Scalar and vector hysteresis simulations using HysterSoft

M Dimian¹ and P Andrei²
¹Department of Computers, Electronics and Automation, Stefan cel Mare University, Suceava 720229, Romania
²Department of Electrical and Computer Engineering, Florida State University, FL 32310, USA
E-mail: pandrei@fsu.edu

Abstract. Hysteresis modeling has become an important research area with many applications in science and engineering. In this article we present a unified and robust simulation framework designed to perform scalar and vector hysteresis modeling. The framework is based on HysterSoft©, which is a simulation platform that can be interfaced with other libraries and simulation programs to model various aspects of hysteresis. We describe the main features of our simulation framework by focusing on scalar and vector hysteresis modeling, direct and inverse modeling, dynamic hysteresis modeling, first-order reversal-curves analysis, identification of the scalar and vector Preisach distribution function using an experimental first-order reversal-curves, noise passage analysis through hysteretic systems, and thermal relaxation in scalar and vector hysteresis. The simulation modules, the user-defined features, and various parameter identification techniques are also presented.

1. Introduction
Scalar and vector hysteresis can be found in many areas of engineering, physics, material science, biology, economics, and social sciences. Due to the wide spectrum of areas in which hysteresis is observed and to the fact that the origins of hysteresis are often multiple and unclear, there exist a large number of theoretical models for hysteresis in the literature. In this article we present, HysterSoft© - a robust simulation environment designed to perform scalar and vector hysteresis modeling [1]. The software was developed at Florida State University (Tallahassee, USA) in collaboration with research groups from other institutions worldwide: the Institute of Industrial Electronics and Material Science at Vienna Institute of Technology (Vienna, Austria), the Department of Computers, Electronics and Automation at Stefan cel Mare University (Suceava, Romania), and Department of Physics at Alexandru Ioan Cuza University (Iasi, Romania).

One of the most important features of HysterSoft© is its universality in the sense that it does not make reference to particular hysteretic systems such as magnetic or mechanic hysteresis, but focuses on the mathematical modeling of hysteresis in general. For this reason HysterSoft© can be used for both educational and research purposes and in a variety of disciplines and its applications may involve finite elements or other physics-based simulations. HysterSoft© is currently employed as the main editor for ferromagnetic and ferroelectric materials in RandFlux©, which is a device and circuit simulator developed at Florida State University [2]. Models defined in HysterSoft© can be easily imported in RandFlux© and used to simulate electronic circuits containing hysteretic inductors, transformers, and capacitors. In addition, HysterSoft© can also be used to compute scalar and vector first order reversal curves diagrams, identify the model parameters from experimental data, conduct temperature and stress...
dependent simulations, perform noise passage analysis in hysteretic systems, etc. Most of these simulations can be performed using any hysteretic model either predefined or user-defined. The program can also be used as a dynamic link library and called from other programs such as Matlab, Simulink, Java, or C++.

A list of the most important problems that can be addressed by using HysterSoft© is given below:

- Scalar and vector hysteretic processes as well as inverse hysteretic problems;
- Dynamic hysteresis phenomena such as rate or frequency dependent hysteretic processes;
- Computation of first order reversal curves (FORCs) and FORC diagrams;
- Identification of the Preisach distribution function using an experimental set of FORCs;
- Parameters identification tools for all implemented models (including the user-defined models);
- Thermal relaxations and after-effect phenomena in both scalar and vector hysteretic systems;
- Noise transfer through hysteretic systems;
- Iterative simulations in which one or more parameters are changing according to the specifications of the user.

The rest of the article is organized as follows. In section 2 we introduce the main hysteresis models that come by default implemented in HysterSoft©. The next two sections focus on how to perform simulations with scalar and vector models of hysteresis, respectively. Section 5 presents how users can define new models of hysteresis in HysterSoft© (user-defined models), section 6 presents how to compute the scalar and vector Preisach distribution function from FORCs, section 7 presents how to use HysterSoft© as a library and how to work with files, and section 8 concludes.

2. Hysteresis models predefined in HysterSoft©

The main models of hysteresis implemented in HysterSoft© are presented in Table I. A detailed description of the constitutive equations for each model, parameter definition, and numerical implementation can be found on the references mentioned in the Table for each model. Users can also easily implement and employ additional models to the program by using predefined templates.

| Scalar models          | Definition               | Can be used in vector hysteresis simulations? | Can be used in inverse modeling? |
|------------------------|--------------------------|-----------------------------------------------|----------------------------------|
| 1 Backslash operator   | [17]                     | yes                                           | yes                              |
| 2 Bouc-Wen model       | [8-10, 18-20]            | yes                                           | yes                              |
| 3 Coleman-Hodgdon model| [14, 15, 21-25]          | yes                                           | yes                              |
| 4 Elastic-plastic (stop) operator | [17] | yes                                           | yes                              |
| 5 Energetic modela     | [16, 26-30]              | yes                                           | yes                              |
| 6 Jiles-Atherton model | [11-13, 31-33]          | yes                                           | yes                              |
| 7 Langevin modelb      | [34]                     | yes                                           | yes                              |
| 8 Limiting-loop proximity model | [35] | yes                                           | yes                              |
| 9 Preisach modelc      | [3-7, 36-38]             | yes                                           | yes                              |
| 10 Rectangular loop operator | [17, 39] | no                                            | no                               |

a Also known as the Hauser model.
b Langevin model is not considered a model of hysteresis however it often represents the building block of other models of hysteresis such as the Jiles-Atherton model
c Preisach model can be defined either as a model in which the Preisach function is defined analytically or point-by-point, on a mesh.
3. Simulations using scalar models of hysteresis

HysterSoft© can perform different types of simulations by using the modules that are implemented in the program. These types of simulations can be performed for any hysteresis model, predefined or user-defined which justifies in part the popularity of the software. In this way, users need to concentrate only on developing the governing equations of the hysteresis model and not focus on adjusting it to a particular application. A module can be selected from the main combo-box in the scalar model window.

3.1. Modules available for scalar models of hysteresis

Module 'Input and output defined using the GUI'. Using this module, users can simulate hysteresis curves by varying the input in the case of direct hysteresis modeling or output in the case of inverse hysteresis modeling using the track bars from the graphical user interface. This module can be used to simulate only static hysteresis curves and visualize the current hysteretic state (e.g. the Preisach plane) in real time. Figure 1 presents such a simulation performed using the Preisach model.

![Figure 1](image)

Figure 1. Tracing hysteresis curves using module 'Input and output defined using the GUI' in the case of the Preisach model. The Preisach distribution is normally distributed along the interaction axis and log-normally distributed along the coercivity axis. The reversible component is also assumed to be normally distributed along the $\alpha=\beta$ axis.

3.1.1 Module 'Input defined from file'. Using this module, users can simulate hysteresis curves using the values of the input (or output in the case of inverse modeling) defined in a file or in the text editor. The input (or output) should be defined one value per line. HysterSoft© will parse these values and compute the hysteretic curves. The user can use symbolic notations for the main physical parameters like in the following listing:

```plaintext
//sample input file
0
P1
-Hc
```
In the above example HysterSoft© will start from 0, apply an input equal to the value of user-defined parameter P1 (which is set in the options property grid), then another value equal to minus the coercive field of the material, and finally 100. The final values of the input and output are stored in the FinalX and FinalY variables. This module can be used in loop simulations, but cannot be used to perform dynamic simulations.

3.1.2. Module 'Input defined analytically (dynamic hysteresis)'. Using this module, users can compute the output variable if the input is defined analytically. The input signal should be defined using the InputSignal editor in C#. When defining the input signal it is convenient to use variables such as tMin, tMax, Hc, P1, P2, P3, P4, etc. in order to easily modify the signal from the property grid. It is also convenient to use such variables when performing loop simulations. Figure 2 presents a rate-dependent simulation performed using the Jiles-Atherton model.

Figure 2. Sample rate-dependent simulations performed with the Jiles-Atherton model for a soft magnetic ferrite. The red line shows the rate-dependent hysteresis loop, while the blue line denotes the "static" major hysteresis loop. The model parameters are $a = 41.8$, $\alpha = 0$, $c = 0.15$, $k = 39.3$ and $y_{sat} = 10^5$.

This module allows performing rate-dependent simulations by setting the EffectiveField or RelaxationTime variables. When the EffectiveField variable is set, HysterSoft© will use the effective field approximation, and compute output $y$ using

$$y = \hat{\gamma}x_{eff},$$

$$x_{eff} = x + F(y, \dot{y}),$$

(1)

(2)
where $x$ is the input, $x_{\text{eff}}$ is the effective value of the input, $\hat{\Gamma}$ is the scalar hysteresis operator, and $F$ is a function of the output variable $y$ and of its derivative with respect to time, $\dot{y}$. Equations (1) and (2) represent a system of nonlinear equations that is solved to compute $y$. In the framework of the Preisach model, this system is a system of integro-differential equations, while in the framework of the Jiles, energetic and Hodgdon models it becomes a system of differential equations. When the RelaxationTime variable is set, HysterSoft© will use the relaxation time approximation and compute $y$ using:

$$\dot{y} = -\frac{y(t) - \hat{\Gamma} x(t)}{\tau},$$

where $\tau$ is a relaxation time parameter and $\hat{\Gamma}$ is any regular static (not rate-dependent) hysteresis operator.

The relaxation time should be a real number, while the effective field should be a string variable that defines the effective field as a function of the normalized rate (i.e. time derivative) of the output variable. For instance, by setting the EffectiveField variable to $1.2*dm$

HysterSoft© will use a total input equal to

$$x_{\text{eff}}(t) = x(t) + 1.2 \frac{\dot{y}(t)}{y_{\text{sat}}}$$

Rate-dependent simulations are particularly convenient for simulating frequency-dependent hysteresis.

3.1.3. Module 'FORCs computations'. Using this module, users can compute a set of first-order reversal-curves. One can define the maximum applied input ($X_{\text{Max}}$), the number of reversal curves ($\text{Reversals}$), and the number of points per curve ($\text{Resolution}$) in the property grid. Dynamic models cannot be used when computing FORCs. If one needs to compute FORCs using a dynamic model it is recommended to define the input signal analytically and use the previous module (0).

Figure 3 presents a sample FORCs computation using the limiting-loop proximity model.

**Figure 3.** A set of 100 FORCs computed using the limiting loop proximity model. The model parameters are $h_0 = 1$, $H_c = 1$, $y_{\text{sat}} = 1$, and $z = 0.9$. 
3.1.4. Module 'Thermal relaxation'. Using this module, users can perform simulations involving thermal relaxation. Thermal relaxation simulations are performed by adding a “thermal noise” to the input of a hysteresis model. The initial state can be obtained in two ways. The first way is to bring the hysteretic system in a given state by running a simulation with any other module (for instance any of the first 3 modules), then copying the hysteretic state, and using it as the initial state for the thermal relaxation simulation. The second way to set the initial state is to use the FirstFieldToApply and SecondFieldToApply variables as follows: before any thermal relaxation simulation HysterSoft\textsuperscript{©} starts from the zero anhysteretic state and applies an input equal to FirstFieldToApply, then another input equal to SecondFieldToApply, and, then, performs the thermal relaxation simulations. If one does not want to use these variables one needs to set the FirstFieldToApply and SecondFieldToApply to NaN.

One can select different types of noise to use in the thermal relaxation including Ornstein-Uhlenbeck, Gaussian, Laplace, Cauchy, and uniformly distributed noise by setting the TypeOfNoise variable. For each type of noise one can set the magnitude of the noise and different other parameters characteristic to the noise selected. In addition, one can change the spectrum of the noise by changing the NormalizeInputSpectrum variable. By default the NormalizeInputSpectrum is set to 1, which means that the input noise is multiplied by 1 (i.e. left unchanged), however the user can change this variable to a frequency dependent function in order to modify the color of the noise. In this way, one can use colored noise inputs such as pink, blue, or violet noises as input noises.

One can also define the number of total averages after which the expected values of the output variable is computed, the experiment time, and the time step of the thermal noise. After the simulation is performed, HysterSoft\textsuperscript{©} will set the ViscosityCoefficient variable to the value of the viscosity coefficient. This module can be used in loop simulations to analyze more complex phenomena such as data collapse, field-cooled or zero-field cooled processes.

Figure 4 presents a sample thermal relaxation simulations performed with the energetic and Preisach models. In both simulations we used a system with saturation equal to 1.62×10\textsuperscript{6} and coercivity approximately equal to 8. The simulations started with the initial anhysteretic state after which the input was increased to 10 we waited for the thermal relaxation to occur.

![Figure 4](image-url)

**Figure 4.** Simulations of thermal relaxation in the energetic model (left) and Preisach model (right), in which the input noise is Ornstein-Uhlenbeck with \( b = 1, \sigma = 5, \) and \( x_0 = 0. \) The parameters for the energetic model are \( c_r = 1.32, g = 8.24, h = 0.217, k = 15.9, y_{\text{sat}} = 1.62 \times 10^6, N_e = 10^6, \) and \( q = 15.3. \) In the case of the Preisach model we used a normal distributions along interaction field and coercivity axes with \( h_{\text{int}} = 13, h_{\text{co}} = 6, y_{\text{sat}} = 1.6 \times 10^6 \) and \( h_{\text{rev}} = 8, \) while the reversible distribution (along the \( \alpha=\beta \) axis) was normal with \( h_{\text{rev}} = 15. \)
3.1.5. Module 'Noise passage analysis'. Using this module, users can analyze the noise passage characteristics through hysteretic systems [40-45]. This module allows one to compute the output spectrum of any model of hysteresis if the input spectrum is given. The input spectrum can be of any of the types described in the previous module. This module can be used in loop simulations.

![Image of Noise spectrum analysis](attachment:image.png)

**Figure 5.** Noise passage analysis performed with the energetic model (left) and Preisach model (right), in which the input noise is Ornstein-Uhlenbeck with the same parameters as in the Figure 4. The model parameters are also identical to the ones used in Figure 4.

3.2. Loop simulations

Loop simulations refer to simulations that are performed multiple times, each time for a different set of model parameters or input signal. For instance, such simulations are convenient to perform when computing the anhysteretic curve, when computing hysteresis curves for different values of model parameters, or when simulating the thermal relaxation, noise passage, or stochastic resonance for different parameters of the input noise.

Loop simulation can be performed using the Loop Simulation Editor. By default, HysterSoft© performs two embraced iteration loops for two different parameters. The parameters and the values that these parameters should take during each simulation can be set in the “Major loop” and “Minor loop” tabs of the “Loop simulation” editor. The user can specify what variables to save on the hard disk, what curves from the output window to save, and what image files to produce after each iteration. The file names can be defined using parameters between two $ characters. For instance

```
Output_Hc_$Hc$.dat
```

will be expended by replacing $Hc$ with the current value of the coercive field.

3.3 Inverse modeling

Hysteresis models provide in general a procedure to compute the output variable $y(t)$ as a function of the input variable $x(t)$. In practical applications, it is often necessary to solve the inverse problem, in which one needs to compute the input variable that results in a given output. This is usually required when solving optimization problems or trying to design a hysteresis system.

Since hysteresis is a nonlinear and multi-valued function, the inverse problem might have no solution, one solution, or multiple solutions. One example is to invert the rectangular hysteresis operator, which, depending on the output variable, has no solution (if $y$ is different from 1 or -1) or infinity of
solutions (if $y$ is either 1 or -1). However, quite often it is possible to invert a hysteresis model relatively easily using the technique described below.

Suppose we have a hysteresis model defined as

$$y(t) = \hat{\Gamma}x(t)$$

where $\hat{\Gamma}$ is the hysteresis operator. We introduce the differential susceptibility of the system as

$$\dot{y}(t) = \hat{\chi}\dot{x}(t)$$

where $\hat{\chi}$ is the susceptibility. We suppose that such susceptibility operator exists, is finite, and has an inverse operator $\hat{\chi}^{-1}$ such that $\hat{\chi}^{-1}\hat{\chi} = 1$. In this case:

$$\dot{x}(t) = \hat{\chi}^{-1}\dot{y}(t)$$

In the case of scalar hysteresis models, $\hat{\chi}$ is the differential susceptibility and $\hat{\chi}^{-1} = 1/\hat{\chi}$. If the hysteresis model is a differential model of hysteresis, the computational overhead when solving the direct problem (6) is equal to the computational overhead when solving the inverse problem (7). Most scalar models of hysteresis used in this book can be written in the form of (6). Hence, it is possible to invert these models by using (7).

HysterSoft© implements inverse modeling for all the hysteresis models for which the differential susceptibility is defined, including the Preisach and the energetic models. Inverse modeling in HysterSoft© can be performed only by using the first two modules ('Input and output defined using the GUI' and 'Input defined from file'). If the values of the output are specified in an external file, the ModelingType variable should be set to InverseModeling.

In the case of user-defined models, inverse modeling can be performed only if the model implements the Susceptibility or the ChangeY functions (see section 5).

Figure 6. Non-iterative determination of parameters for the Energetic model [47].
3.4. Parameter identification tools

HysterSoft© implements three types of methods to determine the model parameters [33, 38, 46]:

1. evolutionary identification techniques such as swarm optimization, genetic algorithms, and the Nelder-Mead method;
2. the Levenberg–Marquardt algorithm;
3. iterative techniques.

In addition, HysterSoft© implements model specific, parameter identification tools. For instance Figure 6 shows the non-iterative parameter identification tool in the case of the energetic model) [47]. The values of the physical parameters that are be used in the identification problem (such as the coercive field, the output saturation, the remanence, initial susceptibility, etc.) can be specified on the graphical user interface.

In the case of user-defined models, the model parameters that appear in the parameter identification editor should be identified using the [IdentificationModelParameter] attribute.

4. Simulations using vector models of hysteresis

By default, HysterSoft© adds a two-dimensional vector model of hysteresis for most scalar models. The vector model is defined using superposition described in [39, 45, 48, 49]. For instance, two-dimensional vector models of hysteresis are constructed as a two-dimensional superposition of scalar models of hysteresis

\[ y(t) = \int_{-\pi/2}^{\pi/2} \hat{\phi} \hat{\Gamma}_\phi [x(t) \cdot \hat{\phi}] d\phi, \]  

where \( \hat{\phi} \) is the unit vector along the direction specified by polar angle \( \phi \), \( x \) is the vector input variable, and \( \hat{\Gamma}_\phi \) is the scalar hysteresis operator along direction \( \phi \). \( \hat{\Gamma}_\phi \) can be any scalar hysteresis operator such as the Preisach, Jiles-Atherton, energetic, or other scalar models. It should be mentioned that model (8) was initially introduced in [4] for the Preisach model. Due to the high computational cost required to integrate (8), this equation is relatively difficult to implement in real-time simulations for the Preisach model, however, it becomes more manageable when used with other models such as the energetic of Jiles-Atherton models.

Similar to the case of scalar models of hysteresis, HysterSoft© has a number of modules that can be used to perform vector simulations. These modules are summarized below.

4.1. Module 'Input and output defined using the GUI'. Simulate vector hysteresis curves by varying the input using the buttons on the graphical user interface (these buttons can also be pressed using the arrow buttons on the keyboard).

4.2. Module 'Input defined from file'. Simulate hysteresis curves by using the input from a file or from a textbox. Both the x and y components of the input should be defined one the same line. HysterSoft© will parse these values and compute the output variable. Notations for various parameters can also be used like in the next example

```plaintext
//sample input file for vector simulations
0, 0
Hcx, 0
Hcy, Hcy
P1, P2
0, -P3
```
In this example HysterSoft© starts from point (0, 0), than applies \((HcX, 0), (HcX, HcY), (P1, P2)\), and finally \((0, -P3)\). The values of \(HcX, HcY, P1, P2, P3\), etc. will be expended to the numerical values defined by the user in the property grid. The final value of the input and output components after the simulation are given by the \(\text{FinalX}_x, \text{FinalX}_y, \text{FinalY}_x, \text{and FinalY}_y\) variables. These variables can be used in loop simulations.

This module can also be used in inverse modeling by setting the \texttt{ModelingType} variable to \texttt{InverseModeling}. In this case, the input file should specify the values of the output of the hysteretic system, and HysterSoft© will compute the corresponding input values.

Figure 7 presents sample two-dimensional simulations using the Preisach and Bouc-Wen models. In both simulations the input was increased along the x-axis from 0 to 100 (in the case of the Preisach model) and to 20 (in the case of Bouc-Wen model) and then rotated in the x-y plane. The parameters of both models are distributed uniformly on the unit circle to simulate an isotropic material.

4.3. Module 'FORCs computations'. Compute vectorial FORCs, by specifying the maximum applied input (\(XMax\)), the number of reversal curves (\(\text{Reversals}\)), the number of points per curve (\(\text{Resolution}\)), and the angle under which the FORCs are computed (\(\text{Angle}\)). These values can be all be specified in the property grid. Dynamic models cannot be used when computing vectorial FORCs.

4.4. Module 'Thermal relaxation'. Perform two-dimensional thermal relaxation simulations. The initial hysteretic state can be specified by using the copy and paste features. The user can also specify the magnitude of the noise, the number of averages, and the number of points in which the thermal relaxation is computed.
Figure 8 presents the results of a two-dimensional simulation of the thermal relaxation using the energetic model. The left figure shows how the initial state was obtained using module 'Input and output defined using the GUI': starting from the anhysteretic point we increased the input along the x-direction to $5 \times 10^5$ and rotated it by 60 degrees in counterclockwise direction. The figure on the right shows the trajectory of the output which started in the initial state 1 and ended in final state 2 after 100 iterations. The input noise was assumed normally distributed standard deviation of $1 \times 10^4$. The trajectory of the output is represented with a blue line, while the input noise with represented in red.

**Figure 8.** Two-dimensional thermal relaxation in the energetic model. The left figure shows how the initial state of the system is obtained. The right figure shows the effect of a normally distributed input noise (represented with red) on the output vector (represented with blue).

5. **Defining new hysteresis models**

New scalar and vector hysteresis models can be easily defined in HysterSoft©. To define a new scalar model, start the “New Scalar Model” editor from the “Scalar Models” menu. Using this editor one can define the equations of the new model, properties that should appear in the property grid, default values for the model parameters, the parameters should appear in the parameter identification tool, etc. The easiest way to learn how to define new hysteresis models is to use examples that come with the installation kit.

The code of the new model should be written in C# and should contain the definition of the model as a class in C#. This class should be inherited from the UserDefinedModel class (which is internally implemented in HysterSoft©). Any user-defined model should define at least one of the following two functions

\[
\text{double Susceptibility(double x, double y, int sensOfVariation)}
\]

or

\[
\text{double ChangeX(double x)}
\]

The first function defines the susceptibility of the model, while the second describes how the output variable changes when the input becomes equal to $x$). The first function is convenient for use in differential models of hysteresis, while the second one in algebraic models. Once any of the above
functions is implemented, the user can perform any types of simulations presented above, including thermal relaxation and noise passage analysis.

Now let us look at an example that comes by default in HysterSoft©. This example implements the Jiles-Atherton model.

```csharp
public class MyModel : UserDefinedModel
{
    // a default constructor is optional
    public MyModel()
    {
        xMax = 100;
    }

    // Define the differential susceptibility
    public override double Susceptibility(double x, double y, int sensOfVariation)
    {
        double l, dl;
        l = SpecialFunctions.Langevin(x+alpha*y)-y;
        dl = SpecialFunctions.dLangevin(x+alpha*y);
        if (sensOfVariation*l <= 0)
            return c*dl;

        // return the differential susceptibility
        return (1-c)*l/(k*(1-c)*sensOfVariation-alpha*dl)+c*dl;
    }

    // Add some properties on the property grid
    private double a = 20;
    [IdentificationModelParameter(1, 100)]
    public double _a { get{return a;} set{a=value;} }

    private double c = 0.1;
    [IdentificationModelParameter(0, 1)]
    public double _c { get{return c;} set{c=value;} }

    private double k = 40;
    [IdentificationModelParameter(1, 100)]
    public double _k { get{return k;} set{k=value;} }

    private double alpha = 0;
    [IdentificationModelParameter(0, 1e-5)]
    public double _alpha { get{return alpha;} set{alpha=value;} }
}
```

The `Susceptibility` function is mandatory (because we did not implement the `ChangeX` function). The rest of all other functions and properties are optional, including the default constructor.

The `[IdentificationModelParameter(...)]` attribute for properties is also optional and tells HysterSoft© that the given property defines a model parameter that should appear in the Parameter Identification Tool. The two parameters of the `IdentificationModelParameter` attribute
denote the minimum and maximum values within which HysterSoft© can search during the parameter identification.

Functions `SpecialFunctions.Langevin` and `SpecialFunctions.dLangevin` are internally defined in HysterSoft© and they compute the Langevin function and the derivative of the Langevin function. The user can use his or her own implementation for these functions.

The `ToString` function defines a name for the current model.

Next let us look at another example that implements the Langevin model, which is an algebraic model.

```csharp
public class MyModel : UserDefinedModel
{
    public override double ChangeX(double x)
    {
        return Ms * SpecialFunctions.Langevin(x / 3);
    }
}
```

Notice that HysterSoft© will always define a variable `Ms`, which is by default equal to 1. Since hysteresis is a history dependent phenomenon, HysterSoft© allows users to recall the previous values of the input and output variables. The following predefined variables can be used from anywhere inside the class definition, including from inside functions `Susceptibility` and `ChangeX`:

- `state.X` – the last value of the input
- `state.Y` – the last value of the output
- `state.XReversal` – the last value of the input reversal
- `state.YReversal` – the last value of the input reversal
- `state.pastX[i]` – the previous `i`-th value of the input
- `state.pastY[i]` – the previous `i`-th value of the output

Notice in the last example that HysterSoft© will not be able to perform inverse modeling simulations because the class does not implement the `Susceptibility` or the `ChangeY` functions.

6. Computation of the Preisach distributions using first-order reversal-curves (FORCs)

FORCs have become an essential tool to characterize hysteretic systems [36, 49-52]. In magnetic materials they provide information about the magnetic interactions and particle coercivities and are often used as the "fingerprints" of a given system.

6.1. Scalar FORCs Analysis

The "Scalar FORCs Analysis" tool in HysterSoft© can be used to compute Everett distribution, the reversible $R(\alpha)$ and irreversible $P(\alpha, \beta)$ Preisach distributions if a set of first-order reversal-curves (FORCs) is provided. The set of FORCs should be given as a 3-column file with the values of the reversal field, current field, and current output for each curve, as in the example below:

```
10 10 1     : this is the first curve which consists of one point
9 9 0.99    : here starts the second reversal-curve which consists of 2 points
9 10 1
8 8 0.98    : here starts the third reversal-curve which consists of 3 points
8 9 0.99
8 10 1
... 
```
The user can specify any number of reversal curves, each having any number of points. The reversal fields and the points on each reversal curve can be distributed non-uniformly, however, in order to increase the accuracy of computations it is strongly recommended to use uniform distributions. HysterSoft© will automatically detect the number of reversal curves and points on each curve. FORCs should be given as ASCII files with extension .forcs.

The Everett distribution can be saved as a .everett file, while the computed Preisach distribution as a .preisach file. The structure of the .preisach file is similar to the structure of the .forcs file and consists of 3 columns giving the values of the reversible component of the Preisach distribution \( R(\alpha) \) when \( \alpha = \beta \) and the irreversible component of the Preisach distribution irreversible \( P(\alpha,\beta) \) when \( \alpha > \beta \). This is a sample .preisach file:

```
10 10 0.01 : R(10) = 0.01
9 9 0.02 : R(9) = 0.02
9 10 0 : the first point of the irreversible component P(9,10) = 0
8 8 0.03 : R(8) = 0.03
8 9 0.1 : the next point of the irreversible component P(8,9) = 0.1
8 10 0.02 : the next point of the irreversible component P(8,10) = 0.02
... 
-10 -10 0.1 : R(-10) = 0.1
-10 -9 -0.1 : P(-10,-9) = 0.1
... 
-10 10 0.1 : P(-10,10) = 0.1
```

The .everett and .forcs files can be loaded by the Preisach model with discrete Preisach distributions and used to perform frequency-dependent simulations or thermal and noise analysis.

6.2. Vector FORCs Analysis

To identify the Preisach distributions of the vector Preisach model, HysterSoft© allows users to load multiple .forcs files and vector distributions. This can be done using the "Load FORCs Analysis" and "Load Multiple FORCs" tools. The FORCs in each angular direction need to be specified in different files and loaded in the software. HysterSoft© can compute the Preisach distribution function for both two-dimensional and three-dimensional FORCs in the case of isotropic hysteretic systems, however it can compute only the Preisach diagram for only two-dimensional FORCs in the case of anisotropic systems. Notice that two-dimensional FORCs should be specified along each polar angle \( \phi \), while three-dimensional FORCs should be specified along each polar and azimuthal angles \( \phi \) and \( \theta \).
Figure 9. Computation of the reversible and irreversible components of the scalar Preisach distribution function from a set of FORCs.

Figure 10. Representation of the Preisach distribution function computed in Figure 9.
7. Other features

7.1. Using HysterSoft© as a library
Most HysterSoft© functions can be called from other Windows applications (such as Matlab, C, Fortran, Java, etc.). There are two ways to call HysterSoft© functions from other programs:

- load the HysteresisLibrary.dll file as a .NET assembly, create a desired model, and call the public functions and properties (recommended)
- use HysterSoft© as a COM object (not recommended; this way is obsolete and is kept for compatibility with old versions of HysterSoft©).

Examples of how to call HysterSoft© from Matlab can be found in the /Matlab subdirectory of the installation directory of HysterSoft©.

7.2. Saving the model parameters
The parameters of any model of hysteresis can be saved in .xml files that later can be re-loaded and used in other simulations. By default, HysterSoft© assigns the .hyst file extension to a model parameter file. Models parameters files are in general compatible from one version of HysterSoft© to another.

7.3. Perform temperature and stress dependent simulations
Temperature and mechanical stress change the shape of the hysteresis loops in most magnetic and electric hysteretic systems. For instance, in the case of magnetic systems, increasing the temperature will usually decrease the value of the coercive field, output remanence and saturation, while increasing the stress will increase the output remanence and make the shape of the major loop more rectangular. For instance, Figure 11 presents the results of increasing stress on the major hysteresis loop computed using the energetic model.

Figure 11. Major hysteresis loops computed using the energetic model at different values of the stress.

8. Conclusion
HysterSoft© is a computer software for the simulation of hysteresis and related phenomena in hysteretic systems. It provides a user-friendly simulation framework, in which various mathematical models of hysteresis can be implemented easily. A number of hysteresis models such as the backlash operator, the Bouc-Wen model, the Coleman-Hodgdon model, the stop (elastic-plastic) operator, the energetic model,
the Jiles-Atherton model, the Langevin model, the limiting-loop proximity model, the Preisach model (including the cases in which the Preisach distribution is specified analytically or using discrete values defined on a mesh), and the rectangular-loop model are already implemented in the program. However, users can also implement and employ additional models by using predefined templates.

HysterSoft© can also be used to compute first-order reversal-curves (FORCs) diagrams, identify the model parameters from experimental data, conduct temperature and stress dependent simulations, perform noise passage analysis in hysteretic systems, etc. The program can be used as a library and called from other programs such as Matlab, Java, or C++. Finally, HysterSoft© can be used to define hysteresis models for the electric permittivity and magnetic permeability. These models can be imported directly in RandFlux© and used to simulate electronic circuits containing hysteretic inductors, transformers, and capacitors. A list of the most important features in HysterSoft© was also presented. Since the submission of this article the authors have finalized and published book [53], which also contains numerous simulation results based on HysterSoft©, as well as a short account of this simulation environment in the Appendix.

9. Acknowledgement
This paper was partially supported by the project "Progress and development through postdoctoral research and innovation in engineering and applied sciences - PRiDE - Contract no. POSDRU/89.1.5/S/57083", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

10. References
[1] HysterSoft, (Florida A&M University and Florida State University, Tallahassee, FL, Florida A&M University and Florida State University).
[2] RandFlux, (Florida A&M University and Florida State University, Tallahassee, FL, Florida A&M University and Florida State University), pp. User manual v. 0.6.
[3] Preisach F 1935 Z Phys 94 277-302.
[4] Mayergoyz I 1991 Mathematical Models of Hysteresis. (Springer-Verlag).
[5] Visintin A 1995 Differential Models of Hysteresis (Applied Mathematical Sciences). (Springer).
[6] Brokate M and Sprekels J 1996 Hysteresis and Phase Transitions. (Springer).
[7] Bertotti G and Mayergoyz I 2005 The Science of Hysteresis. (Elsevier).
[8] Bouc R 1971 Acustica 24 (1) 16-25.
[9] Wen Y K 1976 J Eng Mech Div-Asce 102 (2) 249-263.
[10] Duhem P 1897 Z. Phys. Chem. 22 543.
[11] Jiles D C and Atherton D L 1983 IEEE Transactions on Magnetics 19 (5) 2183-2185.
[12] Jiles D C and Atherton D L 1984 Journal of Applied Physics 55 (6) 2115-2120.
[13] SPICE, (http://bwrc.eecs.berkeley.edu/classes/icbook/spice/, University of California at Berkeley).
[14] Coleman B D and Hodgdon M L 1987 Archive for Rational Mechanics and Analysis 99 (4) 375-396.
[15] Coleman B D and Hodgdon M L 1986 International Journal of Engineering Science 24 (6) 897-919.
[16] Hauser H 1994 Journal of Applied Physics 75 (5) 2584-2596.
[17] Visintin A 2005 Mathematical models of hysteresis. (Elsevier Inc.).
[18] Bouc R, presented at the Proceedings of the Fourth Conference on Nonlinear Oscillation, Prague, Czechoslovakia, 1967 (unpublished).
[19] Wen Y K 1976 J Struct Div-Asce 102 (5) 1160-1162.
[20] Ikhouane F and Rodellar J 2007 Systems with hysteresis. (John Wiley & Sons, Ltd.).
[21] Hodgdon M L 1988 IEEE Transactions on Magnetics 24 (6) 3120-3122.
[22] Hodgdon M L 1988 IEEE Transactions on Magnetics 24 (1) 218-221.
[23] Stancu A, Caltun O and Andrei P 1997 Journal De Physique IV 7 (C1) 209-210.
[24] Andrei P, Stancu A and Adedoyin A 2006 Journal of Optoelectronics and Advanced Materials 8 (3) 988-990.
[25] Andrei P and Adedoyin A 2010 Mathematics and Computers in Simulation 80 (6) 1045-1055.
[26] Hauser H 1994 Journal of Magnetism and Magnetic Materials 133 (1-3) 32-35.
[27] Hauser H 2004 Journal of Applied Physics 96 (5) 2753-2767.
[28] Andrei P, Stoleriu L and Hauser H 2000 Journal of Applied Physics 87 (9) 6555-6557.
[29] Grossinger R, Hauser H, Stoleriu L and Andrei P 2000 IEEE Transactions on Magnetics 36 (5) 3357-3359.
[30] Andrei P, Stancu A, Hauser H and Fulmek P 2007 Journal of Optoelectronics and Advanced Materials 9 (4) 1137-1139.
[31] Jiles D C and Atherton D L 1986 Journal of Magnetism and Magnetic Materials 61 (1-2) 48-60.
[32] Andrei P and Stancu A 1999 Journal of Magnetism and Magnetic Materials 206 (3) 160-164.
[33] Andrei P, Oniciuc L, Stancu A and Stoleriu L 2007 Journal of Magnetism and Magnetic Materials 316 (2) E330-E333.
[34] Kittel C 2005 Introduction to Solid State Physics 8th ed. (John Wiley & Sons, Inc.).
[35] de Almeida L A L, Deep G S, Lima A M N and Neff H 2003 IEEE Transactions on Magnetics 39 (1) 523-526.
[36] Ionita V, Petrescu L, Bordianu A, Tabara O, 2013 Adv. Electr. Comp. Eng. 13 (2), 121-126.
[37] Andrei P and Stancu A 2000 IEEE Transactions on Magnetics 36 (4) 1982-1989.
[38] Mayorgoyz I 2003 Mathematical Models of Hysteresis and Their Applications: Electromagnetism. (Academic Press).
[39] Adedoyin A, Dimian M and Andrei P 2009 IEEE Transactions on Magnetics 45 (10) 3934-3937.
[40] Dimian M, Adedoyin A, Gindulescu A and Andrei P 2009 IEEE Transactions on Magnetics 45 (11) 5231-5234.
[41] Dimian M, Adedoyin A, Gindulescu A and Andrei P 2010 IEEE Transactions on Magnetics 46 (2) 266-269.
[42] Dimian M and Andrei P 2011 Journal of Applied Physics 109 (7) 07D330-333
[43] Dimian M, Andrei P, Manu O and Popa V 2011 IEEE Transactions on Magnetics 47 (10) 3825-3828.
[44] Andrei P and Adedoyin A 2008 Journal of Applied Physics 103 (7) 07D913.
[45] Jiles D C, Thoelke J B and Devine M K 1992 IEEE Transactions on Magnetics 28 (1) 27-35.
[46] Andrei P and Adedoyin A 2009 Journal of Applied Physics 105 (7) 07D523.
[47] Stancu A, Stoleriu L and Andrei P 2007 Journal of Magnetism and Magnetic Materials 316 (2) E309-E312.
[48] Stoleriu L, Andrei P and Stancu A 2008 Journal of Applied Physics 103 (7) 07D923-923
[49] Stancu A and Andrei P 2006 Physica B-Condensed Matter 372 (1-2) 72-75.
[50] Stancu A, Andrei P and Stoleriu L 2006 Journal of Applied Physics 99 (8) 08D702.
[51] Stoleriu L, Postolache P, Andrei P and Stancu A 2010 Optoelectron Adv Mat 4 (4) 505-508.
[52] Dimian M, Andrei P 2014 Noise-Driven Phenomena in Hysteretic Systems. (Springer).