1 = cd_mulRe(cd_add(cd_add(mpTr1, mpTr2), cd_add(mpTr3, mpTr4)), k/(((Twg/2)-(pi/4/k)) * 8*pi));
    MT1 = cd_mulRe(cd_add(cd_add(mpTl1, mpTl2), cd_add(mpTl3, mpTl4)), -k/(((Twg/2)-(pi/4/k)) * 8*pi));
*/

mynewtemp1 = cd_divRe(cd_add(cd_add(cd_add(Mr1, Mr2), cd_add(Ml1, Ml2)), cd_add(MTr, MT1)), mu_ssg*Ts*widthinc);
mynewtemp2 = cd_divRe(Mc, mu_air*Ts*widthinc);
mynewtemp4 = cd_add(temp1, temp2);
mynewtemp3 = cd_divRe(cd_add(cd_add(cd_mul(f1, RR1), cd_mul(f2, RR2)), cd_mul(f3, RR3)), RRs);
mynewtemp3 = cd_mul(temp1, temp4);
mynewYYY2=cd_div(temp4,RRs);
mynewtemp1=cd_divRe(cd_mul(f2,cd_add(Mr2,MTr)),mu_ssg*Ts*widthinc);
mynewtemp2=cd_divRe(cd_mul(f1,cd_add(Ml2,Mtl)),mu_ssg*Ts*widthinc);
mynewYYY1=cd_add(cd_add(temp1, temp2), temp3);
*/

//now that we added these inside the frequency loop, take them out of here

for (myfreqcount=0;myfreqcount<=points;myfreqcount++){
    // CAUTION!!! CAUTION!!! CAUTION!!!
    // the magnetic reluctances corresponding to the physical dimensions
    // THIS IS WHERE I WILL BE INSERTING THE NEW, EDDY-CURRENT DEFINITION
    S

    // BEWARE!!! BEWARE!!! BEWARE!!!
    skindepth=sqrt(2/(*freq+myfreqcount))/mu_ssg/sigma_ssg);
    //here, I assumed freq is omega (which it is) and the
    //stator has the same conductivity as the rctor (which it should)
    P rti=cd_mulRe(cd_add(one,imag j),Ts/skindepth);
    rti eddyfactor=cd_divRe(cd_div(cd_divRe(cd_mul(P rti,cd_sinh(P_rti)),cd_mul(cd_sinh(cd_divRe(P rti,2.0)),4.0)),4.0));
    Rt1=cd_mulRe(rti eddyfactor,Lo/mu_ssg/Ts*widthinc/2.0); // new value
    of the reluctance
    // notice that the reluctance is now complex!!!!

    a conductor
    Rs.Re=Ls/mu_air/Ts/widthinc; // this stays the same, since air is not
    // Rs should also be converted into a complex number

    // here is Rp being converted into the eddy current
    // formalism. Notice that Rp is also complex now
    P rp=cd_mulRe(cd_add(one,imag j),Tp/skindepth);
    rp eddyfactor=cd_divRe(cd_div(cd_divRe(cd_mul(P rp,cd_sinh(P_rp)),cd_mul(cd_sinh(cd_divRe(P rp,2.0)),4.0)),4.0));
    Rp=cd_mulRe(rp eddyfactor,(Ts+(Lpnew/2.0))/mu_ssg/Tp/widthinc);
    // !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    // Repeat the same procedure with Rc now:
    // the question is: WHAT Thickness T SHALL WE USE FOR
    // THE CORNER PIECE???

    // My guess is to use an approximate effective length (for the
    // corner-turn, use sqrt(Tbi^2+(Twg/2)^2)
    // here is the guess for the effective width:
    realtemp1=((Lpnew/2)+(Lo/2)+(sqrt(2)*min(Tp,Ts)))/determineR(Tp,Ts,Lo/2,Lpnew/2.0);
    P rc=cd_mulRe(cd_add(one,imag j),realtemp1/skindepth);
    rc eddyfactor=cd_divRe(cd_div(cd_mul(P rc,cd_sinh(P rc)),cd_mul(cd_sinh(cd_divRe(P rc,2.0)),4.0)),4.0);
    Rc=cd_mulRe(rc eddyfactor,determineR(Tp,Ts,Lo/2,Lpnew/2.0)/mu_ssg/widthinc)/cornerres(Tp/2.0, Ts, Lo/2, Lp, widthinc, mu_ssg); // cornerres(Tp/2.0, Ts, Lo/2, Lp, widthinc, mu_ssg));
    //
    if ((abs(mu_bi-mu_ssg)/mu_ssg) > 0.05){
        P rbi c=cd_mulRe(cd_add(one,imag j),((Tp+Tbi)/2.0)/skindepth);
        rbi c eddyfactor=cd_divRe(cd_div(cd_mul(P rbi c,cd_sinh(P_rbi c)),cd_mul(cd_sinh(cd_divRe(P rbi_c,2.0)),4.0)),4.0);
        P rbi nc=cd_mulRe(cd_add(one,imag j),Tp/skindepth);
        rbi nc eddyfactor=cd_divRe(cd_div(cd_mul(P rbi nc,cd_sinh(P rbi nc)),cd_mul(cd_sinh(cd_divRe(P rbi nc,2.0)),4.0)),4.0);
    }
    Rbi=cd_add(cmulcd_mulRe(rbi nc eddyfactor,(Lpnew/2.0)/Tp/mu_ssg/widt

hinc)), cd mulRe(rbi_c_eddyfactor, (determineR(Tp, Tbi, Twg/2.0, Lpnew/2.0/100.0)/mu_b/widthinc));
}

else{
    // THE QUESTION IS: WHAT IS THE EFFECTIVE THICKNESS TO CONSIDER HERE
    // TO USE IN THE EXPRESSION FOR P????
    // My guess is to use an approximate effective length (for the
    // corner-turn, use sqrt(Tbi^2+(Twg/2)^2)
    // here is the guess for the effective width:
    realtemp=((Lpnew/2)+(T WG/2)+(sqrt(2)*min(Tbi,Tp)))/determineR(Tp, Tbi, Twg/2, Lpnew/2);
    P rbi_c=cd mulRe(cd add(one, imag j), realtemp/skindepth);
    rbi_c eddyfactor=cd divRe(cd divRe(cd mul(P rbi_c, cd sinh(P rbi_c)), cd sinh(cd divRe(P rbi_c, 2.0))), 4.0);
    realtemp=determineR(Tp, Tbi, Twg/2, Lpnew/2)/mu_ssg/widthinc;
    Rbi=cd_mulRe(rbi_c_eddyfactor, realtemp);
}

// old RRs changed from this:
//RRs.Re=2*Rtl+Rs+2*(Rc+Rbi);
//RRs.Im=0;
// into this:
RRs=cd_add(cd_add(cd_mulRe(Rtl, 2.0), Rs), cd_mulRe(cd_add(Rc, Rbi), 2));

// RR1 is changed from this:
//RR1.Re=Rc+Rbi;
//RR1.Im=Rbi;
// into this:
RR1=cd_add(Rc, cd_mul(cd_add(one, imag_j), Rbi));

// RR2 changes from this:
//RR2.Re=-Rc-Rbi;
//RR2.Im=Rbi;
// into this:
RR2=cd_add(cd_neg(Rc), cd_mul(cd_add(imag_j, cd_neg(one)), Rbi));

// finally, RR3 changes from this:
//RR3.Re=Rbi;
//RR3.Im=Rbi;
// into this:
RR3=cd_mul(cd_add(one, imag_j), Rbi);

mynewtempl=cd divRe(cd add(cd_add(cd_add(Mrl, Mr2), cd_add(Mll, M12)), MTl)), mu_ssg*Ts*widthinc);
mynewtemp2=cd divRe(Mc, mu_air*Ts*widthinc);
mynewtemp4=cd add(mynewtempl, mynewtemp2);
mynewtemp1=cd divRe(cd_add(cd_mul(f1, RR1), cd_mul(f2, RR2)), cd_mul(f3, RR3)), RRs);

mynewtemp3=cd_mul(cd_add(MTr, MTl)), mu_ssg*Ts*widthinc);
mynewtemp2=cd divRe(Mc, mu_air*Ts*widthinc);
mynewtemp4=cd add(mynewtempl, mynewtemp2);
mynewtemp1=cd divRe(cd_add(cd_mul(f1, RR1), cd_mul(f2, RR2)), cd_mul(f3, RR3)), RRs);

YYY2=cd div(mynewtemp4, RRs);
mynewtemp1=cd divRe(cd_mul(f2, cd_add(Mr2, MTr)), mu_ssg*Ts*widthinc);
mynewtemp2=cd divRe(cd_mul(f1, cd_add(Ml2, MTl)), -mu_ssg*Ts*widthinc);

YYY1=cd_add(cd_add(mynewtemp1, mynewtemp2), mynewtemp3);

mygamma air.Re=k*k*mm*mm;
mygamma cond.Re=k*k*mm*mm;
mygamma rs.Re=k*k*mm*mm;
mygamma air.Im=mu_air*sigma_air*(freq+myfreqcount);
mygamma cond.Im=mu_cond*sigma_cond*(freq+myfreqcount);
mygamma rs.Im=mu_rs*sigma_rs*(freq+myfreqcount);

mygamma air=cd sqrt(mygamma air);
mygamma cond=cd sqrt(mygamma cond);
mygammars = cd_sqrt(mygammars);

AA = cd_div(one, cd_cosh(cd_mulRe(mygamma_air,g)));

templ = cd_div(mygammars, cd_mul(mygammars, cd_sinh(cd_mulRe(mygammars, b)));

temp2 = cd_add( cd_div(cd_div(mymu_air, cd_tanh(cd_mulRe(mygamma_air,g)) , mygamma_air)),
            cd_div(cd_div(mymu_rs, cd_tanh(cd_mulRe(mygamma_rs,b))) , mygamma_rs));

temp3 = cd_sub(temp2, cd_mul(AA, cd_div(cd_div(mymu_air, cd_sinh(cd_mulRe(mygamma_air,g))) , mygamma_air)));

BB = cd_div(temp1, temp3);

temp3 = cd_div(mymu_cond, cd_mul(mygamma_cond, cd_sinh(cd_mulRe(mygamma_cond, mygamma_rs, b)) , mygamma_rs));

BB = cd_div(temp1, temp3);

temp2 = cd_add( cd_div(cd_div(mymu_cond, cd_tanh(cd_mulRe(mygamma_cond,a))) , mygamma_cond),
            cd_div(cd_div(mymu_rs, cd_tanh(cd_mulRe(mygamma_air,d))) , mygamma_air));

temp3 = cd_sub(temp2, cd_mul(BB, cd_div(cd_div(mymu_rs, cd_sinh(cd_mulRe(mygamma_air,g))) , mygamma_air)));

CC = cd_div(temp1, temp3);

temp3 = cd_div(mymu_air, cd_mul(mygamma_air, cd_sinh(cd_mulRe(mygamma_air, d)) , mygamma_air));

CC = cd_div(temp1, temp3);

temp2 = cd_add( cd_div(cd_div(mymu_air, cd_sinh(cd_mulRe(mygamma_air,d))) , mygamma_air),
            cd_div(cd_div(mymu_air, cd_tanh(cd_mulRe(mygamma_air,d))) , mygamma_air));

EE = cd_div(temp1, temp3);

temp1 = cd_div(one, YYY1);

temp2 = cd_div(cd_div(cd_mulRe(imag_j, k*mm*mu_air), cd_tanh(cd_mulRe(mygamma_air,d))) ,
            mygamma_air);

temp3 = cd_div(cd_div(cd_mul(myqamma_air, cd_sinh(cd_mulRe(mygamma_air,d)))) ,
            mygamma_air);

DD = cd_add_add(temp1, temp2, temp3);

Hst = cd_div(cd_mul(YYY2, negI), cd_mul(YYY1, DD));

H6x = cd_mul(EE, Hst);

temp1 = cd_div(cd_mulRe(imag_j, k*mm, mygamma_air);

temp2 = cd_add( cd_div(cd_neg(Hst), cd_sinh(cd_mulRe(mygamma_air,d))) ,
            cd_div(H6x, cd_tanh(cd_mulRe(mygamma_air,d)))));

H6y = cd_mul(temp1, temp2);

H10x = cd_mul(cd_mul(BB, CC, H6x);

temp1 = cd_div(cd_mulRe(imag_j, k*mm, mygamma_air);

temp2 = cd_add( cd_div(cd_mul(AA, H10x), cd_sinh(cd_mulRe(mygamma_air,g))) ,
            cd_div(cd_neg(H10x), cd_tanh(cd_mulRe(mygamma_air,g)))));

H11y = cd_mul(temp1, temp2);

H11y = cd_mul(temp1, temp2);

// this time, we multiply by L, but remember to multiply by 2L and
// the local motor radius to get the total torque...

temp1 = cd_mulRe(cd_mul(conj(H6y), H6x), 0.5*mu_air*L*widthinc*mReff);

temp2 = cd_mulRe(cd_mul(conj(H11y), H11x), 0.5*mu_air*L*widthinc*mReff);

temp3 = cd_mulRe(cd_sub(cd_mul(conj(H6y), H6y), cd_mul(conj(H6x), H6x), 0.25*mu_air*L*widthinc);

temp4 = cd_mulRe(cd_sub(cd_mul(conj(H11y), H11y), cd_mul(conj(H11x), H11x), 0.25*mu_air*L*widthinc);

*(X pt+myfreqcount) = *(X pt+myfreqcount) - temp1.Re - temp2.Re;

*(Y pt+myfreqcount) = *(Y pt+myfreqcount) - temp3.Re - temp4.Re;
\[
d \text{div}(H6x, cd \sinh(cd \text{mulRe}(mygamma \text{air}, d))) ;
B5 = \text{cd mul}(\text{temp1}, \text{temp2})
H8x = \text{cd mul}(\text{CC}, H6x);
\phi 1 = \text{cd mul}(\text{M}, \text{cd div}(\text{cd add}(\text{cd mul}(B5, \text{cd add}(\text{cd add}(\text{cd mul}(RR1, f1)
\text{, cd mul}(RR2, f2)))), \text{neg1}, RR3));
\phi 1\text{phase} = \text{cd mul}(\text{cd mul}(f1, B5), \text{cd_neg}(\text{imag}_j));
\phi 2 = \text{cd mul}(f2, B5);
\phi 3\text{phase} = \text{cd mul}(\text{cd mul}(f4, B5), \text{cd_neg}(\text{imag}_j)); \quad \text{// check this!!!}
\text{temp1} = \text{cd divRe}(\phi 1, \text{widthinc*Ts});
*(B RtlRe+\text{myfreqcount}) = *(B RtlRe+\text{myfreqcount}) + \text{temp1}.Re;
*(B RtlIm+\text{myfreqcount}) = *(B RtlIm+\text{myfreqcount}) + \text{temp1}.Im;
\text{temp1} = \text{cd divRe}(\text{cd add}(\phi 1, \phi 2), \text{widthinc*Ts});
*(B Rtl2Re+\text{myfreqcount}) = *(B Rtl2Re+\text{myfreqcount}) + \text{temp1}.Re;
*(B Rtl2Im+\text{myfreqcount}) = *(B Rtl2Im+\text{myfreqcount}) + \text{temp1}.Im;
\text{temp1} = \text{cd add}(\phi 12, \phi 2, \phi 3\text{phase});
\phi 3 = \text{cd div}(\text{cd mul}(\text{imag}_j, \text{K*mm*mu rs}), \text{mygamma rs});
\text{temp2} = \text{cd add}(\text{cd div}(\text{cd neg}(H8x), \text{cd_tanh}(\text{cd_mulRe}(\text{mygamma rs}, b))));
\text{cd div}(H10x, \text{cd sinh}(\text{cd_mulRe}(\text{mygamma rs}, b)));}
\text{temp3} = \text{cd mul}(\text{temp1}, \text{temp2});
\text{temp1} = \text{cd divRe}(\text{cd mul}(\text{ftop}, \text{temp3}), 2*\text{widthinc*b});
\text{temp1} = \text{cd divRe}(\text{cd add}(\phi 1\text{phase}, \phi 2), \phi 3\text{phase});
*(B attopRe+\text{myfreqcount}) = *(B attopRe+\text{myfreqcount}) + \text{temp1}.Re;//\text{temp1.Re}
*(B attopIm+\text{myfreqcount}) = *(B attopIm+\text{myfreqcount}) + \text{temp1}.Im;//\text{temp1.Im}
\text{temp1} = \text{cd divRe}(\text{cd div}(\text{cd add}(\text{cd add}(\text{cd add}(\text{cd add}(\phi 1, \phi 2), \phi 3\text{phase}), \text{cd_mul}(\phi s, \text{imag}_j)), \text{cd_neg}(\text{cd_add}(\text{one}, \text{imag}_j)))), \text{widthinc*T bi});
*(B biRe+\text{myfreqcount}) = *(B biRe+\text{myfreqcount}) + \text{temp1}.Re;
*(B biIm+\text{myfreqcount}) = *(B biIm+\text{myfreqcount}) + \text{temp1}.Im;
\}
\}
B Rtl1magsum=0.0;
B Rtl2magsum=0.0;
B bimagsum=0.0;
B attopmagsum=0.0;
for (\text{myfreqcount=0;} \text{myfreqcount<points;} \text{myfreqcount++}){
B Rtl1.Re=*(B RtlRe+\text{myfreqcount});
B Rtl1.Im=*(B RtlIm+\text{myfreqcount});
B Rtl2.Re=*(B Rtl2Re+\text{myfreqcount});
B Rtl2.Im=*(B Rtl2Im+\text{myfreqcount});
//Bcorner.Re = *(BcornerRe+\text{myfreqcount});
//Bcorner.Im = *(BcornerIm+\text{myfreqcount});
B attop.Re = *(B attopRe+\text{myfreqcount});
B attop.Im = *(B attopIm+\text{myfreqcount});
B bi.Re = *(B biRe+\text{myfreqcount});
B bi.Im = *(B biIm+\text{myfreqcount});
B Rtl1magnew=sqrt((B Rtl1.Re*B Rtl1.Re)+(B Rtl1.Im*B Rtl1.Im));
B Rtl2magnew=sqrt((B Rtl2.Re*B Rtl2.Re)+(B Rtl2.Im*B Rtl2.Im));
//Bcornermagnew=sqrt((B Rtl1.Re*Bcorner.Re)+(B Rtl1.Im*Bcorner.Im)+(B Rtl2.Re*Bcorner.Re)+(B Rtl2.Im*Bcorner.Im))/\text{are}
asum;
B attopmagnew=sqrt((B attop.Re*B attop.Re)+(B attop.Im*B attop.Im));
B bimagnew=sqrt((B bi.Re*B bi.Re)+(B bi.Im*B bi.Im));
B Rtl1magsum=B Rtl1magsum+B Rtl1magnew;
B Rtl2magsum=B Rtl2magsum+B Rtl2magnew;
B bimagsum=B bimagsum+B bimagnew;
B attopmagsum=B attopmagsum+B attopmagnew;
if (B Rt1magnew > B Rt1magd) {
    B_Rt1magd = B_Rt1magnew;
}
if (B Rt2magnew > B Rt2magd) {
    B_Rt2magd = B_Rt2magnew;
}
    // if (Bcornermagnew > Bcornermagd) {
    //     Bcornermagd = Bcornermagnew;
    // }
    if (B attopmagnew > B attopmagd) {
        B_attopmagd = B_attopmagnew;
    }
    if (B bimagnew > B bimagd) {
        B_bimagd = B_bimagnew;
    }
}

// calculating core losses... Underlying assumption -- worst saturation occurs
// at the frequency of the torque peak (i.e., stator excitation frequency = 80 kHz
// and the effective rotor electrical frequency = 8 kHz at 10% slip

B_Rt1.Re = *(B_Rt1Re + 67);
B_Rt1.Im = *(B_Rt1Im + 67);
B_Rt2.Re = *(B_Rt2Re + 67);
B_Rt2.Im = *(B_Rt2Im + 67);
//Bcorner.Re = *(BcornerRe + 67);
//Bcorner.Im = *(BcornerIm + 67);
B_attop.Re = *(B_attopRe + 67);
B_attop.Im = *(B_attopIm + 67);

B_Rt1magnew = sqrt((B_Rt1.Re*B_Rt1.Re) + (B_Rt1.Im*B_Rt1.Im));
B_Rt2magnew = sqrt((B_Rt2.Re*B_Rt2.Re) + (B_Rt2.Im*B_Rt2.Im));
//Bcornermagnew = sqrt((Bcorner.Re*Bcorner.Re) + (Bcorner.Im*Bcorner.Im)) / areasum;
B_attopmagnew = sqrt((B_attop.Re*B_attop.Re) + (B_attop.Im*B_attop.Im));

corelosssta = corelosssta + 2*poles*2*(B_Rt1magnew + B_Rt2magnew)* (Lo/2)*Ts*widthinc*lossat80;
corelossrot = corelossrot + B_attopmagnew*b*2*pi*mReff*widthinc*lossat8;
// note: should I divide the Battopmagnew by two to estimate the average B at the rotor?...

B_corneermagsum = 0.0;
for (myfreqcount = 0; myfreqcount <= points; myfreqcount++) {
    Bcorner.Re = *(BcornerRe + myfreqcount);
    Bcorner.Im = *(BcornerIm + myfreqcount);
    Bcorneermagnew = sqrt((Bcorner.Re*Bcorner.Re) + (Bcorner.Im*Bcorner.Im)) / areasum;
    if (Bcorneermagnew > Bcorneermagd) {
        Bcorneermagd = Bcorneermagnew;
    }
}

// addition to the core loss ...

Bcorner.Re = *(BcornerRe + 67);
Bcorner.Im = *(BcornerIm + 67);
Bcorneermagnew = sqrt((Bcorner.Re*Bcorner.Re) + (Bcorner.Im*Bcorner.Im)) / areasum;

corelosssta = corelosssta + 2*poles*2*Bcorneermagnew*areasum*(Ts+Lp)*lossat80 + Bcorneermagnew*pi*(maxReff+minReff)*areasum*lossat80;

//
*(B_Rtlmag)=B_Rtlmagd;
*(B_Rt2mag)=B_Rt2magd;
*(Bcornermag)=Bcornermagd;
*(B_attopmag)=B_attopmagd;
*(B_bimag)=B_bimagd;

*(mycorelosssta)=corelosssta;
*(mycorelossrot)=corelossrot;

// for the time being, use the worst-case scenario values for B's... once you find
// out how to determine the frequency of the maximum, calculate the core losses f or each
// dR and add.
*(B_Rtlavemag)=Ts*(Lo/2.0)*[(maxReff-minReff)*2*poles*2;
*(B_Rt2avemag)=Ts*(Lo/2.0)*[(maxReff-minReff)*2*poles*2;
*(B_corneravemag)=(Ts+Lp)*area*sum*2*poles;
*(B_biaavemag)=((maxReff*maxReff)-(minReff*minReff))*PI*Tbi;
*(B_attopavemag)=((maxReff*maxReff)-(minReff*minReff))*PI*b;

*/

free dvector(B_RtlRe,1,100);
free dvector(B_RtlIm,1,100);
free dvector(B_Rt2Re,1,100);
free dvector(B_Rt2Im,1,100);
free dvector(BcornerRe,1,100);
free dvector(BcornerIm,1,100);
free dvector(B_attopRe,1,100);
free dvector(B_attopIm,1,100);*/
free(B_RtlRe);
free(B_RtlIm);
free(B_Rt2Re);
free(B_Rt2Im);
free(BcornerRe);
free(BcornerIm);
free(B_attopRe);
free(B_attopIm);
free(B_bRe);
free(B_bIm);

// the gateway function

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
    *mycorelossrot, *B_bimag;

g=mxGetScalar(prhs[0]);
b=mxGetScalar(prhs[1]);
a=mxGetScalar(prhs[2]);
d=mxGetScalar(prhs[3]);
mu_rs=mxGetScalar(prhs[4]);
mu_ssg=mxGetScalar(prhs[5]);
mu_cond=mxGetScalar(prhs[6]);
mu air=mxGetScalar(prhs[7]);
sigma cond=mxGetScalar(prhs[8]);
sigma rs=mxGetScalar(prhs[9]);
sigma air=mxGetScalar(prhs[10]);
minReff=mxGetScalar(prhs[11]);
maxReff=mxGetScalar(prhs[12]);
IA=mxGetScalar(prhs[13]);
wire=mxGetScalar(prhs[14]);
wirew=mxGetScalar(prhs[15]);

freq=mxGetPr(prhs[16]);

Ts=mxGetScalar(prhs[17]);
Lsmid=mxGetScalar(prhs[18]);
Lp=mxGetScalar(prhs[19]);
Rpoints=mxGetScalar(prhs[20]);
poles=mxGetScalar(prhs[21]);

lossat8O=mxGetScalar(prhs[22]);
lossat8=mxGetScalar(prhs[23]);

rmatrixpc=mxGetPr(prhs[24]);
row row rmatrixpc=mxGetScalar(prhs[25]);
col col rmatrixpc=mxGetScalar(prhs[26]);
Tbi=mxGetScalar(prhs[27]);
mu bi=mxGetScalar(prhs[28]);

lossdata=mxGetPr(prhs[29]);
row row lossdata=mxGetScalar(prhs[30]);
col col lossdata=mxGetScalar(prhs[31]);
sigmassg=mxGetScalar(prhs[32]);

plhs[0]=mxCreateDoubleMatrix(1, 100, mxREAL);
plhs[1]=mxCreateDoubleMatrix(1, 100, mxREAL);
plhs[2]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[3]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[4]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[5]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[6]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[7]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[8]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[9]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[10]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[11]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[12]=mxCreateDoubleMatrix(1, 1, mxREAL);
plhs[13]=mxCreateDoubleMatrix(1, 1, mxREAL);

X pt=mxGetPr(plhs[0]);
Y pt=mxGetPr(plhs[1]);
B Rtlmag=mxGetPr(plhs[2]);
B Rt2mag=mxGetPr(plhs[3]);
B Bcornermag=mxGetPr(plhs[4]);
B attopmag=mxGetPr(plhs[5]);
mycorelosst=mxGetPr(plhs[6]);
mycorelossrot=mxGetPr(plhs[7]);
B biavemag=mxGetPr(plhs[8]);
B Btlavemag=mxGetPr(plhs[9]);
B B2avemag=mxGetPr(plhs[10]);
B Bcorneravemag=mxGetPr(plhs[11]);
B biavemag=mxGetPr(plhs[12]);
B attopavemag=mxGetPr(plhs[13]);

// call the computational routine
mydesign(X pt, Y pt, B_Rt1mag, B_Rt2mag, Bcornermag, B_atttopmag, mycorelosssta, m ycorelossrot, B_bimag);

} double getRentry(long kk, long ll)
{
    return(*(rmatrixpc+(((ll-1)*((long) row_rmatrixpc))+kk-1) ));
}

double determineR(double A, double B, double C, double D)
{
double AA, BB, CC, DD, temp, AAmul, AAhigh, AAlow, BBhigh, BBlow;
double CChigh, CClow, DDeff, DDeff;
double R1, R2, R3, R4, R5, R6, R;
long AIndexedlow, AIndexedhigh;
long RRindexedClowDDlow, RRindexedClowDDhigh, RRindexedCChighDDlow, RRindexedCChighDDhigh;

AA = A/B;
if (AA < 0.1){
    temp=A;
    A=B;
    B=temp;
    temp=C;
    C=D;
    D=temp;
    AA=A/B;
}
if (AA > 18.0){
    AA = 18.0; // after 18.0, you don't gain anything
}
BB = 1.0;
CC = C/B;
DD = D/B;
}
if (AA > 4.999999999999){
    AAmul=1.0;
    AIndexedlow=50+((long) floor((AA*AAmul)))-5;
    AIndexedhigh=50+((long) ceil((AA*AAmul)))-5;
}
else{
    AAmul=10.0;
    AIndexedlow=((long) floor((AA*AAmul)));
    AIndexedhigh=((long) ceil((AA*AAmul)));
}
AAlow=floor(AA/AAmul)/AAmul;
AAhigh=(AAlow*1e-10)+(ceil(AA/AAmul)/AAmul);
CClow=floor(CC/(AA/10.0))*AA/10.0;
CChigh=(CClow*1e-10)+(ceil(CC/(AA/10.0))*AA/10.0);
DDlow=floor(DD*10.0)/10.0;
DDhigh=(DDlow*1e-10)+(ceil(DD*10.0)/10.0);

if ((CC<=AA) & (DD<2*BB)){
    RRindexedClowDDlow=((long) 60.0*(max(flo0or(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(floor(DD/0.1),1.0)-1.0);
    RRindexedCChighDDlow=((long) 60.0*(ceil(CC/(AA/10.0))-1.0)+2.0+3.0*max(floor(DD/0.1),1.0)-1.0);
    RRindexedClowDDhigh=((long) 60.0*(max(flo0or(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(DD/0.1)-1.0);
    RRindexedCChighDDhigh=((long) 60.0*(ceil(CC/(AA/10.0))-1.0)+2.0+3.0*ceil(DD/0.1)-1.0);
}
xAA=1.0-(AA-AAlow)/(AAhigh-AAlow);
xCC=1.0-(CC-CClow)/(CChigh-CClow);
xDD=1.0-(DD-DDlow)/(DDhigh-DDlow);
R1 = xAA*(getRentry(AAindexlow, RRindexCClowDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDlow));
R2 = xAA*(getRentry(AAindexlow, RRindexCHighDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCHighDDlow));
R3 = xAA*(getRentry(AAindexlow, RRindexCClowDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDhigh));
R4 = xAA*(getRentry(AAindexlow, RRindexCHighDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCHighDDhigh));
R5 = xCC*R1 + (1-xCC)*R2;
R6 = xCC*R3 + (1-xCC)*R4;
R = xDD*R5 + (1-DD)*R6;
}

} else if ((CC>AA) && (DD<=2*BB)){

CCeff = AA;
CCLow = floor(CCeff/(AA/10.0))*AA/10.0;
CChigh = ceil(CCeff/(AA/10.0))*AA/10.0;

RRindexCClowDDlow = ((long)60.0*max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(floor(DD/0.1),1.0)-1.0;
RRindexCClowDDhigh = ((long)60.0*max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CCeff/(AA/10.0))-1.0;
RRindexCHighDDlow = ((long)60.0*max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CCeff/(AA/10.0))-1.0;
RRindexCHighDDhigh = ((long)60.0*max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CCeff/(AA/10.0))-1.0;

R1 = xAA*(getRentry(AAindexlow, RRindexCClowDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDlow));
R2 = xAA*(getRentry(AAindexlow, RRindexCHighDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCHighDDlow));
R3 = xAA*(getRentry(AAindexlow, RRindexCClowDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDhigh));
R4 = xAA*(getRentry(AAindexlow, RRindexCHighDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCHighDDhigh));
R5 = xCC*R1 + (1-xCC)*R2;
R6 = xCC*R3 + (1-xCC)*R4;
R = xDD*R5 + (1-DD)*R6 + ((CC-CCeff)/AA);
}

} else if ((CC<=AA) && (DD>2*BB)){

DDeff = 2*BB;
DCLow = floor(DDeff*10.0)/10.0;
DCHigh = ceiling(DDeff*10.0)/10.0;

RRindexCClowDDlow = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(floor(DDeff/0.1),1.0)-1.0;
RRindexCClowDDhigh = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(ceil(CC/(AA/10.0))-1.0);
RRindexCCHighDDlow = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CC/(AA/10.0))-1.0;
RRindexCCHighDDhigh = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CC/(AA/10.0))-1.0;

R1 = xAA*(getRentry(AAindexlow, RRindexCClowDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDlow));
R2 = xAA*(getRentry(AAindexlow, RRindexCCHighDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCCHighDDlow));
R3 = xAA*(getRentry(AAindexlow, RRindexCClowDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDhigh));
R4 = xAA*(getRentry(AAindexlow, RRindexCCHighDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCCHighDDhigh));
R5 = xCC*R1 + (1-xCC)*R2;
R6 = xCC*R3 + (1-xCC)*R4;
R = xDD*R5 + (1-DD)*R6 + ((DD-DDeff)/BB);
}

} else if ((CC<=AA) && (DD>2*BB)){

DDeff = 2*BB;
DCLow = floor(DDeff*10.0)/10.0;
DCHigh = ceiling(DDeff*10.0)/10.0;

RRindexCClowDDlow = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(floor(DDeff/0.1),1.0)-1.0;
RRindexCClowDDhigh = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*max(ceil(CC/(AA/10.0))-1.0);
RRindexCCHighDDlow = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CC/(AA/10.0))-1.0;
RRindexCCHighDDhigh = ((long)60.0*max(floor(CC/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(CC/(AA/10.0))-1.0;

R1 = xAA*(getRentry(AAindexlow, RRindexCClowDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDlow));
R2 = xAA*(getRentry(AAindexlow, RRindexCCHighDDlow)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCCHighDDlow));
R3 = xAA*(getRentry(AAindexlow, RRindexCClowDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCClowDDhigh));
R4 = xAA*(getRentry(AAindexlow, RRindexCCHighDDhigh)) + (1-xAA)*(getRentry(AAindexhigh, RRindexCCHighDDhigh));
R5 = xCC*R1 + (1-xCC)*R2;
R6 = xCC*R3 + (1-xCC)*R4;
R = xDD*R5 + (1-DD)*R6 + ((DD-DDeff)/BB);
}
else if ((CC>AA) && (DD>2*BB)) {
    CCeff=AA;
    DDeff=2*BB;
    CClow=floor(CCeff/(AA/10.0))*AA/10.0;
    CChigh=ceil(CClow*1.0)+floor(CCeff/(AA/10.0))*AA/10.0;
    DDllow=floor(DDeff*10.0)/10.0;
    DDbighigh=ceil(DDbighigh*10.0)/10.0;
    RRindexCClowDDlow=((long)60.0*(max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*
    max(floor(DDbighigh*10.0)-1.0);  
    RRindexCChighDDlow=((long)60.0*(ceil(CCeff/(AA/10.0))-1.0)+2.0+3.0*
    *ceil(DDbighigh*10.0)-1.0);  
    RRindexCClowDDhigh=((long)60.0*(max(floor(CCeff/(AA/10.0)),1.0)-1.0)+2.0+3.0*ceil(DDbighigh*10.0)-1.0);  
    R=1-xAA*(getRentry(AAindexlow, RRindexCClowDDlow))+(1-xAA)*(getRentry(AAindexhigh, RRindexCChighDDlow));  
    R2=xAA*(getRentry(AAindexlow, RRindexCChighDDlow))+(1-xAA)*(getRentry(AAindexhigh, RRindexCChighDDlow));  
    R3=xAA*(getRentry(AAindexlow, RRindexCClowDDhigh))+(1-xAA)*(getRentry(AAindexhigh, RRindexCChighDDhigh));  
    R4=xAA*(getRentry(AAindexlow, RRindexCChighDDhigh))+(1-xAA)*(getRentry(AAindexhigh, RRindexCChighDDhigh));  
    R5=xCC*R1+(1-xCC)*R2;  
    R6=xCC*R3+(1-xCC)*R4;  
    R=xDD*R5+(1-xDD)*R6+(max(CCeff,AA)+((DD-DBeff)/BB));  
}

return(R);  
}

double cornerres(double A, double B, double C, double D, double mywidthinc, double my_ssg) {

double alpha, gamma, integral2, sumR, forus;

double Ato4, Ato3, Ato2, Bto4, Bto3, Bto2, Cto4, Cto3, Cto2, Dto4, Dto3, Dto2;
    Ato4= A*A*A*A;  
    Ato3= A*A*A;  
    Ato2= A*A;  
    Bto4= B*B*B*B;  
    Bto3= B*B*B;  
    Bto2= B*B;  
    Cto4= C*C*C*C;  
    Cto3= C*C*C;  
    Cto2= C*C;  
    Dto4= D*D*D*D;  
    Dto3= D*D*D;  
    Dto2= D*D;  
    alpha=35*C* ( (1715*D*Ato4 - 5425*D*Bto4 - 1300*Bto3*Dto2 - 1638*Bto3*Ato2 + 882* 
    Ato4*B + 700*B*Dto2*Ato2 + 1890*D*Ato2*Bto2) / (47628*Bto3*Ato3 + 404250*D*Ato2*C*Bto2 
    + 112000*Bto2*Ato2 + 84525*D*Bto4*C + 37800*Bto3*Dto2*A + 30000*B*Dto2*Ato2 + 
    112000*B*Dto2*C*Ato2 + 28000*Bto3*Dto2*C + 37800*B*Ato3*Cto2 + 141120*B*Ato4*C + 35280 
    *Bto3*Ato2*C + 35280*D*Ato3*Bto2 + 28000*D*Ato3*Cto2 + 84525*D*Ato4*C + 141120*D*Bto4*A) 
    );  
    gamma=-35* ( 5425*Ato4*C+1300*Ato3*Cto2+1638*Ato2*Bto2-1890*Ato2*C*Bto2-700*Bto2*A 
    *Cto2-882*Bto4*A-1715*Bto4*C) * ( D ) / (47628*Bto3*Ato3+404250*D*Ato2*C*Bto2+112000*D 
    *Bto2*A*Bto2 + 84525*D*Bto4*C + 37800*Bto3*Dto2*A + 30000*B*Dto2*Ato2 + 112000*B 
    Dto2*C*Ato2 + 28000*Bto3*Dto2*C + 37800*B*Ato3*Cto2 + 141120*B*Ato4*C + 35280*Bto3 
    *Ato2*C + 35280*D*Ato3*Bto2 + 28000*D*Ato3*Cto2 + 84525*D*Ato4*C + 141120*D*Bto4*A ));  
    integral2=(7*Ato2*gamma*alpha +31*Bto2+16*Bto2*gamma*alpha-7*Bto2*gamma+4*Bto2*alpha 
    *alpha+13*Ato2*gamma +4*Ato2*gamma*alpha+31*Ato2+16*Ato2*alpha*alpha+7*Ato2*alpha
motordesign.c

-7*Bto2*gamma*alpha +13*Bto2*alpha)/A/B/90;
sumR=(integral2 + (C/B) + (D/A));
forus=integral2/mymu_ssg/mywidthinc;
return(forus);
}

double getlossentry(long kk, long ll)
{
    return(*(lossdata+(((ll-1)*((long) row_lossdata))+kk-1) ));
}

double determineloss(double lossfreq, double lossB)
{
    long lossfreqcount, lossBcountup, lossBcountdown;
    double losscontribup, losscontribdown, xx, yy;
    lossfreqcount=0;
    lossBcountup=0;
    lossBcountdown=0;
    while true{
        lossfreqcount=lossfreqcount+1;
        if (lossfreqcount > row_lossdata){
            return(-999999999999.9);
        }
        if (lossfreq-1e-10 < getlossentry(lossfreqcount,1)){
            break;
        }
    }
    while true{
        if ((lossfreqcount+lossBcountdown+1) < row lossdata){
            if (((lossB-1e-10 < getlossentry(lossfreqcount+lossBcountdown,2)) || (getlos
essentry(lossfreqcount+lossBcountdown+1,1)-1e-10 > getlossentry(lossfreqcount+lossBco
toundown,1)))){
                break;
            }
            lossBcountdown=lossBcountdown+1;
        }
        losscontribdown=getlossentry(lossfreqcount+lossBcountdown,2);
        if ((double)lossfreqcount > 1.01){
            while true{
                lossBcountup=lossBcountup-1;
                if (((lossB-1e-10 > getlossentry(lossfreqcount+lossBcountup,2)) || (getlos
essentry(lossfreqcount+lossBcountup+1,1)+1e-10 < getlossentry(lossfreqcount+lossBcountup-
1,1)))){
                    lossBcountup=lossBcountup+1;
                    losscontribup=getlossentry(lossfreqcount+lossBcountup,2);
                    break;
                }
            }
        }
    else{
        lossBcountup=0;
        losscontribup=losscontribdown;
    }
    xx=lossfreq-getlossentry(lossfreqcount+lossBcountup,1)+1e-10;
    yy=getlossentry(lossfreqcount+lossBcountdown,1)+1e-10-lossfreq;
    if ((losscontribup-losscontribdown/losscontribdown) < 0.00001){

16
return(losscontribdown);
}  
else{
    return((losscontribup*yy+losscontribdown*xx)/(xx+yy));
}  
}
THIS IS THE C-CODE FOR THE SIMULATION SOFTWARE BASED ON MODEL II

written by

HUR KOSER

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <math.h>
//#include "F:/MSDEV/include/nrutil.h"
//#include "F:/MSDEV/include/nrutil.c"
#include "G:\Matlab\extern\include\mex.h"
#include "F:\Matlab\extern\include\matrix.h"
#include <cdmath.h>
#define Pi 3.14159265358979

define the global variables

double Jext amp, k, dt, omegadt, ie x, ie y;
double **x ez, **ez, **hx, **hy, **bx, **by, **ca matrix, **cb x matrix;
double **cb y matrix, **cc_matrix, **db_x_matrix, **db_y_matrix, **mu_matrix;

the gateway function

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{

long n p, n t, n a, m a, m p, m g, m c, m bi, m r;
double stampmatrix(long, long, double, double);
double seriescond(double, double);
double setboundary(long);

// check for proper number of input and output arguments
if (nrhs != 19){
    mexErrMsgTxt("19 input arguments required.");
}  
if (nlhs > 1){
    mexErrMsgTxt("Too many output arguments.");
}

// Check data type of input argument
if (!mxIsDouble(prhs[0])){
    mexErrMsgTxt("Input argument must be of type double.");
}

// m is the total number of points in the discretization

Jext amp = mxGetScalar(prhs[0]);
k = mxGetScalar(prhs[1]);
dt = mxGetScalar(prhs[2]);
omegadt = mxGetScalar(prhs[3]);
ie x = mxGetScalar(prhs[4]);
ie y = mxGetScalar(prhs[5]);
ex = mxGetPr(prhs[6]);
ez = mxGetPr(prhs[7]);
hx = mxGetPr(prhs[8]);
hy = mxGetPr(prhs[9]);
bx = mxGetPr(prhs[10]);
by = mxGetPr(prhs[11]);
ca matrix = mxGetPr(prhs[12]);
cb x matrix = mxGetPr(prhs[13]);
cb y matrix = mxGetPr(prhs[14]);
cc_matrix = mxGetPr(prhs[15]);
\[
\text{db } x \text{ matrix } = \text{mxGetPr(prhs[16])};
\text{db } y \text{ matrix } = \text{mxGetPr(prhs[17])};
\text{mu \_matrix } = \text{mxGetPr(prhs[18])};
\]

\[
\text{mu \_vector } = \text{mxGetPr(prhs[18])};
\]

\[
\text{skindepth } = \sqrt{2/w/\text{sigma elec ss}/\text{mu ss}};
\text{skindepth\_wafer } = \sqrt{2/w/\text{sigma\_ss}/\text{mu\_ss\_wafer}};
\]

\[
\begin{align*}
\text{n p } & = 10; \\
\text{n t } & = 20; \\
\text{n a } & = 20; \\
\text{m a } & = 10; \\
\text{m p } & = 20; \\
\text{m g } & = 20; \\
\text{m c } & = 20;
\end{align*}
\]

\[
\begin{align*}
\text{m bi } & = 21; \\
\text{m r } & = 21; \\
& \text{// then simplify calculation by not computing anything beyond 7 skindepths -- 0.1\% effect}
\end{align*}
\]

\[
\begin{align*}
& \text{if (Trotor } > 7*\text{skindepth})\
& \quad \text{Trotor } = 7*\text{skindepth}; \\
& \quad \text{m \_r } = 21;
\end{align*}
\]

\[
\begin{align*}
& \text{// same thing for the back iron}
& \text{if (Tbi } > 7*\text{skindepth \_wafer})\
& \quad \text{Tbi } = 7*\text{skindepth \_wafer}; \\
& \quad \text{m \_bi } = 21;
\end{align*}
\]

\[
\begin{align*}
\text{n p } & = \text{max}(10, (\text{long})\text{ceil}(4*(Tp/\text{skindepth \_wafer}))); \\
\text{m a } & = \text{max}(10, (\text{long})\text{ceil}(4*(Tt/\text{skindepth}))); \\
\text{/*n p } & = 4; \\
\text{n t } & = 4; \\
\text{n a } & = 4; \\
\text{m a } & = 4; \\
\text{m p } & = 4; \\
\text{m g } & = 4; \\
\text{m c } & = 4;
\end{align*}
\]

\[
\begin{align*}
\text{m bi } & = 21; \\
\text{m r } & = 21; \\
& /*
\end{align*}
\]

\[
\text{totalpoints } = ((2*\text{n p}) + (2*\text{n t}) + \text{n a})*(\text{m r } + \text{m c } + \text{m g } + \text{m a } + \text{m p } + \text{m bi})
\]

\[
\text{totalentries } = 5*\text{totalpoints } + 20*((2*\text{n p}) + (2*\text{n t}) + \text{n a } + \text{m r } + \text{m c } + \text{m g } + \text{m a } + \text{m p } + \text{m bi})
\]

\[
\begin{align*}
\text{plhs[0]} & = \text{mxCreateDoubleMatrix(totalentries, 1, mxREAL)}; \\
\text{plhs[1]} & = \text{mxCreateDoubleMatrix(totalentries, 1, mxREAL)}; \\
\text{plhs[2]} & = \text{mxCreateDoubleMatrix(totalentries, 1, mxREAL)}; \\
\text{plhs[3]} & = \text{mxCreateDoubleMatrix(totalentries, 1, mxREAL)};
\end{align*}
\]
newmodelingsparse.c

plhs[4]=mxCreateDoubleMatrix(totalpoints, 1, mxREAL);
plhs[5]=mxCreateDoubleMatrix(30, 1, mxREAL);

rowvector=mxGetPr(plhs[0]);
colvector=mxGetPr(plhs[1]);
realvector=mxGetPr(plhs[2]);
imagvector=mxGetPr(plhs[3]);
RHS=mxGetPr(plhs[4]);
endpointinfo=mxGetPr(plhs[5]);

////////////////////////////////////////////////////
// NOW, we can start stamping our conductivity matrix.
////////////////////////////////////////////////////

//%% corner pole section (sections 1 and 5)
hx p = Tp/(n p);
hy p = Tc/(m a);
hz p = Width;
G p x = hy p*hz p/hx p;
G p y = hx p*hz p/hy p;

//%% tooth section (sections 2 and 4)
hx t = Lt/(n t);
hy t = Tc/(m a);
hz t = Width;
G t x = hy t*hz t/hx t;
G t y = hx t*hz t/hy t;

//%% tooth air gap section (section 3)
hx a = Ls/(n a);
hy a = Tc/(m a);
hz a = Width;
G a x = hy a*hz a/hx a;
G a y = hx a*hz a/hy a;

//%% pole section (sections 6 and 7)
hx pp = Tp/(n p);
hy pp = Lp/(m p);
hz pp = Width;
G pp x = hy pp*hz pp/hx pp;
G pp y = hx pp*hz pp/hy pp;

//%% back iron corner section (sections 8 and 10)
hx bic = Tp/(n p);
hy bic = Tbi/(m bi);
hz bic = Width;
G bic x = hy bic*hz bic/hx bic;
G bic y = hx bic*hz bic/hy bic;

//%% back iron main section (section 9a)
hx bi a = Lt/(n t);
hy bi a = Tbi/(m bi);
hz bi a = Width;
G bi a x = hy bi a*hz bi a/hx bi a;
G bi a y = hx bi a*hz bi a/hy bi a;

//%% back iron main section (section 9b)
hx bi b = Ls/(n a);
hy bi b = Tbi/(m bi);
hz bi b = Width;
G bi b x = hy bi b*hz bi b/hx bi b;
G bi b y = hx bi b*hz bi b/hy bi b;
// back iron main section (section 9c)
hx bi c = Lt/(n t);
hy bi c = Tbi/(m bi);
hz bi c = Width;
G bi c x = hy bi c*hz bi c/hx bi c;
G bi c y = hx bi c*hz bi c/hy bi c;

// wire gap section (section 11a)
hx w a = Lt/(n t);
hy w a = Lp/(m p);
hz w a = Width;
G w a x = hy w a*hz w a/hx w a;
G w a y = hx w a*hz w a/hy w a;

// wire gap section (section 11b)
hx w b = Ls/(n a);
hy w b = Lp/(m p);
hz w b = Width;
G w b x = hy w b*hz w b/hx w b;
G w b y = hx w b*hz w b/hy w b;

// wire gap section (section 11c)
hx w c = Lt/(n t);
hy w c = Lp/(m p);
hz w c = Width;
G w c x = hy w c*hz w c/hx w c;
G w c y = hx w c*hz w c/hy w c;

// air gap section (section 12)
hx 12 = Tp/(n p);
hy 12 = Gap/(m g);
hz 12 = Width;
G 12 x = hy 12*hz 12/hx 12;
G 12 y = hx 12*hz 12/hy 12;

// air gap section (section 13)
hx 13 = Lt/(n t);
hy 13 = Gap/(m g);
hz 13 = Width;
G 13 x = hy 13*hz 13/hx 13;
G 13 y = hx 13*hz 13/hy 13;

// air gap section (section 14)
hx 14 = Ls/(n a);
hy 14 = Gap/(m g);
hz 14 = Width;
G 14 x = hy 14*hz 14/hx 14;
G 14 y = hx 14*hz 14/hy 14;

// air gap section (section 15)
hx 15 = Lt/(n t);
hy 15 = Gap/(m g);
hz 15 = Width;
G 15 x = hy 15*hz 15/hx 15;
G 15 y = hx 15*hz 15/hy 15;

// air gap section (section 16)
hx_16 = Tp/(n p);
hy 16 = Gap/(m_g);
hz 16 = Width;
G 16 x = hy 16*hz 16/hx 16;
G 16 y = hx 16*hz 16/hy 16;

//%%% air gap section (section 17)
hx 17 = Tp/(n p);
hy 17 = Tcond/(m_c);
hz 17 = Width;
G 17 x = hy 17*hz 17/hx 17;
G 17 y = hx 17*hz 17/hy 17;

//%%% air gap section (section 18)
hx 18 = Lt/(n t);
hy 18 = Tcond/(m_c);
hz 18 = Width;
G 18 x = hy 18*hz 18/hx 18;
G 18 y = hx 18*hz 18/hy 18;

//%%% air gap section (section 19)
hx 19 = Ls/(n a);
hy 19 = Tcond/(m_c);
hz 19 = Width;
G 19 x = hy 19*hz 19/hx 19;
G 19 y = hx 19*hz 19/hy 19;

//%%% air gap section (section 20)
hx 20 = Lt/(n t);
hy 20 = Tcond/(m_c);
hz 20 = Width;
G 20 x = hy 20*hz 20/hx 20;
G 20 y = hx 20*hz 20/hy 20;

//%%% air gap section (section 21)
hx 21 = Tp/(n p);
hy 21 = Tcond/(m_c);
hz 21 = Width;
G 21 x = hy 21*hz 21/hx 21;
G 21 y = hx 21*hz 21/hy 21;

//%%% air gap section (section 22)
hx 22 = Tp/(n p);
hy 22 = Trotor/(m_r);
hz 22 = Width;
G 22 x = hy 22*hz 22/hx 22;
G 22 y = hx 22*hz 22/hy 22;

//%%% air gap section (section 23)
hx 23 = Lt/(n t);
hy 23 = Trotor/(m_r);
hz 23 = Width;
G 23 x = hy 23*hz 23/hx 23;
G 23 y = hx 23*hz 23/hy 23;

//%%% air gap section (section 24)
hx 24 = Ls/(n a);
hy 24 = Trotor/(m_r);
hz 24 = Width;
G_24_x = hy_24*hz_24/hx_24;
G 24 y = hx 24*hz 24/hy 24;

//%%%% air gap section (section 25)
hx 25 = Lt/(n t);
hy 25 = Trotor/(m_r);
hz 25 = Width;
G 25 x = hy 25*hz 25/hx 25;
G 25 y = hx 25*hz 25/hy 25;

//%%%% air gap section (section 26)
hx 26 = Tp/(n p);
hy 26 = Trotor/(m_r);
hz 26 = Width;
G 26 x = hy 26*hz 26/hx 26;
G 26 y = hx 26*hz 26/hy 26;

//%%% stamping section 1 only (no interconnections with the other sections yet)

totalsecpoints1 = n p*m_a;
pointsuml=pointsum1+1;
pointsum2=pointsum2+totalsecpoints1;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long) (ii-pointsum1+1) % (n p) != 0)){
        stampvalue = seriescond(2*G p x/mu vector[ii],2*G_p_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G p y/mu vector[ii],2*G_p_y/mu_vector[ii+n p]);
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii+pointsum1-n p)>=0){
        stampvalue = seriescond(2*G p y/mu vector[ii],2*G_p_y/mu_vector[ii-n p]);
        stampmatrix(ii, ii-n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_elec_ss*hx_p*hy_p*hz_p);
}
endofsec1=pointsum2;

//%%%% end of stamping section 1

//%%% stamping section 2 only (no interconnections with the other sections yet)

totalsecpoints2 = n t*m a;
pointsuml=pointsum1+totalsecpoints1;
pointsum2=pointsum2+totalsecpoints2;
for (ii=pointsum1; (ii<=pointsum2); ii++){
cond countx=0.0;
cond county=0.0;
if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n t) != 0))
   {
      stampvalue = seriescond(2*G_t_x/mu_vector[ii],2*G_t_x/mu_vector[ii+1]);
      stampmatrix(ii, ii+1, -stampvalue, 0.0);
      cond_countx = cond_countx + stampvalue;
   }
if ((ii+n t)<pointsum2+1)
   {
      stampvalue = seriescond(2*G t y/mu_vector[ii],2*G_t_y/mu_vector[ii+n t]);
      stampmatrix(ii, ii+n t, -stampvalue, 0.0);
      cond_county = cond_county + stampvalue;
   }
if (((ii)>pointsum1) && ((long)(ii-pointsum1+1) % (n t) !=1 ))
   {
      stampvalue = seriescond(2*G_t_x/mu_vector[ii],2*G_t_x/mu_vector[ii-1]);
      stampmatrix(ii, ii-1, -stampvalue, 0.0);
      cond_countx = cond_countx + stampvalue;
   }
if ((ii-pointsum1-n t)>=0)
   {
      stampvalue = seriescond(2*G t y/mu_vector[ii],2*G_t_y/mu_vector[ii-n t]);
      stampmatrix(ii, ii-n t, -stampvalue, 0.0);
      cond_county = cond_county + stampvalue;
   }
   stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_elec_ss*hx_t*hy_t*hz_t);
endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;

endofsec2=pointsum2;
newmodelingsparse.c

//////////////////////////////////////////////////////////////////////////////////
//**** stamping section 4 only (no interconnections with the other sections yet)
//////////////////////////////////////////////////////////////////////////////////

totalsecpoints4 = n t*m a;
pointsuml=pointsuml+totalsecpoints3;
pointsum2=pointsum2+totalsecpoints4;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n t) != 0)){
        stampvalue = seriescond(2*G_t_x/mu_vector[ii],2*G_t_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if (ii+n t)<pointsum2+1){
        stampvalue = seriescond(2*G_t_y/mu_vector[ii],2*G_t_y/mu_vector[ii+n_t]);
        stampmatrix(ii, ii+n t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n t) !=1 )){
        stampvalue = seriescond(2*G_t_x/mu_vector[ii],2*G_t_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n t)>=0){
        stampvalue = seriescond(2*G_t_y/mu_vector[ii],2*G_t_y/mu_vector[ii-n_p]);
        stampmatrix(ii, ii-n_p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
}
endofsec4=pointsum2;

//////////////////////////////////////////////////////////////////////////////////
//**** end of stamping section 4
//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////////
//**** stamping section 5 only (no interconnections with the other sections yet)
//////////////////////////////////////////////////////////////////////////////////

totalsecpoints5 = n p*m a;
pointsuml=pointsuml+totalsecpoints4;
pointsum2=pointsum2+totalsecpoints5;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n p) != 0)){
        stampvalue = seriescond(2*G_p_x/mu_vector[ii],2*G_p_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if (ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_p_y/mu_vector[ii],2*G_p_y/mu_vector[ii+n_p]);
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n p) !=1 )){
        stampvalue = seriescond(2*G_p_x/mu_vector[ii],2*G_p_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n p)>=0){
        stampvalue = seriescond(2*G_p_y/mu_vector[ii],2*G_p_y/mu_vector[ii-n_p]);
        stampmatrix(ii, ii-n_p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
}
newmodelingsparse.c

```c

} stampmatrix(ii,ii,cond_countx+cond_county, w*sigma_elec_ss*hx_pp*hy_pp*hz_pp);
}
endofsec5=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//***** end of stamping section 5
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//***** stamping section 6 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints6 = n*p*m;  
pointsum1=pointsum1+totalsecpoints5;  
pointsum2=pointsum2+totalsecpoints6;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && (long)(ii-pointsum1+l) % (n p) != 0 ){
        stampvalue = seriescond(2*G_ppx/mu_vector[ii],2*G_ppx/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_ppy/mu_vector[ii],2*G_ppy/mu_vector[ii+n p]
    )
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if ((ii-pointsum1-n p)>=0){
        stampvalue = seriescond(2*G_ppy/mu_vector[ii],2*G_ppy/mu_vector[ii-n p]
    )
        stampmatrix(ii, ii-n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county, w*sigma_elec_ss*hx_pp*hy_pp*hz_pp);
}
endofsec6=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//***** end of stamping section 6
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//***** stamping section 7 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints7 = n*p*m;  
pointsum1=pointsum1+totalsecpoints6;  
pointsum2=pointsum2+totalsecpoints7;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && (long)(ii-pointsum1+l) % (n p) != 0 ){
        stampvalue = seriescond(2*G_ppx/mu_vector[ii],2*G_ppx/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_ppy/mu_vector[ii],2*G_ppy/mu_vector[ii+n p]
    )
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
```
cond_county = cond_county + stampvalue;
}
if (((ii)>pointsuml) && ((long)(ii-pointsuml+1) % (n p) !=1 )){
    stampvalue = seriescond(2*G_pp_x/mu_vector[ii],2*G_pp_x/mu_vector[ii-1]);
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii-pointsuml-n p)>=0){
    stampvalue = seriescond(2*G_pp_y/mu_vector[ii],2*G_pp_y/mu_vector[ii-n_p]);
}
    stampmatrix(ii, ii-n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_elec_ss*hx_pp*hy_pp*hz_pp);
}
endofsec7=pointsum2;

//*******************************************************
//**** end of stamping section 7
//*******************************************************

//*******************************************************
//**** stamping section 8 only (no interconnections with the other sections yet)
//*******************************************************
totalsecpoints8 = n*p*m bi;
pointsum1=pointsum1+totalsecpoints7;
pointsum2=pointsum2+totalsecpoints8;
for (ii=pointsum1; (ii<pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsuml+1) % (n p) != 0)){
        stampvalue = seriescond(2*G_bic_x/mu_vector[ii],2*G_bic_x/mu_vector[ii+1] );
    }
    stampmatrix(ii, ii+1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii+n p)<pointsum2+1){
    stampvalue = seriescond(2*G_bic_y/mu_vector[ii],2*G_bic_y/mu_vector[ii+n_p]);
    stampmatrix(ii, ii+n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
if (((ii)>pointsuml) && ((long)(ii-pointsuml+1) % (n p) !=1 )){
    stampvalue = seriescond(2*G_bic_x/mu_vector[ii],2*G_bic_x/mu_vector[ii-1]);
}
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii-pointsuml-n p)>=0){
    stampvalue = seriescond(2*G_bic_y/mu_vector[ii],2*G_bic_y/mu_vector[ii-n_p]);
    stampmatrix(ii, ii-n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
}
endofsec8=pointsum2;

//*******************************************************
//**** end of stamping section 8
//*******************************************************

//******************** stamping section 9a only (no interconnections with the other sections yet)***********************
//********************

```
totalsecpoints9a = n*t*m bi;
pointsuml=pointsuml+totalsecpoints8;
pointsum2=pointsum2+totalsecpoints9a;
for (ii=pointsum1; (ii<=pointsum2); ii++)
{
  cond_countx=0.0;
  cond_county=0.0;
  if (((ii)<pointsum2) & ((long)(ii-pointsum1+1) % (n*t) != 0)){
    stampvalue = seriescond(2*G_bia_x/mu_vector[ii],2*G_bia_x/mu_vector[ii+1]);
    stampmatrix(ii, ii+1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
  }
  if (((ii+n*t)<pointsum2+1)){
    stampvalue = seriescond(2*G_bia_y/mu_vector[ii],2*G_bia_y/mu_vector[ii+n*t]);
    stampmatrix(ii, ii+n*t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
  }
  if ((((ii)>pointsum1) & ((long)(ii-pointsum1+1) % (n*t) !=1 ))){
    stampvalue = seriescond(2*G_bia_x/mu_vector[ii],2*G_bia_x/mu_vector[ii-1]);
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
  }
  if (((ii-pointsum1-n*t)>=0){
    stampvalue = seriescond(2*G_bia_y/mu_vector[ii],2*G_bia_y/mu_vector[ii-n*t]);
    stampmatrix(ii, ii-n*t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
  }
  stampmatrix(ii, ii,cond_countx+cond_county,w*sigma_ss*hx_bia*hy_bia*hz_bia)
}
endofsec9a=pointsum2;

// end of stamping section 9a

//%%% stamping section 9b only (no interconnections with the other sections yet)

```
stampvalue = seriescond(2*G\text{bi\_b\_y}/\mu\text{vector}[ii], 2*G\text{bi\_b\_y}/\mu\text{vector}[ii-n a]);
    stampmatrix(ii, ii-n a, -stampvalue, 0.0);
    cond\_county = cond\_county + stampvalue;
}
    stampmatrix(ii, ii, cond\_countx+cond\_county, w*\sigma\text{ss}*hx\text{bi\_b}*hy\text{bi\_b}*hz\text{bi\_b})
;
endofsec9b=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of stamping section 9b
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% stampping section 9c only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints9c = n*t*m\text{bi};
pointsum1=pointsum1+totalsecpoints9b;
pointsum2=pointsum2+totalsecpoints9c;
for (ii=pointsum1; (ii<pointsum2); ii++)
    {
        cond\_countx=0.0;
        cond\_county=0.0;
        if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) \% (n\ t) != 0))
            {
                stampvalue = seriescond(2*G\text{bi\_c\_x}/\mu\text{vector}[ii], 2*G\text{bi\_c\_x}/\mu\text{vector}[ii+1]);
                stampmatrix(ii, ii+1, -stampvalue, 0.0);
                cond\_countx = cond\_countx + stampvalue;
            }
        if ((ii+n\ t)<pointsum2+1)
            {
                stampvalue = seriescond(2*G\text{bi\_c\_y}/\mu\text{vector}[ii], 2*G\text{bi\_c\_y}/\mu\text{vector}[ii+n\ t]);
                stampmatrix(ii, ii+n\ t, -stampvalue, 0.0);
                cond\_county = cond\_county + stampvalue;
            }
        if (((ii)>pointsum1) && ((long)(ii-pointsum1+1) \% (n\ t) !=1 ))
            {
                stampvalue = seriescond(2*G\text{bi\_c\_x}/\mu\text{vector}[ii], 2*G\text{bi\_c\_x}/\mu\text{vector}[ii-1]);
                stampmatrix(ii, ii-1, -stampvalue, 0.0);
                cond\_countx = cond\_countx + stampvalue;
            }
        if ((ii-pointsum1-n\ t)>=0)
            {
                stampvalue = seriescond(2*G\text{bi\_c\_y}/\mu\text{vector}[ii], 2*G\text{bi\_c\_y}/\mu\text{vector}[ii-n\ t]);
                stampmatrix(ii, ii-n\ t, -stampvalue, 0.0);
                cond\_county = cond\_county + stampvalue;
            }
    }
endofsec9c=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% end of stamping section 9c
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%% finally, for section 9, connect subsections
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for (ii=1; (ii<=m\text{bi}); ii++)
    {
        firstindex = endofsec8+ii*(n\ t);  
        secondindex = endofsec9a+1+(ii-1)*(n\ a);
        stampvalue = seriescond(2*G\text{bi\_a\_x}/\mu\text{vector}[firstindex], 2*G\text{bi\_b\_x}/\mu\text{vector}[secondindex]);
        stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
        stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    }
newmodelingsparse.c

// also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);

for (ii=1; (ii<=m bi); ii++){
    firstindex = endofsec9a+ii*(n a);
    secondindex = endofsec9b+1+(ii-1)*(n t);
    stampvalue = seriescond(2*Gbi_b_x/mu_vector[firstindex], 2*G_bi_c_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);  // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// end of stamping section 9

//peating section 10 only (no interconnections with the other sections yet)

 totalsecpoints10 = n p*m bi;
 pointsum1=pointsum1+totalsecpoints9c;
 pointsum2=pointsum2+totalsecpoints10;
 for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii) <pointsum2) && ((long) (ii-pointsum1+1) % (n p) != 0)){
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-n p)<pointsum2+1){
        stampvalue = seriescond(2*Gbicx/mu_vector[ii],2*G_bic_x/mu_vector[ii+n_p]);
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)>pointsum) && ((long) (ii-pointsum1+1) % (n p) !=1 )){
        stampvalue = seriescond(2*Gbicx/mu_vector[ii],2*G_bic_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n p)>=0){
        stampvalue = seriescond(2*Gbicy/mu_vector[ii],2*G_bic_y/mu_vector[ii-n_p]);
        stampmatrix(ii, ii-n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii+cond_countx+cond_county,w*sigma_ss*hx_bic*hy_bic*hz_bic);
}
endofsec10=pointsum2;

// end of stamping section 10

//peating section 11a only (no interconnections with the other sections yet)
newmodelingsparse.c

totalsecpoints1la = n * m * p;
pointsum1=pointsum1+totalsecpoints10;
pointsum2=pointsum2+totalsecpoints1la;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long) (ii-pointsum1+1) % (n * t) != 0)){
        stampvalue = seriescond(2*G_w_a_x/mu_vector[ii],2*G_w_a_x/mu_vector[ii+1])
    };
    stampmatrix(ii, ii+1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii+n * t)<pointsum2+1){
    stampmatrix(ii, ii+n * t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
if (((ii)>pointsum1) && ((long) (ii-pointsum1+1) % (n * t) != 1 )){
    stampvalue = seriescond(2*G_w_a_x/mu_vector[ii],2*G_w_a_x/mu_vector[ii-1])
};
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii-pointsum1-n * t)>=0){
    stampvalue = seriescond (2*G_w_b_x/mu_vector[ii],2*G_w_b_x/mu_vector[ii-n * t])
};
    stampmatrix(ii, ii-n * t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
endofsec11a=pointsum2;

//*****************************************************************************
// end of stamping section 11a
//*****************************************************************************

//*****************************************************************************
//***** stamping section 11b only (no interconnections with the other sections yet)
//*****************************************************************************

totalsecpoints1lb = n * a * m * p;
pointsum1=pointsum1+totalsecpoints1la;
pointsum2=pointsum2+totalsecpoints1lb;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long) (ii-pointsum1+1) % (n * a) != 0)){
        stampvalue = seriescond(2*G_w_b_x/mu_vector[ii],2*G_w_b_x/mu_vector[ii+1])
    };
    stampmatrix(ii, ii+1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii+n * a)<pointsum2+1){
    stampmatrix(ii, ii+n * a, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
if (((ii)>pointsum1) && ((long) (ii-pointsum1+1) % (n * a) !=1 )){
    stampvalue = seriescond(2*G_w_b_x/mu_vector[ii],2*G_w_b_x/mu_vector[ii-1])
};
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii - pointsum1 - n a) >= 0) {
    stampvalue = seriescond(2*G_w_b_y/mu_vector[ii], 2*G_w_b_y/mu_vector[ii-n_a]);
    stampmatrix(ii, ii-n_a, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
} stampmatrix(ii, ii, cond_countx + cond_county, w*sigma_o*hx_b*hy_b*hz_b);
}
endofsec1lb = pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% end of stamping section 1lb
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% stamping section llc only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpointsllc = n t*m p;
pointsum1 = pointsum1 + totalsecpointsllb;
pointsum2 = pointsum2 + totalsecpointsllc;
for (ii=pointsum1; (ii<=pointsum2); ii++) {
    cond_countx = 0.0;
    cond_county = 0.0;
    if (((ii) < pointsum2) && ((long) (ii-pointsuml+l) % (n t) ) != 0)) {
        stampvalue = seriescond(2*G_w_c_x/mu_vector[ii], 2*G_w_c_x/mu_vector[ii+1])
    );
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if (((ii) + n t) < pointsum2+1) { 
        stampvalue = seriescond(2*G_w_c_y/mu_vector[ii], 2*G_w_c_y/mu_vector[ii+n_t])
    );
        stampmatrix(ii, ii+n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii) > pointsuml) && ((long) (ii-pointsuml+1) % (n t) != 1)) {
        stampvalue = seriescond(2*G_w_c_x/mu_vector[ii], 2*G_w_c_x/mu_vector[ii-1])
    );
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1 - n t) >= 0) {
        stampvalue = seriescond(2*G_w_c_y/mu_vector[ii], 2*G_w_c_y/mu_vector[ii-n_t])
    );
        stampmatrix(ii, ii-n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii, ii, cond_countx + cond_county, w*sigma_o*hx_b*hy_b*hz_b);
}
endofsec1lc = pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% end of stamping section llc
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% finally, for section 11, connect subsections
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for (ii=1; (ii<=m p); ii++) {
    firstindex = endofsec10+ii*(n t);
    secondindex = endofsec1la+1*(ii-1)*(n a);
    stampvalue = seriescond(2*G_w_a_x/mu_vector[firstindex], 2*G_w_b_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%% also, update the diagonals
}
newmodelingsparse.c

```c
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the
diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=1; (ii<=m); ii++){
    firstindex = endofsec1a+ii*(n a);
    secondindex = endofsec1b+1+(ii-1)*(n t);
    stampvalue = seriescond(2*G_w_b_x/mu_vector[firstindex], 2*G_w_c_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    //%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the di
agonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%%% end of stamping section 11

//%%%% stamping section 12 only (no interconnections with the other sections yet)

for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) & (long)(ii-pointsum1+1) % (n p) != 0){
        stampvalue = seriescond(2*G_12_x/mu_vector[ii], 2*G_12_y/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_12_y/mu_vector[ii], 2*G_12_y/mu_vector[ii+n p]
    );
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)>pointsum1) & (long)(ii-pointsum1+1) % (n p) != 1 ){
        stampvalue = seriescond(2*G_12_x/mu_vector[ii], 2*G_12_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n p)>=0){
        stampvalue = seriescond(2*G_12_y/mu_vector[ii], 2*G_12_y/mu_vector[ii-n p]
    );
        stampmatrix(ii, ii-n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_o*hx_12*hy_12*hz_12);
}
endofsec12=pointsum2;

//%%%% end of stamping section 12

//%%%% stamping section 13 only (no interconnections with the other sections yet)
```
newmodelingsparse.c

totalsecpoints13 = n * m * g;
pointsum1 = pointsum1 + totalsecpoints12;
pointsum2 = pointsum2 + totalsecpoints13;
for (ii = pointsum1; ii <= pointsum2; ii++) {
   cond_countx = 0.0;
   cond_county = 0.0;
   if (((ii) < pointsum2) && ((long) (ii - pointsum1 + 1) % (n * t) != 0)) {
      stampvalue = seriescond(2 * G_13_x / mu_vector[ii], 2 * G_13_x / mu_vector[ii + 1]);
      stampmatrix(ii, ii + 1, -stampvalue, 0.0);
      cond_countx = cond_countx + stampvalue;
   }
   if ((ii + n * t) < pointsum2 + 1) {
      stampvalue = seriescond(2 * G_13_y / mu_vector[ii], 2 * G_13_y / mu_vector[ii + n * t])
  );
   stampmatrix(ii, ii + n * t, -stampvalue, 0.0);
   cond_county = cond_county + stampvalue;
}
if (((ii) < pointsum1) && ((long) (ii - pointsum1 + 1) % (n * t) != 1)) {
   stampvalue = seriescond(2 * G_13_x / mu_vector[ii], 2 * G_13_x / mu_vector[ii - 1]);
   stampmatrix(ii, ii - 1, -stampvalue, 0.0);
   cond_countx = cond_countx + stampvalue;
}
if ((ii - pointsum1 - n * t) >= 0) {
   stampvalue = seriescond(2 * G_13_y / mu_vector[ii], 2 * G_13_y / mu_vector[ii - n * t])
   stampmatrix(ii, ii - n * t, -stampvalue, 0.0);
   cond_county = cond_county + stampvalue;
}
endofsecl3 = pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% end of stamping section 13
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%% stamping section 14 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints14 = n * a * m * g;
pointsum1 = pointsum1 + totalsecpoints13;
pointsum2 = pointsum2 + totalsecpoints14;
for (ii = pointsum1; ii <= pointsum2; ii++) {
   cond_countx = 0.0;
   cond_county = 0.0;
   if (((ii) < pointsum2) && ((long) (ii - pointsum1 + 1) % (n * a) != 0)) {
      stampvalue = seriescond(2 * G_14_x / mu_vector[ii], 2 * G_14_x / mu_vector[ii + 1]);
      stampmatrix(ii, ii + 1, -stampvalue, 0.0);
      cond_countx = cond_countx + stampvalue;
   }
   if ((ii + n * a) < pointsum2 + 1) {
      stampvalue = seriescond(2 * G_14_y / mu_vector[ii], 2 * G_14_y / mu_vector[ii + n * a])
   );
   stampmatrix(ii, ii + n * a, -stampvalue, 0.0);
   cond_county = cond_county + stampvalue;
}
if (((ii) < pointsum1) && ((long) (ii - pointsum1 + 1) % (n * a) != 1)) {
   stampvalue = seriescond(2 * G_14_x / mu_vector[ii], 2 * G_14_x / mu_vector[ii - 1]);
   stampmatrix(ii, ii - 1, -stampvalue, 0.0);
   cond_countx = cond_countx + stampvalue;
}
if ((ii - pointsum1 - n * a) >= 0) {
   stampvalue = seriescond(2 * G_14_y / mu_vector[ii], 2 * G_14_y / mu_vector[ii - n * a])
   );
   stampmatrix(ii, ii - n * a, -stampvalue, 0.0);
   cond_county = cond_county + stampvalue;
}
newmodelingsparse.c

    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_o*hx_14*hy_14*hz_14);
    }
    endofsec14=pointsum2;

    //**********************************************************************
    //**** end of stamping section 14
    //**********************************************************************

    //**********************************************************************
    //**** stamping section 15 only (no interconnections with the other sections yet)
    //**********************************************************************

    totalsecpoints15 = n_t*m_g;
    pointsum1=pointsum1+totalsecpoints14;
    pointsum2=pointsum2+totalsecpoints15;
    for (ii=pointsum1; (ii<=pointsum2); ii++){
        cond_countx=0.0;
        cond_county=0.0;
        if (((ii)<pointsum2) && ((long) (ii-pointsum1+l) % stampvalue = 0)){
            stampvalue = seriescond(2*G_15_x/mu_vector[ii],2*G_15_x/mu_vector[ii+1]);
            stampmatrix(ii, ii+1, -stampvalue, 0.0);
            cond_countx = cond_countx + stampvalue;
        }
        if ((ii+n_t)<pointsum2+1){
            stampvalue = seriescond(2*G_15_y/mu_vector[ii],2*G_15_y/mu_vector[ii+n_t]);
            stampmatrix(ii, ii+n_t, -stampvalue, 0.0);
            cond_county = cond_county + stampvalue;
        }
        if (((ii)>pointsum1) && ((long) (ii-pointsum1+l) % (n_t) != 0)){
            stampvalue = seriescond(2*G_15_y/mu_vector[ii],2*G_15_y/mu_vector[ii-1]);
            stampmatrix(ii, ii-1, -stampvalue, 0.0);
            cond_countx = cond_countx + stampvalue;
        }
        if ((ii-pointsum1-n_t)>=0){
            stampvalue = seriescond(2*G_15_y/mu_vector[ii],2*G_15_y/mu_vector[ii-n_t]);
            stampmatrix(ii, ii-n_t, -stampvalue, 0.0);
            cond_county = cond_county + stampvalue;
        }
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_o*hx_15*hy_15*hz_15);
    endofsec15=pointsum2;

    //**********************************************************************
    //**** end of stamping section 15
    //**********************************************************************

    //**********************************************************************
    //**** stamping section 16 only (no interconnections with the other sections yet)
    //**********************************************************************

    totalsecpoints16 = n_p*m_g;
    pointsum1=pointsum1+totalsecpoints15;
    pointsum2=pointsum2+totalsecpoints16;
    for (ii=pointsum1; (ii<=pointsum2); ii++){
        cond_countx=0.0;
        cond_county=0.0;
        if (((ii)<pointsum2) && ((long) (ii-pointsum1+l) % (n_p) != 0)){
            stampvalue = seriescond(2*G_16_x/mu_vector[ii],2*G_16_x/mu_vector[ii+1]);
            stampmatrix(ii, ii+1, -stampvalue, 0.0);
            cond_countx = cond_countx + stampvalue;
        }
        if ((ii+n_p)<pointsum2+1){
            stampvalue = seriescond(2*G_16_y/mu_vector[ii],2*G_16_y/mu_vector[ii+n_p]);
            stampmatrix(ii, ii+n_p, -stampvalue, 0.0);
cond_county = cond_county + stampvalue;
}
if (((ii)>pointsuml) && ((long)(ii-pointsuml+1) % (n p) !=1 )){
    stampvalue = seriescond(2*G_16_x/mu_vector[ii],2*G_16_x/mu_vector[ii-1]);
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if (((ii-pointsuml-n p) >=0) {
    stampvalue = seriescond(2*G_16_y/mu_vector[ii],2*G_16_y/mu_vector[ii-np]_17);
    stampmatrix(ii, ii-n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_o*hx_17*hy_17*hz_17);
endofsec16=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//**** end of stamping section 16
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//**** stamping section 17 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints17 = n p*m c;
pointsum1=pointsum1+totalsecpoints16;
pointsum2=pointsum2+totalsecpoints17;
for (ii=pointsuml; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsuml+1) % (n p) != 0)){
        stampvalue = seriescond(2*G_17_x/mu_vector[ii],2*G_17_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if (((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_17_y/mu_vector[ii],2*G_17_y/mu_vector[ii+n p]_17);
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii-pointsuml) && ((long)(ii-pointsuml+1) % (n p) !=1 )){
        stampvalue = seriescond(2*G_17_x/mu_vector[ii],2*G_17_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if (((ii-pointsuml-n p) >=0) {
        stampvalue = seriescond(2*G_17_y/mu_vector[ii],2*G_17_y/mu_vector[ii-np]_17);
        stampmatrix(ii, ii-n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_c*hx_17*hy_17*hz_17);
}
endofsec17=pointsum2;

//**** end of stamping section 17
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//**** stamping section 18 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints18 = n t*m c;
pointsum1=pointsum1+totalsecpoints17;
pointsum2=pointsum2+totalsecpoints18;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n t) != 0)){
        stampvalue = seriescond(2*G_18_x/mu_vector[ii], 2*G_18_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n t)<pointsum2+1){
        stampvalue = seriescond(2*G_18_y/mu_vector[ii], 2*G_18_y/mu_vector[ii+n_t]);
        stampmatrix(ii, ii+n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n t) != 1 )){
        stampvalue = seriescond(2*G_18_x/mu_vector[ii], 2*G_18_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n t)>=0){
        stampvalue = seriescond(2*G_18_y/mu_vector[ii], 2*G_18_y/mu_vector[ii-n_t]);
        stampmatrix(ii, ii-n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_c*hx_18*hy_18*hz_18);
}
endofsec18=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% stamping section 18
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%}

//%%% stamping section 19 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints19 = n a*m c;
pointsum1=pointsum1+totalsecpoints18;
pointsum2=pointsum2+totalsecpoints19;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n a) != 0)){
        stampvalue = seriescond(2*G_19_x/mu_vector[ii], 2*G_19_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n a)<pointsum2+1){
        stampvalue = seriescond(2*G_19_y/mu_vector[ii], 2*G_19_y/mu_vector[ii+n_a]);
        stampmatrix(ii, ii+n a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n a) != 1 )){
        stampvalue = seriescond(2*G_19_x/mu_vector[ii], 2*G_19_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n a)>=0){
        stampvalue = seriescond(2*G_19_y/mu_vector[ii], 2*G_19_y/mu_vector[ii-n_a]);
        stampmatrix(ii, ii-n a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_c*hx_19*hy_19*hz_19);
newmodelingsparse.c

endofsec19=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%% end of stamping section 19
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%% stamping section 20 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints20 = n t*m c;
pointsum1=pointsum1+totalsecpoints19;
pointsum2=pointsum2+totalsecpoints20;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long) (ii-pointsum1+1) % (n t) != 0)){
        stampvalue = seriescond(2*G_20_x/mu_vector[ii],2*G_20_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n t)<pointsum2+1){
        stampvalue = seriescond(2*G_20_y/mu_vector[ii],2*G_20_y/mu_vector[ii+n_t] )
    );
    stampmatrix(ii, ii+n t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
    }
    if (((ii)>pointsum1) && ((long) (ii-pointsum1+1) % (n p) !=1 )){
        stampvalue = seriescond(2*G_20_x/mu_vector[ii],2*G_20_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n t)>=0){
        stampvalue = seriescond(2*G_20_y/mu_vector[ii],2*G_20_y/mu_vector[ii-n_t] );
    );
    stampmatrix(ii, ii-n t, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_c*hx_20*hy_20*hz_20);
}
endofsec20=pointsum2;

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%% end of stamping section 20
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
//%%%%% stamping section 21 only (no interconnections with the other sections yet)
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totalsecpoints21 = n p*m c;
pointsum1=pointsum1+totalsecpoints20;
pointsum2=pointsum2+totalsecpoints21;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long) (ii-pointsum1+1) % (n p) != 0)){
        stampvalue = seriescond(2*G_21_x/mu_vector[ii],2*G_21_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampvalue = seriescond(2*G_21_y/mu_vector[ii],2*G_21_y/mu_vector[ii+n_p])
    );
    stampmatrix(ii, ii+n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
    }
if (((ii)>pointsum1)) && ((long)(ii-pointsum1+1) % (n p) !=1 )){
    stampvalue = seriescond(2*G 21 x/mu vector[ii],2*G_21_x/mu_vector[ii-1]);
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
} 
if (((ii-pointsum1-n p)>=0){
    stampvalue = seriescond(2*G 21_y/mu_vector[ii],2*G_21_y/mu_vector[ii-n p] )
    stampmatrix(ii, ii-n p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
} 
} stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_c*h_x_21*h_y_21*h_z_21);
endofsec21=pointsum2;

end of stamping section 21

//%%% end of stamping section 22 only (no interconnections with the other sections yet)

//%%% stamping section 22 only (no interconnections with the other sections yet)

//%%% stamping section 23 only (no interconnections with the other sections yet)
pointsum1=pointsum1+totalsecpoints22;
pointsum2=pointsum2+totalsecpoints23;
for (ii=pointsum1; (ii<=pointsum2); ii++)
{
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n_t) ! = 0))
    {
        stampvalue = seriescond(2*G_23_x/mu_vector[ii],2*G_23_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n_t)<pointsum2+1){
        stampvalue = seriescond(2*G_23_y/mu_vector[ii],2*G_23_y/mu_vector[ii+n_t]);
        stampmatrix(ii, ii+n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n_t) !=1))
    {
        stampvalue = seriescond(2*G_23_x/mu_vector[ii],2*G_23_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n_t)>=0){
        stampvalue = seriescond(2*G_23_y/mu_vector[ii],2*G_23_y/mu_vector[ii-n_t]);
        stampmatrix(ii, ii-n_t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if ((ii+n_a)<pointsum2+1){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii+n_a]);
        stampmatrix(ii, ii+n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n_a) != 0))
    {
        stampvalue = seriescond(2*G_24_x/mu_vector[ii],2*G_24_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n_a)<pointsum2+1){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii+n_a]);
        stampmatrix(ii, ii+n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if ((ii-pointsum1-n_a)>=0){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii-n_a]);
        stampmatrix(ii, ii-n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_elec_ss*hx_23*hy_23*hz_23);
}
endofsec23=pointsum2;

//%%%% end of stamping section 23

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%
//%%%% stamping section 24 only (no interconnections with the other sections yet)

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%

totalsecpoints24 = n_a*m_r;
pointsum1=pointsum1+totalsecpoints23;
pointsum2=pointsum2+totalsecpoints24;
for (ii=pointsum1; (ii<=pointsum2); ii++)
{
    cond_countx=0.0;
    cond_county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n_a) ! = 0))
    {
        stampvalue = seriescond(2*G_24_x/mu_vector[ii],2*G_24_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n_a)<pointsum2+1){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii+n_a]);
        stampmatrix(ii, ii+n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if (((ii)<pointsum1) && ((long)(ii-pointsum1+1) % (n_a) !=1))
    {
        stampvalue = seriescond(2*G_24_x/mu_vector[ii],2*G_24_x/mu_vector[ii-1]);
        stampmatrix(ii, ii-1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii-pointsum1-n_a)>=0){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii-n_a]);
        stampmatrix(ii, ii-n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    if ((ii+n_a)<pointsum2+1){
        stampvalue = seriescond(2*G_24_y/mu_vector[ii],2*G_24_y/mu_vector[ii+n_a]);
        stampmatrix(ii, ii+n_a, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    stampmatrix(ii,ii,cond_countx+cond_county,w*sigma_elec_ss*hx_24*hy_24*hz_24);
newmodelingsparse.c

}
endofsec24=pointsum2;

//end of stamping section 24

//stamping section 25 only (no interconnections with the other sections yet)

totalsecpoints25 = n t*m r;
pointsum1=pointsum1+totalsecpoints24;
pointsum2=pointsum2+totalsecpoints25;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n t) != 0)){
        stampvalue = seriescond(2*G 25_x/mu_vector[ii],2*G_25_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n t)<pointsum2+1){
        stampmatrix(ii, ii+n t, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
    else {
        for ((ii+1)<=pointsum2+1){
            stampmatrix(ii, ii+1, -stampvalue, 0.0);
            cond_countx = cond_countx + stampvalue;
        } 
    }
    cond_county = cond_county + stampvalue;
}
endofsec25=pointsum2;

//end of stamping section 25

//stamping section 26 only (no interconnections with the other sections yet)

totalsecpoints26 = n p*m r;
pointsum1=pointsum1+totalsecpoints24;
pointsum2=pointsum2+totalsecpoints25;
for (ii=pointsum1; (ii<=pointsum2); ii++){
    cond countx=0.0;
    cond county=0.0;
    if (((ii)<pointsum2) && ((long)(ii-pointsum1+1) % (n p) != 0)){
        stampvalue = seriescond(2*G 26_x/mu_vector[ii],2*G_26_x/mu_vector[ii+1]);
        stampmatrix(ii, ii+1, -stampvalue, 0.0);
        cond_countx = cond_countx + stampvalue;
    }
    if ((ii+n p)<pointsum2+1){
        stampmatrix(ii, ii+n p, -stampvalue, 0.0);
        cond_county = cond_county + stampvalue;
    }
if (((ii) > pointsum1) && ((long) (ii-pointsum1 + 1) % (n_p) != 1))
{
    stampvalue = seriescond(2 * G_26_x/mu_vector[ii], 2 * G_26_x/mu_vector[ii-1]);
    stampmatrix(ii, ii-1, -stampvalue, 0.0);
    cond_countx = cond_countx + stampvalue;
}
if ((ii-pointsum1-n_p) >= 0){
    stampvalue = seriescond(2 * G_26_y/mu_vector[ii], 2 * G_26_y/mu_vector[ii-n_p]);
    stampmatrix(ii, ii-n_p, -stampvalue, 0.0);
    cond_county = cond_county + stampvalue;
}
stampmatrix(ii, ii, cond_countx+cond_county, w*sigma_elec_ss*hx_26*hy_26*hz_26);
endofsec26 = pointsum2;

//****************************************************
// end of stamping section 26
//****************************************************

//<<< Now, connect individual sections together

// sections 1-2
for (ii=1; (ii<=m); ii++){
    firstindex = ii*n_p;
    secondindex = endofsec1+1+(ii-1)*(n_t);
    stampvalue = seriescond(2 * G_p_x/mu_vector[firstindex], 2 * G_t_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// sections 4-5
for (ii=1; (ii<=m); ii++){
    firstindex = endofsec3+ii*(n_t);
    secondindex = endofsec4+1+(ii-1)*(n_p);
    stampvalue = seriescond(2 * G_t_x/mu_vector[firstindex], 2 * G_p_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// sections 1-6
for (ii=0; (ii<=(n_p)-1); ii++){
    firstindex = endofsec1-ii;
    secondindex = endofsec5+(n_p)-ii;
    stampvalue = seriescond(2 * G_p_y/mu_vector[firstindex], 2 * G_pp_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}
newmodelingsparse.c

ondindex);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 5-7
//%%%%%%%%%%%%%%%%

for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec5-ii;
    secondindex = endofsec6+(n p)-ii;
    stampvalue = seriescond(2*G_pp_y/mu_vector[firstindex], 2*G_pp_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 6-8
//%%%%%%%%%%%%%%%%

for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec6-ii;
    secondindex = endofsec7+(n p)-ii;
    stampvalue = seriescond(2*G_pp_y/mu_vector[firstindex], 2*G_bic_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 7-10
//%%%%%%%%%%%%%%%%

for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec7-ii;
    secondindex = endofsec9c+(n p)-ii;
    stampvalue = seriescond(2*G_pp_y/mu_vector[firstindex], 2*G_bic_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 8-9
//%%%%%%%%%%%%%%%%

for (ii=1; (ii<=m bi); ii++){
    firstindex = endofsec7+ii*n p;
    secondindex = endofsec8+(ii-1)*(n t);
    stampvalue = seriescond(2*G_bic_x/mu_vector[firstindex], 2*G_bia_x/mu_vector[secondindex]);


```c

for (ii=1; (ii<=m bi); ii++){
    firstindex = endofsec9b+(ii)*(n t);
    secondindex = endofsec9c+1+(ii-1)*n p;
    stampvalue = seriescond(2*G_bic_x/muvector[firstindex], 2*G_bic_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=1; (ii<=m g); ii++){
    firstindex = endofsec11c+ii*(n p);
    secondindex = endofsec12+1+(ii-1)*n p;
    stampvalue = seriescond(2*G_12_x/muvector[firstindex], 2*G_13_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=1; (ii<=m g); ii++){
    firstindex = endofsec12+ii*(n p);
    secondindex = endofsec13+1+(ii-1)*n a;
    stampvalue = seriescond(2*G_13_x/muvector[firstindex], 2*G_14_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=1; (ii<=m g); ii++){
    firstindex = endofsec14+ii*(n a);
    secondindex = endofsec15+1+(ii-1)*(n t);
    stampvalue = seriescond(2*G_14_x/muvector[firstindex], 2*G_15_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
}
```


stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
    }

    //%%% sections 15-16
    //%%% sections 17-18
    for (ii=1; (ii<=m g); ii++){
        firstindex = endofsec14+ii*(n t);
        secondindex = endofsec15+1+(ii-1)*(n p);
        stampvalue = seriescond(2*G_15_x/muvector[firstindex], 2*G_16_x/muvector[secondindex]);
        stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
        stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
        //%%% also, update the diagonals
        stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
        stampmatrix(secondindex, secondindex, stampvalue, 0.0);
    }

    //%%% sections 18-19
    for (ii=1; (ii<=m c); ii++){
        firstindex = endofsec16+ii*(n p);
        secondindex = endofsec17+1+(ii-1)*(n t);
        stampvalue = seriescond(2*G_17_x/muvector[firstindex], 2*G_18_x/muvector[secondindex]);
        stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
        stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
        //%%% also, update the diagonals
        stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
        stampmatrix(secondindex, secondindex, stampvalue, 0.0);
    }

    //%%% sections 19-20
    for (ii=1; (ii<=m c); ii++){
        firstindex = endofsec18+ii*(n a);
        secondindex = endofsec19+1+(ii-1)*(n t);
        stampvalue = seriescond(2*G_19_x/muvector[firstindex], 2*G_20_x/muvector[secondindex]);
        stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
        stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
        //%%% also, update the diagonals
        stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
        stampmatrix(secondindex, secondindex, stampvalue, 0.0);
    }
// also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);

// sections 20-21
for (ii=1; (ii<=m c); ii++){
  firstindex = endofsec19+ii*(n t);
  secondindex = endofsec20+1+(ii-1)*(n p);
  stampvalue = seriescond(2*G_20_x/muvector[firstindex], 2*G_21_x/mu_vector[secondindex]);
  stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
  stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
  // also, update the diagonals
  stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
  stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// sections 22-23
for (ii=1; (ii<=m r); ii++){
  firstindex = endofsec21+ii*(n p);
  secondindex = endofsec22+1+(ii-1)*(n t);
  stampvalue = seriescond(2*G_22_x/mu_vector[firstindex], 2*G_23_x/mu_vector[secondindex]);
  stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
  stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
  // also, update the diagonals
  stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
  stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// sections 23-24
for (ii=1; (ii<=m r); ii++){
  firstindex = endofsec22+ii*(n t);
  secondindex = endofsec23+1+(ii-1)*(n a);
  stampvalue = seriescond(2*G_23_x/mu_vector[firstindex], 2*G_24_x/mu_vector[secondindex]);
  stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
  stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
  // also, update the diagonals
  stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
  stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// sections 24-25
for (ii=1; (ii<=m r); ii++){
  firstindex = endofsec23+ii*(n a);
  secondindex = endofsec24+1+(ii-1)*(n t);
  stampvalue = seriescond(2*G_24_x/mu_vector[firstindex], 2*G_25_x/mu_vector[secondindex]);
  stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
  stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
// also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term

}  

// sections 25-26

for (ii=1; (ii<=m r); ii++){
    firstindex = endofsec24+i*(n t);
    secondindex = endofsec25+i+(ii-1)*(n p);
    stampvalue = seriescond(2*G_25_x/muvector[firstindex], 2*G_26_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);  // also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);  // this ADDS to the diagonal term

}  

// Next setup is to input the cyclic boundary conditions

// sections 5-1

for (ii=1; (ii<=m a); ii++){
    firstindex = endofsec4+i*(n p);
    secondindex = 1+(ii-1)*(n p);
    stampvalue = seriescond(2*G_p_x/muvector[firstindex], 2*G_p_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);  // also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);  // this ADDS to the diagonal term

}  

// sections 7-6

for (ii=1; (ii<=m p); ii++){
    firstindex = endofsec6+i*(n p);
    secondindex = endofsec5+i+(ii-1)*(n p);
    stampvalue = seriescond(2*G_pp_x/muvector[firstindex], 2*G_pp_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);  // also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);  // this ADDS to the diagonal term

}  

// sections 10-8

for (ii=1; (ii<=m bi); ii++){
    firstindex = endofsec9c+i*(n p);
    secondindex = endofsec7+i+(ii-1)*(n p);
    stampvalue = seriescond(2*G_bic_x/muvector[firstindex], 2*G_bic_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);  //
// %%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// %%%% sections 16-12
for (ii=1; (ii<=m_g); ii++){
    firstindex = endofsec15+ii*(n_p);
    secondindex = endofsec11+1+(ii-1)*(n_p);
    stampvalue = seriescond(2*G_16_x/muvector[firstindex], 2*G_12_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);
    // %%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    // this ADDS to the diagonal term
}

// %%%% sections 21-17
for (ii=1; (ii<=m_c); ii++){
    firstindex = endofsec20+ii*(n_p);
    secondindex = endofsec16+1+(ii-1)*(n_p);
    stampvalue = seriescond(2*G_21_x/muvector[firstindex], 2*G_17_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);
    // %%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    // this ADDS to the diagonal term
}

// %%%% sections 26-22
for (ii=1; (ii<=m_r); ii++){
    firstindex = endofsec25+ii*(n_p);
    secondindex = endofsec21+1+(ii-1)*(n_p);
    stampvalue = seriescond(2*G_26_x/muvector[firstindex], 2*G_22_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, 0.0, stampvalue);
    stampmatrix(secondindex, firstindex, 0.0,-stampvalue);
    // %%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    // this ADDS to the diagonal term
}

// %%%% The next thing to do is to attach different materials together

// %%%% sections 2-3
for (ii=1; (ii<=m_a); ii++){
    firstindex = endofsec1+ii*(n_t);
    secondindex = endofsec2+1+(ii-1)*(n_a);
    stampvalue = seriescond(2*G_t_x/muvector[firstindex], 2*G_a_x/muvector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    // %%%% also, update the diagonals
}
for (ii=1; (ii<=m a); ii++){
    firstindex = endofsec2+ii*(n a);
    secondindex = endofsec3+1+(ii-1)*(n t);
    stampvalue = seriescond(2*G_a_x/mu_vector[firstindex], 2*G_t_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 6-11
for (ii=1; (ii<=m p); ii++){
    firstindex = endofsec5+ii*(n p);
    secondindex = endofsec10+1+(ii-1)*(n t);
    stampvalue = seriescond(2*G_pp_x/mu_vector[firstindex], 2*G_w_a_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 11-7
for (ii=1; (ii<=m p); ii++){
    firstindex = endofsec11b+ii*(n t);
    secondindex = endofsec6+1+(ii-1)*(n p);
    stampvalue = seriescond(2*G_w_c_x/mu_vector[firstindex], 2*G_pp_x/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 2-11
for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec2-ii;
    secondindex = endofsec10+(n t)-ii;
    stampvalue = seriescond(2*G_t_y/mu_vector[firstindex], 2*G_w_a_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 3-11
for (ii=0; (ii<=(n a)-1); ii++){
    firstindex = endofsec3-ii;
    secondindex = endofsec6+(n a)-ii;
    stampvalue = seriescond(2*G_a_y/mu_vector[firstindex], 2*G_t_y/mu_vector[secondindex]);
newmodelingsparse.c

secondindex = endofsec11a+(n a)-ii;
stampvalue = seriescond(2*G_a_y/mu_vector[firstindex], 2*G_w_b_y/mu_vector[secondindex]);
stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
//%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 4-11
for (ii=0; (ii<=(n t)-1); ii++){
firstindex = endofsec4-ii;
secondindex = endofsec11b+(n t)-ii;
stampvalue = seriescond(2*G_t_y/mu_vector[firstindex], 2*G_w_c_y/mu_vector[secondindex]);
stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
//%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 11a-9a
for (ii=0; (ii<=(n t)-1); ii++){
firstindex = endofsec11a-ii;
secondindex = endofsec8+(n t)-ii;
stampvalue = seriescond(2*G_w_a_y/mu_vector[firstindex], 2*G_bi_a_y/mu_vector[secondindex]);
stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
//%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 11b-9b
for (ii=0; (ii<=(n a)-1); ii++){
firstindex = endofsec11b-ii;
secondindex = endofsec9a+(n a)-ii;
stampvalue = seriescond(2*G_w_b_y/mu_vector[firstindex], 2*G_bi_b_y/mu_vector[secondindex]);
stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
//%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 11c-9c
for (ii=0; (ii<=(n t)-1); ii++){
firstindex = endofsec11c-ii;
secondindex = endofsec9b+(n t)-ii;
stampvalue = seriescond(2*G_w_c_y/mu_vector[firstindex], 2*G_bi_c_y/mu_vector[secondindex]);
stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
//%%% also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
stampmatrix(secondindex, secondindex, stampvalue, 0.0);
newmodelingsparse.c

//%%% sections 12-1
for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec12-ii;
    secondindex = n p-ii;
    stampvalue = seriescond(2*G_12_y/mu-vector[firstindex], 2*G_p_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 13-2
for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec13-ii;
    secondindex = endofsec1+n t-ii;
    stampvalue = seriescond(2*G_13_y/mu_vector[firstindex], 2*G_t_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 14-3
for (ii=0; (ii<=(n a)-1); ii++){
    firstindex = endofsec14-ii;
    secondindex = endofsec2+n a-ii;
    stampvalue = seriescond(2*G_14_y/mu_vector[firstindex], 2*G_a_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 15-4
for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec15-ii;
    secondindex = endofsec3+n t-ii;
    stampvalue = seriescond(2*G_15_y/mu_vector[firstindex], 2*G_t_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

//%%% sections 16-5
for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec16-ii;
    secondindex = endofsec4+n p-ii;
    stampvalue = seriescond(2*G_16_y/mu_vector[firstindex], 2*G_p_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
newmodelingsparse.c

ondindex));
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
  }

  //%%% sections 17-12
  for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec17-ii;
    secondindex = endofsec12+n p-ii;
    stampvalue = seriescond(2*G_17_y/mu_vector[firstindex], 2*G_12_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
  }

  //%%% sections 18-13
  for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec18-ii;
    secondindex = endofsec12+n t-ii;
    stampvalue = seriescond(2*G_18_y/mu_vector[firstindex], 2*G_13_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
  }

  //%%% sections 19-14
  for (ii=0; (ii<=(n a)-1); ii++){
    firstindex = endofsec19-ii;
    secondindex = endofsec13+n a-ii;
    stampvalue = seriescond(2*G_19_y/mu_vector[firstindex], 2*G_14_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
  }

  //%%% sections 20-15
  for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec20-ii;
    secondindex = endofsec14+n t-ii;
    stampvalue = seriescond(2*G_20_y/mu_vector[firstindex], 2*G_15_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec21-ii;
    secondindex = endofsec15+n p-ii;
    stampvalue = seriescond(2*G_21_y/mu_vector[firstindex], 2*G_16_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec22-ii;
    secondindex = endofsec16+n p-ii;
    stampvalue = seriescond(2*G_22_y/mu_vector[firstindex], 2*G_17_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec23-ii;
    secondindex = endofsec17+n t-ii;
    stampvalue = seriescond(2*G_23_y/mu_vector[firstindex], 2*G_18_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=0; (ii<=(n a)-1); ii++){
    firstindex = endofsec24-ii;
    secondindex = endofsec18+n a-ii;
    stampvalue = seriescond(2*G_24_y/mu_vector[firstindex], 2*G_19_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    //%%% also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0); // this ADDS to the diagonal term
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

for (ii=0; (ii<=(n t)-1); ii++){
    firstindex = endofsec25-ii;
    secondindex = endofsec19+n t-ii;
    stampvalue = seriescond(2*G_25_y/mu_vector[firstindex], 2*G_20_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);

newmodelingsparse.c

/// also, update the diagonals
stampmatrix(firstindex, firstindex, stampvalue, 0.0);  // this ADDS to the diagonal term
            stampmatrix(secondindex, secondindex, stampvalue, 0.0);

/// sections 26-21
for (ii=0; (ii<=(n p)-1); ii++){
    firstindex = endofsec26+ii;
    secondindex = endofsec20+n p-ii;
    stampvalue = seriescond(2*G_26_y/mu_vector[firstindex], 2*G_21_y/mu_vector[secondindex]);
    stampmatrix(firstindex, secondindex, -stampvalue, 0.0);
    stampmatrix(secondindex, firstindex, -stampvalue, 0.0);
    /// also, update the diagonals
    stampmatrix(firstindex, firstindex, stampvalue, 0.0);
    stampmatrix(secondindex, secondindex, stampvalue, 0.0);
}

// LINKING SECTIONS IS DONE! FINALLY, STAMP IN THE SOURCES
for (ii=0; (ii<totalpoints); ii++){
    RHS[ii] = 0.0;
}

for (kk=1; (kk<=(m p)-2); kk++){
    for (ii=2; (ii<=(n t)); ii++){
        RHS[endofsec10+ii+kk*(n_t)-1] = hx_w_a*hy_w_a*hz_w_a*current;
    }
    for (ii=1; (ii<=(n a)); ii++){
        RHS[endofsec11a+ii+kk*(n_a)-1] = hx_w_b*hy_w_b*hz_w_b*current;
    }
    for (ii=1; (ii<=(n t)-1); ii++){
        RHS[endofsec11b+ii+kk*(n_t)-1] = hx_w_c*hy_w_c*hz_w_c*current;
    }
}

// Define A to be zero at the bottom
for (ii=0; (ii<=(n p)-1); ii++){
    setboundary(endofsec8-ii);
}

for (ii=0; (ii<=(n t)-1); ii++){
    setboundary(endofsec9a-ii);
}

for (ii=0; (ii<=(n a)-1); ii++){
    setboundary(endofsec9b-ii);
}
for (ii=0; (ii<=(n t)-1); ii++){  
  setboundary(endofsec9c-ii);
  
} // top of section 9c
for (ii=0; (ii<=(n p)-1); ii++){  
  setboundary(endofsec10-ii);
  
} // top of section 10

// Define A to be zero at the top
for (ii=1; (ii<=(n p)); ii++){  
  setboundary(endofsec21+ii);
  
} // top of section 22
for (ii=1; (ii<=(n t)); ii++){  
  setboundary(endofsec22+ii);
  
} // top of section 23
for (ii=1; (ii<=(n a)); ii++){  
  setboundary(endofsec23+ii);
  
} // top of section 24
for (ii=1; (ii<=(n t)); ii++){  
  setboundary(endofsec24+ii);
  
} // top of section 25
for (ii=1; (ii<=(n p)); ii++){  
  setboundary(endofsec25+ii);
  
} // top of section 26

// at the end, make sure you are not passing any zero indices
for (ii=entrynumber; (ii<totalentries); ii++){  
  rowvector[ii] = 1;
  colvector[ii] = 1;
}

// end point information
endpointinfo[0]=endofsec1;
endpointinfo[1]=endofsec2;
endpointinfo[2]=endofsec3;
endpointinfo[3]=endofsec4;
endpointinfo[4]=endofsec5;
endpointinfo[6]=endofsec7;
endpointinfo[7]=endofsec8;
endpointinfo[8]=endofsec9a;
endpointinfo[9]=endofsec10;
endpointinfo[10]=endofsec11a;
endpointinfo[12]=endofsec13;
endpointinfo[13]=endofsec14;
endpointinfo[14]=endofsec15;
endpointinfo[15]=endofsec16;
endpointinfo[16]=endofsec17;
endpointinfo[17]=endofsec18;
endpointinfo[18]=endofsec19;
endpointinfo[19]=endofsec20;
endpointinfo[20]=endofsec21;
endpointinfo[21]=endofsec22;
endpointinfo[22]=endofsec23;
endpointinfo[23]=endofsec24;
endpointinfo[24]=endofsec25;
endpointinfo[25]=endofsec26;
endpointinfo[26]=endofsec9b;
endpointinfo[27]=endofsec9c;
endpointinfo[28]=endofsec11b;
endpointinfo[29]=endofsec11c;

// see if you can solve the system of equations from here, calling MATLAB

double stampmatrix(long rowcoor, long colcoor, double entryreal, double entryimag)
{
    rowvector[entrynumber] = rowcoor;
    colvector[entrynumber] = colcoor;
    realvector[entrynumber] = entryreal;
    imagvector[entrynumber] = entryimag;
    entrynumber++;
    return(0.0);
}

double seriescond(double x, double y)
{
    return(x*y/(x+y));
}

double setboundary(long rownumber)
{
    long kk;
    for (kk=0; (kk<totalentries); kk++){
        if (rowvector[kk] == rownumber)
            if (colvector[kk] == rownumber){
                realvector[kk] = 1.0;
            }
            else{
                realvector[kk] = 0.0;
            }
        imagvector[kk] = 0.0;
    }
    return(0.0);
}
THIS IS THE C-CODE FOR THE SIMULATION SOFTWARE BASED ON MODEL III

written by
HUR KOSER

#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <malloc.h>
#include <string.h>
#include <math.h>
#include "nrutil.h"
#include "nrutil.c"
#include "stdafx.h"

void fdtd2d(void);
double findsigmaT(double);
void waitabit(void);
void waitaminute(void);

int main(int argc, char* argv[])
{
    //printf("Hello World!\n");
    fdtd2d();
    return 0;
}

void fdtd2d(void)
{
    long n, x, y, indexcount, ie x n, ie y n;
    double Reff, Ls, Jext amp, freq, deltaT;
    double nmax, dt, omegadt, omega;
    double **dx matrix m, **dy matrix m, **Jz1, **Jz2;
    double **ez m, **hx m, **hy m, **bx m, **by m, **mu m;
    double **xez m, **ca matrix m, **cb x matrix m, **cb y matrix m, **cc matrix_m;
    double **db x matrix m, **db y matrix_m, **eps matrix_m, **sigma_matrix_m;
    double Jext sum, timepast;
    double Jextfactl, Jextfact2, JJsinl, JJsin2, JJcosl, JJcos2, sin omegadtl, sinom 
        // double cosomegadtl, cosomegadt2;
    char firstlinechar[1];
    int foundworksheetspot = 0, no go = 0, decimal, sign;
    double *torquevector, *timevector, *energyvector, *energyvectorss;
    long torquecount, y gap, shallIcompute;
    double poles, Width, sumofstuff, mu_o, magB;//, rotorarea, statorvolume, statorar 
            ea;
    double Hpoint1, Hpoint2, Hpoint3, Hpoint4, Hpoint5, Hpoint6;
    double Hpoint7, Hpoint8, Hpoint9, Hpoint10, Hpoint11, Hpoint12;
    double Bpoint7, Bpoint8, Bpoint9, Bpoint10, Bpoint11, Bpoint12;
    double Hpoint1wafer, Hpoint2wafer, Hpoint3wafer, Hpoint4wafer, Hpoint5wafer, Hpo 
        int6wafer;
    double Hpoint7wafer, Hpoint8wafer, Hpoint9wafer, Hpoint10wafer, Hpoint11wafer, Hp 
        int12wafer;
    double Bpoint1wafer, Bpoint2wafer, Bpoint3wafer, Bpoint4wafer, Bpoint5wafer, Bpo 
        int6wafer;
    double Bpoint7wafer, Bpoint8wafer, Bpoint9wafer, Bpoint10wafer, Bpoint11wafer, Bp 
        int12wafer;
    double temp magh, JJ, effective area;
    double slope1, slope2, slope3, slope4, slope5, slope6, slope7, slope8, slope9, s1 
        ope10;
    double slope11, slope12, slope13;
    double slope1wafer, slope2wafer, slope3wafer, slope4wafer, slope5wafer, slope6waf 
        er;
double slope7wafer, slope8wafer, slope9wafer, slope10wafer;
double slope11wafer, slope12wafer, slope13wafer;
double cc, epsz, endtime, out len, eps out, sigma out, mu out;
double eps ss, sigma ss, mu ss, eps wafer, sigma wafer, mu wafer;
double eps gap, mu_gap, sigma_gap, eps_cond, sigma_cond, sigma_cu, mu_cond, energysum, energysum ss;
double wirewidth, wireheight, Lt, Tt, Lp, Lbi, Gap, Tcond, TotalL, Tp, Trotor, Tb
i, rotor cond, stator cond, rotor ss;
long m out, m r, m c, n g, m a, m p, m bi, n p, n t, n a;
double dx p, dx t, dx a, dy r, dy cond, dy gap, dy a, dy p, dy bi, dy_out;
long locx1, locx2, locx3, locx4, locx5, locx6, locx7, locx8, locx9;
long locy1, locy2, locy3, locy4, locy5, locy6, locy7, locy8;
double mintemp, dt temp, distance, kucuk, Pi = 3.14159265358979, meshfactor = 1.0
, waferrelaxation = 500.0, niferelaxation = 1.0;
long torquepoints = 100, ii, linecount;
float foofool, foofoo2, foofoo3, foofoo4, foofoo5, foofoo6, foofoo7, foofoo9, foo
foo10, foofoo11, copperthick;
long foofoo8;
char line[200], resultsfile[400], filepathlog[400], filepathworksheet[400], esasfilepath[400];
FILE *worksheet, *resultssheet, *logsheet, *locfile;
timp-t Itime;

// first and foremost, determine where the results and the log files will be written
if ((locfile = fopen("C:/locfile.txt", "r")) != NULL)
{
  // attempt to read a necessary amount of characters from file
  if (fscanf(locfile, "%s", esasfilepath) == EOF)
  {
    printf("End of locfile reached!\n");
    no go = 1;
    foundworksheetspot = 1;
  }
  else
  {
    printf("Read from file: %s \n", esasfilepath);
  }
  fclose(locfile);
}
else
{
  printf("Houston, we have a file opening problem!\n");
  printf("Tried to read from: %s\n", esasfilepath);
}

while (1)
{
  no go = 0;
  while (!no go)
  {
    //determine the location of the worksheet
    strcpy(filepathworksheet, esasfilepath);
    if ((worksheet = fopen(cat(filepathworksheet,"/testworksheet.txt"), "r") != NULL)
    {
      // attempt to read a necessary amount of characters from file
      foundworksheetspot = NULL;
      linecount = 0;
      while (!foundworksheetspot)
      {
        linecount++;
        if (fscanf(worksheet, "%d \t %e \t %e \t %e \t %e \t %e \t %e \t %e \t %ld \t %e \t %e \t %e \t %e
", shallIcompute, &foofoo1, &foofoo2, &foofoo3, &foofoo4, &foofoo5, &foofoo6, &foofoo7, &foofoo8, &foofoo9, &foofoo10, &foofoo11) == EOF)
```c
{  
    printf("End of worksheet reached!\n");
    no go = 1;
    foundworksheetspot = 1;
} else  
{  
    //printf("Read from file: %d %e %e %e %e %e %e %e %ld \n", shallIcompute, foofoo1, foofoo2, foofoo3, foofoo4, foofoo5, foofoo6, foofoo7, foofoo8);
    //printf("Line[0]: %d\n", shallIcompute);
    //now see if this worksheet line has been worked at already ...
    if (shallIcompute == 0)  
    {  
        foundworksheetspot = 1;
        printf("Found an assignment on line %d ... Proceeding...\n", linecount);
    }
}
fclose(worksheet);
}
else  
{  
    printf("Houston, we have a file opening problem!");
}

///// READ THE NECESSARY PARAMETERS FROM FILE
Reff = foofoo1;
Ls = foofoo2;
Jext amp = foofoo3;
freq = foofoo4;
deltaT = foofoo5;
Gap = foofoo6;
endtime = foofoo7;
torquepoints = foofoo8;
copperthick = foofoo9;
rotor cond = foofoo10;
stator cond = foofoo11;
waitabit();
if (!no_go) {

    // Fundamental constants
    cc = 5e3;
    mu_o = 4*Pi*1e-7;
    epsz = 1/(cc*cc*mu_o);
    
    // Inputer parameters
    omega = 2.0*Pi*freq;
    Width = 1e-3;
    poles = 6;

    // Geometry parameters
    // outside gap
    out len = 100e-6;
    m out = 10;
    sigma out = 1e7;
    mu out = 5000.0*mu_o;
    eps out = epsz;

    // ...
}
```


```c
///////// stator //////////
sigma ss = stator cond; // this could be a parameter
mu ss = 9000.0*mu_o;
eps ss = epsz;

///////// wafer //////////
sigma wafer = 1.82e6;
mu wafer = 9000.0*mu_o;
eps wafer = epsz;

///////// air gap //////////
sigma gap = 2e4;
mu gap = 1.0*mu_o;
eps gap = epsz;

///////// conductor //////////
mu cond = 1.0;
sigma cu = findsigmaT(deltaT); // change this to incorporate temperature
mu cond = 1.0*mu_o;
eps cond = epsz;

printf("Cu cond thickness: %e\n", copperthick);

///// DEFINE THE GEOMETRY NOW //////////
//poles = 6.0;
wirewidth = 300e-6;
wireheight = 65e-6;
Lt = (wirewidth-Ls)/2.0; // length of each teeth (constant across radial cross-section)
Tt = 100e-6; // thickness of the teeth
Lp = 240e-6;//310e-6; // Pole height
Lbi = wirewidth;
Tcond = copperthick;//10e-6; // rotor conductor thickness
TotalL = 2.0*Pi*Reff/(poles/2.0); // total length of the symmetrical cross section
Tp = ((TotalL/4.0) - wirewidth)/2.0; // half width of the poles

n p = (long)ceil(Tp*meshfactor/10e-6);
n t = (long)ceil(Lt*meshfactor/10e-6);
n a = (long)ceil(Ls*meshfactor); //cell(Ls/10e-6);
m a = (long)ceil(Tt*meshfactor/10e-6);
m p = (long)ceil(Lp*meshfactor/10e-6);
m g = (long)ceil(Gap*meshfactor/10e-6);
m_c = (long)ceil(1*meshfactor);

TRotor = 200e-6;
m r = (long)ceil(20*meshfactor);
Tbi = 600e-6;
m bi = (long)ceil(60*meshfactor);

///////////////////////////////

// here is the discretization
dx p = Tp/n p;
dx t = Lt/n t;
dx a = Ls/n a;
dy r = T rotor/m r;
dy cond = T cond/m c;
dy gap = Gap/m g;
dy a = T t/m a;
dy p = Lp/m p;
dy bi = T bi/m bi;
dy out = out_len/m out;

ie x n = (long)(n p*4 + n t*4 + n a*2 + 2);
ie_y n = (long)((m out+1)*2 + m r + m c + m g + m a + m p + m bi);
```
DEFINE THE MATERIAL PROPERTY AND GRID MATRICES

locx1 = (n_p+1);
locx2 = (locx1+n_t);
locx3 = (locx2+n_a);
locx4 = (locx3+n_t);
locx5 = (locx4+(n_p*2));
locx6 = (locx5+n_t);
locx7 = (locx6+n_a);
locx8 = (locx7+n_t);
locx9 = (locx8+(n_p+l));
locy1 = (m_out+1);
locy2 = (locy1+m_x);
locy3 = (locy2+m_c);
locy4 = (locy3+m_g);
locy5 = (locy4+m_m);
locy6 = (locy5+m_p);
locy7 = (locy6+m_b);
locy8 = (locy7+(m_out+1));

// DEFINE THE HEAT GENERATING ROTOR/STATOR AREA
//rotorarea = (Trotor+Tcond)*(2.0*Pi*Reff/poles);
//statorarea = (2.0*Tt*((2.0*Lt)+(2.0*Tp))) + (2.0*Tp*(2.0*Pi*Reff/poles));

// to use the matrix notation, define corresponding matrix variables
eg m = dmatrix(1,ie_x n, 1, ie_y n);
hx m = dmatrix(1,ie_x n, 1, ie_y n);
hy m = dmatrix(1,ie_x n, 1, ie_y n);
bg m = dmatrix(1,ie_x n, 1, ie_y n);
by m = dmatrix(1,ie_x n, 1, ie_y n);
mu m = dmatrix(1,ie_x n, 1, ie_y n);
xex m = dmatrix(1,ie_x n, 1, ie_y n);
ca matrix m = dmatrix(1,ie_x n, 1, ie_y n);
cb x matrix m = dmatrix(1,ie_x n, 1, ie_y n);
cb y matrix m = dmatrix(1,ie_x n, 1, ie_y n);
cc matrix m = dmatrix(1,ie_x n, 1, ie_y n);
db x matrix m = dmatrix(1,ie_x n, 1, ie_y n);
db y matrix m = dmatrix(1,ie_x n, 1, ie_y n);
Jz1 = dmatrix(1,ie_x n, 1, ie_y n);
Jz2 = dmatrix(1,ie_x n, 1, ie_y n);
dx matrix m = dmatrix(1,ie_x n, 1, ie_y n);
dy matrix m = dmatrix(1,ie_x n, 1, ie_y n);
eps matrix m = dmatrix(1,ie_x n, 1, ie_y n);
sigma matrix m = dmatrix(1,ie_x n, 1, ie_y n);
torquevector = dvector(1,torquepoints);
timevector = dvector(1,torquepoints);
energyvector = dvector(1,torquepoints);
energyvector_ss = dvector(1,torquepoints);

// first, dx, and dy matrices //
for (y=1;y<=ie_y n;y++){
   for (x=1;x<=locx1;x++){
      dx_matrix_m[x][y] = dx_p;
   }
   for (x=locx1+l;x<=locx2;x++){
      dx_matrix_m[x][y] = dx_t;
   }
}
for (x=locx2+1;x<=locx3;x++){
    dx_matrix_m[x][y] = dx_a;
}  
for (x=locx3+1;x<=locx4;x++){
    dx_matrix_m[x][y] = dx_t;
}  
for (x=locx4+1;x<=locx5;x++){
    dx_matrix_m[x][y] = dx_p;
}  
for (x=locx5+1;x<=locx6;x++){
    dx_matrix_m[x][y] = dx_t;
}  
for (x=locx6+1;x<=locx7;x++){
    dx_matrix_m[x][y] = dx_a;
}  
for (x=locx7+1;x<=locx8;x++){
    dx_matrix_m[x][y] = dx_t;
}  
for (x=locx8+1;x<=locx9;x++){
    dx_matrix_m[x][y] = dx_p;
}  
}

for (x=1;x<=ie x n;x++){
    for (y=1;y<=locy1;y++){
        dy_matrix_m[x][y] = dy_out;
    }
    for (y=locy1+1;y<=locy2;y++){
        dy_matrix_m[x][y] = dy_r;
    }
    for (y=locy2+1;y<=locy3;y++){
        dy_matrix_m[x][y] = dy_cond;
    }
    for (y=locy3+1;y<=locy4;y++){
        dy_matrix_m[x][y] = dy_gap;
    }
    for (y=locy4+1;y<=locy5;y++){
        dy_matrix_m[x][y] = dy_a;
    }
    for (y=locy5+1;y<=locy6;y++){
        dy_matrix_m[x][y] = dy_p;
    }
    for (y=locy6+1;y<=locy7;y++){
        dy_matrix_m[x][y] = dy_b1;
    }
    for (y=locy7+1;y<=locy8;y++){
        dy_matrix_m[x][y] = dy_out;
    }
}

// NOW, define sigma and epsilon matrices for the motor //  
for (x=1;x<=ie x n;x++){
    for (y=1;y<=locy1;y++){
        eps matrix m[x][y] = eps out;
        sigma_matrix_m[x][y] = sigma_out;
    }
    for (y=locy1+1;y<=locy2;y++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = rotor_ss;
    }
    for (y=locy2+1;y<=locy3;y++){
        eps matrix m[x][y] = eps cond;
        sigma_matrix_m[x][y] = sigma_cu;
    }
    for (y=locy3+1;y<=locy4;y++){
        eps matrix m[x][y] = eps gap;
        sigma_matrix_m[x][y] = sigma_gap;
    }
for (y=locy6+1;y<=locy7;y++){
    eps matrix m[x][y] = eps wafer;
    sigma_matrix_m[x][y] = sigma_wafer;
}
for (y=locy7+1;y<=locy8;y++){
    eps matrix m[x][y] = eps out;
    sigma_matrix_m[x][y] = sigma_out;
}

// teeth section
for (y=locy4+1;y<=locy5;y++){
    for (x=1;x<=locx2;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
    for (x=locx2+1;x<=locx3;x++){
        eps matrix m[x][y] = eps gap;
        sigma_matrix_m[x][y] = sigma_gap;
    }
    for (x=locx3+1;x<=locx6;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
    for (x=locx6+1;x<=locx7;x++){
        eps matrix m[x][y] = eps gap;
        sigma_matrix_m[x][y] = sigma_gap;
    }
    for (x=locx7+1;x<=locx9;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
}

// poles section
for (y=locy5+1;y<=locy6;y++){
    for (x=1;x<=locx1;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
    for (x=locx1+1;x<=locx4;x++){
        eps matrix m[x][y] = eps cond;
        sigma_matrix_m[x][y] = sigma_cond;
    }
    for (x=locx4+1;x<=locx5;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
    for (x=locx5+1;x<=locx8;x++){
        eps matrix m[x][y] = eps cond;
        sigma_matrix_m[x][y] = sigma_cond;
    }
    for (x=locx8+1;x<=locx9;x++){
        eps matrix m[x][y] = eps ss;
        sigma_matrix_m[x][y] = sigma_ss;
    }
}

// calculate the largest dt possible //
mintemp = 1e9;
for (y=1;y<=ie y;y++){
    for (x=1;x<=ie x;x++){
        distance = sqrt((1/dx_matrix_m[x][y])*dx_matrix_m[x][y]));
        dt temp = ((1/cc)/2.0)/distance;
        if (dt temp < mintemp){
            mintemp = dt_temp;
        }
    }
}
dt = mintemp; // play around with this time step
nmax = (long) ceil(endtime/dt);

// determine where there is external current applied
for (y=1;y<ie y;n;y++)
  for (x=1;xx<ie x;n;x++)
    Jz1[x][y] = 0.0;
    Jz2[x][y] = 0.0;
}

for (y=locy5+1;y<locy6;y++)
  for (x=locx1+1;xx<locx4;x++)
    Jz1[x][y] = 1.0;
    Jz2[x][y] = 1.0;
}

// now, determine the effective area over which current is applied
effective_area = 0.0;
for (y=1;y<ie y;n;y++)
  for (x=1;xx<ie x;n;x++)
    effective_area += dx_matrix_m[x][y]*dy_matrix_m[x][y];
}

JJ = Jext_amp*sqrt(2)/effective_area;

indexcount = 0;
for (y=1;y<ie y;n;y++)
  for (x=1;xx<ie x;n;x++)
    ez_matrix_m[x][y] = 0.0; // (ez+indexcount);
    hx_matrix_m[x][y] = 0.0; // (hx+indexcount);
    hy_matrix_m[x][y] = 0.0; // (hy+indexcount);
    bx_matrix_m[x][y] = 0.0; // (bx+indexcount);
    by_matrix_m[x][y] = 0.0; // (byfoo+indexcount);
    mu_matrix_m[x][y] = mu_o; // (mu_matrix+indexcount);
    // xez_matrix_m[x][y] = *(xez+indexcount);
    kucuk = (dt*sigma_matrix_m[x][y])/(eps_matrix_m[x][y]*2.0);
    ca_matrix_m[x][y] = (1.0-kucuk)/(1.0+kucuk); // (ca_matrix+indexcount);
    cb_x_matrix_m[x][y] = -dt/(dy_matrix_m[x][y]*eps_matrix_m[x][y] * (1.0+kucuk)); // (cb x_matrix+indexcount);
    cb_y_matrix_m[x][y] = dt/(dx_matrix_m[x][y]*eps_matrix_m[x][y] * (1.0+kucuk)); // (cb y_matrix+indexcount);
    // Jz1[x][y] = *(Jz1_matrix+indexcount);
    // Jz2[x][y] = *(Jz2_matrix+indexcount);
    cc_matrix_m[x][y] = -dt*(Jz1[x][y] + Jz2[x][y])/(2.0*eps_matrix_m[x][y] * (1.0+kucuk)); // (cc_matrix+indexcount);
    // db_x_matrix_m[x][y] = *(db x_matrix+indexcount);
    // db_y_matrix_m[x][y] = *(db y_matrix+indexcount);
    // dx_matrix_m[x][y] = *(dx_matrix+indexcount);
    // dy_matrix_m[x][y] = *(dy_matrix+indexcount);
    indexcount++;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// define the frequency with which torque will be computed
omegadt = omega*dt;
when = (long)ceil((double)nmax/(double)torquepoints);//ceil(1/(omega
dt/(2.0*Pi)))*32.0)
);
torquecount = 0;
y gap = locy3+1;
printf("nmax = %e", nmax);
// DEFINE THE POINTS ALONG THE BH CURVE
/*Hpointlwafer = 1.0e-1;
Hpoint2wafer = 4.6444e0;
Hpoint3wafer = 2.7367e1;
Hpoint4wafer = 4.1e1;
Hpoint5wafer = 6.3722e1;
Hpoint6wafer = 9.0989e1;
Hpoint7wafer = 1.1371e2;
Hpoint8wafer = 1.5461e2;
Hpoint9wafer = 2.2278e2;
Hpoint10wafer = 3.1367e2;
Hpointllwafer = 3.8183e2;
Hpoint12wafer = 4.5e2;
Bpointlwafer = 6.6036e-6;
Bpoint2wafer = 1.3689e-2;
Bpoint3wafer = 2.7783e-1;
Bpoint4wafer = 4.1201e-1;
Bpoint5wafer = 5.3882e-1;
Bpoint6wafer = 6.1147e-1;
Bpoint7wafer = 6.4447e-1;
Bpoint8wafer = 6.7839e-1;
Bpoint9wafer = 7.0727e-1;
Bpoint10wafer = 7.2790e-1;
Bpoint11wafer = 7.3817e-1;
Bpoint12wafer = 7.4620e-1;/* //THESE POINTS WERE ADOPTIONS FOR
// WAFER NIFE FROM ELECTROPLATED NIFE, DO NOT USE THEM ANY MORE!!!!@@
!
Hpointlwafer = waferrelaxation*0.0004;
Hpoint2wafer = waferrelaxation*0.001;
Hpoint3wafer = waferrelaxation*0.003;
Hpoint4wafer = waferrelaxation*0.005;
Hpoint5wafer = waferrelaxation*0.008;
Hpoint6wafer = waferrelaxation*0.01;
Hpoint7wafer = waferrelaxation*0.02;
Hpoint8wafer = waferrelaxation*0.04;
Hpoint9wafer = waferrelaxation*0.1;
Hpoint10wafer = waferrelaxation*0.2;
Hpoint11wafer = waferrelaxation*1.0;
Hpoint12wafer = waferrelaxation*2.0;
Bpointlwafer = 30e-4;
Bpoint2wafer = 100e-4;
Bpoint3wafer = 600e-4;
Bpoint4wafer = 2000e-4;
Bpoint5wafer = 3500e-4;
Bpoint6wafer = 4000e-4;
Bpoint7wafer = 5500e-4;
Bpoint8wafer = 6000e-4;
Bpoint9wafer = 6800e-4;
Bpoint10wafer = 7100e-4;
Bpoint11wafer = 7300e-4;
Bpoint12wafer = 7.4620e-1;
slopelwafer = Hpointlwafer/Bpointlwafer;
slope2wafer = (Hpoint2wafer-Hpoint1wafer)/(Bpoint2wafer-Bpoint1wafer)
;
slope3wafer = (Hpoint3wafer-Hpoint2wafer)/(Bpoint3wafer-Bpoint2wafer)
;
slope4wafer = (Hpoint4wafer-Hpoint3wafer)/(Bpoint4wafer-Bpoint3wafer)
slopewafer = (Hpointwafer-Hpoint4wafer)/(Bpointwafer-Bpoint4wafer);
slopewafer = (Hpointwafer-Hpoint5wafer)/(Bpointwafer-Bpoint5wafer);
slopewafer = (Hpointwafer-Hpoint6wafer)/(Bpointwafer-Bpoint6wafer);
slopewafer = (Hpointwafer-Hpoint7wafer)/(Bpointwafer-Bpoint7wafer);
slopewafer = (Hpointwafer-Hpoint8wafer)/(Bpointwafer-Bpoint8wafer);
slopewafer = (Hpointwafer-Hpoint9wafer)/(Bpointwafer-Bpoint9wafer);
slopewafer = (Hpointwafer-Hpoint10wafer)/(Bpointwafer-Bpoint10wafer);
slopewafer = (Hpointwafer-Hpoint11wafer)/(Bpointwafer-Bpoint11wafer);
slopewafer = (Hpointwafer-Hpoint12wafer)/(Bpointwafer-Bpoint12wafer);

// electroplated NiFe B-H points
Hpoint1 = niferelaxation*1.0e-1;
Hpoint2 = niferelaxation*4.64440e;
Hpoint3 = niferelaxation*2.7367e1;
Hpoint4 = niferelaxation*4.1e1;
Hpoint5 = niferelaxation*6.3722e1;
Hpoint6 = niferelaxation*9.0989e1;
Hpoint7 = niferelaxation*1.1371e2;
Hpoint8 = niferelaxation*1.5461e2;
Hpoint9 = niferelaxation*2.278e2;
Hpoint10 = niferelaxation*3.1367e2;
Hpoint11 = niferelaxation*3.8183e2;
Hpoint12 = niferelaxation*4.5e2;

Bpoint1 = (0.86038/0.75)*6.6036e-6;
Bpoint2 = (0.86038/0.75)*1.3689e-2;
Bpoint3 = (0.86038/0.75)*2.7783e-1;
Bpoint4 = (0.86038/0.75)*4.1201e-1;
Bpoint5 = (0.86038/0.75)*5.3882e-1;
Bpoint6 = (0.86038/0.75)*6.1147e-1;
Bpoint7 = (0.86038/0.75)*6.4447e-1;
Bpoint8 = (0.86038/0.75)*6.7839e-1;
Bpoint9 = (0.86038/0.75)*7.0727e-1;
Bpoint10 = (0.86038/0.75)*7.2790e-1;
Bpoint11 = (0.86038/0.75)*7.3817e-1;
Bpoint12 = (0.86038/0.75)*7.4620e-1;

// the following commented line is STUPID, because you multiplied
// H points with the scaling factor, too.
/*Hpoint1 = (0.86038/0.75)*Hpoint1wafer;
Hpoint2 = (0.86038/0.75)*Hpoint2wafer;
Hpoint3 = (0.86038/0.75)*Hpoint3wafer;
Hpoint4 = (0.86038/0.75)*Hpoint4wafer;
Hpoint5 = (0.86038/0.75)*Hpoint5wafer;
Hpoint6 = (0.86038/0.75)*Hpoint6wafer;
Hpoint7 = (0.86038/0.75)*Hpoint7wafer;
Hpoint8 = (0.86038/0.75)*Hpoint8wafer;
Hpoint9 = (0.86038/0.75)*Hpoint9wafer;
Hpoint10 = (0.86038/0.75)*Hpoint10wafer;
Hpoint11 = (0.86038/0.75)*Hpoint11wafer;
Hpoint12 = (0.86038/0.75)*Hpoint12wafer;

Bpoint1 = (0.86038/0.75)*Bpoint1wafer;
Bpoint2 = (0.86038/0.75)*Bpoint2wafer;
Bpoint3 = (0.86038/0.75)*Bpoint3wafer;
Bpoint4 = (0.86038/0.75)*Bpoint4wafer;
Bpoint5 = (0.86038/0.75)*Bpoint5wafer;
Bpoint6 = (0.86038/0.75)*Bpoint6wafer;
Bpoint7 = (0.86038/0.75)*Bpoint7wafer;*/
Bpoint8 = (0.86038/0.75)*Bpoint8wafer;
Bpoint9 = (0.86038/0.75)*Bpoint9wafer;
Bpoint10 = (0.86038/0.75)*Bpoint10wafer;
Bpoint11 = (0.86038/0.75)*Bpoint11wafer;
Bpoint12 = (0.86038/0.75)*Bpoint12wafer; */
slope1  = Hpoint1/Bpoint1;
slope2  = (Hpoint2-Hpoint1)/(Bpoint2-Bpoint1);
slope3  = (Hpoint3-Hpoint2)/(Bpoint3-Bpoint2);
slope4  = (Hpoint4-Hpoint3)/(Bpoint4-Bpoint3);
slope5  = (Hpoint5-Hpoint4)/(Bpoint5-Bpoint4);
slope6  = (Hpoint6-Hpoint5)/(Bpoint6-Bpoint5);
slope7  = (Hpoint7-Hpoint6)/(Bpoint7-Bpoint6);
slope8  = (Hpoint8-Hpoint7)/(Bpoint8-Bpoint7);
slope9  = (Hpoint9-Hpoint8)/(Bpoint9-Bpoint8);
slope10 = (Hpoint10-Hpoint9)/(Bpoint10-Bpoint9);
slope11 = (Hpoint11-Hpoint10)/(Bpoint11-Bpoint10);
slope12 = (Hpoint12-Hpoint11)/(Bpoint12-Bpoint11);
slope13 = 1/mu_o;

//printf("1 geldi!!!\n");
for (n=1;n<=nmax;n++)
{
    // this is the time iteration

    timepast = dt*n;
    JJextfact1 = (1.0-exp(-2e6*dt*n))*JJ;
    sin omegadt1 = sin(omegadt*n + (Pi/4));
    cos omegadt1 = cos(omegadt*n + (Pi/4));
    JJsin1 = JJextfact1*sin omegadt1;
    JJcos1 = JJextfact1*cos omegadt1;

    JJextfact2 = (1.0-exp(-2e6*dt*(n-1)))*JJ;
    sin omegadt2 = sin(omegadt*(n-1) + (Pi/4));
    cos omegadt2 = cos(omegadt*(n-1) + (Pi/4));
    JJsin2 = JJextfact2*sin omegadt2;
    JJcos2 = JJextfact2*cos omegadt2;

    for (y=2;y<=ie_y-1;y++)
    {
        // this is the space iteration for ez

        for (x=2;x<=ie_x-1;x++)
        {
            // ez update equation

            ez m[x][y] = a matrix m[x][y] * ez m[x][y]
            + cb x matrix m[x][y] * (hx m[x][y+1] - hx m[x][y])
            + cb y matrix m[x][y] * (hy m[x+1][y] - hy m[x][y])
            + cc_matrix_m[x][y] * Jext sum;
        }
    }
}

//for (x=1;x<=ie x;n++;)
     // here are the antisymmetric boundary conditions on either side

     // ez m[x][1] = 0.0;
     // ez_m[x][ie_y_n] = 0.0;

     // NOTE THAT NO E FIELD ON TOP OR BOTTOM IS AUTOMATICALLY SATISFI

     //} for (y=1;y<=ie y;n;y++)
     // here are the antisymmetric boundary conditions on either s

     ez m[1][y] = -ez m[ie x n-1][y];
     ez_m[ie_x_n][y] = -ez_m[2][y];

ED!!!
// NOTE THAT NO E FIELD ON TOP OR BOTTOM IS AUTOMATICALLY SATISFIED!!

// update db x matrix and db y matrix here
for (y=2;y<ie y-1;y++){
    // this is the space iteration for ez
    for (x=2;x<ie x-1;x++){
        
        db y matrix m[x][y] = 2.0*dt/(dx_matrix_m[x][y] + (mu_m[x -1][y]/mu m[x][y]) * dx matrix m[x-1][y]);
        db x matrix m[x][y] = -2.0*dt/(dy_matrix_m[x][y] + (mu_m[x][y-1]/mu m[x][y]) * dy matrix_m[x][y-1]);
    }
}

// deal with the first indices!!!!!!
for (x=1;x<ie x n;x++){
    db x matrix m[x][1] = -2.0*dt/(dy_matrix_m[x][1] + (mu_m[x][1]/mu m[x][y])*dy matrix_m[x][y]);
}

// end of db_matrix update

// for the rotor NiFe
for (y=locy1+1;y<locy2;y++){
    // this is the space iteration for hy
    for (x=1;x<ie x n;x++){
        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y]);
        if (magB < Bpoint1){
            temp_magh = slope1*magB;
        } else if ((magB >= Bpoint1) && (magB < Bpoint2)){
            temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
        } else if ((magB >= Bpoint2) && (magB < Bpoint3)){
            temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
        } else if ((magB >= Bpoint3) && (magB < Bpoint4)){
            temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
        } else if ((magB >= Bpoint4) && (magB < Bpoint5)){
            temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
        }
    }
}
temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
} else if ((magB >= Bpoint5) && (magB < Bpoint6)) {
    temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
} else if ((magB >= Bpoint6) && (magB < Bpoint7)) {
    temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
} else if ((magB >= Bpoint7) && (magB < Bpoint8)) {
    temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
} else if ((magB >= Bpoint8) && (magB < Bpoint9)) {
    temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
} else if ((magB >= Bpoint9) && (magB < Bpoint10)) {
    temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
} else if ((magB >= Bpoint10) && (magB < Bpoint11)) {
    temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
} else if ((magB >= Bpoint11) && (magB < Bpoint12)) {
    temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
} else if (magB >= Bpoint12) {
    temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
}

mu_m[x][y] = (magB+le-9*mu_o)/(temp_magh+le-9);

} // for the left, top side of the stator NiFe
for (y=locy4+1;y<=locy5;y++) { // this is the space iteration for
    for (x=1;x<=locx2;x++) {
        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y]);
        if (magB < Bpoint1) {
            temp_magh = slope1*magB;
        } else if ((magB >= Bpoint1) && (magB < Bpoint2)) {
            temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
        } else if ((magB >= Bpoint2) && (magB < Bpoint3)) {
            temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
        } else if ((magB >= Bpoint3) && (magB < Bpoint4)) {
            temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
        } else if ((magB >= Bpoint4) && (magB < Bpoint5)) {
            temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
        } else if ((magB >= Bpoint5) && (magB < Bpoint6)) {
            temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
        } else if ((magB >= Bpoint6) && (magB < Bpoint7)) {
            temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
        } else if ((magB >= Bpoint7) && (magB < Bpoint8)) {
            temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
        } else if ((magB >= Bpoint8) && (magB < Bpoint9)) {
            temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
        } else if ((magB >= Bpoint9) && (magB < Bpoint10)) {
            temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
        }
else if ((magB >= Bpoint10) && (magB < Bpoint11))
    temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
else if ((magB >= Bpoint11) && (magB < Bpoint12))
    temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
else if (magB >= Bpoint12)
    temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
}

mu_m[x][y] = (magB+1e-9*mu_o)/(temp_magh+1e-9);
}

// for the center, top side of the stator NiFe
for (y=locy4+1;y<=locy5;y++) // this is the space iteration for hy
for (x=locx3+1;x<=locx6;x++){
    magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y]);
    if (magB < Bpoint1) {
        temp_magh = slope1*magB;
    } else if ((magB >= Bpoint1) && (magB < Bpoint2)) {
        temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
    } else if ((magB >= Bpoint2) && (magB < Bpoint3)) {
        temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
    } else if ((magB >= Bpoint3) && (magB < Bpoint4)) {
        temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
    } else if ((magB >= Bpoint4) && (magB < Bpoint5)) {
        temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
    } else if ((magB >= Bpoint5) && (magB < Bpoint6)) {
        temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
    } else if ((magB >= Bpoint6) && (magB < Bpoint7)) {
        temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
    } else if ((magB >= Bpoint7) && (magB < Bpoint8)) {
        temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
    } else if ((magB >= Bpoint8) && (magB < Bpoint9)) {
        temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
    } else if ((magB >= Bpoint9) && (magB < Bpoint10)) {
        temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
    } else if ((magB >= Bpoint10) && (magB < Bpoint11)) {
        temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
    } else if ((magB >= Bpoint11) && (magB < Bpoint12)) {
        temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
    } else if (magB >= Bpoint12) {
        temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
    }

    mu_m[x][y] = (magB+1e-9*mu_o)/(temp_magh+1e-9);
}

// for the right, top side of the stator NiFe
for (y=locy4+1;y<=locy5;y++) // this is the space iteration for hy
for (x=locx7+1;x<=ie_x_n;x++){
```c
magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y])

if (magB < Bpoint1)
    temp_magh = slope1*magB;
else if (((magB >= Bpoint1) && (magB < Bpoint2))
    temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
else if (((magB >= Bpoint2) && (magB < Bpoint3))
    temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
else if (((magB >= Bpoint3) && (magB < Bpoint4))
    temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
else if (((magB >= Bpoint4) && (magB < Bpoint5))
    temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
else if (((magB >= Bpoint5) && (magB < Bpoint6))
    temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
else if (((magB >= Bpoint6) && (magB < Bpoint7))
    temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
else if (((magB >= Bpoint7) && (magB < Bpoint8))
    temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
else if (((magB >= Bpoint8) && (magB < Bpoint9))
    temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
else if (((magB >= Bpoint9) && (magB < Bpoint10))
    temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
else if (((magB >= Bpoint10) && (magB < Bpoint11))
    temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
else if (((magB >= Bpoint11) && (magB < Bpoint12))
    temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
else if ((magB >= Bpoint12)
    temp_magh = slope13*(magB-Bpoint12) + Hpoint12;

mu_m[x][y] = (magB+le-9*muo)/(temp_magh+le-9);

// for the left half-pole of the stator NiFe
for (y=locy5+1;y<=locy6;y++){
    for (x=1;x<=locxl;x++){
        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y])
        if (magB < Bpoint1)
            temp_magh = slope1*magB;
        else if (((magB >= Bpoint1) && (magB < Bpoint2))
            temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
        else if (((magB >= Bpoint2) && (magB < Bpoint3))
            temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
        else if (((magB >= Bpoint3) && (magB < Bpoint4))
            temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
        else if (((magB >= Bpoint4) && (magB < Bpoint5))
            temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
```
else if ((magB >= Bpoint5) && (magB < Bpoint6)) {
    temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
}
else if ((magB >= Bpoint6) && (magB < Bpoint7)) {
    temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
}
else if ((magB >= Bpoint7) && (magB < Bpoint8)) {
    temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
}
else if ((magB >= Bpoint8) && (magB < Bpoint9)) {
    temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
}
else if ((magB >= Bpoint9) && (magB < Bpoint10)) {
    temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
}
else if ((magB >= Bpoint10) && (magB < Bpoint11)) {
    temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
}
else if ((magB >= Bpoint11) && (magB < Bpoint12)) {
    temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
}
else if (magB >= Bpoint12) {
    temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
}
}
mu_m[x][y] = (magB+1e-9*mu_o)/(temp_magh+1e-9);
}

// for the center pole of the stator NiFe
for (y=locy5+1;y<=locy6;y++) { // this is the space iteration for hy
   for (x=locx4+1;x<=locx5;x++){

        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y]);

       if (magB < Bpoint1) {
           temp_magh = slope1*magB;
       }
       else if ((magB >= Bpoint1) && (magB < Bpoint2)) {
           temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
       }
       else if ((magB >= Bpoint2) && (magB < Bpoint3)) {
           temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
       }
       else if ((magB >= Bpoint3) && (magB < Bpoint4)) {
           temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
       }
       else if ((magB >= Bpoint4) && (magB < Bpoint5)) {
           temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
       }
       else if ((magB >= Bpoint5) && (magB < Bpoint6)) {
           temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
       }
       else if ((magB >= Bpoint6) && (magB < Bpoint7)) {
           temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
       }
       else if ((magB >= Bpoint7) && (magB < Bpoint8)) {
           temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
       }
       else if ((magB >= Bpoint8) && (magB < Bpoint9)) {
           temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
       }
       else if ((magB >= Bpoint9) && (magB < Bpoint10)) {
           temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
       }
       else if (magB >= Bpoint10) {
           temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
       }
   }
}
else if ((magB >= Bpoint11) && (magB < Bpoint12)){
    temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
} else if (magB >= Bpoint12){
    temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
}

mu_m[x][y] = (magB+le-9*mu_o)/(temp_magh+le-9);

} // for the right half-pole of the stator NiFe
for (y=locy5+1;y<=locy6;y++){ // this is the space iteration for
    for (x=locx8+1;x<=ie_x_n;x++){
        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y]);
        if (magB < Bpoint1){
            temp_magh = slope1*magB;
        } else if ((magB >= Bpoint1) && (magB < Bpoint2)){
            temp_magh = slope2*(magB-Bpoint1) + Hpoint1;
        } else if ((magB >= Bpoint2) && (magB < Bpoint3)){
            temp_magh = slope3*(magB-Bpoint2) + Hpoint2;
        } else if ((magB >= Bpoint3) && (magB < Bpoint4)){
            temp_magh = slope4*(magB-Bpoint3) + Hpoint3;
        } else if ((magB >= Bpoint4) && (magB < Bpoint5)){
            temp_magh = slope5*(magB-Bpoint4) + Hpoint4;
        } else if ((magB >= Bpoint5) && (magB < Bpoint6)){
            temp_magh = slope6*(magB-Bpoint5) + Hpoint5;
        } else if ((magB >= Bpoint6) && (magB < Bpoint7)){
            temp_magh = slope7*(magB-Bpoint6) + Hpoint6;
        } else if ((magB >= Bpoint7) && (magB < Bpoint8)){
            temp_magh = slope8*(magB-Bpoint7) + Hpoint7;
        } else if ((magB >= Bpoint8) && (magB < Bpoint9)){
            temp_magh = slope9*(magB-Bpoint8) + Hpoint8;
        } else if ((magB >= Bpoint9) && (magB < Bpoint10)){
            temp_magh = slope10*(magB-Bpoint9) + Hpoint9;
        } else if ((magB >= Bpoint10) && (magB < Bpoint11)){
            temp_magh = slope11*(magB-Bpoint10) + Hpoint10;
        } else if ((magB >= Bpoint11) && (magB < Bpoint12)){
            temp_magh = slope12*(magB-Bpoint11) + Hpoint11;
        } else if (magB >= Bpoint12){
            temp_magh = slope13*(magB-Bpoint12) + Hpoint12;
        }
        mu_m[x][y] = (magB+le-9*mu_o)/(temp_magh+le-9);
    }
}
// for the wafer NiFe
for (y=locy6+1;y<=locy7;y++){ // this is the space iteration for
    for (x=1;x<=ie_x_n;x++){
        magB = sqrt(bx_m[x][y]*bx_m[x][y] + by_m[x][y]*by_m[x][y])
    }
}
if (magB < Bpoint1wafer){
    temp_magh = slopelwafer*magB;
} else if ((magB >= Bpoint1wafer) && (magB < Bpoint2wafer))
    temp_magh = slope2wafer*(magB-Bpoint1wafer) + Hpoint1wafer;
else if ((magB >= Bpoint2wafer) && (magB < Bpoint3wafer))
    temp_magh = slope3wafer*(magB-Bpoint2wafer) + Hpoint2wafer;
else if ((magB >= Bpoint3wafer) && (magB < Bpoint4wafer))
    temp_magh = slope4wafer*(magB-Bpoint3wafer) + Hpoint3wafer;
else if ((magB >= Bpoint4wafer) && (magB < Bpoint5wafer))
    temp_magh = slope5wafer*(magB-Bpoint4wafer) + Hpoint4wafer;
else if ((magB >= Bpoint5wafer) && (magB < Bpoint6wafer))
    temp_magh = slope6wafer*(magB-Bpoint5wafer) + Hpoint5wafer;
else if ((magB >= Bpoint6wafer) && (magB < Bpoint7wafer))
    temp_magh = slope7wafer*(magB-Bpoint6wafer) + Hpoint6wafer;
else if ((magB >= Bpoint7wafer) && (magB < Bpoint8wafer))
    temp_magh = slope8wafer*(magB-Bpoint7wafer) + Hpoint7wafer;
else if ((magB >= Bpoint8wafer) && (magB < Bpoint9wafer))
    temp_magh = slope9wafer*(magB-Bpoint8wafer) + Hpoint8wafer;
else if ((magB >= Bpoint9wafer) && (magB < Bpoint10wafer))
    temp_magh = slope10wafer*(magB-Bpoint9wafer) + Hpoint9wafer;
else if ((magB >= Bpoint10wafer) && (magB < Bpoint11wafer))
    temp_magh = slope11wafer*(magB-Bpoint10wafer) + Hpoint10wafer;
else if ((magB >= Bpoint11wafer) && (magB < Bpoint12wafer))
    temp_magh = slope12wafer*(magB-Bpoint11wafer) + Hpoint11wafer;
else if (magB >= Bpoint12wafer)
    temp_magh = slope13wafer*(magB-Bpoint12wafer) + Hpoint12wafer;
}

mu_m[x][y] = (magB+1e-9*mu_0)/(temp_magh+1e-9);

for (y=1;y<=ie_y_n;y++) { // this is the space iteration for hy
for (x=1;x<=ie_x_n;x++){
    hx_m[x][y] = bx_m[x][y]/mu_m[x][y];
    hy_m[x][y] = by_m[x][y]/mu_m[x][y];
}

if ((n % when) == 0){ // change with when
    sumofstuff = 0.0;
    for (x=2;x<=ie_x_n-1;x++){
        sumofstuff = sumofstuff + mu_o*hx_m[x][y_gap]*hy_m[x][y_gap]*dx_matrix_m[x][y_gap];
    }
}

printf("Time past: %e Count: %d\n",timepast,torquecount);
torquevector[torquecount] = Reff*Width*poles*sumofstuff;
timevector[torquecount] = timepast;

//////// TEMPERATURE ADDITION ///////////

energyssum = 0.0;
energyssum = 0.0;
for (y=locy1+1;y<=locx2;y++){
    for (x=1;x<=locx9;x++){
        energyssum = energysum + rotor_ss*ez_m[x][y]*ez_m[x][y]*dx_matrix_m[x][y];
    }
}
for (y=locy2+1;y<=locx3;y++){
    for (x=1;x<=locx9;x++){
        energyssum = energyssum + sigma_wafer*ez_m[x][y]*ez_m[x][y]*dx_matrix_m[x][y];
    }
}

// stator addition
// backiron wafer
for (x=1;x<=ie_x_n;x++){
    for (y=locy6+1;y<=locy7;y++){
        energyssum = energysum + sigma_ss*ez_m[x][y]*dx_matrix_m[x][y]*dy_matrix_m[x][y];
    }
}

// teeth section
for (y=locy4+1;y<=locy5;y++){
    for (x=1;x<=locx2;x++){
        energyssum = energysum + sigma_ss*ez_m[x][y]*dx_matrix_m[x][y]*dy_matrix_m[x][y];
    }
}
for (x=locx3+1;x<=locx6;x++){
    for (y=locy6+1;x<=locx9;x++){
        energyssum = energysum + sigma_ss*ez_m[x][y]*dx_matrix_m[x][y]*dy_matrix_m[x][y];
    }
}
for (x=locx7+1;x<=locx9;x++){
    for (y=locy4+1;x<=locx6;x++){
        energyssum = energysum + sigma_ss*ez_m[x][y]*dx_matrix_m[x][y]*dy_matrix_m[x][y];
    }
}
for (y=locy5+1;x<=locy6;y++){
for (x=1; x <= locx1; x++) {
    *ez m[x][y] * dx matrix m[x][y] * dy matrix m[x][y];
}

end of stator addition

**** END OF TEMPERATURE ADDITION ***************

/channel of the time looping

} // end of the time looping

// at the end of the simulation, change the worksheet file
// to reflect the fact that the corresponding simulation
// terminated successfully

if (((worksheet = fopen(filepathworksheet, "r+")) != NULL)) {
    // attempt to read a necessary amount of characters from file
    foundworksheetspot = NULL;
    linecount = 0;
    while (!foundworksheetspot) {

}
if (fread(firstlinechar, sizeof(char), 1, worksheet))
{
    if (firstlinechar[0] == '0')
    {
        fseek(worksheet, -1, SEEK_CUR);
        firstlinechar[0] = '1';
        fwrite(firstlinechar, sizeof(char), 1, worksheet);
        foundworksheetspot = 1;
        //////////////////////////////////////////////////////////////////////////
        // write the computed data to disk
        //////////////////////////////////////////////////////////////////////////
        printf("Now writing computed data to disk! ...
");
        strcpy(filepathlog, esasfilepath);
        if ((logsheet = fopen(strcat(filepathlog, "/testlog.txt"), "a")) != NULL)
        {
            time(&ltime);
            fprintf(logsheet, "Finished computing for Reff:%e Ls:%e current:%e freq:%e deltaT:%e Gap:%e on %s
", Reff, Ls, Jext_amp, freq, deltaT, Gap, ctime(&ltime));
            fclose(logsheet);
        }
        else
        {
            printf("Houston, we have a log file opening problem!\n");
        }
        // determine the name of the file to write to
        strcat(resultsfile, "/Reff ");
        strcat(resultsfile, ecvt(Reff*1e6,4, &decimal, &sign));
        strcat(resultsfile, " Ls ");
        strcat(resultsfile, ecvt(Ls*1e6,2, &decimal, &sign));
        strcat(resultsfile, " freq ");
        strcat(resultsfile, ecvt(freq*le-3,3, &decimal, &sign));
        strcat(resultsfile, " curr ");
        strcat(resultsfile, ecvt(Jext_amp,1, &decimal, &sign));
        strcat(resultsfile, " deltaT ");
        strcat(resultsfile, ecvt(deltaT,3, &decimal, &sign));
        strcat(resultsfile, " Gap ");
        strcat(resultsfile, ecvt(Gap*le6,2, &decimal, &sign));
        strcat(resultsfile, " ss ");
        strcat(resultsfile, ecvt(sigma_ss,2, &decimal, &sign));
        strcat(resultsfile, ".m ");
        if ((resultssheet = fopen(resultsfile, "w")) != NULL)
        {
            fprintf(resultssheet, "Reff = %e ;\n", Reff);
            fprintf(resultssheet, "Ls = %e ;\n", Ls);
            fprintf(resultssheet, "current = %e ;\n", Jext_amp);
            fprintf(resultssheet, "deltaT = %e ;\n", deltaT);
            fprintf(resultssheet, "Gap = %e ;\n", Gap);
            fprintf(resultssheet, "freq = %e ;\n", freq);
            fprintf(resultssheet, "stator nife cond = %e ;\n", sigma_ss);
            fprintf(resultssheet, "rotor nife cond = %e ;\n", rotor_ss);
            fprintf(resultssheet, "time torque = ");
            for (ii=1; ii<=torquecount; ii++)
            {
                fprintf(resultssheet, "%e %e %e %e; ... \n", timevector[ii],
energyvector[ii], energyvector_ss[ii]);
            }
            fprintf(resultssheet, "]\n");
            fclose(resultssheet);
        }
        else
        {
            printf("Houston, we have a log file opening problem!\n");
        }
    }
}
} else
{
    if (fgets(line, 200, worksheet) == NULL)
    {
        foundworksheetspot = 1;
        printf("End of file reached in an attempt to modify it!!!");
    }
}
}
else
{
    foundworksheetspot = 1;
    printf("Error in reading the first line character!! \n");
}
fclose(worksheet);
}
else
{
    printf("Houston, we have a file opening problem!");
}

// now, you can free the memory
free dmatrix(ez m,1,ie x n, 1, ie y n);
free dmatrix(hx m,1,ie x n, 1, ie y n);
free dmatrix(hy m,1,ie x n, 1, ie y n);
free dmatrix(bx m,1,ie x n, 1, ie y n);
free dmatrix(by m,1,ie x n, 1, ie y n);
free dmatrix(mu m,1,ie x n, 1, ie y n);
free dmatrix(xe z m,1,ie x n, 1, ie y n);
free dmatrix(ca matrix m,1,ie x n, 1, ie y n);
free dmatrix(cb x matrix m,1,ie x n, 1, ie y n);
free dmatrix(cb y matrix m,1,ie x n, 1, ie y n);
free dmatrix(cc matrix m,1,ie x n, 1, ie y n);
free dmatrix(db x matrix m,1,ie x n, 1, ie y n);
free dmatrix(db y matrix m,1,ie x n, 1, ie y n);
free dmatrix(Jz1,1,ie x n, 1, ie y n);
free dmatrix(Jz2,1,ie x n, 1, ie y n);
free dmatrix(dx matrix m,1,ie x n, 1, ie y n);
free dmatrix(dy matrix m,1,ie x n, 1, ie y n);
free dmatrix(eps matrix m,1,ie x n, 1, ie y n);
free dmatrix(sigma matrix m,1,ie x n, 1, ie_y_n);
free dvect or(torquevector,1,torquepoints);
free dvect or(timevector,1,torquepoints);
free dvect or(energyvector,1,torquepoints);
free_dvect or(energyvector ss,1,torquepoints);

waitabit();

} // close the big if on no go
else   // if the worksheet is complete
{
    printf("No more simulations left to do! \n");
    no_go = 1;
}
double findsigmaT(double deltatemp)
{
    double cu_slope, cu_intercept;
    cu_slope = 7.0477e-11;
    cu_intercept = -3.8506e-9;
    // the following numbers according to Sang Soo's measurements
    // cu_slope = 9.5e-11;
    // cu_intercept = -7.5e-9;
    return (1/(cu_slope * (deltatemp + 300.0) + cu_intercept));
}

void waitabit(void)
{
    time_t entertime, timenow;
    time(&entertime);
    time(&timenow);
    while (difftime(timenow, entertime) < 2){
        time(&timenow);
    }
}

void waitaminute(void)
{
    time_t entertime, timenow;
    time(&entertime);
    time(&timenow);
    while (difftime(timenow, entertime) < 60){
        time(&timenow);
    }
}