Efficient Implementation Algorithm for a Homogenized Energy Model with Thermal Relaxation

Thomas R. Braun[*] and Ralph C. Smith[†]

Center for Research in Scientific Computation / Department of Mathematics
North Carolina State University, Raleigh, NC 27695

ABSTRACT

In this paper, we present a new algorithm to implement the homogenized energy hysteresis model with thermal relaxation for both ferroelectric and ferromagnetic materials. The approach conserves most of the accuracy of the original algorithm, but enables all `erfc` and `exp` functions to be calculated in advance, thereby requiring that only basic mathematical operations be performed in real time. This is done without a significant increase in memory usage. Using this approach, execution time of the model has been seen to improve by a factor of 70 for some applications, whereas the error only increases by five ten thousandths (0.05%) of the saturation polarization/magnetization. The model with negligible relaxation is also given, as it is used to illustrate some optimizations. Emphasis is placed on the efficient computation of these models, and theoretical development is left to the references.

Keywords: Ferromagnetic, ferroelectric, thermal relaxation, thermal activation, real-time, homogenized energy, hysteresis model

1. BACKGROUND

The homogenized energy framework provides a powerful and flexible mechanism to model ferroelectric and ferromagnetic hysteresis. In this framework, Gibb's energy minimization is considered at the lattice level, using the kernel

$$\bar{P}(E) = \frac{E}{\eta} + P_R\delta, \tag{1}$$

where E is the electric field, $\bar{P}$ is the average polarization, P_R is the polarization at remanence, η is the inverse susceptibility, $\delta = -1$ for negatively oriented dipoles, and $\delta = 1$ for those dipoles with positive orientation. For magnetic materials, the kernel is

$$\bar{M}(H) = \frac{H\ \mu_0}{\eta} + M_R\delta, \tag{2}$$

where H is the magnetic field, M is the magnetization, M_R is the magnetization at remanence, and μ_0 is the permeability. In practice, however, η is the result of a data fit, and μ_0 is combined with η when performing the parameter estimation for magnetic materials. This allows the same model to be used for both classes of materials. For simplicity, this paper refers solely to electric fields and polarizations, with the understanding that everything presented here applies not only to electric but also to magnetic materials.

To account for material nonhomogenuities and interactions between dipoles, both the coercive field value at which δ changes and the interaction field between dipoles are assumed to be manifestations of an underlying distribution. The requirements placed on these distributions are minimal: the coercive field distribution E_c is strictly non-nonnegative, the interactive field distribution E_I is symmetric about 0, and both distributions are bounded by a decreasing exponential. Including these distributions gives the model

$$P(E) = \int_0^\infty \int_{-\infty}^\infty v_c(E_c)v_i(E_I)\bar{P}(E + E_I; E_c)dE_I dE_c, \tag{3}$$

[*]Email: tbraun@pobox.com; Telephone: 919-854-2746
[†]Email: rsmith@eos.ncsu.edu; Telephone: 919-515-7552

```
# Inputs: $E$, $E_c$, $w_c$, $N_c$, $E_I$, $w_I$, $N_I$, $P_R$, $\eta$, $\delta$
# Output: $P$
For $k = 0 \dots \text{length}(E) - 1$
  $P[k] = 0$
  For $i = 0 \dots N_c - 1$
    For $j = 0 \dots N_I - 1$
      If $E_I[j] + E[k] + E_c[i]\delta[i,j] > 0$
        $\delta[i,j] = 1$
        $P[k] = P[k] + w_c[i]w_I[j]((E_I[j] + E[k])/\eta + P_R)$
      Else
        $\delta[i,j] = -1$
        $P[k] = P[k] + w_c[i]w_I[j]((E_I[j] + E[k])/\eta - P_R)$
      End If
    End For
  End For
End For
```

Algorithm 1. Algorithm used to implement the homogenized energy model with negligible thermal relaxation,

where v_c and v_i represent the distributions of coercive and interaction fields, respectively. For computation, the distribution of parameters is considered at a predetermined number of quadrature points, and in general the only restriction on the quadrature method is that an even number of quadrature points are needed on the E_I axis. This assures accurate modelling of the material in the depoled state. Implementing the quadrature method gives the model

$$P(E) = \sum_{i=1}^{N_c} \sum_{j=1}^{N_I} w_c[i] w_I[j] \bar{P}(E + E_I; E_c), \tag{4}$$

where N_c is the number of coercive field quadrature points, N_I is the number of interaction field quadrature points, $w_c = v_c \times$ *coercive quadrature weights*, and w_I is defined similarly for the interaction field. The [ ] notation represents array indexing. A complete discussion of the development of this model can be found in [1, 3, 2, 4], and a pseudocode implementation is given in Algorithm 1 and Table 1.

Algorithm 1 assumes the effects of thermal relaxation are negligible. Considering a material at absolute temperature T with volume V, the effect of thermal relaxation or activation depends on the ratio kT/V, where k is Boltzmann's constant. When this value is very small (equivalently, when the relative lattice volume is sufficiently large), thermal relaxation is negligible and the previous model may be used. When this is not the

Name	Type	Description
E	Vector	Input electric field values – constant sampling rate assumed
E_c	Vector	Quadrature points for coercive field distribution
w_c	Vector	Quadrature weights times distribution levels for E_c integration
N_c	Scalar integer	Number of coercive field quadrature points/weights
E_I	Vector	Quadrature points for interactive field distribution
w_I	Vector	Quadrature weights times distribution levels for E_I integration
N_I	Scalar integer	Number of interactive field quadrature points/weights
P_R	Scalar	Polarization at remanence – determined from parameter estimation
Δt	Scalar	Time between successive samples of E
η	Scalar	Determined from parameter estimation
β	Scalar	Determined from parameter estimation – Note $\beta = \sqrt{2}kT/V$
τ	Scalar	Determined from parameter estimation
ϵ	Scalar	Small positive constant, on order of 1×10^{-3}
f	Scalar	Extremely small positive constant, on order of machine accuracy limits
δ	$N_c \times N_I$ matrix	Initial state of the material – 1 if domain is positive; -1 if negative
x_+	$N_c \times N_I$ matrix	Initial state of the material – 1 if domain is positive; 0 if negative
r	Scalar integer	Resolution increase for shifting operations

Table 1. Input parameters used by various algorithms in this paper. Note: some algorithms do not use all parameters.

case, we consider the Boltzmann relation

$$\mu(G) = C\mathtt{exp}\left(\frac{-G(P)V}{kT}\right), \tag{5}$$

where G is the Gibb's energy. The local average polarization is given by replacing $\bar{P}$ in [4] with

$$\bar{P}(E) = x_+\bar{P}_+ + x_-\bar{P}_-, \tag{6}$$

where x_+ and x_- are the fraction of moments having positive and negative orientations, respectively. This can be simplified slightly by realizing that $x_- = 1 - x_+$. The average polarizations are given by

$$P_+ = \int_{P_I}^{\infty} P\mu(G(P))dP, \quad P_- = \int_{-\infty}^{P_I} P\mu(G(P))dP, \tag{7}$$

where P_I is the inflection point. The moment fractions evolve according to the differential equation

$$\dot{x}_+ = -p_{+-}x_+ + p_{-+}x_-, \tag{8}$$

where p_{+-} and p_{-+} represent the likelihood of dipoles switching from positive to negative and from negative to positive, respectively. These likelihoods are given by

$$p_{+-} = \frac{\int_{P_I-\epsilon}^{P_I+\epsilon} \mathtt{exp}(-G(P)V/(kT))dP}{\tau\int_{P_I-\epsilon}^{\infty} \mathtt{exp}(-G(P)V/(kT))dP}, \quad p_{-+} = \frac{\int_{-P_I-\epsilon}^{-P_I+\epsilon} \mathtt{exp}(-G(P)V/(kT))dP}{\tau\int_{-\infty}^{-P_I-\epsilon} \mathtt{exp}(-G(P)V/(kT))dP}. \tag{9}$$

Details regarding the development of these equations is given in [4], and the resulting model is implemented in Algorithm 2 with parameters as described in Table 1. Note that the calculation of these terms involves evaluating complementary error (`erfc`) and exponential (`exp`) functions. This significantly reduces the computational efficiency of the model when thermal relaxation is included.

```
# Inputs: $E$, $E_c$, $w_c$, $N_c$, $E_I$, $w_I$, $N_I$, $P_R$, $\eta$, $\beta$, $\tau$, $\epsilon$, $f$, $x_+$
# Output: $P$
For $i = 0 \ldots N_c - 1$
  $P_I[i] = P_R - E_c[i]/\eta$
End For
For $k = 0 \ldots \text{length}(E) - 1$
  $P[k] = 0$
  For $i = 0 \ldots N_c - 1$
    For $j = 0 \ldots N_I - 1$
      $P^-_{min} = (E[k] + E_I[j])/\eta - P_R$
      $P^+_{min} = (E[k] + E_I[j])/\eta + P_R$
      If $E_c[i] - E[k] - E_I[j] > 0$
        $p_{-+} = (\mathtt{erfc}((P^-_{min} + P_I[i] - \epsilon)/\beta) -$
            $\mathtt{erfc}((P^-_{min} + P_I[i] + \epsilon)/\beta))/$
            $(\tau\mathtt{erfc}((P^-_{min} + P_I[i] - \epsilon)/\beta) + f)$
      Else
        $p_{-+} = 1/\tau$
      End If
      If $E_c[i] + E[k] + E_I[j] > 0$
        $p_{+-} = (\mathtt{erfc}((P_I[i] - \epsilon - P^+_{min})/\beta) -$
            $\mathtt{erfc}((P_I[i] + \epsilon - P^+_{min})/\beta))/$
            $(\tau\mathtt{erfc}((P_I[i] - \epsilon - P^+_{min})/\beta) + f)$
      Else
        $p_{+-} = 1/\tau$
      End If
      $x_+[i,j] = \dfrac{x_+[i,j] + p_{-+}\Delta t}{1 + \Delta t(p_{+-} + p_{-+})}$
      $\bar{P}_- = -\dfrac{\beta\mathtt{exp}(-((-P_I[i] - \epsilon - P^-_{min})/\beta)^2)}{\sqrt{\pi}\mathtt{erfc}((P_I[i] + \epsilon + P^-_{min})/\beta) + f} + P^-_{min}$
      $\bar{P}_+ = \dfrac{\beta\mathtt{exp}(-((P_I[i] + \epsilon - P^+_{min})/\beta)^2)}{\sqrt{\pi}\mathtt{erfc}((P_I[i] + \epsilon - P^+_{min})/\beta) + f} + P^+_{min}$
      $P[k] = P[k] + w_c[i]w_I[j](x_+[i,j]\bar{P}_+ + (1 - x_+[i,j])\bar{P}_-)$
    End For
  End For
End For
```

Algorithm 2. Algorithm for the homogenized energy model which includes thermal relaxation.

2. REPETITIVE OPERATIONS

In a first step to improve the efficiency of the model, several repetitive operations can be removed. For simplicity, consider first the model with negligible relaxation. For each iteration and each mesh point,

$$P[k] = P[k] + w_c[i]w_I[j]\left(\frac{E_I[j] + E[k]}{\eta} \pm P_R\right) \tag{10}$$

must be calculated. Note that E_I does not change through the course of the algorithm, and that dE is the same for all mesh points in a given temporal iteration. Thus, each point in E_I may be divided through by η as part of algorithm setup, and $E[k]$ divided just once per iteration (this division will later be removed). This process transforms (10) into

$$\begin{aligned} P[k] &= P[k] + w_c[i]w_I[j](E_I[j] + E[k] \pm P_R) \\ &= P[k] + w_c[i]w_I[j]E_I[j] + w_c[i]w_I[j]E[k] \pm w_c[i]w_I[j]P_R. \end{aligned} \tag{11}$$

Note that each mesh point adds three terms to $P[k]$. The first, $w_c[i]w_I[j]E_I[j]$, is independent of the input field, and a scalar value for $\sum_i \sum_j w_c[i]w_I[j]E_I[j]$ may be calculated and stored a priori. The second does depend on the input field, but only as a scalar multiple. As such, this term can be simplified to one multiplication per temporal iteration by calculating the scalar term $\sum_i \sum_j w_c[i]w_I[j]$ in advance. In actuality, we compute $(\sum_i \sum_j w_c[i]w_I[j]E_I[j])/\eta$ and $(\sum_i \sum_j w_c[i]w_I[j])/\eta$, to combine this simplification with the division mentioned above. The final term is $\pm w_c[i]w_I[j]P_R$, where the sign depends on the field level and the current state of the material. This must still be done in iteration. However, we may pre-multiply either w_c or w_I by P_r, and remove a multiplication from the loop. Further, the multiplication by $w_c[i]$ need not be performed for every mesh point, but only once per row (only when i changes). This requires accumulating the $w_I[j]$ terms into a temporary register, but this action is typically much more efficient than a multiplication. Incorporating these optimizations into the negligible relaxation model yields Algorithm 3. While some of these changes would be done anyway by an optimizing compiler (exactly which ones depends on the compiler), most compilers would not perform all these steps, and making these changes can give a noticeable improvement in computational efficiency. We found that Algorithm 3 runs almost twice as quickly as Algorithm 1 when both were compiled under the gcc compiler with optimizations enabled. More performance comparisons may be found in Section 4.

```
# Inputs: $E$, $E_c$, $w_c$, $N_c$, $E_I$, $w_I$, $N_I$, $P_R$, $\eta$, $\delta$
# Output: $P$
# Initial Setup – not input field dependent
$addit = 0$
$w_{sum} = 0$
For $i = 0 \ldots N_c - 1$
  For $j = 0 \ldots N_I - 1$
    $addit = addit + w_c[i]w_I[j]E_I[j]$
    $w_{sum} = w_{sum} + w_c[i]w_I[j]$
  End For
End For
$addit = addit/\eta$
$w_{sum} = w_{sum}/\eta$
For $j = 0 \ldots N_I - 1$
  $w_I[j] = w_I[j]P_R$
End For
# Begin Iteration
For $k = 0 \ldots \text{length}(E) - 1$
  $dE = E[k] - E[0]$
  $P[k] = addit + w_{sum}dE$
  For $i = 0 \ldots N_c - 1$
    $out = 0$
    For $j = 0 \ldots N_I - 1$
      If $E_I[j] + dE + E_c[i]\delta[i,j] > 0$
        $\delta[i,j] = 1;$
        $out = out + w_I[j]$
      Else
        $\delta[i,j] = -1$
        $out = out - w_I[j]$
      End If
    End For
    $P[k] = P[k] + w_c[i]\ out$
  End For
End For
```

Algorithm 3. Implementation algorithm for the Homogenized energy model with negligible thermal relaxation – repeated operations removed.

To remove repeated operations from the relaxation model, we do not calculate P_I, P^-_{min}, and P^+_{min} directly, as in Algorithm 2. This change effects the equations for p_{-+}, p_{+-}, $\bar{P}_-$, and $\bar{P}_+$. For example,

$$\begin{aligned} p_{-+} &= \frac{\texttt{erfc}((P^-_{min} + P_I[i] - \epsilon)/\beta) - \texttt{erfc}((P^-_{min} + P_I[i] + \epsilon)/\beta)}{\tau\texttt{erfc}((P^-_{min} + P_I[i] - \epsilon)/\beta) + f} \\ &= \frac{1}{\tau}\left(1 - \frac{\texttt{erfc}((E[k] + E_I[j] - E_c[i])/(\eta\beta) + \epsilon/\beta)}{\texttt{erfc}((E[k] + E_I[j] - E_c[i])/(\eta\beta) - \epsilon/\beta) + f}\right), \end{aligned} \tag{12}$$

where the fact that f is essentially 0 has been used to remove an $\texttt{erfc}$ calculation. We now see that division by η and β can be done in advance (for all terms except $E[k]$) or once per temporal iteration (for $E[k]$). The same holds for p_{+-}. Further, let

$$\begin{aligned} \bar{P}_- &= \widehat{P}_- + P^-_{min} = \widehat{P}_- + \frac{E[k] + E_I[j]}{\eta} - P_R, \\ \widehat{P}_- &= -\frac{\beta\texttt{exp}(-((E_c[i] - E_I[j] - E[k])/(\eta\beta) - \epsilon/\beta)^2)}{\sqrt{\pi}\texttt{erfc}((-E_c[i] + E_I[j] + E[k])/(\eta\beta) + \epsilon/\beta) + f}, \end{aligned} \tag{13}$$

$$\begin{aligned} \bar{P}_+ &= \widehat{P}_+ + P^+_{min} = \widehat{P}_+ + \frac{E[k] + E_I[j]}{\eta} + P_R, \\ \widehat{P}_+ &= \frac{\beta\texttt{exp}(-((-E_c[i] - E_I[j] - E[k])/(\eta\beta) + \epsilon/\beta)^2)}{\sqrt{\pi}\texttt{erfc}((-E_c[i] - E_I[j] - E[k])/(\eta\beta) + \epsilon/\beta) + f}. \end{aligned} \tag{14}$$

This allows η and β to be divided through in advance for these equations, and simplifies the polarization relation to

$$\begin{aligned} P[k] &= P[k] + w_c[i]w_I[j](x_+[i,j]\bar{P}_+ + (1 - x_+[i,j])\bar{P}_-) \\ &= P[k] + w_c[i]w_I[j]\left(x_+[i,j]\left(\widehat{P}_+ - \widehat{P}_- + 2P_R\right) + \widehat{P}_- + \left(\frac{E_I[j]}{\eta} - P_R\right) + \frac{E[k]}{\eta}\right). \end{aligned} \tag{15}$$

This in turn allows the $w_c[i]w_I[j](E_I[j]/\eta - P_R)$ and $w_c[i]w_I[j]E[k]/\eta$ to be calculated in advance, as was done with the negligible relaxation model. Incorporating these optimizations yields Algorithm 4. Numerical experiments demonstrate that Algorithm 4 runs about 15% to 20% faster than Algorithm 2 (again, more detailed results can be seen in Section 4). However, more significant then the speed increase given by Algorithm 4 is the framework it establishes for the more significant optimizations given in the next section. It should be noted that Algorithms 3 and 4 are equivalent (within numerical limits) to Algorithms 1 and 2, respectively; we have not lost any accuracy by making these changes.

3. CALCULATION OF LIKELIHOODS AND AVERAGE POLARIZATIONS

Whereas some improvement in the relaxation algorithm was gained in the previous section, Algorithm 4 is still many times slower than Algorithm 3. Much of this time is spent in calculating the $\texttt{erfc}$ and $\texttt{exp}$ functions, which occur in the equations for p_{-+}, p_{+-}, $\widehat{P}_-$, and $\widehat{P}_+$. Examining the equations for p_{-+} and $\widehat{P}_-$, the only non-constant terms are $E[k]$, E_I, and E_c, and these always occur together in the same pattern. As such, we write them as a function of a single variable

$$E_\mu = E[k] + E_I[j] - E_c[i]. \tag{16}$$

This gives p_{-+} and $\widehat{P}_-$ as functions of a single variable, which are calculated as

$$\widehat{P}_- = -\frac{\beta\texttt{exp}(-(-E_\mu - \epsilon)^2)}{\sqrt{2}\texttt{erfc}(E_\mu + \epsilon) + f}, \tag{17}$$

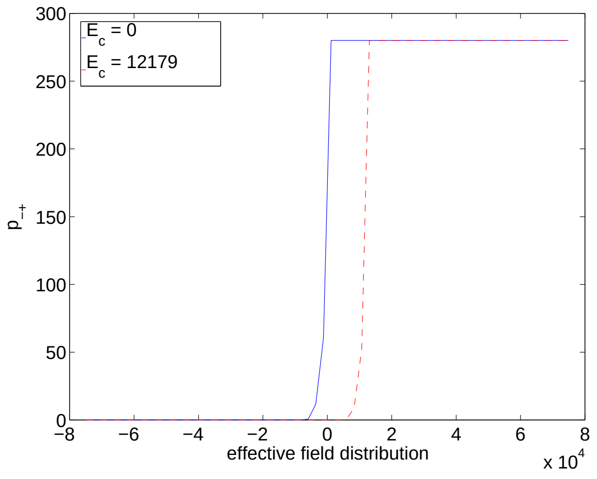
```
# Inputs: $E$, $E_c$, $w_c$, $N_c$, $E_I$, $w_I$, $N_I$, $P_R$, $\eta$, $\beta$, $\tau, \epsilon$, $f$, $x_+$
# Output: $P$
# Initial Setup – not input field dependent
$\epsilon = \epsilon/\beta$
For $j = 0 \ldots N_I - 1$
  $E_I[j] = E_I[j]/(\eta\beta)$
End For
$addit = 0$
$w_{sum} = 0$
For $i = 0 \ldots N_c - 1$
  $E_c[i] = E_c[i]/(\eta\beta)$
  For $j = 0 \ldots N_I - 1$
    $addit = addit + w_c[i]w_I[j]E_I[j]$
    $w_{sum} = w_{sum} + w_c[i]w_I[j]$
  End For
End For
$addit = \beta\ addit - P_R w_{sum}$
$w_{sum} = w_{sum}\beta$
# Begin Iteration
For $k = 0 \ldots \text{length}(E) - 1$
  $E[k] = E[k]/(\eta\beta)$
  $P[k] = addit + w_{sum}E[k]$
  For $i = 0 \ldots N_c - 1$
    $out = 0$
    For $j = 0 \ldots N_I - 1$
      $tmp = \texttt{erfc}(E[k] + E_I[j] - E_c[i] + \epsilon)$
      If $E_c[i] - E[k] - E_I[j] > 0$
        $p_{-+} = \frac{1}{\tau}(1 - \frac{tmp}{\texttt{erfc}(E[k] + E_I[j] - E_c[i] - \epsilon) + f})$
      Else
        $p_{-+} = \frac{1}{\tau}$
      End If
      $\widehat{P}_- = -\frac{\beta \texttt{exp}(-(-E[k] - E_I[j] + E_c[i] - \epsilon)^2)}{\sqrt{\pi}\ tmp + f}$
      $tmp = \texttt{erfc}(-E[k] - E_I[j] - E_c[i] + \epsilon)$
      If $E_c[i] + E[k] + E_I[j] > 0$
        $p_{+-} = \frac{1}{\tau}(1 - \frac{tmp}{\texttt{erfc}(-E[k] - E_I[j] - E_c[i] - \epsilon) + f})$
      Else
        $p_{+-} = \frac{1}{\tau}$
      End If
      $\widehat{P}_+ = \frac{\beta \texttt{exp}(-(-E[k] - E_I[j] - E_c[i] + \epsilon)^2)}{\sqrt{\pi}\ tmp + f}$
      $x_+[i,j] = \frac{x_+[i,j] + p_{-+}\Delta t}{1 + \Delta t(p_{+-} + p_{-+})}$
      $out = out + w_I[j](x_+[i,j](\widehat{P}_+ - \widehat{P}_- + 2P_R) + \widehat{P}_-)$
    End For
    $P[k] = P[k] + w_c[i]\ out$
  End For
End For
```

Algorithm 4. Implementation of the homogenized energy algorithm including thermal relaxation – repeated operations removed.

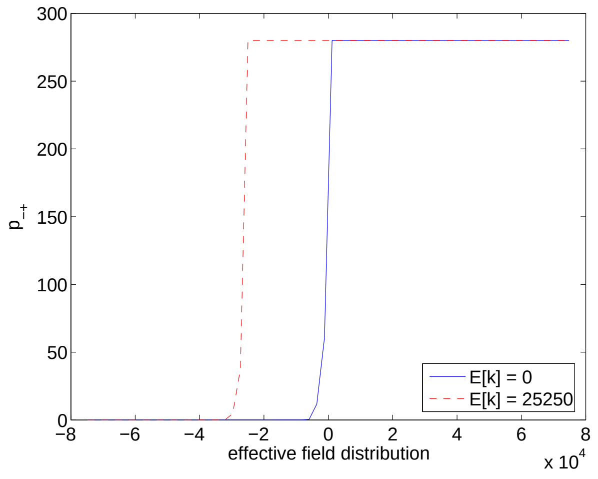
$$\begin{aligned}
&\text{If } E_\mu > 0 \\
&\qquad p_{-+} = \frac{1}{\tau} \\
&\text{Else} \\
&\qquad p_{-+} = \frac{1}{\tau}\left(1 - \frac{\texttt{erfc}(E_\mu + \epsilon)}{\texttt{erfc}(E_\mu - \epsilon) + f}\right).
\end{aligned} \tag{18}$$

This fact is illustrated in Figure 1. Notice that changing E_c, E_I, or $E[k]$ shifts the resulting likelihood function, but does not otherwise change the shape of the function. If the solutions of (17) and (18) are known for all $E_\mu \in \mathbb{R}$, then the loop iterations need only look up the the values of p_{-+} and $\widehat{P}_-$, rather than perform the complex calculations on each iteration. The same process holds for p_{+-} and $\widehat{P}_+$, except that $E_\mu = -E[k] - E_I[j] - E_c[i]$.

Of course, it is impossible to calculate and store in memory the solution of (17) or (18) for all $E_\mu \in \mathbb{R}$. However, it is possible to bound both the range and resolution needed for E_μ, thereby obtaining a vector that can be calculated and stored in memory. To bound the range, one of two options may be used. First, if the input field level is bounded by an arbitrary number, a minimum and maximum possible E_μ may be calculated. To do this, we note that the minimum and maximum values of E_c and E_I are known from the quadrature points. The minima and maxima can be added together to bound E_μ. A second approach to bound E_μ is to recall that the distribution for E_c is bounded by a decaying exponential. As such, for $E_\mu > E_c[N_c - 1]$, we may assume the distribution is close enough to 0 to be treated as such, and therefore all dipoles are oriented in the positive direction. Likewise, for values of $E_\mu < -E_c[N_c - 1]$, all dipoles are oriented in the negative direction. When all



(a)



(b)

Figure 1. (a) Shift caused by a different coercive field value with 0 applied field. (b) Shift caused by different applied fields with $E_c = 0$.

dipoles line up uniformly, the material acts as a single domain and behaves in a linear fashion as given by the kernel (1). Thus, we only need to know the values of p_{-+} and $\widehat{P}_-$ for $E_\mu \in [-E_c[N_c - 1], E_c[N_c - 1]]$. Values outside this range will simply give 0 for $\widehat{P}_-$ and $\widehat{P}_+$ and either 0 or $1/\tau$ for p_{-+} and p_{+-} (depending on whether the out of range value is positive or negative). The first approach yields a slightly simpler final model, but places an additional constraint on the input field level. The second approach requires no additional constraints, but does require checking for out-of-bounds values (which can never occur in the first approach). To maintain generality, the second approach is employed here, with one modification. To reduce the number of out-of-bounds calculations (at the expense of memory usage), we allow $E_\mu \in [-E_c[N_c-1]-E_I[N_I-1], E_c[N_c-1]+E_I[N_I-1]]$. As mentioned above, all the points just added to the range of E_μ will give repetitive values for the likelihoods and average polarizations. However, given an $E[k]$ and $E_c[i]$, we may now iterate over all values of E_I, checking only the first for an out of bounds condition. All other values that do not give $E_\mu \in [-E_c[N_c - 1], E_c[N_c - 1]]$ will receive the same out of bounds values as before, without additional conditional logic to check their bounds.

We have limited the range of E_μ that must be considered, but that by itself still gives infinitely many possible values. The resolution of E_μ must now be set. This problem should be familiar to anyone who has designed numerical methods. The higher the resolution, the more accuracy is obtained, but the more memory is needed. If the resolution is set arbitrarily, a very large amount of memory is needed to give accurate results (potentially many tens to a couple hundred thousand points or more for each of p_{-+}, p_{+-}, $\widehat{P}_-$, and $\widehat{P}_+$). This is unacceptably large for most real-time applications. To improve this, we must restrict the quadrature method over E_I to one of the Newton-Coates formulas (to maintain the even number of quadrature points, it must be an even degree formula such as trapezoid or Simpson's 3/8 rule over an odd number of intervals). This restriction yields equally spaced quadrature points. The resolution can now be set as some positive integer divisor r of the E_I stepsize. Quantization error will still be introduced when E_c and $E[k]$ are not zero; however, no additional quantization is added by iterating over E_I. This reduces the amount of error significantly. While the exact amount of error depends on the parameters being used, in test cases we've run less than a thousand points need to be calculated and stored for p_{-+}, p_{+-}, $\widehat{P}_-$, and $\widehat{P}_+$ in order to limit the increased quantization error to values well below those introduced by data collection and quadrature method. Two such test cases are explored in Section 4.

The final relaxation algorithm, including the optimizations described in this section, is given in Algorithm 5. A few things should be noted. No `exp` or `erfc` functions need to be computed in real time – all can be calculated and stored in advance. In addition, the number of additions, subtractions, multiplications and divisions needed per iteration has been reduced. Experiments detailed in Section 4 indicate this algorithm runs as little as 1.6 times slower than the optimized negligable relaxation model (Algorithm 3), although the exact difference does

```
# Inputs: $E$, $E_c$, $w_c$, $N_c$, $E_I$, $w_I$, $N_I$, $P_R$, $\eta$, $\beta$, $\tau$, $\epsilon$,
#         $f$, $x_+$, $r$
# Output: $P$
# Initial Setup – not input field dependent
$\epsilon = \epsilon/\beta$
$addit = 0$
$w_{sum} = 0$
For $i = 0 \ldots N_c - 1$
  For $j = 0 \ldots N_I - 1$
    $addit = addit + w_c[i]w_I[j]E_I[j]$
    $w_{sum} = w_{sum} + w_c[i]w_I[j]$
  End For
End For
$addit = addit/eta - P_R w_{sum}$
$w_{sum} = w_{sum}/eta$
$e_{step} = (E_I[1] - E_I[0])/r$
$N_r = N_I\ r$
If $E_c[N_c - 1] > E_I[E_I - 1]$
  $increase = \text{ceil}((E_c[N_c - 1] - E_I[N_I - 1])/e_{step}) + N_r$
Else
  $increase = N_r - \text{floor}((E_I[N_I - 1] - E_c[N_c - 1])/e_{step})$
End If
$step = e_{step}/(\eta\beta)$
$N_\mu = 2\ increase + N_r$
$eff = E_I[0]/(\eta\beta)$
For $j = 0 \ldots N_\mu - 1$
  $E_\mu = eff - step\ increase + step\ j$
  $tmp_- = \texttt{erfc}(E_\mu + \epsilon)$
  $tmp_+ = \texttt{erfc}(-E_\mu + \epsilon)$
  If $E_\mu > 0$
    $p_{-+}[j] = 1/\tau$
    $p_{+-}[j] = (1 - tmp_+/(\texttt{erfc}(-E_\mu - \epsilon) + f))/\tau$
  Else
    $p_{-+}[j] = (1 - tmp_{-/}\,(\texttt{erfc}(E_\mu - \epsilon) + f))/\tau$
    $p_{+-}[j] = 1/\tau$
  End If
  $\widehat{P}_-[j] = -\beta\texttt{exp}(-(E_\mu + \epsilon)^2)/(\sqrt{\pi}(tmp_- + f))$
  $\widehat{P}_+[j] = \beta\texttt{exp}(-(-E_\mu + \epsilon)^2)/(\sqrt{\pi}(tmp_+ + f))$
End For
# Begin Iteration
For $k = 0 \ldots \text{length}(E)$
  $P[k] = addit + w_{sum}E[k]$
  $k_{shift} = E[k]/e_{step}$
  For $i = 0 \ldots N_c - 1$
    $i_{shift} = E_c[i]/e_{step}$
    $j_- = \text{round}(increase + k_{shift} - i_{shift})$
    If $j_- < 0$ Then $j_- = 0$ End If
    If $j_- + N_r \geq N_\mu$ Then $j_- = N_\mu - N_r - 1$ End If
    $j_+ = \text{round}(increase + k_{shift} + i_{shift})$
    If $j_+ < 0$ Then $j_+ = 0$ End If
    If $j_+ + N_r \geq N_\mu$ Then $j_+ = N_\mu - N_r - 1$ End If
    $out = 0$
    For $j = 0 \ldots N_I - 1$
      $j_- = j_- + r$
      $j_+ = j_+ + r$
      $x_+[i,j] = \dfrac{x_+[i,j] + p_{-+}[j_-]\ \Delta t}{1 + \Delta t(p_{-+}[j_-] + p_{+-}[j_+])}$
      $out = out + w_I[j](x_+[i,j](P_+[j_+] - P_-[j_-] + 2P_R)+$
          $P_-[j_-])$
    End For
    $P[k] = P[k] + w_c[i]\ out$
  End For
End For
```

Algorithm 5. Improved algorithm for the homogenized energy model with thermal relaxation.

depend on the parameters input and the compiler/hardware being used. Regardless, this is a vast improvement over the 100 times slower execution exhibited by Algorithm 4. It should also be noted that, while the E_I quadrature has been constrained to a Newton-Coates formula, E_c has not had any constraints placed on it, other than those mentioned in Section 1. Finally, note that the choice of constraining and working along E_I is arbitrary. An analogous model could be developed with constraints placed on E_c instead.

4. PERFORMANCE AND ERROR COMPARISONS

The exact performance of Algorithm 5 depends on the hardware and model parameters in use. Table 2 provides a performance comparison for an example ferroelectric material (from [3]). All five algorithms were tested with the same input of approximately 15000 field points, and used Simpson's 3/8 rule on 21 quadrature intervals in both the E_c and E_I distributions (64 quadrature points). Models were compiled and run on two different platforms to compare the effects of different architectures. All algorithms were implemented as c language MATLAB mex

	Linux		**Sun**	
	seconds	relative time	seconds	relative time
Negligible relaxation				
Original (Algorithm 1)	2.2	1.8	13.1	1.6
Optimized (Algorithm 3)	1.3	1	8.3	1
Thermal relaxation				
Original (Algorithm 2)	153.1	122.5	318.3	38.4
Intermediate (Algorithm 4)	127.5	102.0	239.3	28.9
Optimized (Algorithm 4)	2.1	1.7	13.1	1.6

Table 2. Execution time of five algorithms presented in this paper. Times are given for two different architectures.

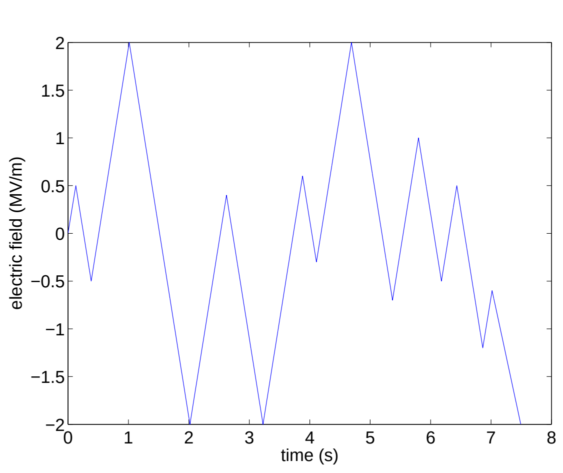
files, and run from that environment. The first test machine used was a 1.7 GHz Pentium IV Xeon running Red Hat Enterprise Linux Workstation 3. Code on this machine was compiled with gcc 3.2.3. The other test machine was a Sunblade 100 with a 500 MHz Sparc processor. The Sun Workshop Compiler version 5.0 was used on this machine. For simplicity, these machines are referred to as linux and sun in the table.

Table 2 makes it clear that Algorithm 5 is far superior computationally to a direct implementation of the homogenized energy model with thermal relaxation, as given in Algorithm 2. The test detailed in the table gives almost a 25 times improvement on the RISC-based SPARC processor, and over a 70 times improvement on the more CISC-designed Pentium IV. Tests with other parameters have yielded slightly less dramatic differences, but still give marked improvement. For example, the same Pentium IV machine gave a 25 times improvement for the computation speed of Algorithm 5 when run on an example ferromagnetic material (from [5]). The reason for this difference is believed to be related to the internal computer architecture, but is not yet fully understood.

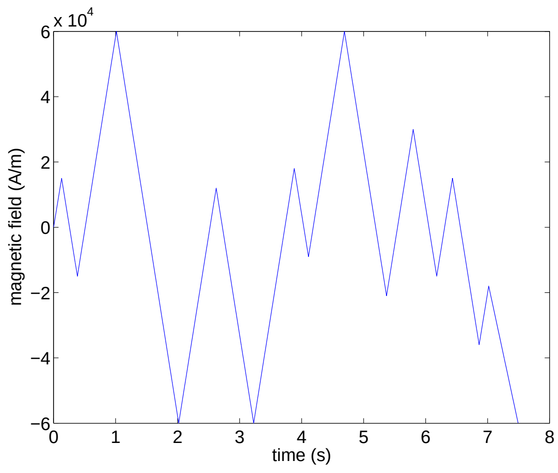
As noted before, the improved run time of Algorithm 5 is not free; increased quantization error has been introduced into the polarization/magnetization calculations. However, this quantization error is quite manageable. Figure 2 illustrates the input, output, and difference between the original relaxation algorithm (Algorithm 2) and the optimized one presented in this paper (Algorithm 5) for $r = 10$, and Table 3 gives the maximum and root mean square (RMS) error for r values between 1 and 64. Notice that in all cases, the maximum error introduced by the optimizations is less than 0.6% (1 part in 167) of the saturation polarization or magnetization. Further, note that error is inversely proportional to r. Thus, error can be roughly halved by doubling r. This should allow the faster algorithm to perform to almost any desired level of accuracy (when compared to the original model). A larger r has very little effect on the computational speed of the algorithm, as it only requires accesses of larger constant arrays (p_{-+}, p_{+-}, $\widehat{P}_-$, and $\widehat{P}_+$). No significant computation difference was seen for r values of 1 to 16 in the ferroelectric example, and no significant difference was seen for any r in Table 3 for the ferromagnetic example. However, the size of likelihood and average polarization/magnetization arrays is proportional to r, so more memory must be utilized for finer resolutions, but is still fairly small for moderate r. With $r = 1$, only 502

	Ferroelectric Example				**Ferromagnetic Example**			
r	max error		RMS error		max error		RMS error	
1	4.42×10^{-4}	0.13%	1.24×10^{-4}	0.035%	2007	0.44%	994	0.22%
2	9.97×10^{-4}	0.28%	1.97×10^{-4}	0.056%	2668	0.58%	1111	0.24%
4	4.74×10^{-4}	0.14%	9.15×10^{-5}	0.028%	1147	0.25%	502	0.11%
8	2.64×10^{-4}	0.075%	4.72×10^{-5}	0.013%	561	0.12%	244	0.053%
10	1.95×10^{-4}	0.056%	3.73×10^{-5}	0.011%	432	0.094%	194	0.042%
16	1.47×10^{-4}	0.042%	2.31×10^{-5}	0.0066%	320	0.070%	121	0.026%
32	9.30×10^{-5}	0.027%	1.27×10^{-5}	0.0036%	182	0.038%	61.3	0.013%
64	6.73×10^{-5}	0.019%	7.78×10^{-6}	0.0022%	136	0.030%	33.0	0.0072%

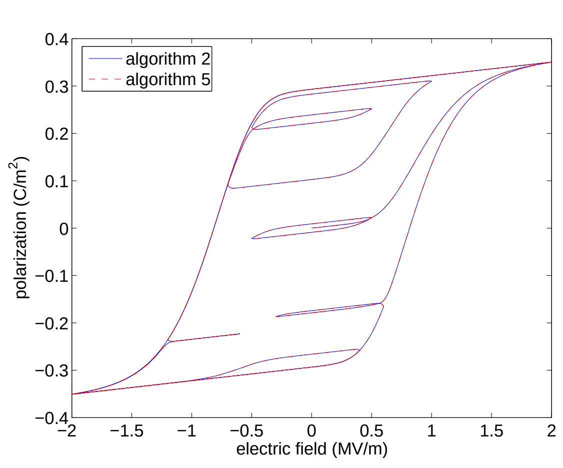
Table 3. Error introduced by Algorithm 5, when compared to the original Algorithm 2. Percentages are of saturation polarization/magnetization.



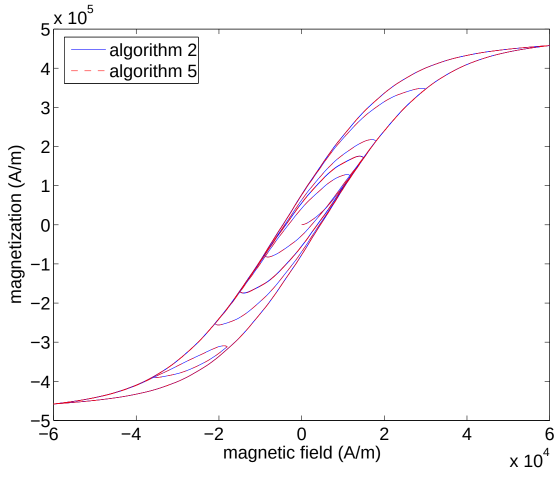
(a) input electric field



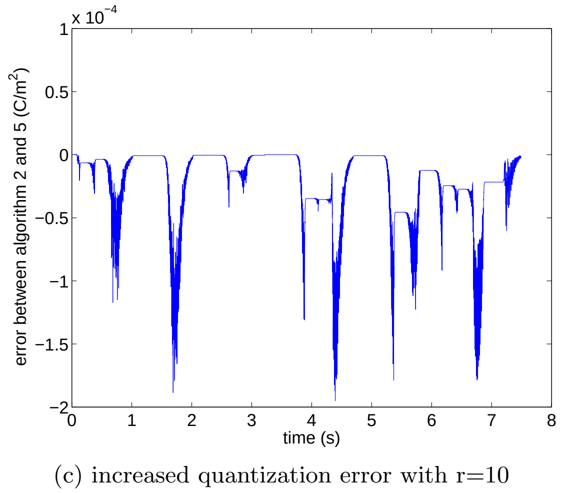
(d) input magnetic field



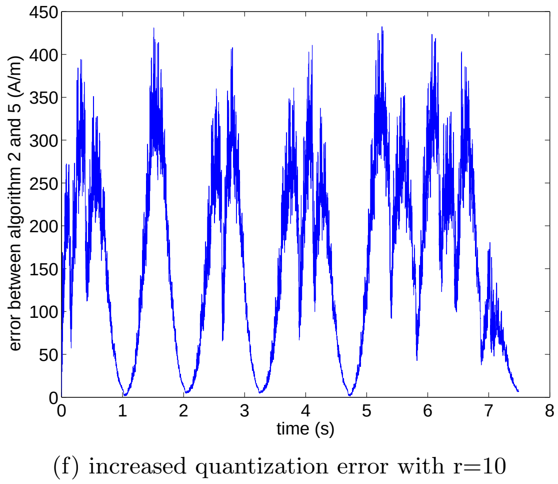
(b) polarization hysteresis loops



(e) magnetization hysteresis loops



(c) increased quantization error with r=10



(f) increased quantization error with r=10

Figure 2. Numerical simulations showing accuracy of thermal relaxation algorithm presented in this paper versus the standard algorithm from the literature. Note difference in scales between results plot and error plot.

values need to be stored for the ferroelectric example (64 quadrature points for each of E_c and E_I). At $r = 10$, this value is up to 5006 values for each of the four arrays. The ferromagnetic example benefits from a maximum E_c less than the maximum E_I and needs only 152 values for $r = 1$ and 1514 values for $r = 10$ (again with 64 quadrature points for each of E_c and E_I). This suggests that memory (and in some cases speed as well) may be optimized in the ferroelectric example by optimizing along E_c instead of E_I (as is mentioned at the end of Section 3), but this is not further explored here. In either example, this level of memory is quite obtainable in most modern microprocessor or gate array devices.

The error figures were obtained by comparing the same quadrature routines in both Algorithms 2 and 5. While this provides a good measure for the quantization error introduced by the changes made herein, it does tend to ignore the biggest drawback to Algorithm 5: the restriction to Newton-Coates quadrature on the E_I distribution. The dominant factor influencing error is often the quadrature method in use. This is especially true around zero applied field where the accuracy is often most desired. Care in choosing a quadrature method is a general numerical necessity, and is not discussed in detail here. However, we note that the speed advantage of Algorithm 5 allows many more quadrature intervals to be considered while still giving a faster execution speed than Algorithm 2. More quadrature intervals do require more memory, and applications that need very high accuracy with memory constrains in the low kilobytes may need to stop their optimizations at Algorithm 4 (which uses less memory than Algorithm 2). For most applications, however, the accuracy lost in Algorithm 5 by requiring Newton-Coates quadrature may be recovered by increasing the number of quadrature intervals, while still maintaining a faster execution speed over the origin relaxation algorithm.

ACKNOWLEDGMENTS

This work supported by the United States Department of Education GAANN fellowship and the Air Force Office of Scientific Research under the grant AFOSR-FA 9550-04-1-0203.

REFERENCES

[1] A.G. Hatch, R.C. Smith and T. De, "Model development and control design for high speed atomic force microscopy," Smart Structures and Materials 2004, Proceedings of the SPIE, Volume 5383, pp. 457–468, 2004.

[2] R.C. Smith, M.J. Dapino and S. Seelecke, "A free energy model for hysteresis in magnetostrictive transducers," *Journal of Applied Physics*, 93, pp. 458-466, 2003.

[3] R.C. Smith, A. Hatch, B. Mukherjee and S. Liu, "A homogenized energy model for hysteresis in ferroelectric materials: General desnity forumulation," CRSC Technical Report CRSC-TR04-23; *Journal of Intelligent Material Systems and Structures*, to appear.

[4] R.C. Smith, "Smart Material Systems: Model Development," SIAM, Philadelphia, 2005.

[5] D.C. Jiles and D.L. Atherton, "Theory of the magnetisation process in ferromagnetics and its application to the magnetomechanical effect," *Journal of Physics D: Applied Physics*, 17, pp. 1265-1281, 1984.