Journal of Applied Physics

# Efficient computation of demagnetizing fields for magnetic multilayers using multilayered convolution

Cite as: J. Appl. Phys. **126**, 103903 (2019); https://doi.org/10.1063/1.5116754
Submitted: 27 June 2019 . Accepted: 20 August 2019 . Published Online: 09 September 2019

Serban Lepadatu

View Online Export Citation CrossMark

© 2019 Author(s).

Journal of Applied Physics ARTICLE scitation.org/journal/jap

# Efficient computation of demagnetizing fields for magnetic multilayers using multilayered convolution

Cite as: J. Appl. Phys. **126**, 103903 (2019); doi: 10.1063/1.5116754
Submitted: 27 June 2019 · Accepted: 20 August 2019 ·
Published Online: 9 September 2019

View Online Export Citation CrossMark

Serban Lepadatu[a)]

## AFFILIATIONS

Jeremiah Horrocks Institute for Mathematics, Physics and Astronomy, University of Central Lancashire, Preston PR1 2HE, United Kingdom

[a)]SLepadatu@uclan.ac.uk

## ABSTRACT

As research into magnetic thin films and spintronics devices is moving from single to multiple magnetic layers, there is a need for micromagnetics modeling tools specifically designed to efficiently handle magnetic multilayers. Here, we show an exact method of computing demagnetizing fields in magnetic multilayers, which is able to handle layers with arbitrary spacing, arbitrary thicknesses, and arbitrary relative positioning between them without impacting the computational performance. The multilayered convolution method is a generalization of the well-known fast Fourier transform-based convolution method used to compute demagnetizing fields in a single magnetic body. In typical use cases, such as multilayered stacks used to study skyrmions, we show that the multilayered convolution method can be up to 8 times faster, implemented both for central processors and graphics processors, compared to the simple convolution method.

*Published under license by AIP Publishing.* https://doi.org/10.1063/1.5116754

## I. INTRODUCTION

Multilayered magnetic structures are currently at the forefront of research into spintronics devices, spurred by applications to nonvolatile magnetic memories and logic as well as the fascinating physics of spin transport across multiple ferromagnetic/nonmagnetic layer interfaces. In particular, skyrmions,[1] stabilized at room temperature in ultrathin magnetic layers through the Dzyaloshinskii-Moriya interaction (DMI),[2,3] have shown great promise as information carriers in spintronics devices, utilizing the spin-Hall effect to efficiently manipulate them with electrical currents.[4] Skyrmion motion has been observed in magnetic multilayered stacks,[5–8] while hybrid chiral skyrmions have been studied in magnetic multilayers.[9,10] Racetrack memory devices have also been proposed[11] based on current-induced domain wall motion in multilayered stacks.[12,13] Moreover, magnetic multilayers with surface exchange coupling allow the modification of dipolar interactions in synthetic antiferromagnetic and synthetic ferrimagnetic tracks,[14–17] resulting in fast domain wall motion and reduced threshold currents.

Numerical micromagnetics[18] modeling plays a very important role in understanding and analyzing experimental results, allowing the reproduction of magnetization dynamics in the presence of magnetic fields as well as spin torques in multilayers.[19] The magnetostatic interaction, which is an essential part of the micromagnetics model, is particularly difficult to evaluate due to its long-range effect, accounting for the majority of simulation time. The use of magnetic multilayers further significantly complicates this, as the spacing and thicknesses of magnetic layers used in many experimental studies make it difficult to discretize the simulation space, while also allowing an efficient simulation. A widely used approach to calculating the demagnetizing fields due to the magnetostatic interaction is based on finite difference discretization and uses fast Fourier transforms (FFTs) to evaluate the convolution sum of a demagnetizing tensor with the magnetization distribution.[20,21] A closely related method allows the calculation of demagnetizing fields from the scalar potential.[22,23] When applied to multilayers, the main difficulty with this approach is the requirement for uniform computational mesh discretization. This poses a problem for magnetic multilayers, where the layer thicknesses and spacings may not readily lend themselves to a uniform discretization of the entire space. Other approaches to calculating the demagnetizing field are available, including tensor grids[24] as well as finite element/boundary element methods[25]—for a review of this class of


Published under license by AIP Publishing.

methods, see Ref. 26. The finite difference method with FFT-based convolution remains very popular, owing to better computational performance compared to finite element methods, particularly for rectangular geometries.[26] Finite element methods are more accurate for curved geometries, although staircase corrections can be used in the finite difference formulation to reduce discretization errors on the demagnetizing field.[27,28]

Freely available software include OOMMF,[29] mumax3,[30] and Fidimag,[31] and all compute demagnetizing fields using the FFT-based convolution method. Here, we introduce a new method specifically designed for magnetic multilayers, which is a generalization of the FFT-based convolution method, termed multilayered convolution. This method has been implemented in Boris Computational Spintronics[32] both for central processors (CPU) and graphics processors (GPU). Multilayered convolution allows computation of demagnetizing fields in multiple layers with arbitrary thicknesses and spacing, without the requirement to uniformly discretize the entire simulation space. In typical use cases, we show that this method results in up to 8 times faster computational speeds compared to FFT-based convolution with uniform discretization, while still being an exact method.

## II. MULTILAYERED CONVOLUTION

In micromagnetics, for a magnetic body with a discrete distribution of magnetization values $\mathbf{M}$ at points in the set $V = \{\mathbf{r}_i;\ i \in P\}$, the demagnetizing fields may be obtained as

$$\mathbf{H}(\mathbf{r}_k) = -\sum_{\mathbf{r}_i \in V} \mathbf{N}(\mathbf{r}_k - \mathbf{r}_i)\mathbf{M}(\mathbf{r}_i), \quad \mathbf{r}_k \in V. \tag{1}$$

The demagnetizing tensor $\mathbf{N}$ has the following components, which may be computed using the formulas given in Newell *et al.*:[33]

$$\mathbf{N} = \begin{pmatrix} N_{xx} & N_{xy} & N_{xz} \\ N_{xy} & N_{yy} & N_{yz} \\ N_{xz} & N_{yz} & N_{zz} \end{pmatrix}. \tag{2}$$

For uniform finite difference discretization, Eq. (1) may be evaluated very efficiently using the convolution theorem:[20,21] the input magnetization and tensor components are transformed using the discrete Fourier transform (DFT), multiplied point-by-point in the transform space, and the output demagnetizing field distribution is obtained by further applying the inverse DFT. Since the demagnetizing tensor only depends on the fixed geometry, it can be obtained in the kernel form by applying the DFT only once in the initialization stage.

When we have a collection of magnetic bodies, $\{V_i;\ i = 1, \ldots, n\}$, one approach to calculating the demagnetizing field distribution is to simply apply Eq. (1) again by taking the union of these separate magnetic bodies into a single magnetic body $V$. For this method to be exact, the discretization cellsize must be chosen so as to divide the separate bodies $V_i$, as well as the empty space between them, into an integer number of cells in each dimension. For most cases of practical interest, this approach is not only restrictive in terms of the geometries that can be reasonably simulated but also inefficient, since the resulting cellsize is typically much smaller than that required to accurately simulate each magnetic body separately. To give examples, we distinguish two cases: (i) magnetic multilayers with large thickness values compared to the separation between the layers and (ii) ultrathin magnetic multilayers with relatively large separation between the layers. Case (i) includes synthetic antiferromagnetic structures,[14–17] while case (ii) occurs most notably in ultrathin magnetic multilayered stacks used to study skyrmions.[5–8,34] With this method, the local and short-range effective field contributions, e.g., due to exchange interaction and magnetocrystalline anisotropy, are computed separately in each computational mesh, while the long-range demagnetizing field is computed on the union of these computational meshes. We term this method supermesh convolution.

With the multilayered convolution approach, we can write the convolution sum as

$$\mathbf{H}(\mathbf{r}'_{kl}) = -\sum_{\substack{i=1,\ldots,n \\ \mathbf{r}_{ij} \in V_i}} \mathbf{N}(\mathbf{r}'_{kl} - \mathbf{r}_{ij}, \mathbf{h}_k, \mathbf{h}_i)\mathbf{M}(\mathbf{r}_{ij}), \quad k = 1, \ldots, n; \quad \mathbf{r}'_{kl} \in V_k \tag{3}$$

In the demagnetizing tensor of Eq. (3), we explicitly specify the cellsize, $\mathbf{h}$, of the two computational meshes the tensor relates. Since in Eq. (3) we have $n$ terms of the form given in Eq. (1), we can again apply the convolution theorem. This time, for each output mesh ($\mathbf{H}$), we have $n$ input meshes ($\mathbf{M}$), together with $n$ kernels. Thus, to calculate the outputs in all $n$ meshes, we require a total of $n^2$ sets of kernel multiplications and $n(n-1)$ summations in the transform space. This is illustrated in Fig. 1. Since the set of $n$ input magnetization distributions is reused for each of the $n$ output field distributions, we only require $n$ applications of the DFT, and similarly the final outputs can be obtained using only $n$ applications of the inverse DFT. This approach is much more

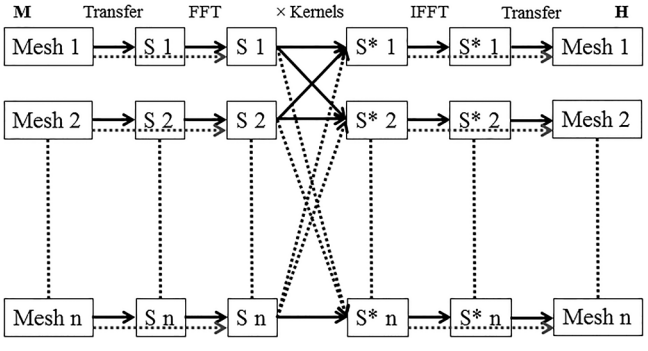


**FIG. 1.** Multilayered convolution algorithm for $n$ computational meshes. The magnetization input of each mesh is transformed separately using a FFT algorithm, either directly (dotted line) or by first transferring to a scratch space with a common discretization cellsize using a weighted average smoother (solid lines). In the transform space, the inputs are multiplied with precomputed kernels for a total of $n^2$ sets of point-by-point multiplications. Finally, the output demagnetizing fields are obtained using an inverse FFT algorithm, which are set directly in the output meshes (dotted line), or transferred using a weighted average smoother if the discretization cellsizes differ (solid lines).


Published under license by AIP Publishing.

efficient than directly summing the interlayer demagnetizing field contributions, as this would require $n^2$ inverse DFTs.

In the 3D mode (all DFTs are three-dimensional), we require all input/output spaces to have the same dimensions and same discretization cellsize. For the 2D mode (all DFTs are two-dimensional in the $xy$ plane), there is no restriction on the thickness of each layer; thus, we only require the $x$ and $y$ components of discretization cellsizes to be the same across the $n$ input/output spaces. In this latter case, it is easy to modify the formulas given in Newell *et al.*[33] to calculate the tensor components for two cells with unequal dimensions—see Appendix A. If the $n$ input spaces have unequal dimensions, we can simply use zero padding to extend them to the largest dimension in the set. A more difficult case arises if the cellsize values differ between the input spaces. In this case, we can use an interpolation method to first transfer the input values to scratch spaces with a common discretization cellsize across the $n$ scratch spaces. Similar remarks apply for the output fields, and this is also illustrated in Fig. 1.

Before discussing the mesh transfer method, we note that in many micromagnetics problems involving multilayers, the simulated magnetic materials used are either the same or with a similar exchange length, which means the required computational cellsize can be set the same without sacrificing computational efficiency. There is a further restriction on the cellsize due to the requirement of integer number of cells in each dimension. For the 2D mode, as mentioned above, there is no restriction on the thickness of each layer as there is just one computational cellsize along the z direction for all computational meshes involved. In the 3D mode, we may also wish to simulate layers with different thickness values. In this case, we can obtain the z component of the common discretization cellsize by dividing the largest mesh $z$ dimension by the largest number of computational cells along the z direction, from the set of $n$ computational meshes. The input magnetization distributions are then transferred using interpolation to the scratch spaces with common discretization, using zero padding where needed. There is no restriction imposed on the cellsize by either the spacing or relative positioning between the layers; thus, multilayered convolution allows for simulations with arbitrary spacing between the layers, which may be inaccessible to supermesh convolution. For example, consider the Pt(5 nm)\FM\Au($d$)\FM\Pt(5 nm) structure from Ref. 8, where FM is Ni(4 Å)\Co(7 Å)\Ni(4 Å), and $d$ is the variable Au spacer thickness. To simulate such a structure using an exact discretization, a 1 Å cellsize in the z direction is required that renders it impractical. Instead, an effective medium approximation may be introduced by considering the FM layer as a whole, in which case a cellsize of 1.5 nm can be used—this still requires discretizing the Au layer, which can be very inefficient for large thicknesses, and also restricts the values of $d$ to multiples of 1.5 nm. With the multilayered convolution method, this structure with the individual Ni and Co layers can be simulated exactly, as it is very efficient (six 2D layers of the required thickness can be set); moreover, the Au layer thickness can be set to any value without impacting the computational performance. Similar considerations apply to the multilayered structures used in Refs. 5–7 as well as the multilayered tracks used in Refs. 14–17. Thus, in many cases, the individual layers may be simulated using 2D transforms, which further results in significant speedup compared to supermesh convolution, the latter requiring a large 3D convolution. The lower DFT dimensions also result in increased numerical accuracy.[35] The need for $n^2$ sets of kernel multiplications may seem excessive, but each set of point-by-point multiplications is much smaller compared to the case of supermesh convolution, which, when taking into account the significantly reduced DFT sizes, allows for a large number of layers to be handled while still providing significant speedup factors.

The mesh transfer procedure uses a weighted average smoother with second order accuracy in space,[36] described as follows. Consider a discrete distribution of magnetization values $\mathbf{M}$ at points $V = \{\mathbf{r}_i;\ i \in P\}$. Let $\mathbf{h}$ be the cellsize of the input mesh, with the set of cells $\{c_i;\ i \in P\}$ centered around the points $\mathbf{r}_i$. To obtain the magnetization value at a point $\mathbf{r}'$ in a cell $c$ with dimensions $\mathbf{h}'$, we introduce the definitions $d_i = |\mathbf{r}' - \mathbf{r}_i|$, $d_V = |\mathbf{h}' + \mathbf{h}|/2$, and $\tilde{d}_i = d_V - d_i$. The weighted average is given as

$$\mathbf{M}(\mathbf{r}') = \sum_{i \in P} w_i \mathbf{M}(\mathbf{r}_i), \tag{4}$$

where

$$\begin{aligned} w_i &= \frac{\tilde{d}_i \delta_i}{\tilde{d}_T}, \\ \delta_i &= \begin{cases} 1, & c_i \cap c \neq \emptyset, \\ 0, & \text{otherwise}, \end{cases} \\ \tilde{d}_T &= \sum_{i \in P} \tilde{d}_i \delta_i. \end{aligned} \tag{5}$$

The weights can be precomputed at the initialization stage, thus speeding up the algorithm at run-time.

Finally, we consider the properties of kernels used for transform space multiplications, which are obtained from the demagnetizing tensors using the DFT. In general, the kernels are complex-valued and use a storage space with $(N_x/2 + 1) \times N_y \times N_z$ points, where $N_x$, $N_y$, and $N_z$ are the DFT sizes in the $x$, $y$, and $z$ dimensions, respectively. The first dimension is always reduced since the input tensor elements are purely real. The demagnetizing tensor elements also have symmetry properties, which in some important special cases allow the kernels to be purely real or purely imaginary (thus resulting in multiplication by a scalar only) as well as use a reduced storage space of $(N_x/2 + 1) \times (N_y/2 + 1) \times (N_z/2 + 1)$ points.[20] The symmetry properties of the demagnetizing tensor components as well as the resulting kernel properties in the cases of interest are summarized in Table I.

At one extreme, we have the self-demagnetizing kernels for 2D and 3D cases (i.e., zero shift between the input and output spaces), which, due to symmetry properties of the tensor components, are purely real and can also be stored using reduced storage space—the remaining elements may be recovered from symmetry properties of the kernels.[20] At the other extreme, we obtain the stray field from one magnetic body at another, with an arbitrary shift between the two spaces. In this case, the kernels are both complex-valued and require the full storage space. While the input tensor elements have symmetries about the $x = 0$, $y = 0$, and $z = 0$ points in each dimension, due to the shift introduced the input tensor symmetries do not carry through to the transform space.


Published under license by AIP Publishing.

**TABLE I.** Convolution kernel properties for the general 2D and 3D cases, as well as special cases, where $N_{dd}$ and $K_{dd}$ refer to the diagonal components (dd = *xx*, *yy*, or *zz*). 2D-Self and 3D-Self refer to the calculation of self-demagnetizing fields. 2D-zShift and 3D-zShift refer to cases where the shift between two computational meshes is along the *z* axis only. In general, the storage space required has $(N_x/2+1)\times N_y \times N_z$ points. For "reduced" storage space (cases indicated in the table), we only need $(N_x/2+1)\times(N_y/2+1)\times(N_z/2+1)$ points. For 3D modes, we require the cellsizes to match ($\mathbf{h}_i = \mathbf{h}_j$) for the computational meshes the kernel relates, while for 2D modes, we only require the *x* and *y* components of the cellsizes to match $[h_{(x,y)i} = h_{(x,y)j}]$. The symmetry properties of tensor components along the *x*, *y*, and *z* axes are indicated, as well as the resulting kernel types after DFT—real, imaginary, or complex.

| Tensor | *x* | *y* | *z* | Kernel (DFT) | 2D-Self "reduced" | 3D-Self "reduced" | 2D-zShift "reduced" $h_{(x,y)i} = h_{(x,y)j}$ | 3D-zShift "reduced" $h_i = h_j$ | 2D-Full "full" $h_{(x,y)i} = h_{(x,y)j}$ | 3D-Full "full" $h_i = h_j$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $N_{dd}$ | Even | Even | Even | $K_{dd}$ | Real | Real | Real | Complex | Complex | Complex |
| $N_{xy}$ | Odd | Odd | Even | $K_{xy}$ | Real | Real | Real | Complex | Complex | Complex |
| $N_{xz}$ | Odd | Even | Odd | $K_{xz}$ | 0 | Real | Imaginary | Complex | Complex | Complex |
| $N_{yz}$ | Even | Odd | Odd | $K_{yz}$ | 0 | Real | Imaginary | Complex | Complex | Complex |

The notable exception is that of a shift along the *z* axis only (cases denoted as 2D-zShift or 3D-zShift in Table I). In this case, the symmetries in the *x* and *y* dimensions are still applicable and the resulting kernel properties are summarized in Table I. Note that for the 3D-zShift case, while the kernels are complex, they can still be stored using reduced storage space since the input to the *z* dimension DFT is either purely real or purely imaginary.

While the multilayered convolution algorithm requires $n^2$ sets of kernel multiplications, typically we do not require storage of $n^2$ kernels due to the redundant information between them. For example, for each kernel that relates an input and output space with a given shift between them, we also need a kernel for the opposite direction shift. For the 2D-zShift case, this may simply be obtained from the first by adjusting signs in the kernel multiplication stage, as resulting from the tensor properties in Table I. Also, since it is only the relative shift between two spaces that is important, not their absolute positions, we can further reduce the required kernel storage in many typical use cases. For example, the most efficient use case is that of regularly spaced multilayers, for which we only need *n* kernels. Finally, a note on implementation, the FFTs in Boris are computed using FFTW3[37] on the CPU and the CUDA 9.2 FFT library[38] on the GPU. Boris is coded in C++14 and is open source.[32] A pseudo-code for the multilayered convolution algorithm is shown in Appendix B.

## III. VALIDATION

To verify the multilayered convolution algorithm, micromagnetics problems have been solved using both the supermesh convolution and multilayered convolution algorithms for all the cases shown in Table I. The most stringent test involves reproducing the exact magnetization dynamics, similar to the approach taken in μMag standard problem 4.[39] For these, the Landau-Lifshitz-Gilbert (LLG) equation was solved with effective field contributions of applied field, exchange interaction field, and demagnetizing field. An example of this is shown in Fig. 2, where the magnetization switching in a $640\times320\,\mathrm{nm}^2$ trilayer $Ni_{80}Fe_{20}$ structure was simulated under a 20 kA/m in-plane magnetic field oriented 5° to the *x* axis. Typical material parameters for $Ni_{80}Fe_{20}$ are used as given in Ref. 40. The starting magnetization state is shown in the inset of Fig. 2(b). The switching field was applied to the top and bottom layers only; thus, the middle layer switches purely due to the dipole field from the outer layers. Here, the outer layers have a thickness of 20 nm, while the middle layer has a thickness of 10 nm, with separation between layers of 1 nm. The supermesh convolution method uses a (5 nm, 5 nm, 1 nm) cellsize in order to accommodate the 1 nm gap between the layers and was computed using mumax3. The Runge-Kutta 4th (RK4) order evaluation method was used with a 100 fs time step due to the stiffness of the LLG equation. For the multilayered convolution, we simply use a 5 nm cubic cellsize in each of the three layers. This was computed using Boris with the RK4 evaluation method using a 500 fs time step. The magnetization switching is plotted for the bottom and middle layers in Fig. 2(b) and 2(c)—the top layer average magnetization dynamics is the same as for the bottom layer due to mirror symmetries. Due to the larger magnetic moments of the top and bottom layers, these are switched toward the applied magnetic field direction. The middle layer, with a smaller magnetic moment, switches due to the large stray fields from the top and bottom layers. As can be seen in Fig. 2, the two convolution methods result in excellent agreement, despite the different cellsize values used to compute the demagnetizing and exchange fields. This problem was computed using the GTX 1050 Ti GPU on Windows 7 x64. In terms of computational performance, the multilayered convolution on Boris is around 18 times faster compared to the supermesh convolution on mumax3, partly due to the much smaller time step required when using the smaller cellsize. In terms of absolute performance per RK4 iteration, the supermesh convolution on Boris is around 1.5 times faster compared to mumax3 on this platform.

Another problem is shown in Fig. 3, where the Néel skyrmion diameter in ultrathin Co layer stacks is computed using both the supermesh and multilayered convolution algorithms. The Co layers are 1 nm thick, of circular shape with 512 nm diameter, and are with a 3 nm nonmagnetic spacer between the layers. The Co layers have strong perpendicular magnetocrystalline anisotropy, in practice arising due to interfacial spin-orbit coupling with a heavy metal layer,[41,] e.g., Pt, which forms part of the nonmagnetic spacer. Material parameters used are the same as given in Ref. 34. The effective field contributions include the applied field, exchange interaction field, interfacial DMI field with DMI exchange constant $D = -1.5\,\mathrm{mJ/m}^2$, uniaxial magnetocrystalline anisotropy field, and demagnetizing field. The skyrmion diameter was obtained by fitting the z skyrmion profile with the function $m_z(r) = \cos(2\arctan(\sinh(R/w)/\sinh(r/w)))$,[42] where *R* is the skyrmion radius and


Published under license by AIP Publishing.

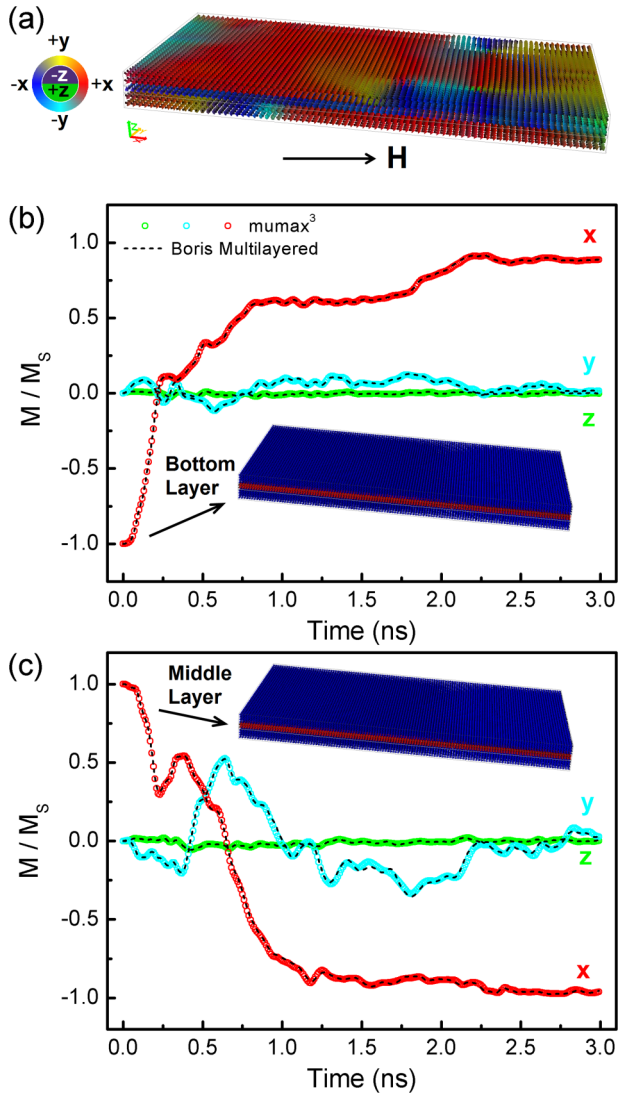


**FIG. 2.** Magnetization switching in a trilayer $Ni_{80}Fe_{20}$ structure using a 20 kA/m in-plane magnetic field oriented 5° to the *x* axis, computed using multilayered convolution (Boris) as well as supermesh convolution (mumax$^3$). The top and bottom layers have a thickness of 20 nm, the middle layer has thickness of 10 nm, with separation between layers of 1 nm, length of 640 nm, and width of 320 nm. (a) Magnetization configuration during the switching event, showing the three separate layers, with magnetization direction arrows color coded using the inset color wheel. (b) and (c) Components of average magnetization as a function of time, plotted for the bottom and middle layers, respectively, showing the starting magnetization configuration in the inset. For supermesh convolution, a (5 nm, 5 nm, 1 nm) cellsize was used—open symbols—while for multilayered convolution, a 5 nm cubic cellsize was used in each layer—dashed lines.

$w = \pi D/4K$ with $K = K_u - \mu_0 M_S^2/2$. Here $K_u$ is the uniaxial magnetocrystalline anisotropy and $M_S$ is the saturation magnetization.[34]

The calculated diameter as a function of out-of-plane magnetic field strength and number of Co layers is shown in Fig. 3(b).

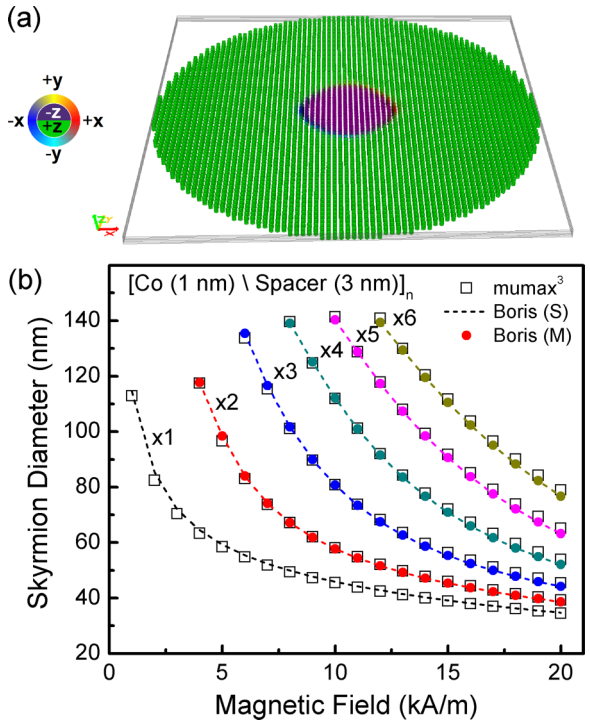


**FIG. 3.** Calculation of average skyrmion diameter in multilayered 512 nm diameter disks as a function of out-of-plane magnetic field and number of Co layers. The Co layers are 1 nm thick with a separation of 3 nm. (a) Skyrmion in a 6-layer stack, with magnetization direction arrows color coded using the inset color wheel. (b) Skyrmion diameter as a function of magnetic field and number of Co layers, computed with Boris using supermesh convolution (dashed lines), multilayered convolution (disks), as well as supermesh convolution with mumax3 (open squares).

Both the supermesh and multilayered convolution algorithms use a cellsize of (4 nm, 4 nm, 1 nm); however, with multilayered convolution, the stray field is only computed in the Co layers alone. As can be seen in Fig. 3(b), the computed diameters are virtually identical for the two methods on Boris, showing the expected inverse dependence on magnetic field strength.[43] We have also computed the skyrmion diameters using supermesh convolution with mumax3, shown as open squares in Fig. 3(b). Again, there is an excellent agreement between the two methods, with differences in diameter up to 2 nm, thus half the in-plane discretization cellsize.

## IV. ALGORITHM PERFORMANCE

The performance comparison between the two algorithms clearly depends on the relative spacing between the layers. At one extreme, we can have a set of magnetic layers with relatively little empty space between them, which can also be exactly discretized without reducing the cellsize dimensions just to accommodate the layer spacing. In this case, the supermesh convolution algorithm is faster. At the other extreme, we have magnetic multilayers with either very small spacing between them relative to the layer thickness values or which otherwise need a very small magnetic cellsize to exactly and


Published under license by AIP Publishing.

uniformly discretize for supermesh convolution. Here, we give a performance comparison for a typical use case, e.g., as arising in multilayered [Pt (3 nm)\Co (1 nm)\Ta (4 nm)]$_n$ stacks used in previous works.[5,34] The same material parameters and effective fields are used for the results in Fig. 3. Two computational platforms were used, GTX 980 Ti GPU with the i7 4790 K CPU on Windows 7 x64 as well as the GTX 1050 Ti GPU with the i7 x980 CPU on Windows 10 x64. The benchmarking results are the average of the results obtained on these 2 computational platforms. The benchmarking results are shown in Fig. 4, plotting the computation time per iteration as a function of number of stack repetitions, namely $n = 1$ up to $n = 17$, for both the supermesh and multilayered convolution algorithms. Both the CPU and GPU implementations of the algorithms are considered. In all cases, the multilayered convolution algorithm results in much faster performance, with speedup factors between 2.5 and 8.

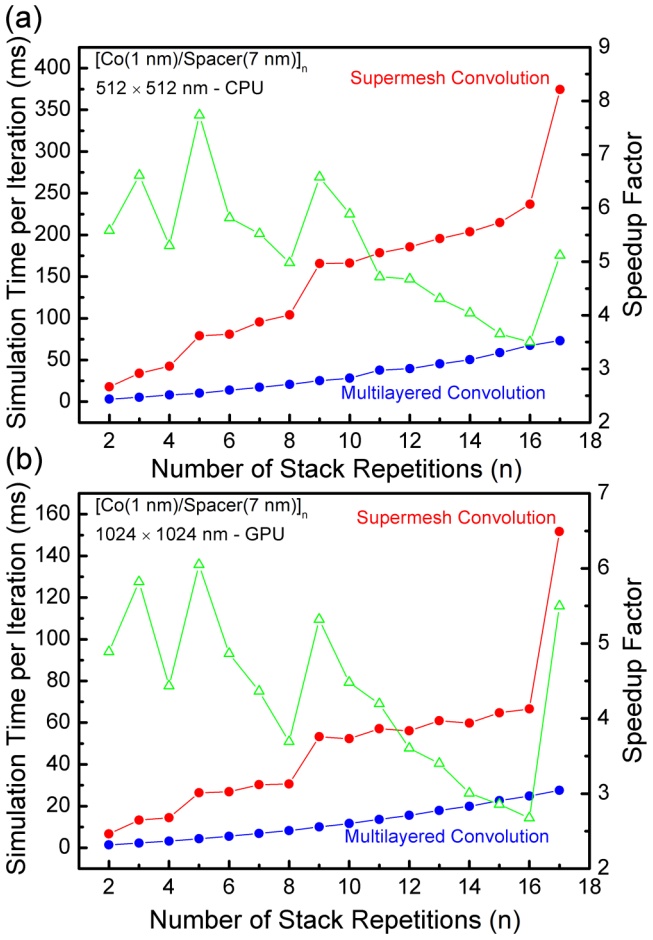


**FIG. 4.** Performance comparison of supermesh and multilayered convolution algorithms in a Co stack as a function of number of Co repetitions, for both (a) CPU implementation for 512 nm diameter disks and (b) GPU implementation for 1024 nm diameter disks. Solid disks show the simulation time per iteration, and empty triangles show the speedup factor of multilayered vs supermesh convolution (simulation time ratio of supermesh to multilayered convolution).

The multilayered convolution simulation time increases smoothly with the number of layers, following a parabolic dependence partly due to the required $n^2$ sets of kernel multiplications indicated in Fig. 1. On the other hand, the supermesh convolution algorithm shows abrupt jumps in simulation time—this is due to the power-of-2 dimensions required by the FFT algorithm. The same benchmarking test was run for the case of 1 nm separation between the layers, which should favor supermesh convolution due to the reduced empty space between the layers. Even in this case, the multilayered convolution algorithm is faster, with an average speedup factor of 1.5 and a maximum speedup factor of 2. The only case where multilayered convolution was slower was for $n = 16$, with a speedup factor of 0.9; however, this jumps to 1.2 for $n = 17$ as the $z$ FFT dimension is doubled for supermesh convolution.

## V. CONCLUSIONS

Here, we have demonstrated a new method of computing demagnetizing fields in magnetic multilayers, which was shown to be a generalization of the FFT-based convolution method used for single magnetic bodies. The multilayered convolution method is able to handle arbitrary spacing and arbitrary relative positioning between the magnetic layers with no impact on the computational performance. Moreover, for thin magnetic layers, which may be simulated using 2D convolution, the multilayered convolution method also allows arbitrary thickness values for the layers in the stack. For 3D convolution, the algorithm is also able to handle layers with different thickness values as well as different $xy$ plane dimensions between the different layers. The algorithm was implemented both for the CPU and GPU. Multilayered convolution is most efficient when the individual layers are thin and are stacked along the z direction. This case occurs very often in practice, and, in particular, for a typical multilayered stack used to study skyrmions, it was shown to be up to 8 times faster compared to the simple convolution method that treats the entire multilayered stack as a single magnetic body.

## APPENDIX A: DEMAGNETIZING TENSOR FORMULAS

Let $\mathbf{s} = (x, y, z)$ be the shift between two rectangular prisms with dimensions (cellsizes) $\mathbf{h_s} = (h_x, h_y, h_{sz})$ and $\mathbf{h_d} = (h_x, h_y, h_{dz})$; thus, the cellsizes are allowed to differ at most in their z dimension. The shift is oriented from the origin corner of the source cellsize with dimensions $\mathbf{h_s}$ to the origin corner of the destination cellsize with dimensions $\mathbf{h_d}$. In this case, the demagnetizing tensors for the $xx$ and $xy$ elements are computed using

$$\begin{aligned} N_{xx}(\mathbf{s}) &= L[f; \mathbf{h_s}, \mathbf{h_d}](\mathbf{s}), \\ N_{xy}(\mathbf{s}) &= L[g; \mathbf{h_s}, \mathbf{h_d}](\mathbf{s}). \end{aligned} \tag{A1}$$

The function $L$ is given as

$$L[\varphi; \mathbf{h_s}, \mathbf{h_d}](\mathbf{s}) = \frac{1}{\tau} \sum_{\varepsilon_1, \varepsilon_2 = -1}^{1} \frac{1}{(-2)^{|\varepsilon_1|+|\varepsilon_2|}} \begin{Bmatrix} -\varphi(x+\varepsilon_1 h_x, y+\varepsilon_2 h_y, z-h_{sz}) \\ -\varphi(x+\varepsilon_1 h_x, y+\varepsilon_2 h_y, z+h_{dz}) \\ +\varphi(x+\varepsilon_1 h_x, y+\varepsilon_2 h_y, z) \\ +\varphi(x+\varepsilon_1 h_x, y+\varepsilon_2 h_y, z-\Delta) \end{Bmatrix}, \tag{A2}$$

where $\tau = \pi h_x \times h_y \times h_{dz}$ and $\Delta = h_{sz} - h_{dz}$.


Published under license by AIP Publishing.

The functions $f$ and $g$ are given below,[33] where $R^2 = x^2 + y^2 + z^2$.

$$f(x,y,z) = \frac{(2x^2 - y^2 - z^2)R}{6} - xyz\arctan\left(\frac{yz}{xR}\right) + \frac{y(z^2 - x^2)}{4}\ln\left(1 + \frac{2y(y+R)}{x^2+z^2}\right) + \frac{z(y^2 - x^2)}{4}\ln\left(1 + \frac{2z(z+R)}{x^2+y^2}\right), \tag{A3}$$

$$g(x,y,z) = -\frac{xyR}{3} - \frac{z^3}{6}\arctan\left(\frac{xy}{zR}\right) - \frac{zy^2}{2}\arctan\left(\frac{xz}{yR}\right) - \frac{zx^2}{2}\arctan\left(\frac{yz}{xR}\right) + \frac{y(3z^2 - y^2)}{12}\ln\left(1 + \frac{2x(x+R)}{y^2+z^2}\right) + \frac{x(3z^2 - x^2)}{12}\ln\left(1 + \frac{2y(y+R)}{x^2+z^2}\right) + \frac{xyz}{2}\ln\left(1 + \frac{2z(z+R)}{x^2+y^2}\right). \tag{A4}$$

The remaining tensor elements may be obtained from $N_{xx}$ and $N_{xy}$ by permuting the dimensions for the **s**, $\mathbf{h_s}$, and $\mathbf{h_d}$ vectors as explained in Ref. 33.

## APPENDIX B: MULTILAYERED CONVOLUTION ALGORITHM PSEUDOCODE

The multilayered convolution algorithm is presented in pseudo-code below. The implementation using C++, both for the CPU and GPU using CUDA, is available as open source in Ref. 32.

**DATA**:
Mesh $M_1, \ldots, M_n$.
Scratch $S_1, \ldots, S_n, S_1^*, \ldots, S_n^*$
Each Mesh and Scratch space has a *rectangle* and *cellsize*
Each Mesh has *magnetisation* and *field* data
Kernel $K_{ij}$ for $i,j = 1, \ldots, n$

**INITIALIZATION**:
**for** $i$ in range 1, …, n **do**
- Set $S_i$ and $S_i^*$ rectangle origin the same as that of $M_i$
- Set $S_i$ and $S_i^*$ rectangle dimensions as the largest dimensions in the set of mesh rectangles $\{M_1, \ldots, M_n\}$
- Set $S_i$ and $S_i^*$ cellsizes as the ratio of their rectangles to the maximum number of computational cells from $\{M_1, \ldots, M_n\}$ in each dimension respectively

**end for**
**for** $i$ in range 1, …, n **do**
    **for** $j$ in range 1, …, n **do**
        $T_{ij}$ = Compute_Newell_Tensor_Elements($S_i$, $S_j$)
        $K_{ij}$ = FFT($T_{ij}$)
    **end for**
**end for**

**procedure** Multiconvolution($M_1, \ldots, M_n, S_1, \ldots, S_n, S_1^*, \ldots, S_n^*$)

    **for** $i$ in range 1, …, n **do**
        **if** $M_i.rectangle = S_i.rectangle$ **and** $M_i.cellsize = S_i.cellsize$ **then**
            $S_i$ = FFT($M_i.magnetisation$)
        **else**
            $S_i$ = Transfer($M_i.magnetisation$)
            $S_i$ = FFT($S_i$)
        **end if**
    **end for**

    **for** $i$ in range 1, …, n **do**
        **for** $j$ in range 1, …, n **do**
            $S_i^*$ += $S_j \times K_j$, using point-by-point multiplication
        **end for**
    **end for**

    **for** $i$ in range 1, …, n **do**
        **if** $M_i.rectangle = S_i^*.rectangle$ **and** $M_i.cellsize = S_i^*.cellsize$ **then**
            $M_i.field$ = Inverse_FFT($S_i^*$)
        **else**
            $S_i^*$ = Inverse_FFT($S_i^*$)
            $M_i.field$ = Transfer ($S_i^*$)
        **end if**
    **end for**
**end procedure**


Published under license by AIP Publishing.

## REFERENCES


[1]A. Bogdanov and D. Yablonskii, "Thermodynamically stable "vortices" in magnetically ordered crystals. The mixed state of magnets," Zh. Eksp. Teor. Fiz. **95**, 178 (1989).
[2]I. Dzyaloshinskii, "A thermodynamic theory of weak ferromagnetism of antiferromagnetics," J. Phys. Chem. Solids **4**, 241–255 (1958).
[3]T. Moriya, "Anisotropic superexchange interaction and weak ferromagnetism," Phys. Rev. **120**, 91–98 (1960).
[4]L. Liu, O. J. Lee, T. J. Gudmundsen, D. C. Ralph, and R. A. Buhrman, "Current-induced switching of perpendicularly magnetized magnetic layers using spin torque from the spin Hall effect," Phys. Rev. Lett. **109**, 096602 (2012).
[5]S. Woo *et al.*, "Observation of room-temperature magnetic skyrmions and their current-driven dynamics in ultra-thin metallic ferromagnets," Nat. Mater. **15**, 501–506 (2016).
[6]W. Legrand *et al.*, "Room-temperature current-induced generation and motion of sub-100 nm skyrmions," Nano Lett. **17**, 2703–2712 (2017).
[7]K. Litzius *et al.*, "Skyrmion Hall effect revealed by direct time-resolved X-ray microscopy," Nat. Phys. **13**, 170–175 (2017).
[8]A. Hrabec *et al.*, "Current-induced skyrmion generation and dynamics in symmetric bilayers," Nat. Commun. **8**, 15765 (2017).
[9]W. Legrand *et al.*, "Hybrid chiral domain walls and skyrmions in magnetic multilayers," Sci. Adv. **4**, eaat0415 (2018).
[10]W. Li *et al.*, "Anatomy of skyrmionic textures in magnetic multilayers," Adv. Mater. **31**, 1807683 (2019).
[11]S. Parkin and S.-H. Yang, "Memory on the racetrack," Nat. Nanotechnol. **10**, 195–198 (2015).
[12]K.-S. Ryu, L. Thomas, S.-H. Yang, and S. Parkin, "Chiral spin torque at magnetic domain walls," Nat. Nanotechnol. **8**, 527–533 (2013).
[13]K.-S. Ryu, L. Thomas, S.-H. Yang, and S. S. P. Parkin, "Current induced tilting of domain walls in high velocity motion along perpendicularly magnetized micron-sized Co/Ni/Co racetracks," Appl. Phys. Express **5**, 093006 (2012).
[14]S. Lepadatu *et al.*, "Very low critical current density for motion of coupled domain walls in synthetic ferrimagnet nanowires," Sci. Rep. **7**, 1640 (2017).
[15]A. Hrabec *et al.*, "Velocity enhancement by synchronization of magnetic domain walls," Phys. Rev. Lett. **120**, 227204 (2018).
[16]A. Hamadeh *et al.*, "Inversion of the domain wall propagation in synthetic ferromagnets," Appl. Phys. Lett. **111**, 022407 (2017).
[17]S.-H. Yang, K.-S. Ryu, and S. Parkin, "Domain-wall velocities of up to 750 ms-1 driven by exchange-coupling torque in synthetic antiferromagnets," Nat. Nanotechnol. **10**, 221–226 (2015).
[18]W. F. Brown, Jr., *Micromagnetics* (Interscience, New York, 1963).
[19]S. Lepadatu, "Unified treatment of spin torques using a coupled magnetisation dynamics and three-dimensional spin current solver," Sci. Rep. **7**, 12937 (2017).
[20]J. E. Miltat and M. J. Donahue, "Numerical micromagnetics: Finite difference methods," in *Handbook of Magnetism and Advanced Magnetic Materials* (John Wiley & Sons, 2007).
[21]N. Hayashi, K. Saito, and Y. Nakatani, "Calculation of demagnetizing field distribution based on fast Fourier transform of convolution," Jpn. J. Appl. Phys. **35**, 6065–6073 (1996).
[22]D. V. Berkov, K. Ramstock, and A. Hubert, "Solving micromagnetic problems towards an optimal numerical method," Phys. Status Solidi A **137**, 207–225 (1993).
[23]C. Abert, G. Selke, B. Kruger, and A. Drews, "A fast finite-difference method for micro-magnetics using the magnetic scalar potential," IEEE Trans. Magn. **48**, 1105–1109 (2012).
[24]L. Exl, W. Auzinger, S. Bance, M. Gusenbauer, F. Reichel, and T. Schrefl, "Fast stray field computation on tensor grids," J. Comput. Phys. **231**, 2840–2850 (2012).
[25]D. Fredkin and T. Koehler, "Hybrid method for computing demagnetizing fields," IEEE Trans. Magn. **26**, 415–417 (1990).
[26]C. Abert, L. Exl, G. Selke, A. Drews, and T. Schrefl, "Numerical methods for the stray-field calculation: A comparison of recently developed algorithms," J. Magn. Magn. Mater. **326**, 176–185 (2013).
[27]S. Lepadatu, "Effective field model of roughness in magnetic nano-structures," J. Appl. Phys. **118**, 243908 (2015).
[28]M. J. Donahue and R. D. McMichael, "Micromagnetics on curved geometries using rectangular cells: Error correction and analysis," IEEE Trans. Mag. **43**, 2878–2880 (2007).
[29]M. J. Donahue and D. G. Porter, OOMMF User's Guide, Version 1.0. Interagency Report NISTIR 6376, 1999.
[30]A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. Van Waeyenberge, "The design and verification of mumax3," AIP Adv. **4**, 107133 (2014).
[31]M.-A. Bisotti, D. Cortés-Ortuño, R. Pepper, W. Wang, M. Beg, T. Kluyver, and H. Fangohr, "Fidimag—A finite difference atomistic and micromagnetic simulation package," J. Open Res. Softw. **6**, 22 (2018).
[32]See https://github.com/SerbanL/Boris2 for Boris Computational Spintronics Open Source User Manual (16 July 2019).
[33]A. J. Newell, W. Williams, and D. J. Dunlop, "A generalization of the demagnetizing tensor for nonuniform magnetization," J. Geophys. Res. Solid Earth **98**, 9551–9555 (1993).
[34]S. Lepadatu, "Effect of inter-layer spin diffusion on skyrmion motion in magnetic multilayers," Sci. Rep. **9**, 9592 (2019).
[35]J. C. Schatzman, "Accuracy of the discrete Fourier transform and the fast Fourier transform," SIAM J. Sci. Comput. **17**, 1150–1166 (1996).
[36]J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods* (Springer-Verlag, 1995).
[37]M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," Proc. IEEE **93**, 216–231 (2005).
[38]See http://www.nvidia.com/object/cuda_home for "Nvidia CUDA Zone."
[39]See https://www.ctcms.nist.gov/~rdm/mumag.org.html for μMag Site Directory (accessed 6 June 2019).
[40]S. Lepadatu, "Interaction of magnetization and heat dynamics for pulsed domain wall movement with Joule heating," J. Appl. Phys. **120**, 163908 (2016).
[41]M. T. Johnson, R. Jungblut, P. J. Kelly, and F. J. A. den Broeder, "Perpendicular magnetic anisotropy of multilayers: Recent insights," J. Magn. Magn. Mater. **148**, 118–124 (1995).
[42]X. S. Wang, H. Y. Yuan, and X. R. Wang, "A theory on skyrmion size," Commun. Phys. **1**, 31 (2018).
[43]J. Sampaio, V. Cros, S. Rohart, A. Thiaville, and A. Fert, "Nucleation, stability and current-induced motion of isolated magnetic skyrmions in nanostructures," Nat. Nanotechnol. **8**, 839–844 (2013).



Published under license by AIP Publishing.