A New Object-Oriented Simulation Tool for Modeling Preisach-Based Magnetic Hysteresis Nonlinearities

PAUL MOSES
School of Electrical and Computer Engineering
University of Oklahoma
110 W. Boyd St., Devon Energy Hall, Norman, Oklahoma 73019
UNITED STATES
pmoses@ou.edu

Abstract: - The Preisach theory of hysteresis is regarded by many experts to be the most accurate physical description of hysteresis behavior, especially in ferromagnetism. Unfortunately, its practical applications and accessibility to students and researchers are limited due to its complex formulation and numerical implementation. Instead, simpler and less accurate hysteresis models or anhysteretic single-value nonlinear models are often employed. This paper proposes a new and relatively simple object-oriented model representation for Preisach theory in MATLAB-Simulink. The developed hysteresis block is highly customizable for modeling complex dynamical systems and is also ideal for use in educational courses to allow students to simulate and further their understanding of ferromagnetism and hysteresis behavior in nonlinear systems.

Key-Words: - Preisach, magnetics, hysteresis, nonlinear model

1 Introduction

Detailed modeling of electromagnetic circuits with ferrite cores is essential in the optimal design of power supplies, transformers, motors, generators and converters. Accurate electromagnetic modeling of dynamic hysteresis nonlinearities has recently been shown to be significant in uncovering complicated behavior in transformers (e.g., inrush and ferroresonance) [1-4]. In an educational setting, object-oriented software tools have also proven useful in facilitating the learning of many power system engineering concepts [5-10]. However, easy-to-use software tools are lacking for students who wish to gain a physical understanding of ferromagnetism, especially with regards to the complicated hysteresis behavior exhibited in many electrical machine components.

The hysteresis phenomenon is observed in many other fields including economics, medical sciences, chemistry and mechanical engineering. Unfortunately, the best hysteresis models have seen little or no use in these fields due to their complexity and inaccessibility to students and researchers [11]. Instead, simpler and less accurate nonlinear models (e.g., single-valued saturation functions) are often employed which do not reach the full potential for optimal electromagnetic circuit analysis and design. In order to address this deficiency, a new object-oriented methodology based on the Preisach model of hysteresis [12] is proposed using the widely accessible MATLAB-Simulink software package.

Very few attempts have been made at developing readable software models of hysteresis for design and analysis applications as well as for educational purposes. Of the more successful attempts, Prigozy [13] and Ngo [14] have published PSPICE models using the Jiles-Atherton hysteresis approach [15]. However, these models are not very successful at reproducing minor hysteresis loops compared to other models of ferromagnetism [16]. More accurate models suffer from problems such as; (1) lack of generality by tailoring to a particular application, (2) numerical implementation is obscured through complicated mathematical formulation, (3) dynamic effects (e.g., minor loop formation) are often oversimplified and inaccurately portrayed, and (4) tuning the model to approximate experimental results is often a long and tedious trial-and-error process.

The Preisach theory of hysteresis [12] is generally regarded by experts to be the most complete and accurate representation of hysteresis, particularly in the field of ferromagnetism. Unfortunately, this powerful model has seen limited applications due to its complicated mathematical formulation. The few reported studies provide very limited or no insight on the methodology of implementation. Some of the more successful publications in this regard are [17-20]. However, the software implementation is not described, putting the model largely out of reach of researchers and designers.

The aim of this paper is to increase the accessibility of the Preisach model by offering a novel object-oriented implementation in the MATLAB-Simulink environment [21]. Using the S-function block capability of MATLAB-Simulink, the Preisach model is realized by embedding the developed numerical code into plug-and-play type Simulink library blocks. The user interface is graphical and intuitive making it a useful tool in expanding users ability to experiment with the model. Furthermore, the proposed implementation allows users to incorporate this powerful model into larger dynamical systems using block-diagrams [10] and Simulink block library. In this way, the approach offers a practical perspective of Preisach-theory and insight into ferromagnetic hysteresis behavior.

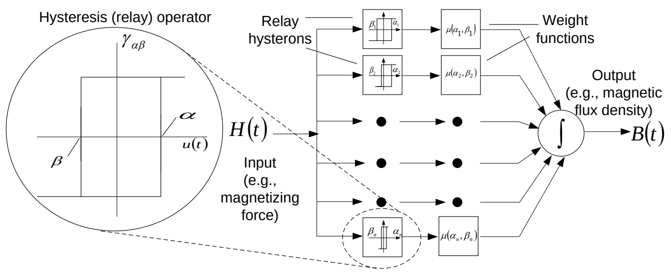
2 Preisach Theory of Hysteresis

Ferenc Preisach proposed his theory of hysteresis in a landmark paper in 1935 [12]. The theory drifted into obscurity until the 1970s when Russian mathematicians Krasnoselskii and Pokrovskii began studying the model for its mathematical properties [22]. It was not until the 1980s that the theory was first developed into a usable hysteresis model [23]. The model has now become one of the most successful mathematical descriptions of hysteresis to date.

2.1 Model Basis

The emphasis of this paper is the practical numerical and object-oriented implementation of the model. Only critical mathematical concepts of Preisach-theory are presented to guide the algorithm development. For greater mathematical insight, the interested reader should refer to [19, 23].

The elementary building blocks for the Preisach model first conceived in [12] is a two-valued, binary type, hysteresis relay operator, or, “hysteron”. These hysteron relays (denoted by $\hat{\gamma}_{\alpha\beta}$) can be viewed to have two switching states when operated on an arbitrary input H (e.g., magnetizing force). That is $\hat{\gamma}_{\alpha\beta}(H(t))$ can take values of +1 (“up”) and -1 (“down”). The threshold value α is the level the input must exceed to cause an “up” switch transition. Conversely, β is the threshold value for the input to decrease to cause a “down” switch transition. Therefore, hysterons operating on inputs H will cause up and down transitions when $H \geq \alpha$ and $H \leq \beta$, respectively. These hysteron transitions are depicted in Fig. 1a.



(a) (b)

Fig. 1. (a) Two-state relay hysteron operator and (b) parallel connection of hysterons operating on arbitrary input.

The Preisach model is built up from a set of hysterons, each with different (α, β) switching transitions threshold values. These hysterons are akin to magnetic dipoles in ferromagnets. It was postulated that if a parallel set of hysteron relays operates on an arbitrary input H (e.g., magnetizing force), and each hysteron is multiplied by its own weighting function $\mu(\alpha, \beta)$, then the sum of the results gives rise to an output B (e.g., magnetic flux density) that exhibits hysteretic properties when plotted against its time-varying input. This concept is illustrated in Fig. 1b. The combined action of all weighted hysterons is summed and called the Preisach operator $\hat{\mathrm{P}}$. The model can thus be expressed as follows,

$$B(H(t)) = \hat{P}(H(t)) = \iint_{\alpha \geq \beta} \mu(\alpha,\beta)\hat{\gamma}_{\alpha\beta} H(t) \mathrm{d}\alpha \mathrm{d}\beta \quad (1)$$

From Fig. 1 and (1), it should become apparent that this model exhibits the property of memory. That is, the state of particular relay hysterons may have been set in the up or down positions temporarily by prior values of a time-varying input. In ferromagnetism, this behavior closely approximates the magnetization history of ferromagnetic material which determines the magnetic memory of a sample. When this is played out over time, repetitive input oscillations (e.g., sinusoidal) can result in recognizable hysteresis behavior.

2.2 Preisach Memory

It has been proven in [23] that only certain selected values of past inputs are necessarily retained to determine the Preisach model output at a given time. A special "wipe-out" property allows selected new values of input to erase Preisach memory. These input values are the dominant extrema of the input waveform such as those depicted in Fig. 2.

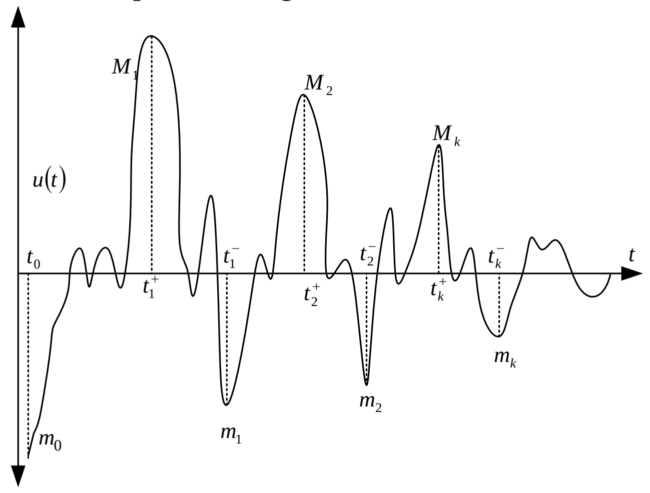


Fig. 2. An example of arbitrary input (e.g., magnetizing force *H*) where only dominant local input extrema in sets $\{M_k\}$ and $\{m_k\}$ are accumulated as required by the Preisach model.

The dominant extrema values are constantly updated depending on whether new extrema are sufficiently large enough to "wipe out" the old extrema values.

For example, in Fig. 2, when the input is raised from an initial minimum $m_0 = -\infty$ at $t = t_0$ to M_1 at $t = t_1^+$, all input extrema up until $t = t_1^+$ are wiped out by maximum M_1. Continuing from $t = t_1^+$ to $t = t_1^-$, the minimum m_1 wipes out all input extrema that occurred between $t = t_1^+$ and $t = t_1^-$. Likewise, when the input arrives at $t = t_2^+$ with M_2, this maximum wipes out all input extrema that occurred between $t = t_1^-$ and $t = t_2^+$. So far, the values $\{m_0, M_1, m_1, M_2\}$ are retained because no input excursions were sufficient enough to wipe out these values. This process continues and an alternating series of maxima $\{M_k\}$ and mimima $\{m_k\}$ are stored. These sets of extrema values can be expressed as,

$$M_k = \max_{[t_{k-1}^-, t']} H(t), \qquad H(t_k^+) = M_k \quad (2)$$

$$m_k = \min_{[t_k^+, t']} H(t), H(t_k^-) = m, \quad (3)$$

resulting in a set of dominant maxima and minima extrema,

$$M = \{M_1, M_2, M_3, ..., M_k\}, m = \{m_0, m_1, m_2, m_3, ..., m_k\}$$
(4)

2.3 Preisach Output Computation

The next step is to simplify (1) by eliminating the double integrals and modifying the weight function $\mu(\alpha,\beta)$ for more efficient computation. Therefore, a new function *F(α, β)* was introduced. For brevity, the full derivation is omitted and can be found in [19, 23]. *F(α, β)* relates the weighting function to a desired hysteresis characteristic. Therefore, (1) is equivalent to

$$B(t) = -\lim_{\alpha \to \infty} F(\alpha, -\alpha) + \left(2\sum_{k=1}^{\infty} \left[F(M_k, m_{k-1}) - F(M_k, m_k) \right] \right) \tag{5}$$

A subtlety of this formula explained in [23] is that the final values in sets $\{M_k\}$ and/or $\{m_k\}$ could take on the present input value depending on whether *H* is increasing or decreasing. For increasing *H*, $M_k = H$ and (5) becomes,

$$B(t) = -\lim_{\alpha \to \infty} F(\alpha, -\alpha) + \left(2\sum_{k=1}^{n(t)-1} \left[F(M_k, m_{k-1}) - F(M_k, m_k) \right] \right) + 2\left[F(M_n, m_{n-1}) - F(M_n, H(t)) \right], \tag{6}$$

and for decreasing *H*, $m_k = H$ and (5) is evaluated as,

$$B(t) = -\lim_{\alpha \to \infty} F(\alpha, -\alpha) + \left(2\sum_{k=1}^{n(t)-1} \left[F(M_k, m_{k-1}) - F(M_k, m_k) \right] \right) + 2F(H(t), m_{n-1}). \tag{7}$$

Therefore from (5)-(7), it is observed that an updating set of input extrema (2)-(4), current input value and knowledge of *F(α, β)* function is all that is required to compute the hysteresis response. The choice of function *F(α, β)* is discussed in [17-20].

3 Model Implementation in MATLAB

This section details the software implementation of Preisach-theory based on the mathematics described in Section II. In order to demonstrate its implementation, one of the most widely used numerical packages MATLAB is used with the object-oriented based simulator Simulink [21]. Using the S-function feature in Simulink, a custom hysteresis block can be developed whose properties and behavior is defined by the developed M-file source code. The S-function template provided in MATLAB help files is modified to encapsulate the Preisach algorithm. In the following, it is assumed the reader is cognizant with a basic understanding of S-functions, which can be found in Simulink help.

3.1 Initialization Stage

From Section II, it is apparent that the numerical code will have to deal with storing prior values of input in order to compute the output. S-function blocks have internal memory storage known as Dwork vectors that can retain information for subsequent time-steps. The main Dwork vectors required will be used as stack buffers to store dominant extrema $\{M_k\}$, $\{m_k\}$. Furthermore, some status flags and vector indices also need to be stored in Dwork vectors for subsequent time steps.

S-functions require Dwork vectors to be pre-allocated to a fixed size. This restriction presents some minor hindrance on the allocation of storage vectors for dominant extrema. This is because the number of elements *n(t)* in (6)-(7) with dominant extrema is always changing depending on the input variations. Therefore, it was decided to preallocate these vectors to a sufficient size such that collected dominant extrema values would not be expect to exceed the buffer size. The Dwork vectors for M_k and m_k are initialized with a finite size of 500 elements. Another Dwork vector (*uSeg*) of size 10000 is for buffering enough input values to scan for dominant extrema. The following is a sample of S-function code of how variables for memory buffers are declared;

```
function DoPostPropSetup(block)
… block.Dwork(1).Name = 'uSeg';
block.Dwork(1).Dimensions  = 10000;
block.Dwork(1).DatatypeID  = 0;
block.Dwork(1).Complexity = 'Real';
block.Dwork(1).UsedAsDiscState = true;
block.Dwork(2).Name = 'MkBuffer';
block.Dwork(2).Dimensions = 500;
block.Dwork(2).DatatypeID = 0;
block.Dwork(2).Complexity    = 'Real';
block.Dwork(2).UsedAsDiscState = true;
block.Dwork(3).Name = 'mkBuffer';
block.Dwork(3).Dimensions = 500;
block.Dwork(3).DatatypeID  = 0;
block.Dwork(3).Complexity   = 'Real';
block.Dwork(3).UsedAsDiscState = true;  …
end
```

Following the declaration of memory buffer sizes and data types, it is necessary to prescribe a sequence of $\{M_k\}$ and $\{m_k\}$ values into the buffers to set an initial magnetization history. In ferromagnetism, this defines the residual flux density. For this work, a decaying alternating series of maxima and minima is stored into the memory buffers which are akin to the action of demagnetizing ferromagnetic material with a slowly decaying alternating current magnetizing field. The following code demonstrates the initialization routine performed once for the entire simulation run;

```
function InitConditions(block)
   % Initialize prior time step (t-1) to negative
   infinity
 block.Dwork(1).Data=
  NaN(1,block.Dwork(1).Dimensions);
 block.Dwork(1).Data(1) = -inf;
   % Place NaNs in buffers and then write initial
   decaying maxima series
 block.Dwork(2).Data=
 NaN(1,block.Dwork(2).Dimensions);
 lock.Dwork(2).Data(1:400)=[0.4:0.001:0.001];
 % Place NaNs in buffers and then write initial
decaying minima series
 block.Dwork(3).Data=
 NaN(1,block.Dwork(3).Dimensions);
 block.Dwork(3).Data(1) = -inf;
 block.Dwork(3).Data(2:401)=[-0.4:0.001:-
 0.001];  … end
```

A few subtleties of the presented code should be pointed out. Firstly, it is assumed that the magnetization history is such that the input is first raised from a large negative value (e.g., state of negative saturation). Therefore, the first value in the decaying sequence of alternating maxima-minima values is the minimum $m_0 = -\infty$. Subsequent minima values are gradually increasing to zero while maxima values decrease toward zero, effectively mimicking the demagnetization process. Note the number of minima values in the set is one more than the number of maxima values in the set, as required by the summation terms in (5).

Furthermore, MATLAB NaNs (not-a-number) values are used to fill the buffers with nonessential elements. This is necessary because of the fixed vector size restriction in S-function blocks. Otherwise, the vector sizes need only to resize themselves according to the number of elements contained in $\{M_k\}$, $\{m_k\}$ (e.g., when new extrema are introduced and old ones are wiped out). On the other hand, the fixed buffer sizes allow for more efficient processing as resizing memory buffers every time step is computationally intensive.

3.2 Main Preisach Algorithm

The full computer code listing of the main routine and all required S-function subroutines is listed in the Appendix. The flow chart for the implemented Preisach algorithm is depicted in Fig. 3. The primary tasks of the main routine are to detect and store new dominant extrema values, perform wiping out of old extrema values, and update the output for the current time step. Prior to simulation run, the solver method, time-steps, run duration and hysteresis block input parameters need to be configured. The main block parameters are the constants and coefficients for *F(α, β)* which shapes the hysteresis loops.

Once the simulation is initiated, the current input value at the block input port and model parameters are passed into the main code contained in preisachHysteresis.m. The initialization stage as described previously is performed which pre-defines the set $\{M_k\}$ and $\{m_k\}$ in Preisach memory for initial magnetization history. In subsequent time steps, $\{M_k\}$ and $\{m_k\}$ are calculated from the previous iteration. A temporary storage buffer named *uSeg* is used to store values from the last global extrema to the present input value. This allows the program to detect whether or not local maximum/minimum turning points (potentially new extrema) have been reached. Therefore, vector *uSeg* will only contain values between two successive turning points. Based on this, and whether the input is increasing or decreasing, the program chooses between two codes to begin detection of dominant maxima and minima values of input.

If the input has passed a local minimum turning point and is increasing, the first step is to find and keep all values of dominant maxima greater than the present input value *H*. Values of $\{M_k\}$ that were not greater than *H* are wiped out from memory. As stipulated in (6), the current input H must be added to the end of $\{M_k\}$ set. As some values of $\{M_k\}$ were wiped out, correspondingly, dominant minima in$\{m_k\}$ that occurred after the last two extrema are wiped out. This is because $\{M_k\}$ and $\{m_k\}$ together must form a sequence of alternating maxima and minima. A similar process is performed to satisfy (7) for a decreasing input, passing a local maximum turning point. In this case, the present input H must be added to the end of both $\{M_k\}$ and $\{m_k\}$ sets.

Finally, the Dwork vectors for dominant extrema $\{M_k\}$, $\{m_k\}$ and input segments (*uSeg*) are updated such that they are stored for the next iteration and the computed output is sent to the Simulink block output port. The entire process is repeated for subsequent time steps until the simulation run time has elapsed.

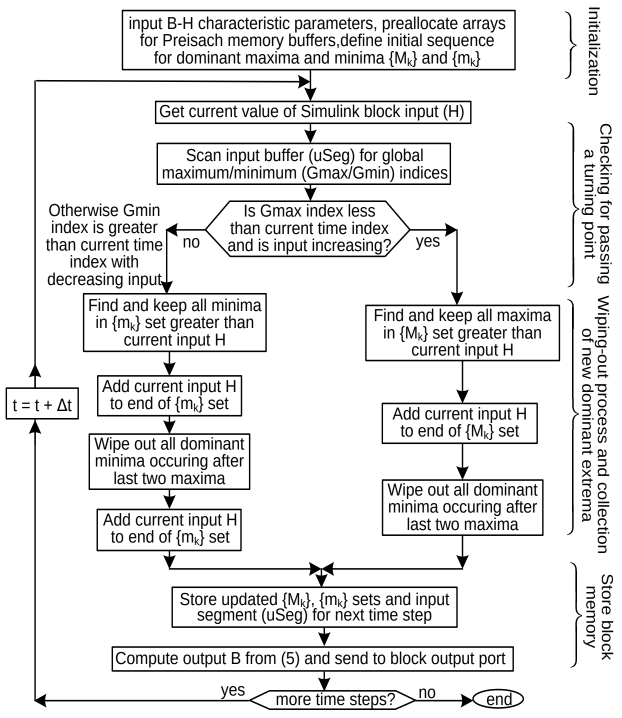
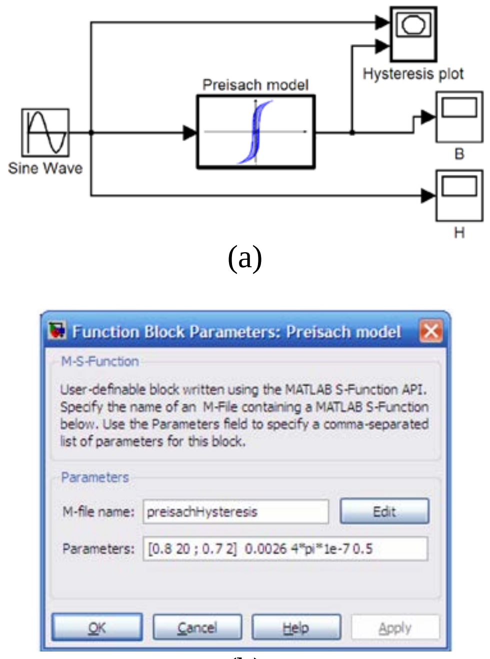

input B-H characteristic parameters, preallocate arrays for Preisach memory buffers,define initial sequence for dominant maxima and minima {M$_k$} and {m$_k$}
Initialization
Get current value of Simulink block input (H)
Scan input buffer (uSeg) for global maximum/minimum (Gmax/Gmin) indices
Checking for passing a turning point
Is Gmax index less than current time index and is input increasing?
Otherwise Gmin index is greater than current time index with decreasing input
no
yes
Find and keep all minima in {m$_k$} set greater than current input H
Add current input H to end of {m$_k$} set
Wipe out all dominant minima occuring after last two maxima
Add current input H to end of {m$_k$} set
Find and keep all maxima in {M$_k$} set greater than current input H
Add current input H to end of {M$_k$} set
Wipe out all dominant minima occuring after last two maxima
Wiping-out process and collection of new dominant extrema
t = t + Δt
Store updated {M$_k$}, {m$_k$} sets and input segment (uSeg) for next time step
Store block memory
Compute output B from (5) and send to block output port
yes
more time steps?
no
end

Fig. 3. Main Preisach algorithm for MATLAB-Simulink S-function

3.3 Object-Oriented Block Definition

In the Simulink environment, the S-function block allows one to link the model-based simulator to numerical code. The extensive Simulink library offers many functions to serve as inputs for the hysteresis block as well as manipulating and plotting outputs as required (Fig. 4a). Therefore, Simulink permits highly complex dynamical systems to be analyzed in a straightforward and intuitive manner. Hence, the developed Preisach hysteresis block can be used for analyzing complex physical systems (e.g., magnetic circuits). The input dialog box for the S-Function block is used to specify the Preisach hysteresis M-file and *F(α, β)* function parameters (Fig. 4b).



Function Block Parameters: Preisach model

M-S-Function

User-definable block written using the MATLAB S-Function API. Specify the name of an M-File containing a MATLAB S-Function below. Use the Parameters field to specify a comma-separated list of parameters for this block.

Parameters

M-file name: preisachHysteresis

Parameters: [0.8 20 ; 0.7 2] 0.0026 4*pi*1e-7 0.5

Fig. 4. (a) Simulink block of Preisach model and (b) parameter dialog box

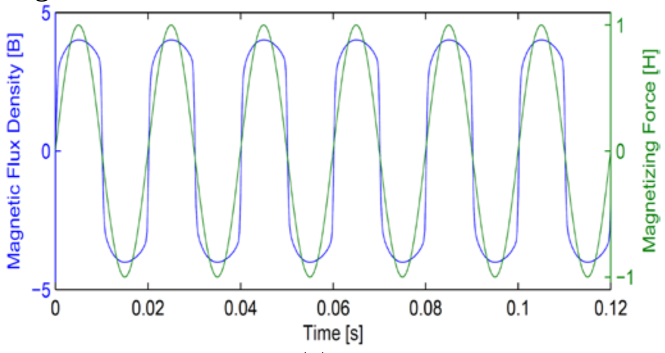
4 Simulation Results

This section presents simulation results demonstrating the Preisach model simulated in MATLAB-Simulink. The XY plotter and scope blocks are used to view the hysteresis loops and input-output waveforms. A fixed step discrete-time solver is used with a step-size of $\Delta t = 0.2$ ms. For demonstration purposes, the *F(α, β)* function and parameters are obtained from [19] for a known *B-H* characteristic where $M_1 = 0.8$, $M_2 = 0.7$, $P_1 = 20$, $P_2 = 2$, $e = 0.5$ and magnetic permeability constant is $\mu_0 = 4\pi.10^{-7}$. Alternatively, the user is free to select other *F(α, β)* functions.

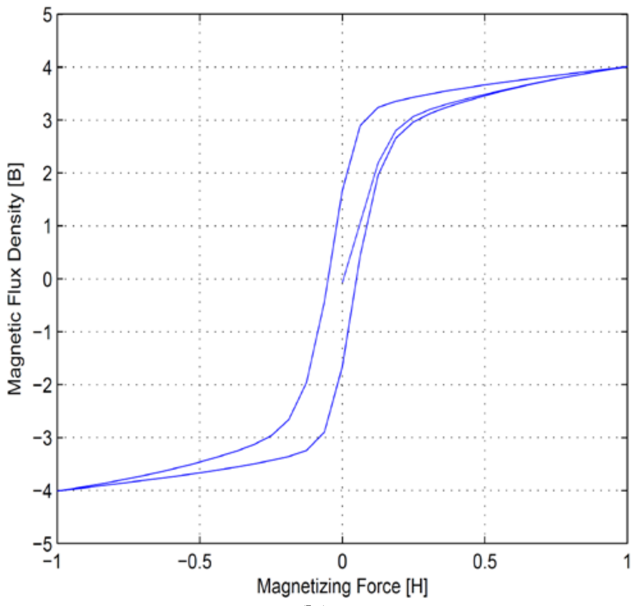
4.1 Sinusoidal Input Excitation

The first simulation case is for a sinusoidal input $H(t) = 1.0\sin(2*\pi*50*t)$ as shown in Fig.

5. This is representative of typical hysteresis behavior in an iron-core inductor or transformer connected to an alternating current source. As expected, the hysteresis trajectory increases from the initial demagnetized state defined by the initial $\{M_k\}$ and $\{m_k\}$ sets and settles into its typical cyclical behavior with positive and negative saturation states.



(a)

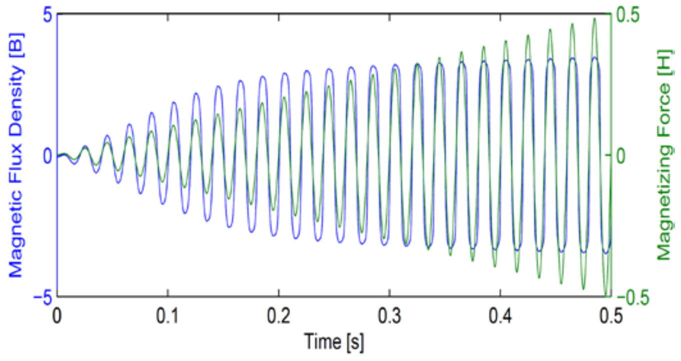


(b)

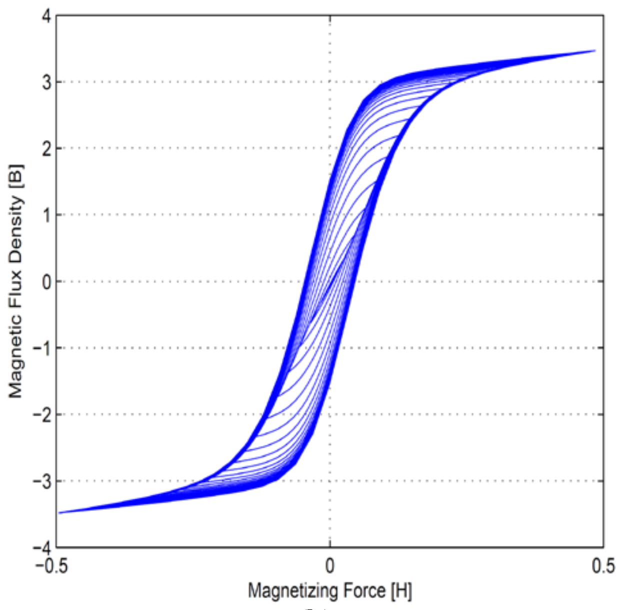
Fig. 5. Simulation of Preisach hysteresis block with sinusoidal input; (a) input-output waveforms for magnetization force *H* and flux density *B*, and (b) corresponding *B*-*H* hysteresis loops.

4.2 Nonsinusoidal Input Excitation

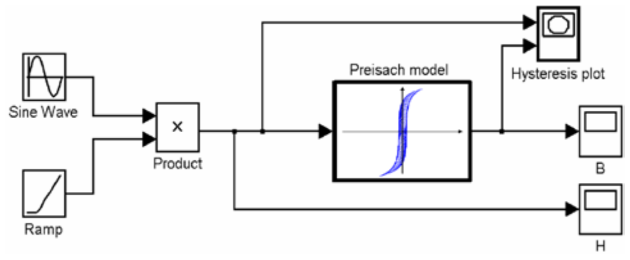
In order to demonstrate the model versatility in Simulink and show the dynamic behavior through formation of minor hysteresis loops, the input is changed by multiplying the sine function with a ramp function. The corresponding hysteresis loops and waveforms for linearly increasing sine function amplitude are observed in Fig. 6. It should be noted that unlike most hysteresis models (e.g., Jiles-Atherton method [13-15]), the Preisach model is capable of forming the minor hysteresis loops independently from the major hysteresis loop.



(a)



(b)



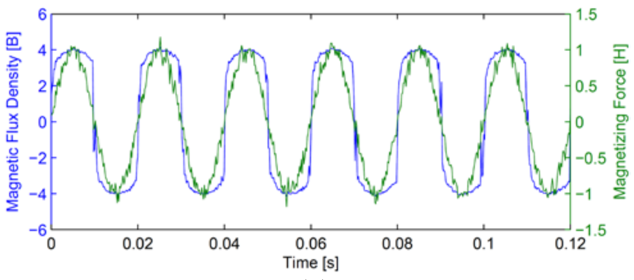
(c)

Fig. 6. Simulation of Preisach hysteresis block under ramped sinusoidal input; (a) input-output waveforms for magnetization force *H* and flux

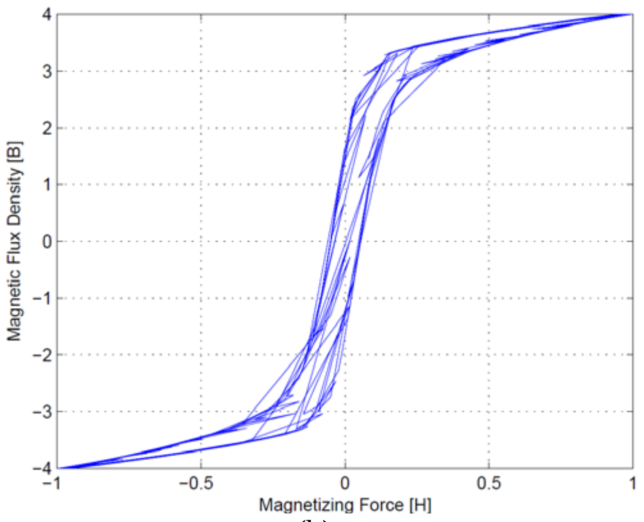
density *B*, (b) *B-H* hysteresis loops and (c) Simulink block diagram.

4.3 Sinusoidal Input with White Noise

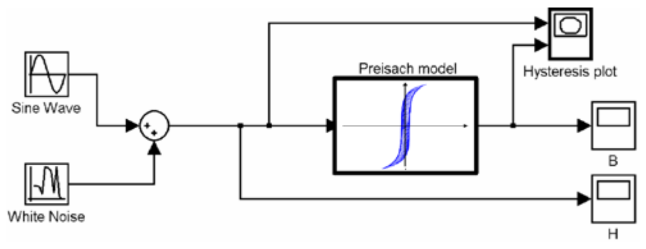
For demonstrating the robustness of the model algorithm, the developed Preisach block is subjected to a simulated noisy input. A white noise Gaussian block is added to the sinusoidal input resulting in the waveforms and hysteresis loops depicted in Fig. 7. Similarly, there are many other block sets in Simulink which can be utilized with the Preisach model block. This opens up a wide range of applications for Presiach theory in Simulink such as modeling dynamics of mechanical systems, control systems, and electromagnetic circuits in electronics and power applications.



(a)



(b)



(c)

Fig. 7. Simulation of Preisach hysteresis block with sinusoidal input subjected to additive white noise; (a) input-output waveforms for magnetization force *H* and flux density B, (b) *B-H* hysteresis loops and (c) Simulink block diagram.

4 Conclusion

The aim of this paper was to demonstrate a new and practical implementation of the Preisach model of hysteresis. The implementation is well suited for education, design and analysis applications. This was prompted by the lack of clear and general numerical implementations of Preisach-theory which has resulted in very narrow and highly specialized usage of this powerful model. Furthermore, nonlinearities of magnetic circuits in power applications are often oversimplified to single-value nonlinear functions which ignore true dynamic hysteresis behavior. Therefore, the developed MATLAB code and object-oriented approach for implementing Preisach-theory generalizes the model to researchers and designers in a wide variety of disciplines, especially within the power engineering field. For example, the presented methodology could be useful in the dynamic modeling of ferrite cores in magnetic circuits such as transformers, inductors and rotating electrical machinery. Furthermore, the ability to use the Simulink engine greatly increases model flexibility for serving as a powerful design tool for students and engineers to experiment with and gain further insight into hysteresis and ferromagnetism.

5 Appendix

S-function code listing for Preisach Hysteresis Model

```
function preisachHysteresis (block) ;
setup(block);  end   % S-Function
function setup(block); % Parameter structure is
      ordered as{ [Mi ; Pi] A μ0 e}
     block.NumDialogPrms  = 4;      %
     MATLAB version 7.8.0.347 (R2009a)
     block.NumInputPorts  = 1;
     block.NumOutputPorts = 1;
```

```
  % Setup functional port properties to
      dynamically inherited.
      block.SetPreCompInpPortInfoToDynamic;
      block.SetPreCompOutPortInfoToDynamic;
      block.InputPort(1).Dimensions  = 1;
      block.InputPort(1).DirectFeedthrough =
      true;
      block.OutputPort(1).Dimensions   = 1;
  % Set block sample time to inherited
      block.SampleTimes = [0 0];
  % Register methods

block.RegBlockMethod('PostPropagationSetup',
@DoPostPropSetup);

block.RegBlockMethod('InitializeConditions',
@InitConditions);
  block.RegBlockMethod('Outputs',
@Output);
   % block.RegBlockMethod('Update',
@Update);  end
function DoPostPropSetup(block) ; % Setup
Dwork
      block.NumDworks = 4;
   % Setting input storage and dominant extrema
       buffers {m_k}, {M_k}
      block.Dwork(1).Name = 'uSeg';
      block.Dwork(1).Dimensions      =
     10000;
      block.Dwork(9).DatatypeID  = 0;
      block.Dwork(1).Complexity =
     'Real';
      block.Dwork(1).UsedAsDiscState = true;
      block.Dwork(2).Name = 'MkBuffer';
      block.Dwork(2).Dimensions=
      500;
      block.Dwork(2).DatatypeID = 0;
      block.Dwork(2).Complexity   =
     'Real';
      block.Dwork(2).UsedAsDiscState = true;
      block.Dwork(3).UsedAsDiscState = true;
      block.Dwork(4).Name = 'slopeFlag';
      block.Dwork(4).Dimensions
      = 1;
      block.Dwork(4).DatatypeID= 0;
      block.Dwork(4).Complexity     = 'Real';
      block.Dwork(3).UsedAsDiscState = true;
end
```

```
function InitConditions(block) ;   % Initialize
   Dwork
  % Input storage buffer for min max scans of
     input segments
      block.Dwork(1).Data =
      NaN(1,block.Dwork(1).Dimensions);
      block.Dwork(1).Data(1) = -inf;
  % Defining initial magnetic history
      block.Dwork(2).Data =
      NaN(1,block.Dwork(2).Dimensions);
      block.Dwork(2).Data(1:400) = [0.4:-
      0.001:0.001];
      block.Dwork(3).Data =
      NaN(1,block.Dwork(3).Dimensions);
      block.Dwork(3).Data(1) = -inf;
      block.Dwork(3).Data(2:401) = [-0.4:0.001:-
      0.001];
  % Assumed initial slope direction (+1 =
      increasing)
      block.Dwork(4).Data = 1;  end
function Output(block) ; % Main function for
     Preisach algorithm
  % Assemble user input block parameters into a
     data structure
      x.Mi = block.DialogPrm(1).Data(:,1);
      x.Pi = block.DialogPrm(1).Data(:,2);  x.A =
      block.DialogPrm(2).Data;
      x.u0 = block.DialogPrm(3).Data;  x.e =
      block.DialogPrm(4).Data;
  % Retrieve present input and Preisach memory
      from storage buffers
      uSeg   = block.Dwork(1).Data.';   MkBuff =
      block.Dwork(2).Data.';
      mkBuff = block.Dwork(3).Data.';
      s = block.Dwork(4).Data;  % Get initial
      slope flag
      u = block.InputPort(1).Data;  % Get current
      input value
      lenUseg = find(~isnan(uSeg),1,'last');
      uSeg(lenUseg+1) = u;  lenUseg = lenUseg
      +1 ;
      % Check if input is increasing (s=1) /
      decreasing (s=0)
      if u>uSeg(lenUseg-1) ; s = 1;   else  ; s = 0;
      end
            [dummy,maxIndx] = nanmax(uSeg);
            [dummy,minIndx] = nanmin(uSeg);
```

```
 % if passed a minimum turning point and
      ascending
if  minIndx<lenUseg && s==1
        len = length(uSeg(minIndx:end));
        uSeg(1:len) = uSeg(minIndx:end);
        uSeg(len+1:end) = NaN;
 % Find index for all values in Mk set still
above current input
indx = MkBuff>u;
  % Get new index of last value in Mk set
   (just before NaNs)
        lastMkIndx = sum(indx);
        MkBuff(1:lastMkIndx) =
        MkBuff(indx);
    % Store current input at the end of the
      Mk buffer set of maxima
      MkBuff(lastMkIndx+1) = u;
     lastMkIndx = lastMkIndx + 1;
   % Assure NaNs are placed after last
     number to end of buffer
        MkBuff(lastMkIndx+1:end) = NaN;
        mkBuff(lastMkIndx+1:end) = NaN;
 % Find index for all values in mk set still
     below current input
     indx = mkBuff<u;
 % Get new index of last value in mk set
      (just before NaNs)
        lastmkIndx = sum(indx);
 % Keep only values not wiped out by input
        mkBuff(1:lastmkIndx) =
      mkBuff(indx);
 % Store current input to the end of mk set
  of minima
      mkBuff(lastmkIndx+1) = u;
      lastmkIndx = lastmkIndx + 1;
 % Assure NaNs are placed after last
      number to end of buffer
     mkBuff(lastmkIndx+1:end) = NaN;
 % Wipe out dominant maxima that
      occurred between the two maxima
     MkBuff(lastmkIndx:end) = NaN;
 % Since  some Mk values are wiped out,
      index of the last Mk value
 % must be updated (just before NaNs) for
     the Mk buffer.
     lastMkIndx = lastmkIndx-1;   end
  % Computing output using (5)
     sumF =
sum(Fab(MkBuff(1:lastMkIndx),mkBuff(1:last
mkIndx-1),x)
                          -
Fab(MkBuff(1:lastMkIndx),mkBuff(2:lastmkIn
dx),x),2);
          B = -Fab(inf,-inf,x) + 2.*sumF ;
          % Update memory buffers for next
            iteration
          block.Dwork(1).Data = uSeg.';
    %  Update input memory segment
          block.Dwork(2).Data = MkBuff.';
   % Update dominant maxima vector
          block.Dwork(3).Data = mkBuff.';
    % Update dominant minima vector
    % Send calculated output to block's output
        port
     block.OutputPort(1).Data = B;  end
function [ y_out ] = Fab( alpha , beta , x )
% Function for half the output increments along
    the 1st-order reversal curves
    A = x.A;  u0 = x.u0; Ms = x.Mi*A/u0; e =
     x.e; P = x.Pi;  % Ref. [8]
    y_out =
sum((repmat(Ms,1,length(alpha))/2).*(tanh(P*a
lpha) -tanh(P*beta) –
(e/2).*((sech(P*beta)).^2.*tanh(P*alpha) -
(sech(P*alpha)).^2.*tanh(P*beta))) , 1 );  end
```

References:

[1] P. S. Moses, M. A. S. Masoum, and H. A. Toliyat, "Impacts of hysteresis and magnetic couplings on the stability domain of ferroresonance in asymmetric three-phase three-leg transformers," *IEEE Transactions on Energy Conversion,* vol. 26, pp. 581-592, 2011.

[2] P. S. Moses, M. A. S. Masoum, and H. A. Toliyat, "Dynamic modeling of three-phase asymmetric power transformers with magnetic hysteresis: no-load and inrush conditions," *IEEE Transactions on Energy Conversion,* vol. 25, pp. 1040-1047, 2010.

[3] P. S. Moses and M. A. S. Masoum, "Modeling ferroresonance in single-

phase transformer cores with hysteresis," in *Proc. 8th WSEAS International Conference on Electric Power Systems, High Voltages, Electric Machines*, Genova, Italy, 2009.

[4] P. S. Moses and M. A. S. Masoum, "Modeling subharmonic and chaotic ferroresonance with transformer core model including magnetic hysteresis effects," *WSEAS Transaction on Power Systems,* vol. 4, pp. 361-371, 2009.

[5] M. Kezunovic, "User-friendly, open-system software for teaching protective relaying application and design concepts," *IEEE Transactions on Power Systems,* vol. 18, pp. 986-992, 2003.

[6] L. Goel, T. T. Lie, A. I. Maswood, and G. B. Shrestha, "Enhancing power engineering education through the use of design modules," *IEEE Transactions on Power Systems,* vol. 11, pp. 1131-1138, 1996.

[7] A. Mehrizi-Sani and R. Iravani, "On the educational aspects of potential functions for the system analysis and control," *IEEE Transactions on Power Systems,* vol. 26, pp. 878-885, 2011.

[8] N. Mohan, A. K. Jain, P. Jose, and R. Ayyanar, "Teaching utility applications of power electronics in a first course on power systems," *IEEE Transactions on Power Systems,* vol. 19, pp. 40-47, 2004.

[9] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems,* vol. 26, pp. 12-19, 2011.

[10] E. Allen, N. LaWhite, Y. Yoon, J. Chapman, and M. Ilic, "Interactive object-oriented simulation of interconnected power systems using SIMULINK," *IEEE Transactions on Education,* vol. 44, pp. 87-94, 2001.

[11] E. Della Torre, "Problems in physical modeling of magnetic materials," *Physica B: Condensed Matter,* vol. 343, pp. 1-9, 2004.

[12] F. Preisach, "Uber die magnetishe nachwerikung," *Zeitschrift fur Physik,* vol. B 94, pp. 277-302, 1935.

[13] S. Prigozy, "PSPICE computer modeling of hysteresis effects," *IEEE Transactions on Education,* vol. 36, pp. 2-5, 1993.

[14] K. D. T. Ngo, "Subcircuit modeling of magnetic cores with hysteresis in PSpice," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 38, pp. 1425-1434, 2002.

[15] D. Jiles and D. Atherton, "Ferromagnetic hysteresis," *IEEE Transactions on Magnetics,* vol. 19, pp. 2183-2185, 1983.

[16] F. Liorzou, B. Phelps, and D. L. Atherton, "Macroscopic models of magnetization," *IEEE Transactions on Magnetics,* vol. 36, pp. 418-428, 2000.

[17] S. Y. R. Hui and J. Zhu, "Numerical modelling and simulation of hysteresis effects in magnetic cores using transmission-line modelling and the Preisach theory," *IEE Proceedings Electric Power Applications,* vol. 142, pp. 57-62, 1995.

[18] H. Lamba and et al., "The effect of circuit parameters on ferroresonant solutions in an LCR circuit," *Journal of Physics A: Mathematical and General,* vol. 31, p. 7065, 1998.

[19] H. Lamba, M. Grinfeld, S. McKee, and R. Simpson, "Subharmonic ferroresonance in an LCR circuit with hysteresis," *IEEE Transactions on*

Magnetics, vol. 33, pp. 2495-2500, 1997.

[20] S. R. Naidu, "Simulation of the hysteresis phenomenon using Preisach’s theory," *IEE Proceedings on Physical Science, Measurement and Instrumentation*, vol. 137, pp. 73-79, 1990.

[21] "MATLAB/Simulink Version 7.8.0.347 (R2009a)," ed. Natick, MA: Math-Works, Inc.

[22] M. Krasnoselskii and A. Pokrovskii, *Systems with hysteresis*: Nauka, Moscow, 1983.

[23] I. D. Mayergoyz, *Mathematical models of hysteresis and their applications*, 1st ed. Amsterdam ; Boston: Elsevier, 2003.