Magnetic Equivalent Circuit Toolbox

User's Manual
Version 3.2, January 2014

Coordinated By:

S.D. Sudhoff (sudhoff@purdue.edu)
School of Electrical and Computer Engineering
Purdue University

Table of Contents

1. Introduction 3
2. Installation 5
3. Data Structures 6
4. Overview of Core Routines 8
5. Core Routines 9
6. Example 1 16
7. Example 2 22
8. Supplemental Routines 26
9. Acknowledgements 26
10. References 27
11. License 27

1. Introduction

The Magnetic Equivalent Circuit (MEC) Toolbox facilitates analysis of electromagnetic devices such as inductors, transformers, and electric machinery. This is Version 3.2 of the toolbox.

In the MEC Toolbox 1.0, a nonlinear magnetic equivalent circuit was solved using a mesh-based analysis with permeability represented as a function of flux density. The MEC Toolbox 2.0 still used this capability. However, it added the provision to represent part of the circuit with nodal equations. This is advantageous in representing components in which the magnetic circuit change undergoes structural changes as some degree of freedom. Examples of such components include linear and rotating electric machinery. In this situation, is often much more convenient to represent the part of the circuit with changing geometry with nodal rather than mesh equations. Note that since this part of the circuit corresponds to air, it is magnetically linear. Thus, in the MEC Toolbox 2.0, the nodal portion of the circuit is assumed to be magnetically linear. The second change of the toolbox is that the definition of the standard branch has been changed. Users of MEC Toolbox 1.0 should consult with the new standard branch definitions when converting code. In many cases, MEC Toolbox 1.0 code will run without change in MEC Toolbox 2.0. However, if flux sources were used in the branches, a sign change will be required. In addition, the arrangement of the elements within the standard branch is different.

The MEC Toolbox 2.0 and later are based on the standard magnetic branch shown in Fig. 1. There are two types of branches – mesh branches which are processed as part of the mesh equations and nodal branches which are represented through nodal equations.

As can be seen, the mesh and nodal branches are nearly identical. The flux into the mesh/nodal branch, flux in the reluctance/permeance, and MMF drop across the branch are designated $\Phi_{mb,i}$ / $\Phi_{nb,i}$, $\Phi_{mR,i}$ / $\Phi_{nR,i}$, and $F_{mb,i}$ / $F_{nb,i}$ respectively, where *i* denotes the branch number, and mesh and nodal branches are numbered separately (thus there may be a mesh branch 3 and a nodal branch 3). MMF and flux sources associated with each branch are designated $F_{s,i}$ and $\Phi_{s,i}$, respectively. Note that the flux source has the opposite sign convention as in MEC Toolbox 1.0.

One difference between that two branches, is that the mesh branch includes a reluctance (which may be set to zero) while the nodal branch is based on a permeance (which may be set to zero). The fact that the permeance can be set to zero makes the nodal branch useful in representing devices in which the structure of the problem is variable – as in the case of rotating electric machinery. It should be noted that the nodal branch is limited to magnetic linear components as it is primarily designed to represent air gap elements.

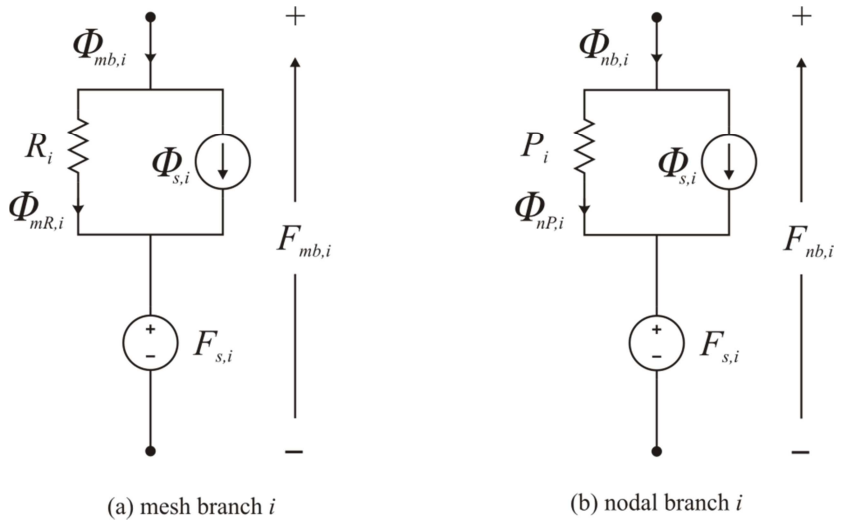

Figure 1. Standard Magnetic Branches

The use of the MEC Toolbox is based on establishing a set of explicit meshes fluxes (those mesh fluxes associated with mesh equations), and explicit nodes (those nodes whose potential are established with nodal equations). Once these are decided upon, a list of explicit mesh branches, explicit node branches, and explicit nodes are established. The explicit node list is used to describe how the explicit nodes are related to the explicit meshes. Then the linear or non-linear magnetic circuit is solved.

In Version 3 of the toolbox, there have been two enhancements. First, this version of the toolbox establishes a material list, so that adding branches based on magnetically nonlinear materials is simpler. Version 3 of the toolbox also introduces a testing function which is useful for debugging. Version 3 retains the mixed mesh/nodal analysis capability and standard branch definitions used in Version 2.

The theoretical development behind the MEC toolbox is set forth in [1], which includes many examples of the use of the toolbox for the analysis of a UI core inductor, and EI core electromagnet, and a single-phase transformer.

The organization of this manual is as follows. First, in Section 2, the installation of the MEC Toolbox is discussed. This is followed by a discussion of the MEC data structure in Section 3. An overview of the core routines of the toolbox is given in Section 4; this is followed by a detailed discussion of the routines in Section 5. Sections 6 and 7 provide examples in the use of the MEC toolbox. Supplemental functions are listed in Section 8. Acknowledgments, references, and license information appear in Sections 8, 9, and 10, respectively.

2. Installation

The MEC Toolbox can be used in MATLAB by adding the location of the toolbox to the MATLAB path. It should be noted that the MEC Toolbox consists of three separate folders, namely *MEC Core*, *MEC Supplemental*, *MEC Examples*, and *MEC Licenses and Documentation*. The *MEC Core* folder contains the core functions of the MEC Toolbox, while *MEC Supplemental* folder contains additional routines that will help users to build and solve the MEC for several inductor and transformer arrangements. The folder *MEC Examples* includes the MATLAB scripts used in Section 8 and 9 of this manual. The folder *Licenses and Documentation* includes license information, this manual, and the example discussed below. The *MEC Core* and *MEC Supplemental* directories should be added to the MATLAB path using the 'path' command. Information on this command can be obtained by typing '`help path`' at the MATLAB prompt.

3. Data Structures

The MEC toolbox places most information into a data structure often denoted MEC. This is a structure of structures. The fields are as follows:

`MEC.Nem`	Number of explicit meshes
`MEC.Nemb`	Number of explicit mesh branches
`MEC.Nen`	Number of explicit nodes
`MEC.Nenb`	Number of explicit node branches
`MEC.mxit`	Maximum number of iterations
`MEC.rec`	Relative error criteria
`MEC.aec`	Absolute error criteria
`MEC.mat`	Non-linear magnetic materials structure
`MEC.mat.muf`	Cell array with pointer to permeability functions (mu and the partial of mu with respect to B as functions of B)
`MEC.mat.mup`	Cell array of structures with parameters for the permeability function
`MEC.mb`	Mesh branch structure. Made of vectors associated with the explicit mesh branch list (members are `MEC.Nemb` by 1)
`MEC.mb.t`	Mesh branch types. 1=linear; 2=non-linear
`MEC.mb.R`	Mesh branch reluctances (1/H)
`MEC.mb.Fs`	Mesh branch MMF sources (At)
`MEC.mb.PHIs`	Mesh branch flux sources (Wb)
`MEC.mb.A`	Mesh branch magnetic cross sections (m^2)
`MEC.mb.l`	Mesh branch magnetic lengths (m)
`MEC.mb.pmub`	Mesh branch pointers to permeability functions
`MEC.mb.MMP`	Mesh branch magnetic material parameters
`MEC.mb.PHIe`	Estimated flux into branches (Wb)
`MEC.mb.pml`	Cell vector with positive mesh list for branch
`MEC.mb.nml`	Cell vector with negative mesh list for branch
`MEC.mb.mi`	Material index for branch (irrelevant if linear material)
`MEC.nb`	Node branch structure. Made up of vectors associated with the explicit node branch list (members are `MEC.Nenb` by 1)
`MEC.nb.pn`	Node branch positive node
`MEC.nb.nn`	Node branch negative node
`MEC.nb.P`	Node branch permeance (H)
`MEC.nb.Fs`	Node branch MMF source (A)
`MEC.nb.PHIs`	Node branch flux source (Wb)
`MEC.psb`	Pole symmetric branch structure. Made up of vectors associated with the explicit node pole symmetric list (members are `MEC.Npsb` by 1)

`MEC.psb.pn1`	Pole symmetric branch positive node 1
`MEC.psb.pn2`	Pole symmetric branch positive node 2
`MEC.psb.P`	Pole symmetric branch permeance (H)
`MEC.psb.Fs`	Pole symmetric branch MMF source (A)
`MEC.psb.PHIs`	Pole symmetric branch flux source (Wb)

`MEC.nl`	Explicit node list structure. Members are (Nen by 1)
`MEC.nl.pml`	Positive mesh list
`MEC.nl.nml`	Negative mesh list

4. Overview of Core Routines

A listing and description of the core routines of the MEC toolbox follows. This will be followed by a detailed description of each routine in Section 5 of this manual.

`mec_enode.m`	Adds an explicit node to a magnetic equivalent circuit
`mec_field_energy.m`	Compute the field energy stored in a magnetic equivalent circuit
`mec_init.m`	Initializes the MEC data structure.
`mec_lmb.m`	Adds a magnetically linear mesh branch
`mec_lnb.m`	Adds a magnetically linear nodal branch
`mec_lp_branch.m`	Adds a magnetically linear permeance to the mesh
`mec_lr_branch.m`	Adds a magnetically linear reluctance to the mesh
`mec_nl_branch.m`	Adds a magnetically nonlinear branch to the mesh
`mec_nl_material.m`	Adds a magnetically nonlinear material to the materials list
`mec_nls_branch.m`	Adds a magnetically non-linear source branch to the mesh
`mec_psb.m`	Adds a pole symmetric branch to the mesh
`mec_solve.m`	Solves the magnetic equivalent circuit
`mec_test.m`	Tests the MEC structure as debugging aid.

5. Core Routines

`mec_enode.m`

Purpose:
Adds an explicit node to a magnetic equivalent circuit.

Function call:
```
MEC = mec_enode(MEC,n,ml)
```

Inputs:

`MEC`	MEC data structure
`n`	node number to add
`ml`	mesh list for node

Outputs:

MEC	updated MEC data structure

`mec_field_energy.m`

Purpose:
Compute the field energy stored in a magnetic equivalent circuit

Function call:
```
Wf = mec_field_energy(PHImb,Fmb,PHInb,Fnb,MEC)
```

Inputs:

`MEC`	MEC data structure (see `mec_init` for description of fields)
`PHImb`	Vector of explicit mesh branch fluxes (`MEC.Nemb` by 1) (Wb)
`Fmb`	Vector of explicit mesh branch MMF drops (`MEC.Nemb` by 1) (At)
`Fnb`	Vector of node branch MMF drops (At)
`PHInb`	Vector of fluxes into nodal branches (Wb)

Outputs:

Wf	Field energy (J)

mec_init.m

Purpose:
This routine is used for setting up the data structure required to do the analysis based on the number of mesh loops and branches of the equivalent circuit.

Function call:
```
MEC = mec_init(Nem,Nemb)
MEC = mec_init(Nem,Nemb,Nnlm)
MEC = mec_init(Nem,Nemb,Nen,Nenb,Nnlm)
MEC = mec_init(Nem,Nemb,Nen,Nenb,Npsb,Nnlm)
```

Inputs:

Nem	Number explicit meshes
Nemb	Number of explicit mesh branches
Nnlm	Number of nonlinear magnetic materials
Nen	Number of explicit nodes
Nenb	Number of explicit nodal branches
Npsb	Number of pole symmetric branches

Outputs:

MEC–	Initialized data structure for the MEC

mec_lmb.m

Purpose:
This routine adds a magnetically linear mesh branch to a magnetic equivalent circuit.

Function call:
```
MEC = mec_lmb(MEC,bn,R,Fs,PHIs,ml)
```

Inputs:

MEC	MEC data structure
Bn	branch number. This should be unique
R	branch reluctance (1/H)
Fs	branch MMF source (A)
PHIs	branch flux source (Wb)
Ml	mesh list for the branch

Outputs:

MEC	Updated MEC data structure

mec_lnb.m

Purpose:
This routine adds a magnetically linear nodal branch to a magnetic equivalent circuit.

Function call:
```
MEC = mec_lnb(MEC,n,pn,nn,P,Fs,PHIs)
```

Inputs:

MEC	MEC data structure
n	branch number. This should be unique for nodal branches
pn	positive explicit node for branch
nn	negative explicit node for branch
P	branch permeance (H)
Fs	branch MMF source (A)
PHIs	branch flux source (Wb)

Outputs:

MEC	Updated MEC data structure

mec_lp_branch.m

Purpose:
This routine adds a linear permeance to the explicit mesh of a magnetic equivalent circuit.

Function Call:
```
MEC = mec_lp_branch(MEC,bn,P,ml)
```

Inputs:

MEC	MEC data structure
bn	explicit mesh branch number. This should be unique.
P	branch permeance (H)
Ml	branch mesh list

Outputs:

MEC	Updated MEC data structure

mec_lr_branch.m

Purpose:
This routine adds a branch with a linear reluctance to the explicit mesh list of a magnetic equivalent circuit.

Function Call:
```
MEC = mec_lr_branch(MEC,bn,R,ml)
```

Inputs:

MEC	MEC data structure
bn	explicit mesh branch number. This should be unique.
R	branch reluctance (1/H)
ml	branch mesh list

Outputs:

MEC	Updated MEC data structure

mec_nl_branch.m

Purpose:
This routine adds a non-linear explicit mesh branch to the mesh of a magnetic equivalent circuit.

Inputs:

MEC	MEC data structure
bn	explicit mesh branch number. This should be unique.
A	effective cross sectional area of the branch (m^2)
l	length of the branch (m)
mi	material index
ml	branch mesh list
PHIe	estimate for the flux into branch (Wb)

Outputs:

MEC	Updated MEC data structure

mec_nl_material.m

Purpose:
Adds a non-linear material type to a magnetic equivalent circuit.

Function Call:
```
MEC = mec_nl_material(MEC,mi,muf,mup)
```

Inputs:

MEC	MEC data structure (see mec_init for documentation)
mi	material index. Should be an integer from 1 to the number of nonlinear materials in the circuit
pmuf	pointer to a function which calculated permeability and the derivative of the permeability with respect to B as a function of B
mup	parameters of permeability function

Outputs:

MEC	Updated MEC data structure

mec_nls_branch.m

Purpose:
This routine adds a non-linear explicit mesh branch with sources to the mesh of a magnetic equivalent circuit.

Inputs:

MEC	MEC data structure
bn	explicit mesh branch number. This should be unique.
A	effective cross sectional area of the branch (m^2)
l	length of the branch (m)
mi	material index
Fs	branch MMF source (A)
PHIs	branch flux source (Wb)
ml	branch mesh list
PHIe	estimate for the flux into branch (Wb)

Outputs:

MEC	Updated MEC data structure

mec_psb.m

Purpose:
Adds a pole symmetric branch to a magnetic equivalent circuit. This is used when, for example, using one pole of an electric machine to represent an entire electric machine. Note this routine is still in testing.

Function Call:
```
MEC = mec_psb(MEC,bn,pn1,pn2,P,Fs,PHIs)
```

Inputs:

MEC	MEC data structure
bn	branch number. This should be unique for pole symmetric branches.
pn1	positive node 1 for branch
pn2	positive node 2 for branch
P	branch permeance (H)
F	branch MMF source (A)
PHIs	branch flux source (Wb)

Outputs:

MEC	Updated MEC data structure

mec_solve.m

Purpose:
Solves the magnetic equivalent circuit.

Function Call:
```
[PHIm,PHImb,PHImR,Fmb,c] = mec_solve(MEC)
[PHIm,PHImb,PHImR,Fmb,Fn,Fnb,PHInb,PHInP,c] = mec_solve(MEC)
[Fn,Fnb,PHInb,c] = mec_solve(MEC)
```

Inputs:

MEC	MEC data structure

Outputs:

PHIm	Vector of explicit mesh fluxes (MEC.Nem by 1) (Wb)
PHImb	Vector of explicit mesh branch fluxes (MEC.Nemb by 1) (Wb)
PHImR	Vector of explicit mesh reluctance fluxes (MEC.Nemb by 1)(Wb)
Fmb	Vector of explicit mesh branch MMF drops (MEC.Nemb by 1) (At)
Fn	Vector of explicit node potentials (MEC.Nen by 1) (At)
Fnb	Vector of node branch MMF drops (At)
PHInb	Vector of fluxes into nodal branches (Wb)
PHInP	Vector of fluxes into nodal branch permeances (Wb)
c	Convergence. 1 if analysis converges; 0 otherwise

mec_test.m

Purpose:
Performs some checks on the MEC data structure as an aide to debugging.

Inputs:

`MEC`	MEC data structure

Outputs:

`ts`	Test status. One if passed, zero if not. Text report is also provided.

6. Example 1

The use of the MEC toolbox will be demonstrated in this section. For this first example, the MEC will be configured for mesh analysis. The use of mixed mesh/nodal analysis is considered in Example 2.

Consider the following EI core inductor shown in Fig. 2. The specific dimensions of the EI core used herein are listed on Table 1.

Table 1- Dimensions of Experimental EI Core

Parameter	Value (mm)	Parameter	Value (mm)
w_e	28.7	w_b	27.3
w_c	57.4	w_i	30.5
w_s	37.0	G	2.96
d_s	48.8	D	119.8

Therein, w_i is the width of the I-core, g is the air gap length, d_s is the depth of the slot, w_b is the width of the bottom part of the E-core, w_e is the width of the end post of the E-core, w_s, is the width of the slot, w_c is the width of the center post of the E-core, and d denotes the depth of the EI core into the page. The core material used was a ferrite material grade 3C90.

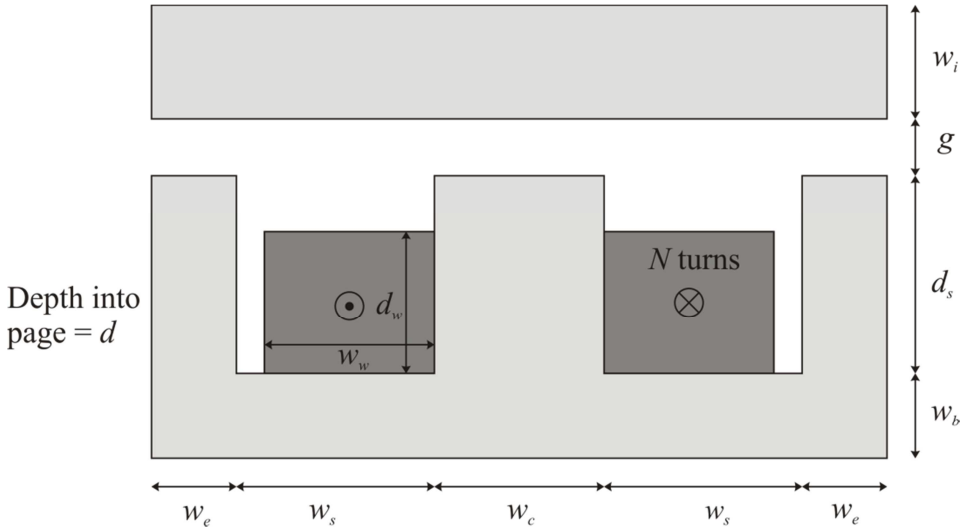

Figure 2. EI Core Inductor Structure

The MEC model of the EI core used herein is shown in Fig. 3. It should be noted here that this is a very simple MEC of the EI core designed to illustrate the use of the MEC toolbox. In this analysis, fringing and leakage fluxes are not considered. A comprehensive MEC of the EI core and the corresponding expression for each permeance have been set forth in [**Error! Reference source not found.**].

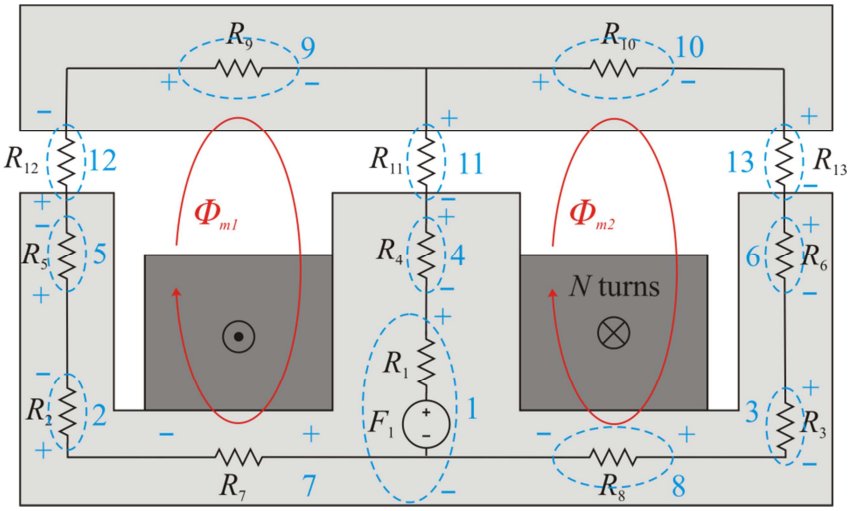

Figure 3. Mesh-based EI Core MEC

In this analysis, thirteen branches are chosen to describe the magnetic elements of the EI core. Since the structure of the device is constant, the circuit is modeled exclusively using mesh analysis in this example. In Fig. 3, each branch of the MEC is indicated with dashed-blue circles along with the direction of the MMF drop in each branch. The direction of the MMF drop may be arbitrarily chosen. The first ten branches are associated with the magnetic material and the last three branches are associated with the air gap.

The branch numbers can be chosen arbitrarily, however the corresponding number for each branch must be unique. In other words, two separate branches may not share the same branch number. For branch 1, 4, and 11, the direction of the MMF drop is chosen such that the flux flowing through the permeance is positively coupled with the mesh Φ_{m1} and negatively coupled with the mesh Φ_{m2}. For branch 2, 5, 7, 9, and 12, the direction of the MMF drop is chosen such that each permeance is positively coupled with the mesh Φ_{m1}. Similarly for branch 3, 6, 8, 10, and 13, the direction of the MMF drop is such that each permeance is positively coupled with mesh Φ_{m2}.

As mentioned in the previous section, the MEC toolbox is developed using a mesh-based analysis. It is interesting to note that the computational performance of the mesh-based MEC far exceeds the nodal-based MEC for nonlinear operating conditions [**Error! Reference source not found.**]. Each mesh corresponds to a possible flux path in the MEC where the direction of each flux can be chosen arbitrarily. For this particular analysis, two meshes (Φ_{m1} and Φ_{m2}) along with their directions are indicated with solid-red lines.

Thus, there are thirteen branches and two meshes in this MEC model of the EI core. From [**Error! Reference source not found.**], the MMF source is given as $F_1 = Ni$, where N is the number of wire turn, wound around the center post, and i is the current through the winding. For the study conducted herein, N has been set to 40 turns and the current i varies from 0 to 100 A.

The script (less header and footer comments) to perform the MEC analysis of the EI core in Fig. 3 is shown below. This script is named `example1.m` and is located in the `MEC Examples` folder of the `MEC 3.2` directory.

```
% Initialize
clear all; close all; clc

% Define some handy numbers
mm = 1e-3;
mu0 = 4*pi*10^-7;

% Specify the dimensions of the EI Core based on
% J.Cale, S.D.Sudhoff, and L.Tan, "Accurately Modeling EI Core Inductors
% Using a High-Fidelity Magnetic Equivalent Circuit Approach",
% IEEE Transactions on Magnetics, Vol.42, No.1, January 2006
we = 28.7 * mm;
wc = 57.4 * mm;
ws = 37.0 * mm;
ds = 48.8 * mm;
wb = 27.3 * mm;
wi = 30.5 * mm;
g  = 2.96 * mm;
d  = 119.8 * mm;
N  = 40;

% Curent range of interest
i  = linspace(0,100,100);
Npoints = length(i);

% Determine the MMF source on the circuit
F1 = N*i;

% Calculate the permeances on the air gap
R11 = g/(mu0*wc*d);
R12 = g/(mu0*we*d);
R13 = R12;

% 3C90 material parameters to calculate permeability as a function of B
M3C90.mur = 22340.9259;
M3C90.muB.a   = [  1.1542      0.049742    0.049644    0.041155];
M3C90.muB.b   = [431.1763       2.29503    15.04824    74.28908];
M3C90.muB.g   = [  0.4742        2.7955     0.59862     0.43996];
M3C90.muB.d   = M3C90.muB.a./M3C90.muB.b;
M3C90.muB.t   = exp(-M3C90.muB.b.*M3C90.muB.g);
M3C90.muB.h   = M3C90.muB.a.*M3C90.muB.t;
M3C90.muB.e   = M3C90.muB.t./(M3C90.muB.t+1);
M3C90.muB.z   = 1./(M3C90.muB.t+1);

% Initial flux linkage and convergence
lambda=NaN(Npoints,1);
c=NaN(Npoints,1);

% Initialize the MEC
MEC = mec_init(2,13,1);

% Establish a list of non-linear materials
MEC = mec_nl_material(MEC,1,@muB,M3C90);

% Branches
MEC= mec_nls_branch(MEC,1,wc*d,(wb/2+ds/2),1,0.0,0.0,[1 -2],0.0);  % MB 1
MEC= mec_nl_branch(MEC,2,we*d,(wb/2+ds/2),1,[1],0.0);              % MB 2
```

```
72    MEC= mec_nl_branch(MEC,3,we*d,(wb/2+ds/2),1,[2],0.0);              % MB 3
73    MEC= mec_nl_branch(MEC,4,wc*d,ds/2,1,[1 -2],0.0);                  % MB 4
74    MEC= mec_nl_branch(MEC,5,we*d,ds/2,1,[1],0.0);                     % MB 5
75    MEC= mec_nl_branch(MEC,6,we*d,ds/2,1,[2],0.0);                     % MB 6
76    MEC= mec_nl_branch(MEC,7,wb*d,(wc/2+ws+we/2),1,[1],0.0);           % MB 7
77    MEC= mec_nl_branch(MEC,8,wb*d,(wc/2+ws+we/2),1,[2],0.0);           % MB 8
78    MEC= mec_nl_branch(MEC,9,wi*d,(wc/2+ws+we/2),1,[1],0.0);           % MB 9
79    MEC= mec_nl_branch(MEC,10,wi*d,(wc/2+ws+we/2),1,[2],0.0);          % MB10
80    MEC= mec_lr_branch(MEC,11,R11,[1 -2]);                             % MB11
81    MEC= mec_lr_branch(MEC,12,R12,[1]);                                % MB12
82    MEC= mec_lr_branch(MEC,13,R13,[2]);                                % MB13
83
84    % Loop over current values
85    for n = 1:Npoints
86
87       % Assign MMF
88       MEC.mb.Fs(1)=F1(n);
89
90       % Solve the MEC
91       [Pm,Pb,PR,Fb,converge] = mec_solve(MEC);
92
93       % Compute total flux linkage
94       lambda(n) = -PR(1)*N;
95
96       % Copy convergence
97       c(n)=converge;
98
99    end
100
101   % Generate the lambda-vs-current curve
102   if (min(c)<1)
103      error('Solution Did Not Converge');
104   end
105
106   % Plot flux linkage versus current
107   plot(i,lambda)
108   xlabel('i, (A)')
109   ylabel('\lambda, (Vs)')
110   grid on
111   axis([0 100 0 0.2]);
```

As can be seen, line 16 clears the workspace, and lines 18-20 define some useful constants. The dimensions of the EI core as assigned in lines 26-34. Lines 36 – 41 of the code concern the MMF source of the MEC. As mentioned previously, the MMF source is given as $F_1 = Ni$, where N is the number of turns, and i is the current in the winding, which is varied linearly from 0 A to 100 A with 100 points.

Line 43 – 46 of the code calculate the linear reluctances in the air gap of the EI core, denoted as $R_{11} - R_{13}$ in Fig. 3. R_{11} is the permeance in the air gap on top of the center post. This quantity can be calculated

$$R_{11} = \frac{g}{\mu_0 w_c d} \tag{1}$$

where $\mu_0 = 4\pi 10^{-7}$ H/m. Similarly for R_{12} and R_{13},

$$R_{12} = R_{13} = \frac{g}{\mu_0 w_e d} \tag{1}$$

Lines 48 – 57 of the code define the structure of parameters associated with the material of the EI core. In this particular example, the material of the EI core is ferrite grade 3C90. These parameters are needed for evaluating the permeability function and its derivative as a function of flux density *B*. It should be noted that this structure is associated with a user defined function which is used to represent the permeability as a function of *B*. Any function may be used – the function itself, and thus the data needed by that function, is not part of the MEC Toolbox. The user provided function should have two inputs and two outputs. The first input should be a structure containing all needed parameters; the second input should be the flux density *B*. The two outputs of the function should be the permeability, and the derivative of the permeability with respect to *B*. For this example, the function `muB.m` calculates these quantities. It is in the `MEC Examples` directory. The mathematics of this function are described in [1] and [6].

Lines 59-61 initialize a flux linkage vector and a vector whose elements will indicate convergence. The elements of both these vectors are initialized to not-a-number.

Lines 63 – 99 of the code build and solve the MEC. It should be noted that the 'for loop' is intended for the purpose of generating the *λ–i* curve by describing and evaluating the MEC at every operating point of the current vector *i*.

On line 64, the MEC is initialized by first calling the `mec_init` function. From previous analysis, it has been determined that the circuit has two meshes, thirteen branches, and one nonlinear material. Other forms of this routine are called when the nodal analysis is also used; this will be illustrated in a second example in the following section.

Line 66-67 of the code defines the magnetic material 1, by passing a pointer to the permeability function and a structure of the function parameters.

Lines 70 processes the first branch of the MEC using the `mec_nls_branch` routine. This branch consists of a nonlinear material with an MMF source and no flux source. The first input of this function is the MEC structure which was initialized by the `mec_init` routine. The output of this function is an updated structure. The next input "1" designates that the information is for branch 1. The next argument, "`wc*d`", is the magnetic cross section and "(`wb/2+ds/2`)" is the magnetic length. The argument '1' denotes the material type, which was defined in line 67. The next arguments, "0.0" and "0.0" are the MMF source and flux source associated with the branch. The MMF source is temporarily set to zero; we will replace this value later on. The argument "[1 -2]" is the mesh list, which in this case indicates that mesh flux 1 enters the positive node of the branch and mesh flux 2 enters the negative node of the branch (hence the – sign). The final argument "0" is an estimate of the flux in the branch. If the flux can be estimated, that estimate can be used to improve speed of convergence.

Lines 71-79 process branches 2-10 using the `mec_nl_branch` routine. This routine is similar to `mec_nls_branch` but is for branches with no MMF or flux sources. Next branches 11-13

are processed using the `mec_rl_branch` in lines 80-82. This routine is for branches which can be represented by linear reluctances. The input arguments to this function are the MEC structure, the branch number, the reluctance of the branch, and the mesh list. The output argument is an updated MEC structure.

The solution to the MEC at each current level occurs in lines 85-99. Each iteration of the for loop corresponds to a different current. In line 88, the MMF associated with branch 1 is assigned to the MMF source component of the mesh branch list. Recall that the MMF source is associated with branch 1. The numerical solution to the MEC is obtained from the call to `mec_solve` in line 91. This routine returns the vector of mesh fluxes (`Pm`), the vector of branch fluxes (`Pb`), the vector of reluctance fluxes (`BR`), the vector of nodal MMFs (`Fb`) and a vector of convergence status (`converge`). The flux linkage associated with the winding is the number of turns times the flux flowing through the reluctance of branch 1 with a change of sign. This is accomplished in line 94. The convergence is stored as an element of a vector in line 97.

In lines 101-104, the convergence of the analysis at all operating points is tested. The final lines of active code plot the flux-linkage versus current characteristics. This is done in lines 106-111. The results are shown in Fig. 4.

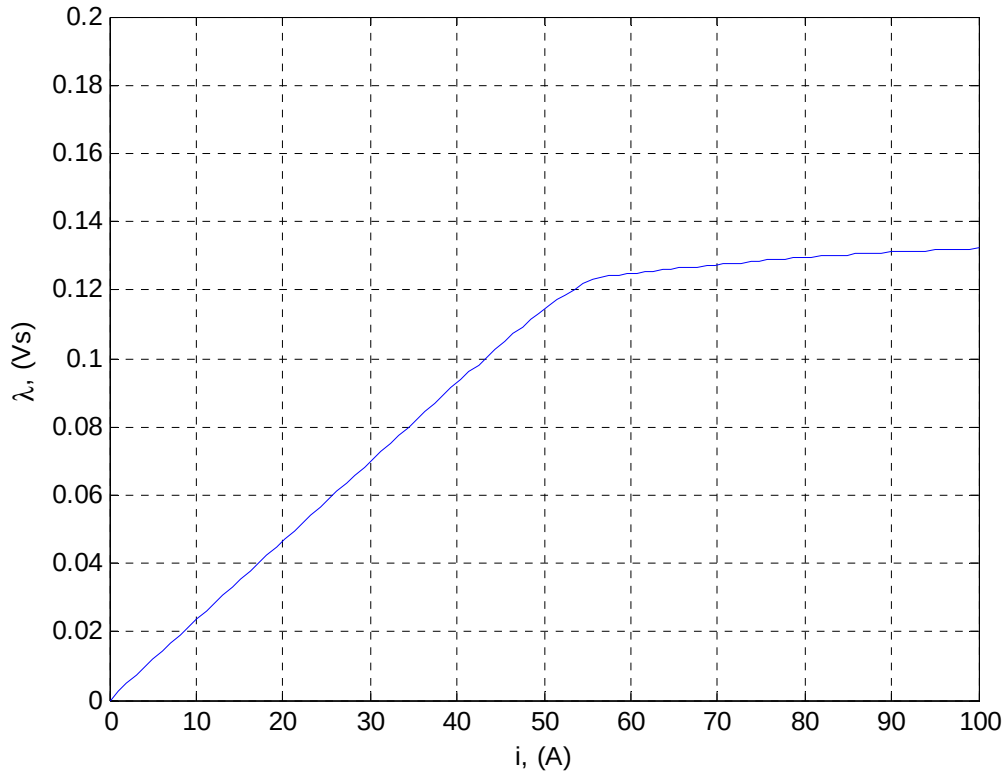

Figure 4. λ–i Characteristics of the EI Core Inductor

7. Example 2

In this example, we will again consider the EI core inductor shown in Fig. 2, and with parameters listed in Table 1. However, in this case, we will use a mixed mesh/nodal formulation. For this problem, the pure mesh formulation in simpler and better. However, a mixed mesh/nodal formulation is considered here to illustrate this type of approach on a simple problem.

The MEC used is illustrated in Fig. 5. Therein, there are four meshes fluxes, Φ_{m1}, Φ_{m2}, Φ_{m3}, and Φ_{m4}. Observe that these mesh loops are open in that they emanate and terminate on nodes. The explicit nodes in the figure are F_{n0}, F_{n1}, F_{n2}, F_{n3}, F_{n4}, F_{n5} and F_{n6}. These are termed explicit nodes because they are explicitly labeled; clearly Fig. 5 has other nodes, but the non-labeled nodes are considered implicit. Note that by definition F_{n0} is at zero potential. In this equivalent circuit, the first 10 branches are taken as mesh branches, and these are identical to those of example 1. However, there are now 3 nodal branches – i.e. branches represented by nodal equations.

One important aspect of dividing the circuit between nodal and mesh portions is that for a branch to a mesh branch, the total flux in that branch must be computable solely in terms of the mesh fluxes. Otherwise, a branch must be represented as a nodal branch. For the purposes of the MEC toolbox, all nodal branches must be magnetically linear.

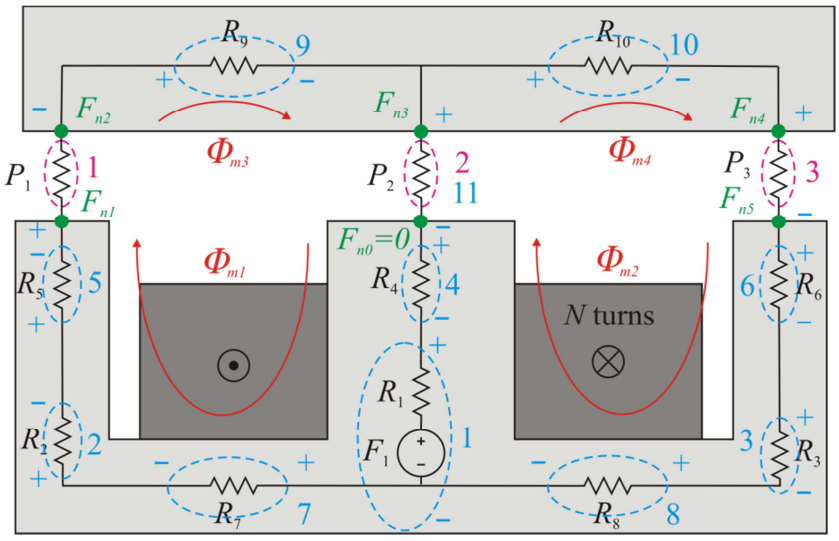

Figure 4. Mixed mesh/nodal EI Core MEC.

The code for this version of the MEC follows:

```
16    % Initialize
17    clear all; close all; clc
18
19    % Define some handy numbers
20    mm = 1e-3;
```

```
21    mu0 = 4*pi*10^-7;
22
23    % Specify the dimensions of the EI Core based on
24    % J.Cale, S.D.Sudhoff, and L.Tan, "Accurately Modeling EI Core Inductors
25    % Using a High-Fidelity Magnetic Equivalent Circuit Approach",
26    % IEEE Transactions on Magnetics, Vol.42, No.1, January 2006
27    we = 28.7 * mm;
28    wc = 57.4 * mm;
29    ws = 37.0 * mm;
30    ds = 48.8 * mm;
31    wb = 27.3 * mm;
32    wi = 30.5 * mm;
33    g  = 2.96 * mm;
34    d  = 119.8 * mm;
35    N  = 40;
36
37    % Curent range of interest
38    i  = linspace(0,100,100);
39    Npoints = length(i);
40
41    % Determine the MMF source on the circuit
42    F = N*i;
43
44    % Calculate the permeances on the air gap
45    P1 = mu0*(we*d)/g;
46    P2 = mu0*(wc*d)/g;
47    P3 = P1;
48
49    % 3C90 material parameters to calculate permeability as a function of B
50    M3C90.mur = 22340.9259;
51    M3C90.muB.a   = [  1.1542      0.049742    0.049644    0.041155];
52    M3C90.muB.b   = [431.1763       2.29503    15.04824    74.28908];
53    M3C90.muB.g   = [  0.4742        2.7955     0.59862     0.43996];
54    M3C90.muB.d   = M3C90.muB.a./M3C90.muB.b;
55    M3C90.muB.t   = exp(-M3C90.muB.b.*M3C90.muB.g);
56    M3C90.muB.h   = M3C90.muB.a.*M3C90.muB.t;
57    M3C90.muB.e   = M3C90.muB.t./(M3C90.muB.t+1);
58    M3C90.muB.z   = 1./(M3C90.muB.t+1);
59
60    % initial flux linkage and convergence
61    lambda=NaN(Npoints,1);
62    c=NaN(Npoints,1);
63
64    % Initialize the MEC
65    MEC = mec_init(4,10,5,3,1);
66
67    % Establish a list of non-linear materials
68    MEC = mec_nl_material(MEC,1,@muB,M3C90);
69
70    % Mesh Branches (MB)
71    MEC = mec_nls_branch(MEC,1,wc*d,(wb/2+ds/2),1,0,0.0,[1 -2],0.0); % MB 1
72    MEC = mec_nl_branch(MEC,2,we*d,(wb/2+ds/2),1,[1],0.0);           % MB 2
73    MEC = mec_nl_branch(MEC,3,we*d,(wb/2+ds/2),1,[2],0.0);           % MB 3
74    MEC = mec_nl_branch(MEC,4,wc*d,ds/2,1,[1 -2],0.0);               % MB 4
75    MEC = mec_nl_branch(MEC,5,we*d,ds/2,1,[1],0.0);                  % MB 5
76    MEC = mec_nl_branch(MEC,6,we*d,ds/2,1,[2],0.0);                  % MB 6
77    MEC = mec_nl_branch(MEC,7,wb*d,(wc/2+ws+we/2),1,[1],0.0);        % MB 7
78    MEC = mec_nl_branch(MEC,8,wb*d,(wc/2+ws+we/2),1,[2],0.0);        % MB 8
79    MEC = mec_nl_branch(MEC,9,wi*d,(wc/2+ws+we/2),1,[3],0.0);        % MB 9
80    MEC = mec_nl_branch(MEC,10,wi*d,(wc/2+ws+we/2),1,[4],0.0);       % MB 10
81
82    % Explicit Nodes (EN)
83    MEC = mec_enode(MEC,1,[1]);                                      % EN 1
```

```
84    MEC = mec_enode(MEC,2,[-3]);                                    % EN 2
85    MEC = mec_enode(MEC,3,[3 -4]);                                  % EN 3
86    MEC = mec_enode(MEC,4,[4]);                                     % EN 4
87    MEC = mec_enode(MEC,5,[-2]);                                    % EN 5
88
89    % Nodal Branches (NB)
90    MEC = mec_lnb(MEC,1,1,2,P1,0,0);                                % NB 1
91    MEC = mec_lnb(MEC,2,3,0,P2,0,0);                                % NB 2
92    MEC = mec_lnb(MEC,3,4,5,P3,0,0);                                % NB 3
93
94     % Test contruction of MEC
95     if (mec_test(MEC)~=1)
96        error('Bad MEC');
97     end
98
99     % Loop over current values
100    for n = 1:Npoints
101
102       % Assign MMF
103       MEC.mb.Fs(1)=F(n);
104
105       % Solve the MEC
106       [PHIm,PHImb,PHImR,Fmb,Fn,Fnb,PHInb,PHInP,c(n)] = mec_solve(MEC);
107
108       % Compute the flux linkage
109       lambda(n) = - PHImR(1)*N;
110
111    end
112
113    % Generate the lambda-vs-current curve
114    if (min(c)<1)
115       error('Solution Did Not Converge');
116    end
117
118    % Plot flux linkage versus current
119    plot(i,lambda)
120    xlabel('i, (A)')
121    ylabel('\lambda, (Vs)')
122    grid on
123    axis([0 100 0 0.2]);
```

The code for this implementation of the MEC is very similar to example 1, and therefore this discussion will focus on the differences between the two examples. The first significant difference occurs in line 65, the call to `mec_init`. Observe the extended argument list, which indicates 4 explicit meshes, 10 mesh branches, 5 non-zero nodes, 3 nodal branches, and 1 type of magnetically nonlinear material.

The exposition of the 10 mesh branches in lines 70-80 is identical to the case for example 1, and does not require further comment; however the next step in the code is significantly different. In lines 82-87 the explicit nodes are declared. Each explicit node is set up with a call to the `mec_enode` routine. The arguments to this routine are the MEC structure, the node number being declared, and the mesh list for the node. The output of this routine is an updated MEC structure. The mesh list for the routine is a list of all mesh flux indices of mesh fluxes which either emanate or terminate on that node. Meshes which terminate on a node are considered

positive. Mesh indices of mesh fluxes which emanate from a node are considered negative. Hence the mesh list for node 1 is [1]. The mesh list for node 3 is [3 -4] or [-4 3] (either form works). Note that it is not necessary to set up node 0.

The next part of the code processes the nodal branches. This occurs in lines 89-92. Nodal branches are set up using the `mec_lnb` routine. The arguments to this routine are the MEC structure, the branch number, the positive node number for the branch, the negative node number for the branch, the permeance of the branch, the MMF source for the branch, and the flux source of the branch, in accordance with the standard branch definition given in Figure 1.

As this point, the remainder of the code could have been identical to example 1, but some additional changes have been made. In lines 94-97, the `mec_test` routine is invoked. This routine inspects the MEC structure writes warning message. While this routine cannot find all errors, it can find many, and is particularly useful while for code debugging. Once an MEC model is mature however, it is advisable to remove the call to this routine as it will slow overall execution.

An additional change in example 2 relative to example 1 is the call to the `mec_solve` routine in line 106. In particular, note the additional output arguments which include `Fn`, the vector of node potentials, `Fnb`, the vector of node branch potentials (see Fig. 1), `PHInb`, the vector of fluxes into the nodal branches, and `PHInP`, the vector of fluxes into the permeance element of the nodal branches.

The remaining aspects of the code are very similar to Example 1 and will not be discussed further. Executing the code yields identical results.

The main advantage of using a mixed mesh/nodal approach is not the situation in this example, but rather a situation in which the permance between sets of nodes vary with a change in geometry – for example in rotating electric machinery where the nodes might be placed on the inside edge of the stator and the outside edge of the rotor. In this situation, note that since the permeance can be set to zero, it is possible to disconnect (and reconnect) stator and rotor nodes as the machine rotates.

8. Supplemental Routines

In addition to the MEC core routines mentioned in Section 4 and 5, the MEC Toolbox also provides additional routines corresponding to several physical models. Detailed documentation of these routines is contained with the m-files.

`BE_core`	Sets up data structure associated with a bobbin E core
`BEE_transformer_mec`	MEC analysis of a bobbin EE core transformer
`bob_wind`	Sets up data structure associated with a bobbin winding
`draw_BEEtrans`	Draws a bobbin EE core transformer
`draw_coil_profileview`	Draws a profile view of a coil
`draw_coil_topview`	Draws a top view of a coil
`draw_U_profileview`	Draws a profile view of a U core
`draw_U_topview`	Draws a top view of a U core
`draw_UUtrans`	Draws a UU core transformer
`ei_inductor_mec`	Comprehensive MEC analysis of an EI core inductor
`ei_inductor_mec2`	Comprehensive MEC analysis of an EI core inductor with a saturation permeance
`muB`	A function which calculates permeability and its derivative using the method set forth in [5]
`P_rct_bndle_fm`	Calculate the internal and external permeance associated with a rectangular bundle of conductor above a ferro- or ferri-magnetic material
`Pint_rct_bndle_air`	Calculates the internal permeance of a rectangular wire bundle in air based on method set forth in [**Error! Reference source not found.**]
`rect_coil`	Calculates the dimensions of a rectangular coil
`resistance`	Determines the ac and dc resistance of the wire given wire cross sectional area, length, and frequency
`u_volume`	Calculates the volume of a U core
`ui_inductor_mec`	MEC analysis of a UI core inductor
`uu_transformer_mec`	MEC analysis of a UU core transformer

9. Acknowledgements

Sponsors for this work include the Office of Naval Research, grants N00014-08-1-0080 (Electric Ship Research and Development Consortium) and N00014-08-1-0397.

10. References

[1] S.D. Sudhoff, Power Magnetic Devices: *A Multi-Objective Design Approach*, *IEEE-Press/Wiley*, 2014

[2] J.Cale, S.D.Sudhoff, L.Tan, "Accurately Modeling EI Core Inductors using a High-Fidelity Magnetic Equivalent Circuit Approach", in *IEEE Transactions on Magnetics*, Vol. 42, No. 1, January 2006, pp. 40–46.

[3] H.W.Derbas, J.M.Williams, A.C.Koenig, S.D.Pekarek, "A Comparison of Nodal- and Mesh-Based Magnetic Equivalent Circuit Models", in *IEEE Transactions on Energy Conversion*, Vol. 24, No. 2, June 2009, pp. 388–396.

[4] B.N.Cassimere, "Modeling and Evolutionary Design of Permanent Magnet Synchronous Machines", *Ph.D. Dissertation*, Purdue University, May 1998.

[5] G.M. Shane, S.D. Sudhoff, "Refinements in Anhysteretic Characterization and Permeability Modeling," IEEE Transactions on Magnetics, vol. 46, no. 11, pg(s) 3834-3843, November 2010

11. License