Journal of Computational Physics 227 (2008) 9913–9932

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Application of the fast multipole method for the evaluation of magnetostatic fields in micromagnetic computations

B. Van de Wiele [a,*], F. Olyslager [b], L. Dupré [a]

[a] Department of Electrical Energy, Systems and Automation, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
[b] Department of Information Technology, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

ARTICLE INFO

Article history:
Received 17 March 2008
Accepted 5 August 2008
Available online 14 August 2008

PACS:
75.40.Mg
41.20.Gz

Keywords:
Fast multipole method
Finite difference
Magnetostatic field
Micromagnetics

ABSTRACT

Micromagnetic simulations are elaborated to describe the magnetic dynamics in ferromagnetic bodies. In these simulations, most of the time is spent on the evaluation of the magnetostatic field in the magnetic material. This paper presents a new numerical finite difference scheme for the evaluation of the magnetostatic field based on the fast multipole method (FMM). The interactions between finite difference cells are described in terms of far and near field interactions. The far field computations are conducted using the spherical harmonic expansion of the magnetostatic field while the near field computations are accelerated using fast Fourier transforms (FFT). The performance of the presented FMM scheme is studied by comparing the scheme with a pure FFT scheme. The FMM scheme is more memory efficient and more flexible then the FFT scheme. It is accurate and still has a good time efficiency.

1. Introduction

Since long research has been performed on the magnetization dynamics in ferromagnetic materials. This ongoing research is based on numerical simulations incorporating the micromagnetic theory and has made it possible to understand the magnetic dynamics at a microscopic time and space scale [1]. The micromagnetic theory has mainly been adopted in the development of storage media, see e.g. [2,3]. However, growing computer resources have opened the opportunity of applying the micromagnetic theory also to larger magnetic bodies [4], where the relation between the microstructure of the material and its magnetic properties is studied.

Two major difficulties are encountered in numerical schemes for the simulation of ferromagnetic bodies: first, the integration of the Landau–Lifshitz equation which describes the dynamic behavior of the magnetic dipoles [5] and second, the computation of the magnetostatic field which describes the long range interaction between the magnetic dipoles. For the integration of the Landau–Lifshitz equation, stable implicit [6,7] and explicit [8] time stepping schemes are presented that keep the amplitude of the magnetic dipoles fixed and ensure a decrease of the total free energy when a constant magnetic field is applied, which are two key properties of the Landau–Lifshitz equation.

This paper focusses on the computation of the magnetostatic field $\mathbf{H}_{ms}$ in the magnetic body. This field satisfies Maxwell equations

* Corresponding author.
E-mail address: ben.vandewiele@ugent.be (B. Van de Wiele).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.08.003

$$\nabla \cdot \mathbf{H}_{ms} = -\nabla \cdot \mathbf{M} \tag{1}$$

$$\nabla \times \mathbf{H}_{ms} = 0, \tag{2}$$

with $\mathbf{M}$ the local magnetization. In a general micromagnetic simulation the evaluation of the magnetostatic field $\mathbf{H}_{ms}$ is the most time consuming part. Moreover, $\mathbf{H}_{ms}$ is evaluated several thousand times during one simulation. Hence, there is a need for fast and memory efficient techniques for the evaluation of the magnetostatic field in large ferromagnetic bodies.

Finite difference (FD) schemes for the evaluation of magnetostatic fields are widely used. In these schemes the magnetic body is discretized in cubic cells. The magnetostatic field values are computed in the centers of each cell. Since the magnetization is considered uniform in a FD cell, analytical expressions can be used to describe the magnetostatic field generated by a FD cell [9,10]. However, the direct computation of magnetostatic fields using analytical expressions is slow. Indeed, when N FD cells are considered, this direct approach needs $\mathcal{O}(N^2)$ computations to compute $\mathbf{H}_{ms}$ in each FD cell. Furthermore, the analytical expressions contain various analytical functions which are time intensive to compute.

Fast Fourier transforms (FFT) have been very popular to reduce the computational cost to $\mathcal{O}(N \log N)$, see e.g. [11,12]. Since very fast implementations of FFT algorithms are developed that can be used as a black box [13], the implementation of FFT based schemes for the evaluation of the magnetostatic field is straight forward. These schemes have some drawbacks: (i) the domain under study has to be a rectangular prism. Hence, more complex geometries have to be embedded in a rectangular domain, adding FD cells containing no magnetic material, which introduces some overhead; (ii) all FD cells have to be equal in size and placed on a regular grid excluding the possibility of adaptive discretization; (iii) FFT based schemes use a large amount of memory resources largely due to the need of zero padding [11]. However, FFT based schemes are very fast and robust so they should be used as a reference to study the performance of other numerical schemes for the evaluation of the magnetostatic field.

In order to overcome (some of) the drawbacks FFT schemes encounter, numerical schemes can be adopted describing the generated magnetostatic fields in terms of multipole expansions. These multipole schemes all go back to the fast multipole method (FMM) introduced by Greengard in [14]. The fast multipole method is adopted to numerous physical problems: electrostatic problems e.g. [15], fluid dynamics e.g. [16], molecular dynamics e.g. [17], electromagnetic scattering problems e.g. [18], acoustics e.g. [19], etc. In the micromagnetic research area the use of multipole methods is not yet well spread. The fast Fourier transform on multipole (FFTM) technique [20] has been developed by Ong et al. as a combination of the FMM and FFT method. They have used this scheme to describe recording techniques e.g. [21]. This method is more flexible then a pure FFT scheme and has a controllable error bound. However, the performance of the FFTM scheme has not yet been compared with e.g. a pure FFT scheme. The method still needs a considerable amount of memory resources, while the authors state in [21] 'However, as the two algorithms scale differently with p, FFTM is not necessarily more efficient than FMM', with p referring to the number of multipole coefficients (see further). Visscher et al. use an FMM algorithm based on Cartesian expansions, see e.g. [22].

This paper presents an FMM scheme based on spherical harmonic expansions for the evaluation of the magnetostatic field in a ferromagnetic body. In the next section we will describe how the geometry is discretized in FD cells. Sections 3 and 4 describe how, respectively the far and near field computations are elaborated. In the fifth section, the performance of the presented scheme is investigated by comparing it with an FFT scheme.

2. Geometry description

The three-dimensional ferromagnetic body can have all possible shapes. It is divided into cubical FD cells using a tree structure defining the FMM tree as described in [14]. The ferromagnetic body is enclosed by a cubic box. This enclosing box is called the *root*: the box on *level zero* of the FMM tree. On a next level (*level one*) the root box is divided into 8 smaller cubical boxes with identical volumes. These are called the *children* of the root box. Vice versa, the root box is the *parent* box of the 8 smaller boxes. This division is performed recursively until the boxes on the lowest level (level LEV) in the tree have the desired dimension. In the case of a non-cubic ferromagnetic body, the boxes that do not contain any magnetic material are neglected. In what follows the body will be considered cubical. This does not affect the applicability of the scheme to non-cubic ferromagnetic bodies. Other definitions used to describe the FMM tree:

- Two boxes are said to be *neighbors* if they are at the same refinement level and share a boundary point (a box is a neighbor of itself).
- Two boxes are said to be *well separated* if they are at the same refinement level and are not neighbors.
- With each box i an *interaction list* is associated, consisting of the children of the neighbors of i's parent which are well separated from box i.

These definitions are illustrated in Fig. 1.

The bookkeeping of the boxes is based on the binary structure of the FMM tree [23]. At each level L the boxes get an identity number (id) from 0 to $8^L - 1$. The binary tree structure allows to determine various quantities using computations that can be implemented at bit-level (using bit shift procedures). This results into bookkeeping computations which are negligible compared with other computations: given the id and the level of a box, the parent of the box, the neighbors of the box

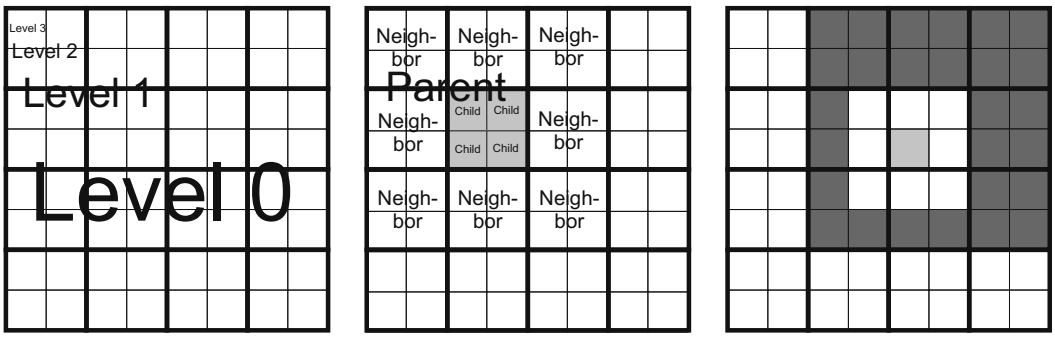

Fig. 1. Definitions used in the FMM theory. The definitions in the middle figure are with respect to the gray colored box. The dark colored boxes in the right figure define the interaction list of the gray colored, central box.

and the center coordinates of the box are found almost instantaneously when the binary structure of the tree is exploited. Vice versa, the *id* of the box containing a point with given coordinates at a given level can be determined also almost instantaneously.

In this implementation of the FMM algorithm, the boxes on the lowest level LEV are called basis boxes. Each basis box itself is further subdivided in cubical FD cells. The number of FD cells in a basis box is 8^{lev}: in each dimension 2^{lev} FD cells. Hence, the total (cubical) geometry contains $8^{LEV+lev}$ FD cells. Fig. 2 shows a geometry described with a tree containing 2 levels $(LEV = 2)$ and basis boxes containing 2^{lev} $(lev = 3)$ FD cells in each direction.

In the FMM theory a distinction is made between boxes that are *far* from each other and boxes that are *near* to each other. Boxes that are far from each other interact via their *far field* which is outlined in Section 3. Neighboring basis boxes interact with each other via near field computations, this is outlined in Section 4. The dimensions of the basis boxes will influence the time spent on near and far field computations. Depending on the difference in computation time of the far field computations and the near field computations an optimal tree can be constructed. Indeed, for a geometry with 8^{tot_lev} FD cells different parameters LEV and lev can be combined such that $LEV + lev = tot\_lev$, i.e. different sizes of basis boxes can be used. The optimal size of the basis boxes depends on the computational time. This discussion is repeated in Section 5.

3. Far field computations

This section explains the computations of the magnetostatic fields due to FD cells that are part of well separated basis boxes. It is shown how the magnetic field is rewritten in terms of expansions of spherical harmonics. The far field computations are based on the classical FMM theory of Greengard [14]. This classical theory is summarized and extended to magnetic dipole sources. The used translation operators are accelerated, exploiting symmetries and convolution structures. It is the intention to develop an algorithm that allows to compute the magnetostatic fields multiple times for different magnetic configurations in a fast way. Hence, generic computations are performed as much as possible during the set up phase of the algorithm in order to avoid duplicating computations.

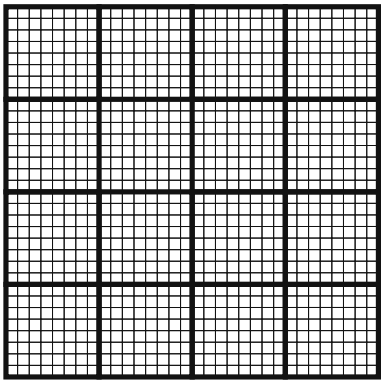

Fig. 2. A tree with LEV = 2 and lev = 3.

3.1. Classical theory

The magnetostatic field in a point $\mathbf{r}$ generated by a FD cell with volume $V'(\mathbf{r} \notin V')$ satisfies Maxwell Eqs. (1) and (2). When Green functions are used, this field can be written as

$$\mathbf{H}_{ms}(\mathbf{r}) = -\frac{M_s}{4\pi}\int_{V'} \nabla\nabla \frac{1}{|\mathbf{r}-\mathbf{r}'|}\cdot \mathbf{m}(\mathbf{r}')\mathrm{d}\mathbf{r}' = \frac{M_s}{4\pi}\nabla \int_{V'} \nabla' \frac{1}{|\mathbf{r}-\mathbf{r}'|}\cdot \mathbf{m}(\mathbf{r}')\mathrm{d}\mathbf{r}' \tag{3}$$

where the ∇' operator acts on $\mathbf{r}'$. In the micromagnetic theory all FD cells are considered to be uniformly magnetized with a fixed magnetization amplitude M_s and a varying magnetization orientation given by the unit vector $\mathbf{m}(\mathbf{r}')$ [1]. The vectors $\mathbf{r}$ and $\mathbf{r}'$ are defined in spherical coordinates as

$$\mathbf{r}' \to (\rho, \alpha, \beta)$$
$$\mathbf{r} \to (r, \theta, \phi).$$

Now the kernel $1/|\mathbf{r}-\mathbf{r}'|$ can be rewritten in an expansion of spherical harmonics, using the spherical harmonic addition theorem for Legendre polynomials $P_n(x)$ [24]. In the case of $\rho < r$ this gives

$$\frac{1}{|\mathbf{r}-\mathbf{r}'|} = \sum_{n=0}^{\infty}\frac{\rho^n}{r^{n+1}}P_n\left(\frac{\mathbf{r}\cdot\mathbf{r}'}{rr'}\right) = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\rho^n Y_n^{-m}(\alpha,\beta)\frac{Y_n^m(\theta,\phi)}{r^{n+1}}. \tag{4}$$

The spherical harmonic $Y_n^m(\theta,\phi)$ is defined as

$$Y_n^m(\theta,\phi) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}(\cos(\theta))\mathrm{e}^{\mathrm{i}m\phi} \tag{5}$$

with

$$P_n^m(x) = (-1)^m(1-x^2)^{m/2}\frac{\mathrm{d}^m}{\mathrm{d}x}P_n(x). \tag{6}$$

The magnetostatic field (3) can now be written in terms of the scalar magnetic potential $\psi(\mathbf{r})$ as

$$\mathbf{H}_{ms}(\mathbf{r}) = \frac{M_s}{4\pi}\nabla\psi(\mathbf{r}). \tag{7}$$

When the addition theorem (4) is introduced, $\psi(\mathbf{r})$ is given by

$$\begin{aligned}\psi(\mathbf{r}) &= \int_{V'}\nabla'\sum_{n=0}^{\infty}\sum_{m=-n}^{n}\rho^n Y_n^{-m}(\alpha,\beta)\frac{Y_n^m(\theta,\phi)}{r^{n+1}}\cdot\mathbf{m}\mathrm{d}V' = \sum_{n=0}^{\infty}\sum_{m=-n}^{n}\left\{\int_{V'}\nabla'\rho^n Y_n^{-m}(\alpha,\beta)\cdot\mathbf{m}\mathrm{d}V'\right\}\frac{Y_n^m(\theta,\phi)}{r^{n+1}}\\ &= \sum_{n=0}^{\infty}\sum_{m=-n}^{n}O_n^m(\mathbf{m})\frac{Y_n^m(\theta,\phi)}{r^{n+1}}.\end{aligned} \tag{8}$$

This defines the expansion coefficients $O_n^m(\mathbf{m})$ of a FD cell with uniform magnetization $\mathbf{M} = M_s\mathbf{m}$. Now analytical expressions will be derived for the expansion coefficients.

3.1.1. Computation of the expansion coefficients $O_n^m(\mathbf{m})$ of a cubical FD cell

Above, the expansion coefficients $O_n^m(\mathbf{m})$ of a cubical FD cell with uniform magnetization $\mathbf{m}$ are defined as

$$O_n^m(\mathbf{m}) = \int_V \nabla\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}V\cdot\mathbf{m}. \tag{9}$$

Applying Gauss' theorem gives

$$O_n^m(\mathbf{m}) = \int_{\partial V}\rho^n Y_n^{-m}(\alpha,\beta)\mathbf{u}\mathrm{d}S\cdot\mathbf{m} \tag{10}$$

with $\mathbf{u}$ the normal unit vector pointing outward of the cell and ∂V the boundary surface of the cell. The explicit computation of the expansion coefficients is done by determining the contributions from each surface of the cubical FD cell with edge length of 2Δ and adding them:

$$\begin{aligned}O_n^m(\mathbf{m}) = &\left(-\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}x\mathrm{d}y\Big|_{z=-\Delta} + \int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}x\mathrm{d}y\Big|_{z=\Delta}\right)m_z\\ &+\left(-\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}x\mathrm{d}z\Big|_{y=-\Delta} + \int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}x\mathrm{d}z\Big|_{y=\Delta}\right)m_y\\ &+\left(-\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}y\mathrm{d}z\Big|_{x=-\Delta} + \int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\rho^n Y_n^{-m}(\alpha,\beta)\mathrm{d}y\mathrm{d}z\Big|_{x=\Delta}\right)m_x.\end{aligned} \tag{11}$$

When $\rho^n Y_n^{-m}(\alpha,\beta)$ is expressed in Cartesian coordinates

$$\rho^n Y_n^{-m}(\alpha,\beta) = (x^2+y^2+z^2)^{n/2}\sqrt{\frac{(n-|m|)!}{(n+|m|)!}}P_n^{|m|}\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right)\left(\frac{x-iy}{\sqrt{x^2+y^2}}\right)^m \tag{12}$$

and the resulting expressions are simplified exploiting some symmetry properties this leads to

$$O_n^m(\mathbf{m}) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\times\begin{cases} n \text{ even} & 0 \\ n \text{ odd}, m \text{ odd} & \begin{aligned} &2m_x\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}(\Delta^2+y^2+z^2)^{\frac{n}{2}}P_n^{|m|}\left(\frac{z}{\sqrt{\Delta^2+y^2+z^2}}\right)\mathcal{R}\mathrm{e}\left(\frac{\Delta-iy}{\sqrt{\Delta^2+y^2}}\right)^m \mathrm{d}y\mathrm{d}z \\ &+2m_y i\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}(x^2+\Delta^2+z^2)^{\frac{n}{2}}P_n^{|m|}\left(\frac{z}{\sqrt{x^2+\Delta^2+z^2}}\right)\mathcal{I}\mathrm{m}\left(\frac{x-i\Delta}{\sqrt{x^2+\Delta^2}}\right)^m \mathrm{d}x\mathrm{d}z \end{aligned} \\ n \text{ odd}, m \text{ even} & \begin{aligned} &m_z\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}(x^2+y^2+\Delta^2)^{\frac{n}{2}}\left[P_n^{|m|}\left(\frac{\Delta}{\sqrt{x^2+y^2+\Delta^2}}\right)\right. \\ &\left.-P_n^{|m|}\left(\frac{-\Delta}{\sqrt{x^2+y^2+\Delta^2}}\right)\right]\left(\frac{x-iy}{\sqrt{x^2+y^2}}\right)^m \mathrm{d}x\mathrm{d}y \end{aligned} \end{cases}. \tag{13}$$

Here and further on, the number of expansion coefficients is truncated to $n < p$. The expansion of the magnetic potential (8) is valid everywhere except inside the circumscribing sphere of the box [14].

The multipole (MP) expansions (13) are the basis of the FMM theory. In the FMM algorithm the MP expansions of the FD cells are combined and translated to define MP expansions of larger groups on lower levels. To compute the magnetostatic field in a group, the outgoing MP expansions from well separated groups are reformulated: the magnetostatic field generated by well separated groups, described by their MP expansions is translated into a local expansion valid in the considered group. These local expansions are translated to local expansions valid in the basis boxes. In what follows the used translation operators will be presented. An extended discussion on the translation of MP expansions and local expansions as well as error bounds can be found in [25].

3.1.2. MP to MP translation operator

To translate a MP expansion O_n^m from the center of a box to a MP expansion M_j^k at the center of its parent, the following translation operator is used

$$M_j^k = \sum_{n=0}^{j}\sum_{m=\max(k+n-j,-n)}^{\min(k+j-n,n)}(-1)^n i^{|k|-|m|-|k-m|}\frac{A_n^m A_{j-n}^{k-m}}{A_j^k} r^n Y_n^{-m}(\theta,\phi)O_{j-n}^{k-m}. \tag{14}$$

$(r,\ \theta,\ \phi)$ are the spherical coordinates of the center of the parent, seen in the coordinate system of the considered box. A_n^m is defined as:

$$A_n^m = \frac{(-1)^n}{\sqrt{(n-m)!(n+m!)}}. \tag{15}$$

For each parent box, 8 MP to MP translations have to be performed. Fig. 3(a) sketches the MP to MP translations.

3.1.3. MP to local translation operator

To translate a MP expansion O_n^m from the center of a box to a local expansion L_j^k at the center of a box at the same level, the following translation operator is used

$$L_j^k = \sum_{n=0}^{p-1}\sum_{m=-n}^{n}(-1)^j i^{|k-m|-|k|-|m|}\frac{A_n^m A_j^k}{A_{j+n}^{m-k}}\frac{Y_{j+n}^{m-k}(\theta,\phi)}{r^{j+n+1}}O_n^m. \tag{16}$$

$(r,\ \theta,\ \phi)$ are the spherical coordinates of the center of the box to where the MP expansion is translated, seen in the coordinate system of the considered box. For each box, 189 MP to local translations have to be performed (except for the boxes near the edge of the computation domain). Fig. 3(b) sketches the MP to local translations.

3.1.4. Local to local translation operator

To translate a local expansion O_n^m from the center of a parent box to a local expansion at the center of a child, the following translation operator is used

$$L_j^k = \sum_{n=j}^{p-1}\sum_{m=k-n+j}^{k-j+n} i^{|m|-|m-k|-|k|}\frac{A_{n-j}^{m-k}A_j^k}{A_n^m} r^{n-j} Y_{n-j}^{m-k}(\theta,\phi)O_n^m. \tag{17}$$

$(r,\ \theta,\ \phi)$ are the spherical coordinates of the center of the child box, seen in the coordinate system of the parent box.

For each parent box, 8 local to local translations have to be performed. Fig. 3(c) sketches the local to local translations.

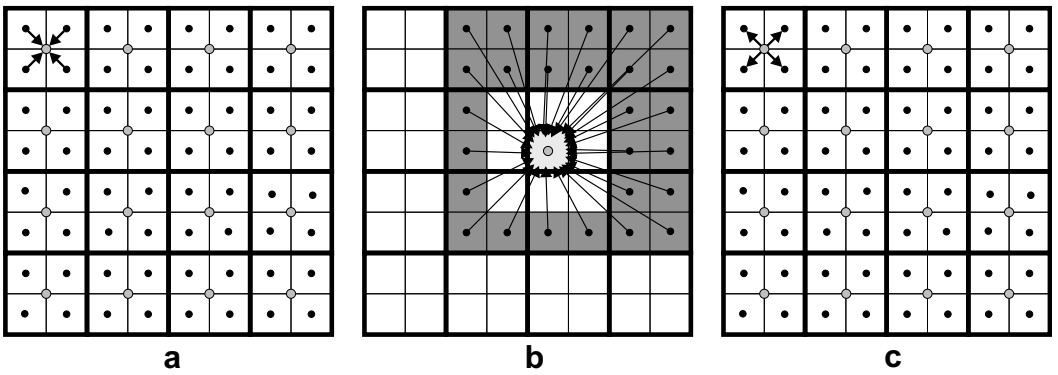

Fig. 3. Sketch of (a) MP to MP translations, (b) MP to local translations and (c) local to local translations.

3.1.5. Local to field translation operator

To compute the field in a point $(r,\ \theta,\ \phi)$ in a basis box with local expansion L_n^m at the center of the basis box, the gradient has to be taken of the scalar potential $\psi(\mathbf{r})$

$$\mathbf{H}_{ms}(\mathbf{r}) = \frac{M_s}{4\pi}\nabla\psi(\mathbf{r}) = \frac{M_s}{4\pi}\sum_{n=0}^{p}\sum_{m=-n}^{n} L_n^m \nabla r^n Y_n^m(\theta,\phi). \tag{18}$$

After the action of the gradient operator on $r^n Y_n^m$ the expression is rotated back to Cartesian coordinates, which results in

$$\mathbf{H}_{ms} = \frac{M_s}{4\pi}\begin{bmatrix} \cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi \\ \sin\phi\sin\theta & \sin\phi\cos\theta & \cos\phi \\ \cos\theta & -\sin\theta & 0 \end{bmatrix} \times \begin{bmatrix} \sum\sum L_n^m n r^{n-1} Y_n^m(\theta,\phi) \\ \sum\sum L_n^m r^{n-1}\sqrt{\frac{(n-|m|)!}{(n+|m|)!}}\Big(-\frac{\cos\theta}{\sin\theta}(n+1)P_n^{|m|}(\cos\theta) \\ \qquad +\frac{1}{\sin\theta}(n-|m|+1)P_{n+1}^{|m|}(\cos(\theta))\Big)\mathrm{e}^{\mathrm{i}m\phi} \\ \sum\sum iL_n^m m\frac{r^{n-1}}{\sin\theta}Y_n^m(\theta,\phi) \end{bmatrix}, \tag{19}$$

where the sums are performed over $n(0 \leqslant n < p)$ and $m(-n \leqslant m \leqslant n)$.

3.1.6. Scaling of the classical translation operators

The expansion coefficients of the spherical harmonics – MP expansions and local expansions – are stored in a vector storage scheme. Since the number of expansion coefficients is truncated to $n < p$ there are p^2 expansion coefficients. In this storage scheme, the computation of the MP coefficients of a FD cell (13) needs $3p^2$ multiplications. Indeed, the integrals in (13) are identical for every FD cell (all FD cells have equal dimensions), thus they can be computed in the set up phase of the algorithm and stored in three vectors $\left(O_{n,x}^m, O_{n,y}^m \text{ and } O_{n,z}^m\right)$. The MP expansion $O_n^m(\mathbf{m})$ of a FD cell with uniform normalized magnetization $\mathbf{m} = [m_x, m_y, m_z]$ is then determined by a linear combination of the three vectors $O_{n,x}^m, O_{n,y}^m$ and $O_{n,z}^m$

$$\mathrm{O}_{\mathrm{n}}^{\mathrm{m}}(\mathbf{m}) = m_x O_{n,x}^m + m_y O_{n,y}^m + m_z O_{n,z}^m. \tag{20}$$

Since the MP and local expansion coefficients are stored in a vector, the action of the MP to MP, the MP to local and the local to local translation operators as given in (14), (16) and (17) can be seen as matrix–vector products which are performed by p^4 multiplications. All translation matrices are computed and stored in the set up phase of the algorithm.

In that way the MP expansion of a basis box containing 8^{lev} FD cells can be computed by first computing the 8^{lev} MP expansion in each FD cell and then translating the 8^{lev} MP expansions to the center of the basis box. This is performed using $8^{\mathrm{lev}} \times 3p^2 + 8^{\mathrm{lev}} \times p^4$ multiplications. When the MP to MP translation operators (14) are denoted by $T_{jk,nm}^q$ $(q = 1,\ldots,8^{\mathrm{lev}})$, this scheme looks like

$$M_j^k = \sum_{q=1}^{8^{\mathrm{lev}}}\sum_n\sum_m T_{jk,nm}^q\left(m_x^q \mathrm{O}_{n,x}^m + m_y^q \mathrm{O}_{n,y}^m + m_z^q \mathrm{O}_{n,z}^m\right). \tag{21}$$

This is accelerated by computing the translated MP expansions $\sum_n\sum_m T_{jk,nm}^q \mathrm{O}_{n,i}^m$ $(q = 1,\ldots,8^{\mathrm{lev}}$ and $i = x,y,z)$ of the FD cells in the set up phase and storing these $3 \times 8^{\mathrm{lev}}$ translated expansions. In that way, the resulting MP expansion is computed by making a linear combination of the translated quantities $\sum_n\sum_m T_{jk,nm}^q \mathrm{O}_{n,i}^m$

$$M_j^k = \sum_{q=1}^{8^{\mathrm{lev}}}\left(m_x^q\sum_n\sum_m T_{jk,nm}^q \mathrm{O}_{n,x}^m + m_y^q\sum_n\sum_m T_{jk,nm}^q \mathrm{O}_{n,y}^m + m_z^q\sum_n\sum_m T_{jk,nm}^q \mathrm{O}_{n,z}^m\right). \tag{22}$$

This scheme needs only $3 \times 8^{\mathrm{lev}} p^2$ multiplications to determine the MP expansion of a basis box.

Applying the local to field translation operator (19) on a local expansion leads to $3p^2$ multiplications relating the 3 components of the magnetostatic field with the p^2 local expansion coefficients, in matrix notation: $\mathbf{H}_{ms} = \mathbf{G}_n^m(r,\theta,\phi)\mathbf{L}_n^m$. The 8^{lev} different local to field translation matrices $\mathbf{G}_n^m(r,\theta,\phi)$ are computed and stored in the set up phase of the algorithm.

3.2. Acceleration of the MP to local translations

In the presented scheme, the computation of the MP expansion of a basis box and the local to field translations are $\mathcal{O}(p^2)$ operations, while the MP to MP, the MP to local and the local to local translations are $\mathcal{O}(p^4)$ operations. In this scheme the number of MP to local translations is very large. Indeed, since each box on each level has an interaction list containing 189 boxes (boundary effects not taken into account) 189 MP to local translations have to be performed per box. This is in contrast to the 8 MP to MP translations and 8 local to local translations performed for each box in the FMM tree. To illustrate this, Table 1 shows the exact number of MP to MP, MP to local an local to local translations in the case of a cubic body discretized using different numbers of levels. N is the total number of basis boxes. From this table it is understood that almost all execution time in the far field computations goes to the translation of MP expansions to local expansions. In what follows, our attention will go to the acceleration of the MP to local translation.

Many accelerated translation schemes are proposed for the MP to local translation operator. In [25] a scheme is described where all translations are performed in the z-direction after a local rotation of the MP coefficients. In this coordinate system the translation operators are diagonal, leading to p^2 translations. However, the rotation of the MP coefficients towards this coordinate system and the back-rotation of the computed local coefficients scale as $\mathcal{O}(p^3)$. These rotations have to be performed on each MP expansion and differs for each direction of the translation. An other faster MP to local translation scheme, also proposed by Greengard in [25] uses the plane wave expansion in stead of the MP expansion to translate the radiation spectrum of a box. Once the plane wave (i.e. exponential) expansion is known for a certain box, the translation operator is diagonal, hence the translation scales as $\mathcal{O}(p^2)$. The computation of the plane wave expansion from the MP expansion and the computation of the local expansion from the plane wave expansion are $\mathcal{O}(p^3)$ calculations. This conversion from MP expansion to plane expansion has to be performed on each MP expansion, but can be used for any translation in any direction. The conversion of the resulting local plane wave expansion into the local expansion is also an $\mathcal{O}(p^3)$ calculation. This scheme is fast, but has some overhead especially for small p the gains are minimal. Other implementations of the MP to local operator are based on the classical $\mathcal{O}(p^4)$ formulation but optimally exploit the use of basic linear algebra subprogram (BLAS) routines [26]. A good overview and performance comparison of different MP to local operator schemes is given in [26]. The different schemes are sketched in Fig. 4.

3.2.1. FFT accelerated scheme: theory

Another scheme, proposed by Elliott and Board Jr. [27] uses the fact that the MP to local translation operator (16) can be written as a convolution, see also [17,28]. This can be seen as follows:

$$L_j^k = \sum_{n=0}^{p-1}\sum_{m=-n}^{n}(-1)^j i^{|k-m|-|k|-|m|}\frac{A_n^m A_j^k}{A_{j+n}^{m-k}}\frac{Y_{j+n}^{m-k}(\theta,\phi)}{r^{j+n+1}}O_n^m \tag{23}$$

$$\frac{(-1)^j i^{|k|}}{A_j^k}L_j^k = \sum_{n=0}^{p-1}\sum_{m=-n}^{n}\frac{i^{|k-m|}}{A_{j+n}^{m-k}}\frac{Y_{j+n}^{m-k}(\theta,\phi)}{r^{j+n+1}}i^{-|m|}A_n^m O_n^m \tag{24}$$

$$\frac{(-1)^j i^{|k|}}{A_{-j}^k}L_{-j}^k = \sum_{n=0}^{p-1}\sum_{m=-n}^{n}\frac{i^{|k-m|}}{A_{-j+n}^{m-k}}\frac{Y_{-j+n}^{m-k}(\theta,\phi)}{r^{-j+n+1}}i^{-|m|}A_n^m O_n^m \tag{25}$$

$$= \sum_{n=0}^{p-1}\sum_{m=-n}^{n}\frac{i^{|k-m|}}{A_{-(j-n)}^{-(k-m)}}\frac{Y_{-(j-n)}^{-(k-m)}(\theta,\phi)}{r^{-(j-n)+1}}i^{-|m|}A_n^m O_n^m. \tag{26}$$

Eq. (26) has a convolution structure. Indeed, entities $(-1)^j i^{|k|}/A_{-j}^k L_{-j}^k$, defined by parameters j and k, are computed through a summation of entities $i^{-|m|}A_n^m O_n^m$, defined by parameters n and m, multiplied by a transfer function that is only defined by the differences $k-m$ and $j-n$. Hence, Eq. (26) can be written as the convolution

$$y_{-n}^m = h_{-n}^m \star x_n^m \tag{27}$$

Table 1
Number of translation operators in a cube discretized using different numbers of levels

# Levels	N	MP to MP	MP to local	Local to local
4	4096	512	56,448	512
5	32,768	4608	640,584	4608
6	262,144	37,376	6,039,504	37,376
7	2,097,152	299,520	52,337,672	299,520
8	16,777,216	2,396,672	435,570,912	2,396,672

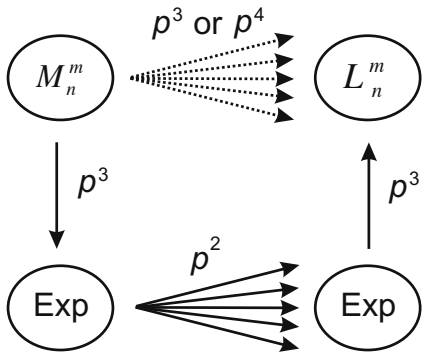

Fig. 4. Sketch of the MP to local translation schemes using the classical matrix–vector multiplication scheme (top, $\mathcal{O}(p^4)$), using rotation based translation operators (top, $\mathcal{O}(p^3)$) and using plane wave based translation operators (bottom).

with

$$y_n^m = \frac{(-1)^n i^{|m|}}{A_n^m} L_n^m \tag{28}$$

$$h_n^m = \frac{i^{|m|}}{A_n^{-m}} \frac{Y_n^{-m}}{r^{n+1}} \tag{29}$$

$$x_n^m = i^{-|m|} A_n^m O_n^m. \tag{30}$$

Here, the negative values $-n$ in y_n^m and the transfer function h_n^m give rise to local expansions and spherical harmonics with negative degrees. Therefore, the entities y_n^m and h_n^m are extended to negative values as follows:

$$y_{-n}^m = y_n^m \tag{31}$$

$$h_{-n}^m = h_n^m. \tag{32}$$

The negative degrees $-n$ affect the alignment of the corresponding matrices $\mathbf{x}$ and $\mathbf{h}$ in coefficient space. For $p = 3$ the transfer matrix $\mathbf{h}$ is

$$\mathbf{h} = \begin{bmatrix} h_0^0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^0 & h_{-3}^0 & h_{-2}^0 & h_{-1}^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^1 & h_{-3}^1 & h_{-2}^1 & h_{-1}^1 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^2 & h_{-3}^2 & h_{-2}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^3 & h_{-3}^3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^{-4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^{-3} & h_{-3}^{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^{-2} & h_{-3}^{-2} & h_{-2}^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{-4}^{-1} & h_{-3}^{-1} & h_{-2}^{-1} & h_{-1}^{-1} \end{bmatrix}. \tag{33}$$

The transfer matrix has values up to degree $2(p-1)$ since these also occur in the MP to local translation operator (16). This so called double height kernel [26] gives rise to matrices with dimensions $2(2p-1) \times 2(2p-1)$. For $p = 3$ the matrices $\mathbf{x}$ and $\mathbf{y}$ in (27) are

$$\mathbf{x} = \begin{bmatrix} x_0^0 & x_1^0 & x_2^0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1^1 & x_2^1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2^{-2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_1^{-1} & x_2^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_0^0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{-2}^0 & y_{-1}^0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{-2}^1 & y_{-1}^1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{-2}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{-2}^{-2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{-2}^{-1} & y_{-1}^{-1} \end{bmatrix}. \tag{34}$$

These matrices need no further zero padding to avoid disturbing side effects originating in the cyclic nature of the Fourier transformation. Moreover, the matrix dimensions can still be reduced by removing the pth to $(2p-1)$th column containing only zeros. In (33) and (34) the fourth, fifth and sixth column can be removed: in general this leads to matrix dimensions of $(3p-2)\times 2(2p-1)$.

3.2.2. FFT accelerated scheme: stability issues

The discussion above seems to appear complete, but the implementation of it leads to numerical instabilities due to the A_n^m factors, the ρ^n term in the MP expansions (9) and the $r^{-(n+1)}$ term in the transfer functions (29), which may cause the coefficients to vary by many orders of magnitude. This results in inaccurate evaluations of the Fourier transforms. To alleviate this problem, the coefficients need to be scaled for a unit box before the FFT method can be applied. To translate the MP expansion of a box with edges 2Δ the elements y_n^m (28), h_n^m (29) and x_n^m (30) have to be redefined as

$$y_n^m = \frac{(-1)^n i^{|m|}}{A_n^m}(2\Delta)^{n+1} L_n^m \tag{35}$$

$$h_n^m = \frac{i^{|m|}}{A_n^{-m}}\left(\frac{2\Delta}{r}\right)^{n+1} Y_n^{-m} \tag{36}$$

$$x_n^m = \frac{i^{-|m|}A_n^m}{(2\Delta)^n} O_n^m. \tag{37}$$

For single precision computations, this solves the problem only for $p < 7$. For larger p values, the quantities A_n^m vary too much in magnitude. In the computation of the magnetostatic field considered here, values of $p < 7$ turn out to be sufficient, see further. When a higher accuracy is required, more expansion coefficients are needed. For these higher p-values the FFT scheme is still applicable, but a block decomposition of the matrices $\mathbf{y}$, $\mathbf{h}$ and $\mathbf{x}$ has to be made [27,17,28] leading to a slower algorithm.

3.2.3. Scaling of the FFT accelerated scheme

The paragraphs above explain how the MP to local translation is accelerated with fast Fourier transforms. In the set up phase of the algorithm the transfer matrices $\mathbf{h}$ are computed, Fourier transformed and stored. These matrices are identical on every level because of the rescaling to transfer matrices of unit boxes. They only differ depending on the (rescaled) vector $\mathbf{r}$ which defines the translation. In total 316 different transfer matrices exist.

Once the aggregation step of the FMM algorithm is performed (i.e. the MP expansions are translated from the lowest level up to the highest level) the MP expansion in every box on every level are known. Before starting the disaggregation step, for each MP expansion the corresponding Fourier transformed matrix $\tilde{\mathbf{x}}$ is computed (the tilde indicates Fourier transformed quantities) and stored. These computations consist of two phases. First the MP expansion values M_n^m are multiplied with $i^{-|m|}A_n^m(2\Delta)^{-n}$ and stored at the corresponding place in the matrix $\mathbf{x}$. These are $\mathcal{O}(p^2)$ computations. The values $i^{-|m|}A_n^m(2\Delta)^{-n}$ differ for every level and are computed in the set up phase of the algorithm. Second, the matrix $\mathbf{x}$ is Fourier transformed. This is performed using FFTW [13] which uses an $\mathcal{O}(p^2\log p)$ algorithm. Since the MP expansion coefficients are not needed any more when the $\tilde{\mathbf{x}}$ is known, both entities can be stored in the same memory space, saving memory requirements.

Once the Fourier transformed matrices $\tilde{\mathbf{x}}$ are computed and stored for every box on every level, the disaggregation step starts. During the disaggregation step, the MP expansions of the boxes in the interaction list (189, considering no side effects) are translated towards the considered box. This is performed by adding the point wise multiplications of the matrices $\tilde{\mathbf{x}}$ of the source boxes with the corresponding Fourier transformed transfer functions $\tilde{\mathbf{h}}$. When no side effects are considered 189 translations consisting of $(3p-2)2(2p-1)$ multiplications have to be performed. Hence, the translations itself require $\mathcal{O}(p^2)$ operations. Once all point wise products are performed and added $\tilde{\mathbf{y}}$ is known.

In the next step $\tilde{\mathbf{y}}$ is Fourier transformed back to real space, which is an $\mathcal{O}(p^2\log p)$ computation. The resulting local expansion values are extracted from the matrix $\mathbf{y}$, divided by the quantity $(-1)^n i^{|m|}(2\Delta)^{n+1}/A_n^m$ and stored. These computations scale as $\mathcal{O}(p^2)$. The values $(-1)^n i^{|m|}(2\Delta)^{n+1}/A_n^m$ differ for every level and are computed in the set up phase of the algorithm.

The total FFT accelerated scheme is shown in Fig. 5. This scheme is much faster since the fast execution of the 189 point wise products exceeds the overhead in the scheme (rescaling, copies, Fourier transformations). An analog scheme can be elaborated to perform the MP to MP and the local to local translations, however, the time gain for these translations is not guaranteed for small p since only 8 translations occur for each box instead of 189. It is not possible to perform the aggregation an disaggregation completely in Fourier space, for a further discussion see [27].

3.3. Exploiting symmetries

The translation operations can still be accelerated when symmetry properties of the spherical harmonics are exploited. The MP and local expansion coefficients satisfy the following symmetries:

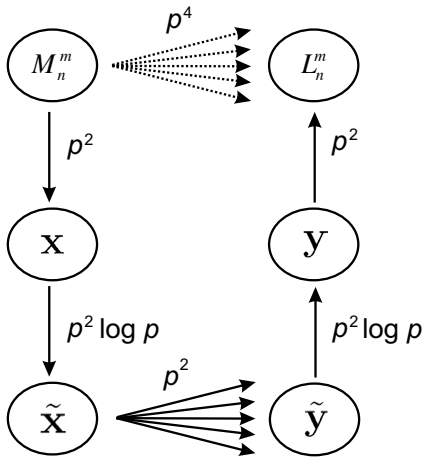

Fig. 5. Sketch of the MP to local translation schemes using the classical matrix–vector multiplication scheme (top), using FFT accelerated translation operators (bottom).

$$M_n^{-m} = \overline{M_n^m} \tag{38}$$

$$L_n^{-m} = \overline{L_n^m}. \tag{39}$$

The redundant negative orders $-m$ can be omitted and only the positive orders $0 \leqslant m < p$ have to be stored. This reduces the memory needed to store a MP expansion and a local expansion from p^2 numbers to $p(p+1)/2$ numbers. In what follows we will comment on the time and memory gains that are obtained when these symmetries are incorporated.

3.3.1. Acceleration of the MP to MP and the local to local translation

Since only the positive orders of the expansion coefficients are stored, the expansions of the source (the child in case of a MP to MP translation and the parent in case of a local to local translation) have to be extended with the negative orders before the actual translation. Second, the translated expansions (only these with positive order m) can be determined by performing a matrix vector product of roughly half the size (the matrix has dimensions $p(p+1)/2 \times p^2$). Compared with the classical translation scheme outlined in (3.1.6), this scheme is roughly twice as fast. However, since the number of MP to MP translations and local to local translations is small compared with the number of MP to local translations, the total time gain is negligible. The most important gain here is the memory reduction obtained for the storage of the MP and local expansions and the translation matrices.

3.3.2. Acceleration of the MP to local translation

In the MP to local translation operator, the symmetries (38) and (39) give also rise to possible memory reduction and faster execution times. Indeed, half of the data in the Fourier transformed matrices $\tilde{\mathbf{x}}, \tilde{\mathbf{h}}$ and $\tilde{\mathbf{y}}$ is conjugated to the other half. Consequently, in the MP to local translation scheme only half of the elements have to be taken into account: when the matrices $\mathbf{x}$ are Fourier transformed, only half of the matrix $\tilde{\mathbf{x}}$ is stored, i.e. $(3p-2)(2p-1)$ elements. During the actual translation, only these elements have to be taken into account for the point wise multiplication. When this is done for all 189 boxes in the interaction list, the second half of the matrix $\tilde{\mathbf{y}}$ is reconstructed based on the first half. After inverse Fourier transforming the matrix, the local expansion is extracted. The MP to local translation scheme is shown in Fig. 6.

In this scheme the number of point wise products is halved in comparison with the scheme outlined in Section 3.2. Since the MP to local translation is responsible for almost the total execution time for the far field computations, this part of the CPU time is roughly halved when this scheme is used. Also the memory needs are reduced: in stead of storing the total number of $(3p-2)2(2p-1)$ elements of the matrix $\tilde{\mathbf{x}}$ in each box, only half of the elements has to be stored.

3.3.3. Acceleration of the local to field translation

In Section 3.1.6 it was outlined that magnetic field components $H_{ms,i}(i = x, y, z)$ are computed by the matrix–vector product $\mathbf{H}_{ms} = \mathbf{G}_n^m \mathbf{L}_n^m$. Since also the elements of the local to field translation matrix $\mathbf{G}$ satisfy the symmetry property

$$G_{n,i}^{-m} = \overline{G_{n,i}^m} \tag{40}$$

the number of multiplications for the determination of the magnetic field from the local expansions is decreased to less then half of the original number of multiplications:

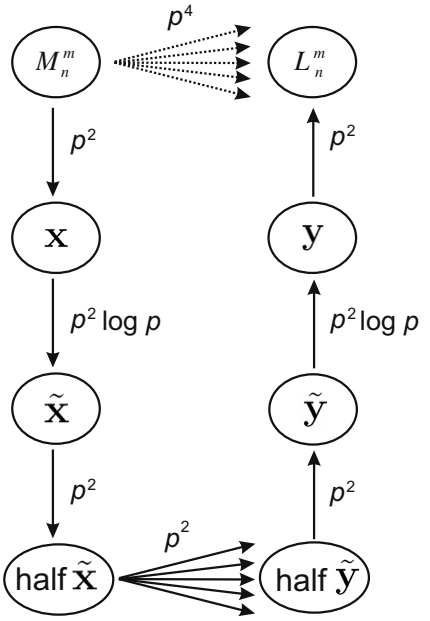

Fig. 6. Sketch of the MP to local translation schemes using the classical matrix–vector multiplication scheme (top), using FFT accelerated translation operators and exploiting the symmetries in the Fourier transformed matrices (bottom). This scheme has half the number of point wise products compared with the scheme of Fig. 5.

$$H_{ms,i} = \sum_{n=0}^{p-1} \left[\mathcal{R}e\left(G_{n,i}^0\right)\mathcal{R}e\left(L_{n,i}^0\right) + 2\sum_{m=1}^{n} \left\{ \mathcal{R}e\left(G_{n,i}^m\right)\mathcal{R}e\left(L_{n,i}^m\right) - \mathcal{I}m\left(G_{n,i}^m\right)\mathcal{I}m\left(L_{n,i}^m\right) \right\} \right]. \tag{41}$$

Hence, the elements with negative order in the transfer matrix do not have to be computed (set up phase) and stored.

4. Near field computations

Up to now all mathematical machinery is provided to calculate the magnetostatic field $\mathbf{H}_{ms}$ originating from FD cells in basis boxes that are well separated, i.e. from FD cells in basis boxes that are not adjacent to the considered basis box. To compute the total magnetic field in a FD cell also the FD cells in the adjacent basis boxes (26 in number) and the FD cells in the considered basis box itself have to be taken into account. This so called *near field* has to be computed and added to the far field contribution. The expression for the magnetostatic field $\mathbf{H}_{ms}(\mathbf{r}_i)$ in the center of a FD cell i originating from N FD cells with uniform normalized magnetization $\mathbf{m}_j$ with box center $\mathbf{r}_{j\,j} = 1,\ldots,N$ and volume V is given by

$$\mathbf{H}_{ms}(\mathbf{r}_i) = -\frac{M_S}{4\pi}\sum_{j=1}^{N}\int_V \nabla\frac{(\mathbf{r}_i - \mathbf{r}_j + \boldsymbol{\rho})\cdot\mathbf{m}_j}{|\mathbf{r}_i - \mathbf{r}_j + \boldsymbol{\rho}|^3}\mathrm{d}\boldsymbol{\rho}. \tag{42}$$

This expression is identical to (3). Applying Gauss' theorem on it gives

$$\mathbf{H}_{ms}(\mathbf{r}_i) = -\frac{M_S}{4\pi}\sum_{j=1}^{N}\int_{\partial V} \frac{(\mathbf{r}_i - \mathbf{r}_j + \boldsymbol{\rho})\cdot\mathbf{m}_j}{|\mathbf{r}_i - \mathbf{r}_j + \boldsymbol{\rho}|^3}\mathbf{u}_S\mathrm{d}\boldsymbol{\rho} \tag{43}$$

with $\mathbf{u}_S$ the normal unit vector pointing outward of the surface ∂V of the FD cell. The integration has to be performed over each cell with identical volume V. Since a classical computation scales $\mathcal{O}(N^2)$, the computations have to be accelerated exploiting the convolution structure of (43). Two numerical schemes exploiting the convolution structure of (43) are possible, both based on FFTs.

4.1. Near interactions with FFTs: Scheme 1

In this scheme the vector $\mathbf{r}_i - \mathbf{r}_j$ between two FD cells is rewritten with respect to the centers of their basis boxes

$$\mathbf{r}_i - \mathbf{r}_j = \mathbf{r}_i' - \mathbf{r}_j' - \boldsymbol{\Lambda}^q. \tag{44}$$

The vector $\boldsymbol{\Lambda}^q$ points from the center of the considered basis box to the center of the qth neighbor (q = 1,...,27). This is shown in Fig. 7. The total magnetostatic field (43) is now written as

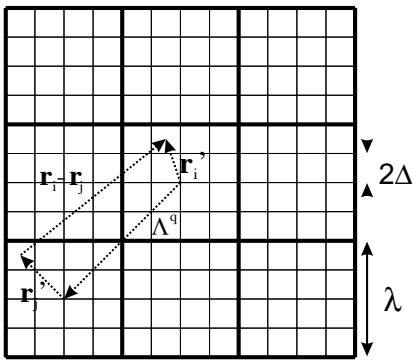

Fig. 7. Vectors used in the near interaction Scheme 1.

$$\mathbf{H}_{ms}(\mathbf{r}_i') = -\frac{M_S}{4\pi}\sum_{q=1}^{27}\sum_{j=1}^{8^{\text{lev}}}\int_{\partial V}\frac{(\mathbf{r}_i\prime - \mathbf{r}_j' - \mathbf{\Lambda}^q + \boldsymbol{\rho})\cdot\mathbf{m}(\mathbf{r}_j)}{|\mathbf{r}_i' - \mathbf{r}_j' - \mathbf{\Lambda}^q + \boldsymbol{\rho}|^3}\mathbf{u}_S \mathrm{d}S. \tag{45}$$

In this near interaction scheme, the FD cells of each basis box are taken into account one basis box after another. Written as a convolution product this becomes

$$\mathbf{H}_{ms}(\mathbf{r}) = \sum_{q=1}^{27}\mathbf{g}(\mathbf{r},\mathbf{\Lambda}^q)\bigstar\mathbf{m}(\mathbf{r}), \tag{46}$$

with $\mathbf{g}(\mathbf{r},\mathbf{\Lambda}^g)$ the Green function which in this case is a symmetrical tensor

$$\mathbf{g}(\mathbf{r},\mathbf{\Lambda}^q) = -\frac{M_S}{4\pi}\int_{\partial V}\frac{\mathbf{r} - \mathbf{\Lambda}^q + \boldsymbol{\rho}}{|\mathbf{r} - \mathbf{\Lambda}^q + \boldsymbol{\rho}|^3}\mathbf{u}_S \mathrm{d}S. \tag{47}$$

With λ the size of the edges of the basis boxes, $\mathbf{\Lambda}^q$ is one of the 27 vectors

$$\mathbf{\Lambda}^q = I\lambda\mathbf{e}_x + J\lambda\mathbf{e}_y + K\lambda\mathbf{e}_z \qquad \begin{cases} I = -1,0,1 \\ J = -1,0,1 \\ K = -1,0,1 \end{cases}. \tag{48}$$

Expressions (43)–(47) make clear that the magnetostatic field $\mathbf{H}_{ms}$ in the FD cells of the considered basis box (the one in the middle of Fig. 7) is computed by:

(1) Zero padding the magnetization vectors of the basis boxes separately.
(2) Fourier transforming the magnetization data of step 1.
(3) Performing the point by point products of the Fourier transformed magnetization vectors (from step 2) with the proper Green function tensor $\mathbf{g}(\mathbf{r},\mathbf{\Lambda}^q)$ and adding the 27 results.
(4) Inverse Fourier transforming the result from step 3 to real space.
(5) Selecting the magnetostatic fields from the data obtained in step 4.

When the edges of the FD cells have length 2Δ, the elements of the Green function tensors $\mathbf{g}(\mathbf{r},\mathbf{\Lambda}^q)$ of (47) are

$$g_{xx}^{I,J,K}(x,y,z) = \frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{x - I\lambda + \Delta}{[(x - I\lambda + \Delta)^2 + (y - J\lambda + \eta)^2 + (z - K\lambda + \zeta)^2]^{3/2}} - \frac{x - I\lambda - \Delta}{[(x - I\lambda - \Delta)^2 + (y - J\lambda + \eta)^2 + (z - K\lambda + \zeta)^2]^{3/2}}\right\}\mathrm{d}\eta\mathrm{d}\zeta \tag{49}$$

$$g_{yy}^{I,J,K}(x,y,z) = \frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{y - J\lambda + \Delta}{[(x - I\lambda + \xi)^2 + (y - J\lambda + \Delta)^2 + (z - K\lambda + \zeta)^2]^{3/2}} - \frac{y - J\lambda - \Delta}{[(x - I\lambda + \xi)^2 + (y - J\lambda - \Delta)^2 + (z - K\lambda + \zeta)^2]^{3/2}}\right\}\mathrm{d}\xi\mathrm{d}\zeta \tag{50}$$

$$g_{zz}^{I,J,K}(x,y,z) = \frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{z - K\lambda + \Delta}{[(x - I\lambda + \xi)^2 + (y - J\lambda + \eta)^2 + (z - K\lambda + \Delta)^2]^{3/2}} - \frac{z - K\lambda - \Delta}{[(x - I\lambda + \xi)^2 + (y - J\lambda + \eta)^2 + (z - K\lambda - \Delta)^2]^{3/2}}\right\}\mathrm{d}\xi\mathrm{d}\eta \tag{51}$$

$$g_{xy}^{I,J,K}(x,y,z)=\frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{y-J\lambda+\eta}{[(x-I\lambda+\Delta)^2+(y-J\lambda+\eta)^2+(z-K\lambda+\zeta)^2]^{3/2}}-\frac{y-J\lambda+\eta}{[(x-I\lambda-\Delta)^2+(y-J\lambda+\eta)^2+(z-K\lambda+\zeta)^2]^{3/2}}\right\}\mathrm{d}\eta\mathrm{d}\zeta \tag{52}$$

$$g_{xz}^{I,J,K}(x,y,z)=\frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{z-K\lambda+\zeta}{[(x-I\lambda+\Delta)^2+(y-J\lambda+\eta)^2+(z-K\lambda+\zeta)^2]^{3/2}}-\frac{z-K\lambda+\zeta}{[(x-I\lambda-\Delta)^2+(y-J\lambda+\eta)^2+(z-K\lambda+\zeta)^2]^{3/2}}\right\}\mathrm{d}\eta\mathrm{d}\zeta \tag{53}$$

$$g_{yz}^{I,J,K}(x,y,z)=\frac{-M_s}{4\pi}\int_{-\Delta}^{\Delta}\int_{-\Delta}^{\Delta}\left\{\frac{z-K\lambda+\zeta}{[(x-I\lambda+\xi)^2+(y-J\lambda+\Delta)^2+(z-K\lambda+\zeta)^2]^{3/2}}-\frac{z-K\lambda+\zeta}{[(x-I\lambda+\xi)^2+(y-J\lambda-\Delta)^2+(z-K\lambda+\zeta)^2]^{3/2}}\right\}\mathrm{d}\xi\mathrm{d}\zeta \tag{54}$$

The Green function tensors are computed and Fourier transformed during the set up phase of the algorithm. The zero padding in the three dimensions is needed to avoid side effects due to the cyclic nature of Fourier transforms. Hence, all Fourier transforms have dimensions $2\cdot 2^{\mathrm{lev}}\times 2\cdot 2^{\mathrm{lev}}\times 2\cdot 2^{\mathrm{lev}}$.

4.2. Near interactions with FFT's: Scheme 2

This scheme uses the expression (43) without redefinitions of vectors. Written as a convolution product, expression (43) looks like

$$\mathbf{H}_{ms}(\mathbf{r})=\mathbf{g}(\mathbf{r})\bigstar\mathbf{m}(\mathbf{r}) \tag{55}$$

with

$$\mathbf{g}(\mathbf{r})=-\frac{M_S}{4\pi}\int_{\partial V}\frac{\mathbf{r}+\boldsymbol{\rho}}{|\mathbf{r}+\boldsymbol{\rho}|^3}\mathbf{u}_S\mathrm{d}S. \tag{56}$$

The elements of this Green tensor are those of (47) with $I=J=K=0$. Hence, expressions (49)–(54) can be used with $I=J=K=0$. In this scheme the magnetic data of all the neighboring basis boxes and the considered basis box itself is translated to the considered basis box in one computation step. Therefor the magnetic data of all the basis boxes have to be rearranged and zero padded. Hence the magnetostatic field $\mathbf{H}_{ms}$ is computed by

(1) Assembling and zero padding the magnetization data.
(2) Fourier transforming the input from step 1.
(3) Performing the point wise products of the Fourier transformed vectors from step 2 with the Green function tensor $\mathbf{g}(\mathbf{r})$
(4) Inverse Fourier transforming the result from step 3.
(5) Selecting the magnetostatic fields from the considered basis box from the result of step 4.

Since in each direction 3 basis boxes are considered in this scheme, the dimension of the block to be Fourier transformed is $3\cdot 2^{\mathrm{lev}}\times 3\cdot 2^{\mathrm{lev}}\times 3\cdot 2^{\mathrm{lev}}$. Then, after zero padding, all Fourier transforms should have dimensions $6\cdot 2^{\mathrm{lev}}\times 6\cdot 2^{\mathrm{lev}}\times 6\cdot 2^{\mathrm{lev}}$. However, the zero padding is needed for not 'spoiling' the Fourier transformed data with side effects due to the cyclic nature of the Fourier transforms. In this scheme, one is only interested in the data computed for the central basis box and not in the magnetostatic fields computed for the neighboring boxes. Hence, the data for these neighboring basis boxes can be spoiled with side effects. Taking this into account, the magnetic data needs only one third of zero padding (2^{lev} zeros) in each direction, reducing the dimensions of the Fourier transforms to $4\cdot 2^{\mathrm{lev}}\times 4\cdot 2^{\mathrm{lev}}\times 4\cdot 2^{\mathrm{lev}}$.

4.3. Computational complexity of the two near interaction schemes

In what follows, the data matrices used in the convolution products will be denoted by a capital. A tilde denotes the Fourier transformed values of the matrices. In that way the matrices containing the zero padded values of the magnetic components of the considered FD cells are denoted by M_x, M_y and M_z while the elements of the Green function tensors (47) and (56) are matrices $G_{xx}, G_{xy}, G_{xz}, G_{yy}, G_{yz}$ and G_{zz}.

The first near interaction scheme is comparable with the MP to local translation in the far field computations. Indeed, in a preparatory step of the near field computations, the magnetization data of all basis boxes is Fourier transformed. This consists of two phases: for the three components x, y, z the magnetization data is copied into a zero padded matrix, this is an $\mathcal{O}(n^3)$ operation (with $n=2^{\mathrm{lev}}$ the number of FD cells in one direction in a basis box). The three matrices are then Fourier transformed ($\mathcal{O}((2n)^3\log 2n)$ operations) and stored. Thus with N_{box} the number of basis boxes, there are $3N_{\mathrm{box}}$ forward Fourier transforms of dimensions $2\cdot 2^{\mathrm{lev}}\times 2\cdot 2^{\mathrm{lev}}\times 2\cdot 2^{\mathrm{lev}}$. These Fourier transformed magnetization values are used (27 times pro basis box) during the actual computation of the magnetostatic field.

The first step in the actual computation of the near fields in a basis box consists of performing the point wise products.

$$\widetilde{H}_{x,i} = \sum_{q=1}^{27} \left[\widetilde{G}^q_{xx,i} \widetilde{M}^q_{x,i} + \widetilde{G}^q_{xy,i} \widetilde{M}^q_{y,i} + \widetilde{G}^q_{xz,i} \widetilde{M}^q_{z,i} \right], \tag{57}$$

$$\widetilde{H}_{y,i} = \sum_{q=1}^{27} \left[\widetilde{G}^q_{xy,i} \widetilde{M}^q_{x,i} + \widetilde{G}^q_{yy,i} \widetilde{M}^q_{y,i} + \widetilde{G}^q_{yz,i} \widetilde{M}^q_{z,i} \right], \tag{58}$$

$$\widetilde{H}_{z,i} = \sum_{q=1}^{27} \left[\widetilde{G}^q_{xz,i} \widetilde{M}^q_{x,i} + \widetilde{G}^q_{yz,i} \widetilde{M}^q_{y,i} + \widetilde{G}^q_{zz,i} \widetilde{M}^q_{z,i} \right]. \tag{59}$$

Since the magnetization data and the Green function values have real values in real space, half of the data is adjoint to the other half in Fourier space. When this property is exploited, each translation of Fourier transformed magnetization values to the considered basis box takes roughly $9(2n)^3/2$ multiplications. For one basis box these computations have to be performed 27 times: once for each neighboring box and once for the considered box itself. So, in total their are $27 \times 9(2n)^3/2 = 972n^3$ multiplications pro basis box. The results are added to each other. The resulting Fourier transformed magnetostatic field values $\widetilde{H}_x$, $\widetilde{H}_y$ and $\widetilde{H}_z$ are then inverse Fourier transformed ($\mathcal{O}((2n)^3 \log 2n)$ operations) and the magnetostatic field values are extracted out of the result ($\mathcal{O}(n^3)$ operations). Thus with N_{box} the number of basis boxes, there are also $3N_{\text{box}}$ inverse Fourier transforms of dimensions $2 \cdot 2^{\text{lev}} \times 2 \cdot 2^{\text{lev}} \times 2 \cdot 2^{\text{lev}}$.

In the second near interaction scheme the forward Fourier transformations of the magnetization data are not computed in a preparatory step since the Fourier transformed magnetizations $\widetilde{M}_x$, $\widetilde{M}_y$ and $\widetilde{M}_z$ are only used once pro basis cell (in contrast with Scheme 1). This also means that this data is not stored in every basis cell, saving roughly $3 \times 8^{\text{lev}+1} N_{\text{box}}$ complex numbers of memory. The first step in the computation of the magnetostatic field in a basis box is the assembly of the 3 zero padded matrices M_x, M_y and M_z containing the magnetization data of the 27 basis boxes ($\mathcal{O}(27 \times n^3)$ operations). In a second step this data is forward Fourier transformed ($\mathcal{O}((4n)^3 \log 4n)$ operations). With N_{box} the number of basis boxes, there are $3N_{\text{box}}$ forward Fourier transforms of dimensions $4 \cdot 2^{\text{lev}} \times 4 \cdot 2^{\text{lev}} \times 4 \cdot 2^{\text{lev}}$. Then the Fourier transformed data is multiplied point wise with the Fourier transformed Green tensor elements

$$\widetilde{H}_{x,i} = \widetilde{G}_{xx,i} \widetilde{M}_{x,i} + \widetilde{G}_{xy,i} \widetilde{M}^{y,i} + \widetilde{G}_{xz,i} \widetilde{M}_{z,i}, \tag{60}$$

$$\widetilde{H}_{y,i} = \widetilde{G}_{xy,i} \widetilde{M}_{x,i} + \widetilde{G}_{yy,i} \widetilde{M}^{y,i} + \widetilde{G}_{yz,i} \widetilde{M}_{z,i}, \tag{61}$$

$$\widetilde{H}_{z,i} = \widetilde{G}_{xz,i} \widetilde{M}_{x,i} + \widetilde{G}_{yz,i} \widetilde{M}^{y,i} + \widetilde{G}_{zz,i} \widetilde{M}_{z,i}. \tag{62}$$

Hence, there are $9(4n)^3/2 = 288n^3$ multiplications pro basis box, which is a much smaller number then in Scheme 1. Moreover, since the matrices $G_{xx}, G_{xy}, G_{xz}, G_{yy}, G_{yz}$ and G_{zz} containing the Green tensor elements are even, their Fourier transformed matrices $\widetilde{G}_{xx}, \widetilde{G}_{xy}, \widetilde{G}_{xz}, \widetilde{G}_{yy}, \widetilde{G}_{yz}$ and $\widetilde{G}_{zz}$ have only strictly real values. This means that the point wise products are not complex $\times$ complex multiplications as in Scheme 1, but real $\times$ complex multiplications, which are performed roughly twice as fast.

The resulting Fourier transformed magnetostatic field values $\widetilde{H}_x, \widetilde{H}_y$ and $\widetilde{H}_z$ are then inverse Fourier transformed ($\mathcal{O}((4n)^3 \log 4n)$ operations) and the field values are extracted out of the result ($\mathcal{O}(n^3)$ operations). Thus there are also $3N_{\text{box}}$ inverse Fourier transforms of dimensions $4 \cdot 2^{\text{lev}} \times 4 \cdot 2^{\text{lev}} \times 4 \cdot 2^{\text{lev}}$.

Hence, comparing the two schemes, one concludes that Scheme 2 is the most memory efficient since the Fourier transformed values are not stored in each basis box. The time efficiency depends on two considerations. First, the number of FFTs is equal: $3N_{\text{box}}$ forward and $3N_{\text{box}}$ inverse FFTs in both schemes, but the dimensions of the Fourier transformed matrices are different. In Scheme 1 the matrix dimensions are half of those in Scheme 2 ($2n \times 2n \times 2n \leftrightarrow 4n \times 4n \times 4n$), resulting in a *much faster execution of the FFTs in the first scheme*. Second, in Scheme 1 there are $972n^3$ complex $\times$ complex point wise products, while in the second scheme there are $288n^3$ real $\times$ complex point wise products, resulting in a *much faster execution of the point wise product in the second scheme*. Which of the two schemes is the fastest depends thus partly on how fast the FFTs are executed.

4.4. Acceleration of the used FFTs

The standard routine in FFTW [13] can be used for the execution of a 3D real to complex FFT. This is a very fast routine to Fourier transform matrices with real elements in 3D. A 3D FFT contains three phases, when transforming a matrix with dimensions $N \times N \times N$ these phases are: (i) Fourier transforming the N^2 arrays in z-direction, (ii) Fourier transforming the N^2 arrays in the y-direction and (iii) Fourier transforming the N^2 arrays in the x-direction. So in total their are $3N^2$ 1D FFTs of arrays of dimension N. As a consequence, when this routine is used to Fourier transform zero padded matrices, 1D FFTs are performed on arrays containing only zeros, which is useless. Omitting the 1D FFTs on arrays containing only zeros accelerates the 3D FFT.

In the case of a zero padded matrix as in near interaction Scheme 1 this is shown in Fig. 8. The three different phases of the 3D FFT are shown. From Fig. 8(a) it is clear that only a quarter of the z-arrays contains non-zero values, thus only $N^2/4$ 1D

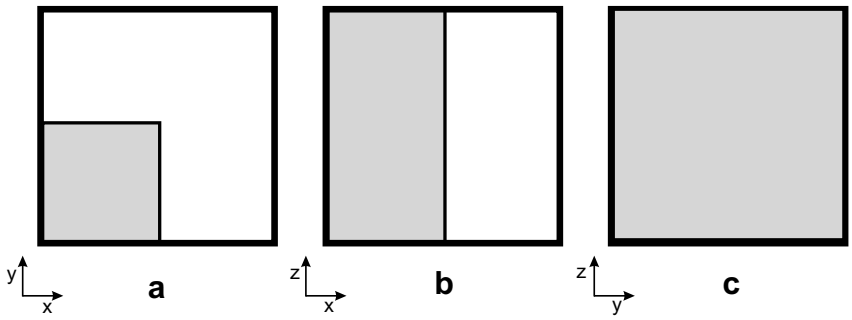

Fig. 8. Overview of the data during the different phases of the 3D forward FFT for a zero padded matrix as in near interaction Scheme 1. The gray areas correspond with arrays in the in-plane direction (a: *z*-direction, b: *y*-direction, c: *x*-direction) containing non-zero values.

FFTs are performed. From Fig. 8(b) it is clear that after the Fourier transformations on the *z*-arrays only half of the *y*-arrays contains non-zero values, thus only $N^2/2$ 1D FFTs are performed. Fig. 8(c) shows that after the Fourier transformations on the *z*- and *y*-arrays all *x*-arrays contain non-zero values, thus all N^2 1D FFTs have to be performed. This means that only $7/4N^2$ 1D FFTs instead of $3N^2$ are performed during the forward Fourier transformations of the matrices in Scheme 1. For the inverse Fourier transforms, this scheme is performed in the opposite direction, starting with Fig. 8(c) and ending with Fig. 8(a). Applying this scheme on the computations of the 3D FFTs in Scheme 1 reduces the time spend on FFTs to about 58% of the original time.

In the case of a zero padded matrix as in near interaction Scheme 2 the three different phases of the forward 3D FFT are shown in Fig. 9. From Fig. 9(a) it is clear that only 9/16 of the *z*-arrays contain non-zero values, thus only $9/16N^2$ 1D FFTs are performed. From Fig. 9(b) it is clear that after the Fourier transformations on the *z*-arrays three quarters of the *y*-arrays contain non-zero values, thus only $3/4N^2$ 1D FFTs are performed. Fig. 9(c) shows that after the Fourier transformations on the *z*- and *y*-arrays all *x*-arrays contain non-zero values, thus all N^2 1D FFTs have to be performed. This means that only $37/16N^2$ instead of $3N^2$ 1D FFTs are performed during the forward Fourier transformations of the magnetization matrices in Scheme 2. Applying this scheme on the computations of the 3D forward FFTs in Scheme 2 reduces the time spend on the forward FFTs to about 77% of the original time.

For the inverse 3D FFTs in near interaction Scheme 2, one is only interested in the results corresponding with magnetostatic fields in the considered basis box, which is the one in the center of Fig. 6, moreover, all other data is spoiled by side effects of the Fourier transformations since the matrices containing the magnetization data where only zero padded with *n* zeros in every direction. Hence one can reduce the number of 1D FFTs to only FFTs on meaningful data, which is data that has influence on the resulting magnetostatic field values of the considered basis box. Fig. 10 shows the arrays that contain meaningful data during the different phases of the inverse 3D FFT. From Fig. 10(a) it is clear that all the *x*-arrays contain meaningful data, thus all N^2 1D (inverse) FFTs have to be performed. From Fig. 10(b) it is clear that after the Fourier transformations on the *x*-arrays, only one quarter of the *y*-arrays contains meaningful data, thus only $N^2/4$ 1D (inverse) FFTs are performed. Fig. 10(c) shows that after the Fourier transformations on the *x*- and *y*-arrays only 1/16 of the *z*-arrays contains meaningful data, thus only $N^2/16$ 1D (inverse) FFTs are performed. This means that only $21/16N^2$ in stead of $3N^2$ 1D (inverse) FFTs are performed during the inverse Fourier transformations of the magnetization matrices in Scheme 2. Applying this scheme on the computations of the 3D inverse FFTs in Scheme 2 reduces the time spend on the inverse FFTs to about 43% of the original time.

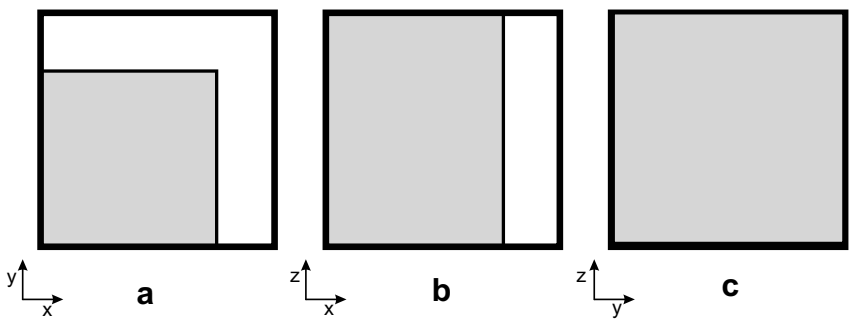

Fig. 9. Overview of the data during the different phases of the 3D forward FFT for a zero padded matrix as in near interaction Scheme 2. The gray areas correspond with arrays in the in-plane direction (a: *z*-direction, b: *y*-direction, c: *x*-direction) containing non-zero values.

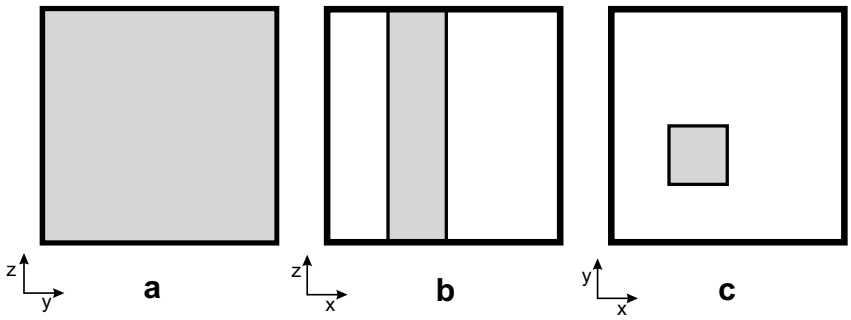

Fig. 10. Overview of the data during the different phases of the 3D inverse FFT as used in near interaction Scheme 2. The gray areas correspond with arrays in the in-plane direction (a: *x*-direction, b: *y*-direction, c: *z*-direction) containing meaningful values.

Table 2
Timing of the two near interaction schemes for one basis box in milliseconds (Fw: forward, Inv: inverse)

	Scheme 1			Scheme 2		
lev	2	3	4	2	3	4
copy **m** (ms)	0.002	0.013	0.083	0.028	0.420	2.43
FFT Fw (ms)	0.007	0.083	0.844	0.114	1.100	11.8
products (ms)	0.540	3.680	52.5	0.078	0.745	6.71
FFT Inv (ms)	0.013	0.093	0.790	0.055	0.570	7.55
copy $\mathbf{H}_{ms}$ (ms)	0.004	0.001	0.05	0.004	0.011	0.12
Total (ms)	0.566	3.870	54.27	0.279	2.846	28.61

4.5. Timing of the two near interaction schemes

Now that the optimal FFT schemes are known, a comparison between the two near interaction schemes is possible to determine the fastest scheme. Table 2 shows the timings of the subroutines and the total timing for the computation of the near interactions for one basis box. One concludes that the second near interaction scheme is by far faster than the first near interaction scheme for all sizes of the basis box. In Scheme 1 almost all computation time goes to the computation of the point wise products while in the second near interaction scheme most of the computation time is divided between the assembling of the magnetization matrices, the forward and inverse Fourier transforms and the point wise products. Moreover, the second scheme needs much less memory. Indeed, in Scheme 1 the three Fourier transformed magnetization matrices have to be stored in each basis box, while this is not the case in the second scheme. It is obvious that the second scheme is used in the algorithm to compute the near interactions. However, since the FFT scheme for the near interactions introduces some overhead, the direct classical computation of the magnetostatic field is faster for basis boxes with lev < 2.

5. Performance study

This section discusses the performance of the presented FMM algorithm. To evaluate the time and memory consumption, comparison is made with a pure FFT scheme. The FFT scheme is based on the expression (43) of the magnetostatic field. As in the near field computations of the FMM scheme, the magnetostatic field is computed by zero padding the three 3D matrices containing the magnetic components of the uniformly magnetized FD cells. In Fourier space, the Fourier transformed magnetization values are multiplied pointwise with the Fourier transformed Green's function elements. The magnetostatic field values are then obtained by inverse Fourier transforming the pointwise products. In contrast with the near interaction computations, all FD cells are included in one big computation to determine the magnetostatic field values in all FD cells at once. The used FFTs are optimized, thus 1D FFTs on arrays containing only zeros are excluded as in Fig. 8. The precision of the FFT scheme corresponds with the machine precision. Since all computations – FMM and FFT – are conducted with single precision, this corresponds with a precision of about 6 digits.

In what follows, simulations are conducted on cubical magnetic bodies to evaluate the performance of the FMM algorithm. The discretization is done as explained in Section 2, so all FD cells are equal in size and placed on a regular grid. As discussed in the introduction, these conditions are optimal for the use of an FFT scheme. Hence, it can be expected that the FFT scheme will outperform the FMM scheme with respect to CPU time. However, the difference in CPU time between both schemes under these FFT suited conditions should be acceptable. The slower execution time of the FMM scheme should be compensated with a smaller memory consumption and more flexible applicability of the FMM scheme. The simulations are performed using one processor of a dual core AMD Opteron 270 (2×2 cores) machine with 8 Gbyte memory.

Table 3
Timing of the FMM algorithm for different sample dimensions

$64 \times 64 \times 64$		$128 \times 128 \times 128$		$256 \times 256 \times 256$		$512 \times 512 \times 512$	
(5–1)	9.17 s	(6–1)	78.83 s	(7–1)	663.1 s		
(4–2)	2.03 s	(5–2)	17.66 s	(6–2)	148.4 s	(7–2)	20 min 21 s
(3–3)	1.57 s	(4–3)	12.88 s	(5–3)	104.6 s	(6–3)	14 min 59 s
(2–4)	1.76 s	(3–4)	14.25 s	(4–4)	115.2 s	(5–4)	18 min 16 s
		(2–5)	17.06 s	(3–5)	138.3 s		
				(2–6)	152.8 s		
FFT	0.264 s	FFT	2.22 s	FFT	22.3 s		

Between brackets is mentioned how the total number of levels is divided between far field and near field computations (LEV–lev). The last row shows the run time for the FFT scheme.

5.1. CPU time and memory consumption

As mentioned above, cubic magnetic bodies will be considered. This geometry is discretized using $8^{\text{tot\_lev}}$ FD cells. For a geometry with $8^{\text{tot\_lev}}$ FD cells different parameters LEV and lev can be combined, i.e. different sizes of basis boxes can be used. The optimal size of the basis boxes depends on the total computation time of the algorithm. Table 3 shows the CPU times for the computations of the magnetostatic fields for samples of different sizes. The used number of MP expansions p is equal to 6, which is the largest possible p-value avoiding stability problems for the MP to local translation (see Section 3.2.2). Between brackets, the number of levels in the far field computations, LEV, and the number of levels in the near field computations, lev, is mentioned. In the last row the CPU times for the FFT algorithm are shown.

For all sample dimensions the FMM simulations with lev $= 3$ need the least execution time. This means that in all simulations the optimal size of the basis boxes is $8 \times 8 \times 8$ FD cells (512 FD cells in total). When compared with the FFT scheme, the FMM scheme is slower, for $64 \times 64 \times 64$ a factor 5.95, for $128 \times 128 \times 128$ a factor 5.80 and for $256 \times 256 \times 256$ a factor 4.70. In the FMM scheme with optimal lev parameter, about 85% of the time goes to near interaction computations.

The memory consumption of the FMM scheme with optimal (LEV–lev) ratio and of the FFT scheme is given in Table 4 for the different sizes of the sample from Table 3. There is a remarkable difference in memory needs between the two algorithms (roughly a factor 11 for large dimensions). The sample with dimensions $512 \times 512 \times 512$ can only be computed with the FMM scheme since only 8 Gbyte memory is available. The difference in memory needs is due to the very large matrices in the FFT scheme used for the FFTs (zero padded magnetization data, zero padded field data and Green tensor data).

The scaling of both algorithms is shown in Fig. 11 over a large range of sample dimensions. The CPU time of the FMM scheme depends almost perfectly linear on the number of FD cells – $\mathcal{O}(N^{1.0188})$ dependence – while the CPU time of the FFT scheme has a small supralinear dependence on the number of FD cells – $\mathcal{O}(N^{1.0928})$ dependence. The CPU time spend on a fast Fourier transform of a matrix depends vastly on the dimensions of the matrix. FFTW performs best for matrices with dimensions that are products of small primes

$$N = 2^a 3^b 5^c 7^d 11^e 13^f \tag{63}$$

with $e + f = 0$ or 1. Other sizes are computed by means of a slow, general purpose algorithm [29]. This explains the jumps in the FFT curve in Fig. 11 for large N values. Indeed, while dimensions grow, the sizes for which the condition (63) is met are more scattered. Hence for larger dimensions, more jumps are expected, which makes the difference in CPU time between the FMM and FFT scheme even smaller for these large dimensions.

5.2. Accuracy

The accuracy of the far field computations depends on the order of multipoles p used in the computations of the far field. The near field computations are computed with an accuracy corresponding with machine precision. Because of the stability problems encountered in the MP to local translations the maximum number of multipoles p is limited to six (in this implementation), see Section 3.2.2. Theoretical considerations about the accuracy of the FMM scheme are given in [25] where error bounds are given. Here, the error on the magnetostatic field values computed with the FMM scheme, is given by comparing the results of the FMM simulations with the results of the FFT simulations.

To make this comparison, magnetic samples in a micromagnetic equilibrium state encountered in the simulation of their hysteresis loops [11] are used. This is done for different sample dimensions. The normalized root-mean-square error for dif-

Table 4
Memory consumption of the FMM scheme with optimal (LEV–lev) division and of the FFT scheme

	$64 \times 64 \times 64$	$128 \times 128 \times 128$	$256 \times 256 \times 256$	$512 \times 512 \times 512$
FMM	16 MB	64 MB	0.48 GB	3.78 GB
FFT	82 MB	654 MB	5.23 GB	41.9 GB (estimation)

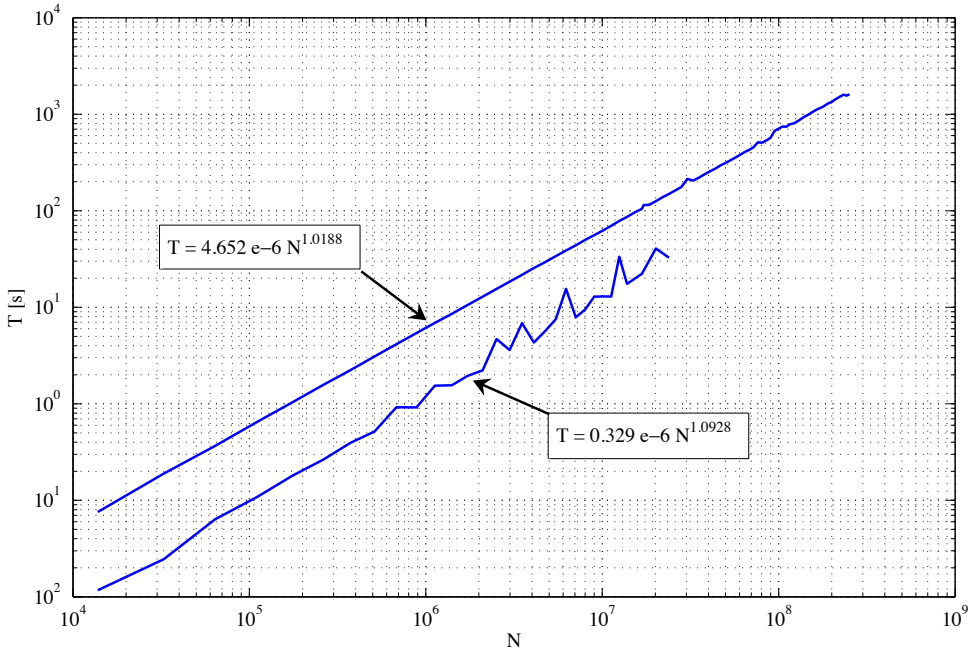

Fig. 11. CPU time (T) for the evaluation of the magnetostatic field in a cubic magnetic body versus the number of FD cells (N) used to discretize the sample. The upper curve: FMM scheme, the lower curve: FFT scheme.

Table 5
Normalized root-mean-square error of the FMM algorithm for different sample dimensions

$64 \times 64 \times 64$		$128 \times 128 \times 128$		$256 \times 256 \times 256$	
(5–1)	2.17 e−3	(6–1)	2.35 e−3	(7–1)	2.48 e−3
(4–2)	2.13 e−3	(5–2)	2.30 e−3	(6–2)	2.44 e−3
(3–3)	1.98 e−3	(4–3)	2.22 e−3	(5–3)	2.37 e−3
(2–4)	1.54 e−3	(3–4)	2.01 e−3	(4–4)	2.24 e−3
		(2–5)	1.49 e−3	(3–5)	1.96 e−3
				(2–6)	1.36 e−3

Between brackets is mentioned how the total number of levels is divided between far field and near field computations (LEV–lev).

ferent LEV/lev parameters and different sample dimensions are shown in Table 4. The normalized root-mean-square error is defined as

$$\text{error} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{|\mathbf{H}_{ms,i}^{\text{FMM}} - \mathbf{H}_{ms,i}^{\text{FFT}}|^2}{|\mathbf{H}_{ms,i}^{\text{FFT}}|^2}} \tag{64}$$

From Table 5 it is clear that the error slightly increases when the dimensions of the sample increase. Further, the error decreases when the size of the basis boxes increases. This is when lev is large. This is because for large basis boxes a relatively large number of interactions is computed using near field computations which have an accuracy corresponding with the machine precision.

The magnetic configurations in different planes of the sample with dimensions of $128 \times 128 \times 128$ FD cells are shown in Fig. 12 together with the magnetostatic field values and the local normalized error in the same planes. The local normalized error depends on the position in the basis boxes: at the edges of the basis boxes, the largest errors occur.

6. Conclusions and prospects

A fast and memory efficient fast multipole algorithm is presented for the evaluation of magnetostatic fields in magnetic bodies that are discretized using FD cells of equal size, placed on a regular grid. The performance of the FMM scheme is determined by comparing it with an FFT scheme. This FFT scheme is perfectly suited and generally used to compute the magnetostatic field in cubical bodies discretized as described above. It is shown that the presented FMM scheme has a very good performance. In most numerical algorithms there is a trade of between memory usage and execution time: small memory

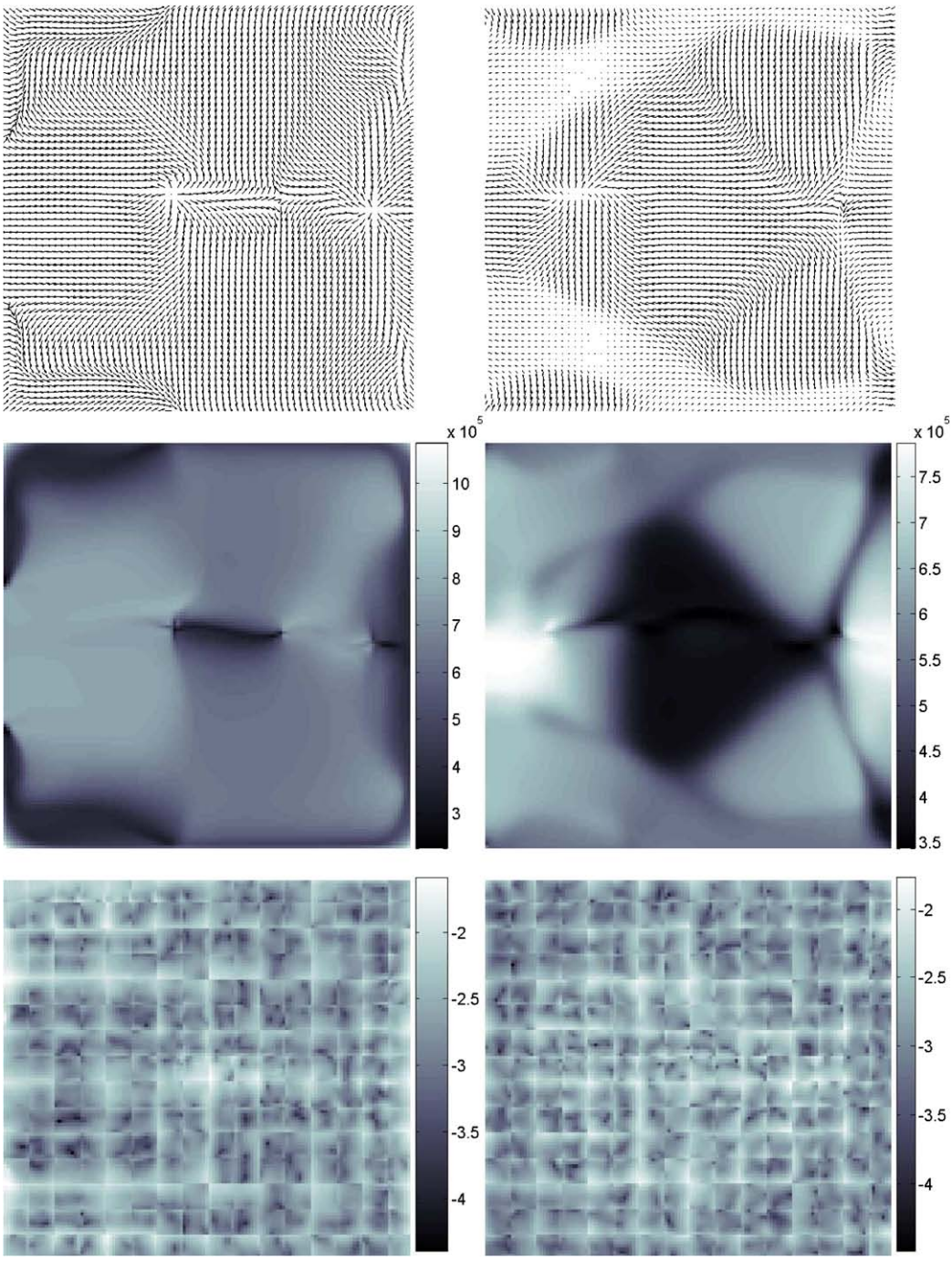

Fig. 12. Magnetization (up), amplitude of the magnetostatic field (middle) and normalized rms error on a logarithmic scale (bottom) in planes $z = 0$ (left) and $z = 16$ (right) of a sample with dimensions $128 \times 128 \times 128$ FD cells.

requirements generally result in a larger execution time. This is also the case in the presented algorithms. However, in the FMM scheme there is a very beneficial trade of: the memory savings are very large – about a factor 11 – compared with the small sacrifices concerning execution time (a factor 4–5). This trade of becomes even better for non-cubical shapes. For instance, when magnetic writing heads are simulated with or without the storage medium itself (see e.g. [30]) the geometry is far more complex. In the FMM scheme only the magnetic bodies are discretized while in the FFT scheme an enclosing rectangular prism has to be discretized (magnetic body and air). Hence the difference in computation time is further decreased. Fig. 11 also shows that for large dimensions the difference in execution time largely depends on the dimensions of the Fourier transforms used in the FFT scheme.

The accuracy of the FMM scheme is sufficient for the micromagnetic simulations. A higher accuracy is reached by increasing the number of expansion coefficients ($p > 6$). In that case only small changes to the MP to local translations are needed [27].

The FMM scheme is far more flexible then an FFT scheme. Curved magnetic bodies for example can easily be discretized in the FMM scheme, but not in the FFT scheme. Indeed, by adjusting the boundaries in the integrals of (13), one can restrict the integrations to only magnetic material present in the FD cell in stead of the whole FD cell. In the near interaction computations, small corrections have to be made to compensate for the FD cells with curved elements. In that way, the magnetic body can have any shape, e.g. ellipsoidal which is a very popular shape to study magnetic reversal processes.

In a more complex variant of this FMM scheme the discretization of the magnetic body can be adaptive, using FD cells of different shape to describe the magnetic configuration state. This is very interesting since magnetic domains are present in the studied ferromagnetic samples. These are large uniformly magnetized regions in the material – which can be discretized with large FD cells – separated from each other by domain walls – which have to be discretized with small FD cells. Extending the presented FMM scheme with adaptive discretization will dramatically improve its performance.

Acknowledgments

This work was supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) SB/53032 and also by FWO project G.0142.08.

References

[1] H. Kronmüller, M. Fähnle, Micromagnetism and the microstructure of ferromagnetic solids, first ed., Cambridge University Press, 2003.
[2] D. Suess, T. Schrefl, S. Fahler, M. Kirschner, G. Hrkac, F. Dorfbauer, J. Fidler, Exchange spring media for perpendicular recording, Applied Physics Letters 87 (1) (2005) 012504.
[3] J. Oti, A micromagnetic model of dual-layer magnetic recording thin films, IEEE Transactions on Magnetics 29 (2) (1993) 1265–1275.
[4] B. Van de Wiele, L. Dupré, F. Olyslager, Influence of space discretization size in 3D micromagnetic modeling, Physica B 403 (2008) 372–375.
[5] L.D. Landau, E.M. Lifshitz, Electrodynamics of continuous media, Pergammon Press X, Oxford – London – New York – Paris, 1960.
[6] M. d'Aquino, C. Serpico, G. Miano, Geometrical integration of Landau–Lifshitz–Gilbert equation based on the mid-point rule, Journal of Computational Physics 209 (2) (2005) 730–753.
[7] X.P. Wang, C.J. Garcia-Cervera, W. E, A Gauss-Seidel projection method for micromagnetic simulations, Journal of Computational Physics 171 (2001) 357–372.
[8] B. Van de Wiele, F. Olyslager, L. Dupré, Fast semianalytical time integration schemes for the Landau–Lifshitz equation, IEEE Transactions on Magnetics 43 (6) (2007) 2917–2919.
[9] M.E. Schabes, A. Aharoni, Magnetostatic fields for a 3-dimensional array of ferromagnetic cubes, IEEE Transactions on Magnetics 23 (1987) 3882–3888.
[10] A.J. Newel, W. Williams, D.J. Dunlop, A generalization of the demagnetizing tensor for nonuniform magnetization, Journal of Geophysical Research – Solid Earth 98 (1993) 9551–9555.
[11] B. Van de Wiele, F. Olyslager, L. Dupré, Fast numerical 3D scheme for the simulation of hysteresis in ferromagnetic grains, Journal of Applied Physics 101 (2007) 073909.
[12] R. Ferré, Large scale micromagnetic calculations for finite and infinite 3D ferromagnetic systems using FFT, Computer Physics Communications 105 (1997) 169–186.
[13] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93 (2) (2005) 216–231.
[14] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, MIT Press, Cambridge, MA, 1988.
[15] Y.C.C. Pan, W.C. Chew, L.X. Wan, A fast multipole-method-based calculation of the capacitance matrix for multiple conductors above stratified dielectric media, IEEE Transactions on Microwave Theory and Techniques 49 (3) (2001) 480–490.
[16] N.A. Gumerov, R. Duraiswami, Fast multipole method for the biharmonic equation in three dimensions, Journal of Computational Physics 215 (1) (2006) 363–383.
[17] J. Kurzak, B.M. Pettitt, Fast multipole methods for particle dynamics, Molecular Simulation 32 (10–11) (2006) 775–790.
[18] D. Pissoort, E. Michielssen, D. Vande Ginste, F. Olyslager, Fast-multipole analysis of electromagnetic scattering by photonic crystal slabs, Journal of Lightwave Technology 25 (9) (2007) 2847–2863.
[19] M. Fischer, U. Gauer, L. Gaul, A multipole Galerkin boundary element method for acoustics, Engineering Analysis with Boundary Elements 28 (2) (2004) 155–162.
[20] E.E. Ong, K.M. Lim, K.H. Lee, H.P. Lee, A fast algorithm for three-dimensional potential fields calculation: fast Fourier transform on multipoles, Journal of Computational Physics 192 (2003) 244–261.
[21] Z.J. Liu, H.H. Long, E.T. Ong, E.P. Li, A fast Fourier transform on multipole algorithm for micromagnetic modeling of perpendicular recording media, Journal of Applied Physics 99(8) 08B903.
[22] G. Brown, T.C. Schulthess, D.M. Apalkov, P.B. Visscher, Flexible fast multipole method for magnetic simulations, IEEE Transactions on Magnetics 40 (4) (2004) 2146–2149.
[23] N.A. Gumerov, R. Duraiswami, E.A. Borovikov, Data structures, optimal choice of parameters and complexity results for generalized multilevel fast multipole methods in d dimensions, Computer Science Technical Report CSTR # 4458.
[24] M. Abramowitz, I.A. Stegun, Handbook of Mathematical Functions, McGraw Hill, 1960.
[25] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numerica 6 (1997) 229–269.
[26] O. Coulaud, P. Fortin, J. Roman, High performance BLAS formulation of the multipole-to-local operator in the fast multipole method, Journal of Computational Physics 227 (2008) 1836–1862.
[27] W.D. Elliott, J.A. Board Jr., Fast Fourier transform accelerated fast multipole algorithm, SIAM Journal on Scientific Computing 17 (2) (1996) 198–415.
[28] P. Fortin, Multipole-to-local operator in the fast multipole method: comparison of FFT, rotations and BLAS improvements, Technical Report, Institut National de Recherche en Informatique et en Automatique, Cedex, France.
[29] M. Frigo, S.G. Johnson, FFTW User Manual. <www.fftw.org>.
[30] K. Takano, Magnetization dynamics of planar writers, IEEE Transactions on Magnetics 40 (1) (2004) 257–262.