Comput. Methods Appl. Mech. Engrg. 198 (2009) 1473–1485

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Comparison between wavelet and fast multipole data sparse approximations for Poisson and kinematics boundary – domain integral equations

J. Ravnik *, L. Škerget, Z. Žunič

Faculty of Mechanical Engineering, University of Maribor, Smetanova ulica 17, SI-2000 Maribor, Slovenia

ARTICLE INFO

Article history:
Received 22 May 2008
Received in revised form 16 December 2008
Accepted 22 December 2008
Available online 6 January 2009

Keywords:
Wavelets
Fast multipole method
Poisson equation
BEM

ABSTRACT

The boundary element method applied on non-homogenous partial differential equations requires calculation of a fully populated matrix of domain integrals. This paper compares two techniques: the fast multipole method and the fast wavelet transform, which are used to reduce the complexity of such domain matrices. The employed fast multipole method utilizes the expansion of integral kernels into series of spherical harmonics. The wavelet transform for vectors of arbitrary length, based on Haar wavelets and variable thresholding limit, is used. Both methods are tested and compared by solving the scalar Poisson equation and the velocity–vorticity vector kinematics equation. The results show comparable accuracy for both methods for a given data storage size. Wavelets are somewhat better for high and low compression ratios, and the fast multipole methods gives better results for moderate compressions. Considering implementation of the methods, the wavelet transform can easily be adapted for any problem, while the fast multipole method requires different expansion for each integral kernel.

1. Introduction

There are numerous techniques for solving partial differential equations. Most of them, such as FVM or FEM, require discretization of the problem domain. The solution is obtained by solving a sparse system of linear equations, with the number of degrees of freedom equal to the number of nodes in the domain. The boundary element method (BEM), on the other hand, uses the fundamental solution of the differential operator and the Green's theorem to write an equivalent boundary integral equation. After discretization of only the boundary of the problem domain, a fully populated system of equations emerges. The number of degrees of freedom is equal to the number of boundary nodes. This reduction of the dimensionality of the problem is a major advantage over the volume based methods. Fundamental solutions are known for a wide variety of differential operators [26], making BEM applicable for solving a wide range of problems.

When dealing with more general linear or non-linear differential operators or with non-homogenous problems, for which the fundamental solutions do not exist, the integral equation can not be written completely by boundary integrals only. A domain integral is also present, requiring the discretization of the domain. In order to write a system of linear equations for boundary unknowns, a matrix of domain integrals must be evaluated. This matrix is fully populated, its order is $n_b \times n_d$, where n_b is the number of nodes on the boundary and n_d is the number of nodes in the domain. The storage requirements and CPU times needed to perform matrix vector operations with such a matrix are huge. The primary advantage of BEM is lost.

The dual reciprocity BEM [15,7] is one of the most popular techniques to eliminate the domain integrals. It uses expansion of the non-homogenous term in terms of radial basis functions. Several other approaches, that enable construction of data sparse approximations of fully populated matrices, are also known. Hackbusch and Nowak [12] developed a panel clustering method, which also enables approximate matrix vector multiplications with decreased amount of arithmetical work. A class of hierarchical matrices was introduced by Hackbusch [10,11] with the aim of reducing the complexity of matrix–vector multiplications. Bebendorf and Rjasanow [2] developed an algebraic approach for solving integral equations using collocation methods with almost linear complexity.

In this paper, we are comparing the following two techniques for avoiding the domain integral matrix problem: the fast multipole method (FMM) and the wavelet transform method (WTM). Both are used to provide a sparse approximation of the fully populated domain matrix. The element count of the sparse approximations scales as $\mathcal{O}(n_d)$. This makes the domain matrix storage requirements lower than the storage requirements of the system matrix, which scales as number of boundary nodes squared $\mathcal{O}(n_b^2)$. Both techniques eliminate the storage and CPU time problems, so BEM can be efficiently applied on any partial differential equation.

* Corresponding author. Tel.: +386 22207745; fax: +386 22207990.
E-mail addresses: jure.ravnik@uni-mb.si (J. Ravnik), leo@uni-mb.si (L. Škerget), zoran.zunic@uni-mb.si (Z. Žunič).

0045-7825/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2008.12.012

The origins of the FMM can be found in a fast algorithm for particle simulations developed by Greengard and Rokhlin [9]. The algorithm decreases the amount of work required to evaluate mutual interaction of particles by reducing the complexity of the problem from quadratic to linear. Ever since, the method was used by many authors for a wide variety of problems using different expansion strategies. Recently, Englund [8] used the FMM to perform matrix vector multiplication in algorithm for crack growth simulations. Schwab and Todor [24] used a kernel independent fast multipole method based algorithm for evaluation of Karhunen–Loève eigenvalues. The boundary integral Laplace equation was accelerated with FMM by Popov et al. [16]. In contrast to the contribution of this paper, where the subject of study is the application of FMM to obtain a sparse approximation of the domain matrix, the majority of work done by other authors dealt with coupling BEM with FMM for the boundary matrices.

The wavelet transform is a recent mathematical tool, developed specially for saving computational time and computer storage. It has been widely used for image compression and signal processing and recently for providing faster solutions of boundary element algorithms. Rathsfeld [18] presented a wavelet algorithm for the boundary element solution of a geodetic boundary value problem. Beylkin et al. [3] proposed the fast wavelet transform algorithm, which ensures $O(n \log n)$ non-zero elements at a fixed thresholding limit, using Daubechies [6] wavelets. In our previous work [19], we developed a wavelet technique that provides sparse approximation of fully populated domain integral matrices of any size. It was used for acceleration of a 2D BEM numerical method for simulations of fluid flow [21]. In this work, the same technique will be improved with the introduction of variable thresholding limit [5] and used in 3D. Its properties will be compared with the newly developed FMM algorithm. The wavelet transform is a purely algebraic technique. It can be applied on any matrix of any size. On the other hand, the FMM requires the expansion of the integral kernel, thus each kernel must be dealt with separately.

The Poisson equation is known to describe many processes in physics and engineering. Its solution is among prime interest among researchers [14]. In this work, we are solving two types of Poisson equations. Firstly, a scalar Poisson equation in 3D, where the integral kernel is the Laplace fundamental solution, and secondly the boundary-domain integral form of the velocity–vorticity vector kinematics equation, where the domain integral kernel is the gradient of the Laplace fundamental solution. In both cases, we set up sparse representations of the full domain matrix by the wavelet and fast multipole methods.

The rest of this paper is organized as follows: in the second section BEM for a scalar Poisson equation is introduced and the third section BEM for the velocity–vorticity vector kinematics equation is explained. Section 4 develops the FMM for the domain matrix and Section 5 the wavelet transform. The techniques are thoroughly tested and examined in Section 6. The main findings and conclusions are summarized in the last section.

2. BEM for the poisson equation

The Poisson equation is a partial differential equation with a diffusion differential operator and a non-homogenous right hand side, i.e.

$$\nabla^2 u(\vec{r}) = b(\vec{r}); \quad \vec{r} \in \Omega, \tag{1}$$

where the unknown scalar field function $u(\vec{r})$ and the non-homogenous source term $b(\vec{r})$ are defined in a domain Ω. The solution can be found when suitable boundary conditions are applied, i.e. known scalar function or its flux ($q = \vec{n} \cdot \vec{\nabla} u$) on the boundary $\Gamma = \partial\Omega$. An integral form of Poisson type equation for a scalar field function $u(\vec{r}) \in \Omega$ is [26]:

$$c(\vec{\xi})u(\vec{\xi}) + \int_\Gamma u(\vec{r})\vec{n} \cdot \vec{\nabla} u^\star d\Gamma = \int_\Gamma q(\vec{r}) u^\star d\Gamma - \int_\Omega b(\vec{r}) u^\star d\Omega; \quad \vec{\xi} \in \Gamma, \tag{2}$$

where $\vec{\xi}$ is the collocation point on the boundary, $\vec{n}$ is the unit normal, c is the geometrical factor and $u^\star = 1/4\pi|\vec{r} - \vec{\xi}|$ is the fundamental solution of the Laplace equation in 3D. The domain is approximated by domain cells $\Omega \approx \sum_{c=1}^{n_c} \Omega_c$ and its boundary by boundary elements $\Gamma \approx \sum_{e=1}^{n_e} \Gamma_e$. Within each domain cell and boundary element the field functions are approximated using shape functions. In this paper, domain cells are hexahedra and boundary elements are parallelepipeds. In each domain cell domain shape functions Φ with 27 nodes are used to achieve quadratic interpolation of function. Boundary shape functions φ with nine continuous nodes are used in boundary elements for quadratic interpolation of function. Flux is interpolated linearly using boundary flux shape functions ϕ with four discontinuous nodes. A sketch of a boundary element and a domain cell is given in Fig. 1 showing locations of nodes. Considering these approximations in Eq. (2) we have:

$$c(\vec{\xi})u(\vec{\xi}) + \sum_{e=1}^{n_e} \sum_{i=1}^{9} u_i^e \int_{\Gamma_e} \varphi_i^e \vec{n} \cdot \vec{\nabla} u^\star d\Gamma = \sum_{e=1}^{n_e} \sum_{i=1}^{4} q_i^e \int_{\Gamma_e} \phi_i^e u^\star d\Gamma - \sum_{c=1}^{n_c} \sum_{i=1}^{27} b_i^c \int_{\Omega_c} \Phi_i^c u^\star d\Omega. \tag{3}$$

The integrals are traditionally named as

$$h_i^{e,\vec{\xi}} = \int_{\Gamma_e} \varphi_i^e \vec{n} \cdot \vec{\nabla} u^\star d\Gamma, \quad g_i^{e,\vec{\xi}} = \int_{\Gamma_e} \phi_i^e u^\star d\Gamma, \quad \beta_i^{c,\vec{\xi}} = \int_{\Omega_c} \Phi_i^c u^\star d\Omega. \tag{4}$$

For a given collocation point $\vec{\xi}$ we must calculate integrals for each internal cell c, each boundary element e and each shape function i. When the collocation point $\vec{\xi}$ is placed into all of boundary nodes, integrals are arranged into matrices and the system of linear equations is written in matrix–vector form. Let [] denote matrices and {} denote vectors. In matrix–vector form Eq. (3) is:

$$[H]\{u\} = [G]\{q\} - [B]\{b\}. \tag{5}$$

Since the integral kernels are non-zero for all collocation points, the matrices of Eq. (5) are unsymmetrical and fully populated. Considering the boundary conditions, the system of Eq. (5) is rearranged so that the unknown values of function and flux are gathered on the left side. A direct solver with LU decomposition is used to solve the resulting system. In order to evaluate the right hand side of the system, we must calculate the domain matrix times vector product $[B]\{b\}$. Since the domain matrix is fully populated, we have lost the advantage of the boundary element method. The non-homogenous part of the Poisson equation requires the discretization of the domain and the calculation of a fully populated domain matrix.

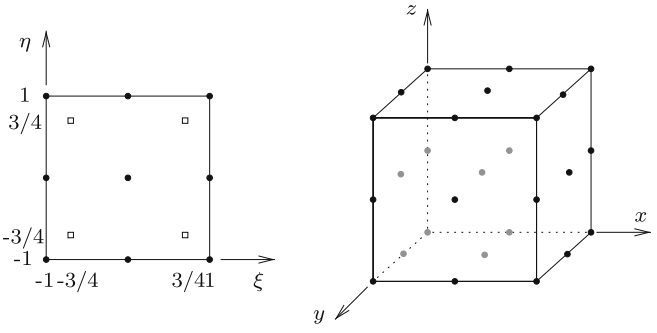

Fig. 1. A boundary element (left) in a local coordinate system with locations of function nodes (circles) and flux nodes (squares). A domain cell (right) with function node locations.

The order of the boundary matrix $[H]$ is the number of boundary nodes (function nodes n_b^u plus flux nodes n_b^q; $n_b = n_b^u + n_b^q$) times the number of boundary function nodes $n_b \times n_b^u$. The order of the boundary matrix $[G]$ is $n_b \times n_b^q$. These sizes are small compared to the order of the domain matrix $[B]$, which is $n_b \times n_d$, since $n_d \gg n_b$ for all 3D geometries meshed by a dense mesh. Thus the number of elements in boundary matrices scales as $\mathcal{O}(n_b^2)$ and the number of elements in the domain integral matrix scales as $\mathcal{O}(n_d \cdot n_b)$. Considering a cube, one can estimate $n_d = N^3$ and $n_b \approx N^2$. So the complexity of the domain integral matrix is $\mathcal{O}(N^5)$, where N is the number of nodes on one edge of our model cube, while the boundary matrices scale as $\mathcal{O}(N^4)$. Since, clearly, the domain integral matrix takes up most of the CPU time and storage space, this paper presents an application of the fast multipole method and wavelet transform method to build a sparse approximation of the domain matrix in such a way that the number of non-zero elements scales linearly as $\mathcal{O}(n_d)$.

3. BEM for the kinematics equation

We are considering flow of an incompressible fluid. Let its velocity vector field be denoted by $\vec{v}$ and its vorticity field by $\vec{\omega} = \vec{\nabla} \times \vec{v}$. The kinematics equation is a vector elliptic partial differential equation of Poisson type, which links the velocity and vorticity fields for every point in space and time. It is equivalent to the Biot–Savart law, which connects the electric current and magnetic field density. For an incompressible fluid it can be stated as

$$\nabla^2 \vec{v} + \vec{\nabla} \times \vec{\omega} = 0, \tag{6}$$

where we must bear in mind, that both velocity and vorticity fields are divergence free.

The kinematics equation is a Poisson equation with a curl of vorticity on the right hand side, so its integral form is analogous to (2):

$$c(\xi)\vec{v}(\xi) + \int_\Gamma \vec{v}(\vec{n}\cdot\vec{\nabla})u^\star d\Gamma = \int_\Gamma u^\star(\vec{n}\cdot\vec{\nabla})\vec{v}d\Gamma + \int_\Omega (\vec{\nabla}\times\vec{\omega})u^\star d\Omega, \quad \xi \in \Gamma. \tag{7}$$

By taking into account the fact that both velocity and vorticity fields are solenoidal, it is possible to rewrite Eq. (7) into a form without derivatives of the velocity and vorticity fields (for derivation, see Ravnik et al. [20] Eqs. (19)–(24)):

$$c(\vec{\xi})\vec{v}(\vec{\xi}) + \int_\Gamma \vec{v}\vec{\nabla}u^\star \cdot \vec{n}d\Gamma = \int_\Gamma \vec{v}\times(\vec{n}\times\vec{\nabla})u^\star d\Gamma + \int_\Omega (\vec{\omega}\times\vec{\nabla}u^\star)d\Omega. \tag{8}$$

In order to use the kinematics equation to obtain boundary vorticity values, we rewrite Eq. (8) in a tangential form to improve the conditioning of the system. This is done by multiplying the equation with a normal in the source point $\vec{n}(\vec{\xi})$:

$$c(\vec{\xi})\vec{n}(\vec{\xi})\times\vec{v}(\vec{\xi}) + \vec{n}(\vec{\xi})\times\int_\Gamma \vec{v}\vec{\nabla}u^\star\cdot\vec{n}d\Gamma = \vec{n}(\vec{\xi})\times\int_\Gamma \vec{v}\times(\vec{n}\times\vec{\nabla})u^\star d\Gamma + \vec{n}(\vec{\xi})\times\int_\Omega(\vec{\omega}\times\vec{\nabla}u^\star)d\Omega. \tag{9}$$

This approach has been proposed by Škerget and was used in 3D by Ravnik et al. [23] and Zunič et al. [27]. Since there are no fluxes in Eq. (9) only boundary function and domain shape functions are required to write its discrete counterpart:

$$c(\vec{\xi})\vec{n}(\vec{\xi})\times\vec{v}(\vec{\xi}) + \vec{n}(\vec{\xi})\times\sum_{e=1}^{n_e}\sum_{i=1}^{9}\vec{v}_i^e\int_{\Gamma_e}\varphi_i^e\vec{\nabla}u^\star\cdot\vec{n}d\Gamma = \vec{n}(\vec{\xi})\times\sum_{e=1}^{n_e}\sum_{i=1}^{9}\vec{v}_i^e\times\int_{\Gamma_e}\varphi_i^e(\vec{n}\times\vec{\nabla})u^\star d\Gamma + \vec{n}(\vec{\xi})\times\sum_{c=1}^{n_c}\sum_{i=1}^{27}\vec{\omega}_i^c\times\int_{\Omega_c}\Phi_i^c\vec{\nabla}u^\star d\Omega. \tag{10}$$

Similarly to the Poisson equation, we may name the integrals as

$$h_i^{e,\vec{\xi}} = \int_{\Gamma_e}\varphi_i^e\vec{\nabla}u^\star\cdot\vec{n}d\Gamma, \quad \vec{h}t_i^{e,\vec{\xi}} = \int_{\Gamma_e}\varphi_i^e(\vec{n}\times\vec{\nabla})u^\star d\Gamma, \quad \vec{\beta}_i^{c,\vec{\xi}} = \int_{\Omega_c}\Phi_i^c\vec{\nabla}u^\star d\Omega. \tag{11}$$

We observe an important difference between the domain integral in (11) and the domain integral (4) of the Poisson equation. Namely, we now have three components of the domain integral leading to three domain matrices. Their kernels are the components of the gradient of the fundamental solution, while the kernel of the domain integral in Eq. (4) is the fundamental solution itself.

In order to write linear systems of equations for the unknown boundary vorticity values, we place the collocation point into every boundary node of the whole computational domain. This yields three full system matrices and several right hand side matrices of the order number of boundary nodes squared and three domain matrices of the order $n_b \times n_d$. The systems are solved using a LU decomposition method.

The same reasoning regarding storage space and CPU time as is Section 2 applies for the kinematics equation as well. The domain integral matrices take up most resources, thus we will apply FMM and WTM for them.

4. Fast multipole method for the domain matrix

Let us consider the domain integrals in Eqs. (4) and (11). Since for each collocation point $\vec{\xi}$ integrals for all domain cells c must be evaluated, we are obviously faced by a $\mathcal{O}(n^2)$ type problem. This is analogous to the problem of interaction of n particles [1], where the origins of the FMM can be found.

4.1. Series expansion

The FMM is based on the fact that it is possible to expand the domain integral kernel, (i.e. the fundamental solution in Eq. (4) and its gradient in Eq. (11)) into a series and by doing so, separate the variables – the collocation point $\vec{\xi}$ and the domain integration point $\vec{r}$. Fig. 2 states the geometry of the problem.

In this work, we will use spherical harmonics to expand the integral kernels into series. Other expansions are also possible, such as Taylor series, Lagrangian polynomials, etc. The Laplace fundamental solution is expanded into a series in the following manner:

$$u^\star = \frac{1}{4\pi|\vec{r}-\vec{\xi}|} = \frac{1}{4\pi\sqrt{r^2 - 2r\xi\cos\gamma + \xi^2}} = \frac{1}{4\pi\xi}\sum_{l=0}^{\infty}\left(\frac{r}{\xi}\right)^l P_l(\cos\gamma), \tag{12}$$

where P_l are the Legendre polynomials and γ is the angle between $\vec{\xi}$ and $\vec{r}$. In order for the series (12) to converge we must have $r/\xi < 1$. This condition is not satisfied for all $\vec{\xi}$ and $\vec{r}$ combinations in an arbitrary domain. However, since the integral kernel depends only on the distance between the collocation and domain points, we may swap $\vec{\xi}$ and $\vec{r}$ in order to meet the convergence criteria. Furthermore, it is also possible to move the origin of the coordinate system so that the series convergence is improved. Each Legendre polynomial can be written as a finite series of spherical harmonics Y. In polar coordinate system, where $\vec{r} = (r, \varphi_r, \theta_r)$ and $\vec{\xi} = (\xi, \varphi_\xi, \theta_\xi)$, we write

$$P_l(\cos\gamma) = \frac{4\pi}{2l+1}\sum_{m=-l}^{l}(-1)^m Y_l^{-m}(\theta_\xi, \varphi_\xi)Y_l^m(\theta_r, \varphi_r). \tag{13}$$

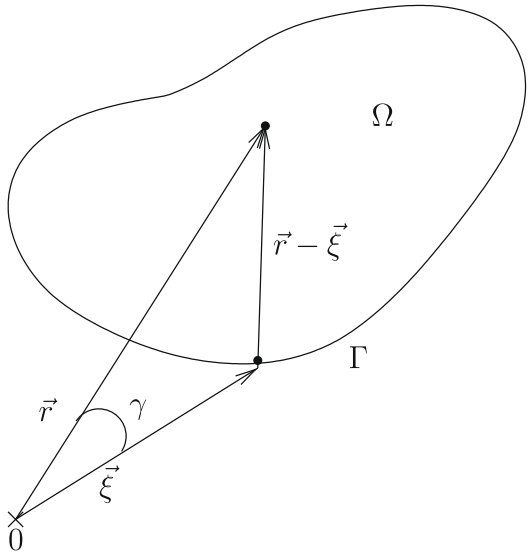

Fig. 2. A problem domain Ω, its boundary Γ, a collocation point $\vec{\xi}$ and a domain integration point $\vec{r}$.

Using Eq. (13) in (12) we obtain an expression for the integral kernel

$$\frac{1}{4\pi|\vec{r}-\vec{\xi}|}=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(-1)^m}{2l+1}\underbrace{\frac{1}{\xi^{l+1}}Y_l^{-m}(\theta_\xi,\varphi_\xi)}_{f(\vec{\xi})}\underbrace{r^lY_l^m(\theta_r,\varphi_r)}_{g(\vec{r})}, \tag{14}$$

where the dependence on the collocation point and domain point are separate. The spherical harmonics are calculated using their relationship to associated Legendre polynomials P_l^m

$$Y_l^m(\theta,\varphi)=\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)e^{im\varphi}. \tag{15}$$

The associated Legendre polynomials are evaluated using recurrence relations as described in Press et al. [17]. Using the same reasoning, we may expand the gradient of the fundamental solution as:

$$\begin{aligned}\vec{\nabla}u^\star&=\vec{\nabla}\frac{1}{4\pi|\vec{r}-\vec{\xi}|}=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(-1)^m}{2l+1}\frac{1}{\xi^{l+1}}Y_l^{-m}(\theta_\xi,\varphi_\xi)\vec{\nabla}\left[r^lY_l^m(\theta_r,\varphi_r)\right]\\&=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(-1)^m}{2l+1}\frac{1}{\xi^{l+1}}Y_l^{-m}(\theta_\xi,\varphi_\xi)\\&\quad\times\left\{lY_l^m(\theta_r,\varphi_r)r^{l-2}\vec{r}+r^l\vec{\nabla}Y_l^m(\theta_r,\varphi_r)\right\}.\end{aligned} \tag{16}$$

The gradient of spherical harmonics may be written as

$$\begin{aligned}\vec{\nabla}Y_l^m(\theta,\varphi)&=\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\vec{\nabla}\left\{P_l^m(\cos\theta)e^{im\varphi}\right\}\\&=\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}\vec{\nabla}\\&\quad\times\left\{P_l^m(\cos\theta)\frac{\partial e^{im\varphi}}{\partial\varphi}\vec{\nabla}\varphi-\sin(\theta)\frac{\partial P_l^m(\cos\theta)}{\partial\cos\theta}\vec{\nabla}\theta\right\}.\end{aligned} \tag{17}$$

The derivatives of associated Legendre polynomials are obtained using the following recurrence relation:

$$\frac{\partial P_l^m(x)}{\partial x}=\frac{lxP_l^m(x)-(l+m)P_{l-1}^m(x)}{x^2-1}, \tag{18}$$

while the polar angles derivatives written in Cartesian coordinate system are

$$\begin{aligned}\vec{\nabla}\theta&=\frac{\sqrt{x^2+y^2}}{x^2+y^2+z^2}\left(\frac{zx}{x^2+y^2},\frac{zy}{x^2+y^2},-1\right),\\\vec{\nabla}\varphi&=\frac{1}{x^2+y^2}(-y,x,0).\end{aligned} \tag{19}$$

Using the above expansions, the domain integrals of Eqs. (4) and (11) may now be written with separate variables, as

$$\beta_i^{c,\vec{\xi}}\approx\sum_{l=0}^{L}\sum_{m=-l}^{l}F_l^m(\vec{\xi})\int_{\Omega_c}G_l^{m,i}(\vec{r})d\Omega, \tag{20}$$

where F and G represent the above derived relationships. We are able to approximately calculate each entry in the domain matrices with the above sum. The number of expansion terms $n_{exp}=(L+1)^2$ in the series controls the accuracy of the approximation.

4.2. Cluster trees

Let us consider a cluster of n_r nearby collocation points and a cluster of n_c nearby domain cells, as illustrated in Fig. 3. These correspond to a $n_r\times n_c$ matrix block, which is a part of the domain matrix. Since the variables in Eq. (20) are separated, it is possible to evaluate two lower order matrix blocks $(n_r\times n_{exp})$ and $(n_{exp}\times n_c)$, where n_{exp} is the number of expansion terms. In the first one expansion terms F are evaluated for all collocation points. In the second one integrals of expansion terms G are evaluated for all domain cells. Multiplication of the two lower order matrix blocks gives the full $n_r\times n_c$ matrix block up to an expansion error, which is defined by the number of terms in the expansion. But this is never done; rather we store the two lower order matrices instead of the full matrix. This technique saves memory if the amount of data, that must be stored in the two lower order matrices, is smaller than the amount of data in the full matrix block, i.e.

$$2(n_rn_{exp}+n_cn_{exp})<n_rn_c; \tag{21}$$

the factor 2 on the left hand side is due to the fact that spherical harmonics are complex and must be stored as such, while real values are stored in the full matrix. As long as the collocation node cluster and the domain cells cluster are far apart from each other the integral kernels are slowly varying functions, so we can expect a low number of expansion terms to yield a suitable approximation. When the clusters are nearby, they should be smaller and a larger number of expansion terms must be used. When the clusters coincide, i.e. the collocation nodes are a part of the integration cells, the kernels are singular. Such cluster pairs are called inadmissible and the corresponding matrix block is evaluated in full, not approximated with two lower order matrices.

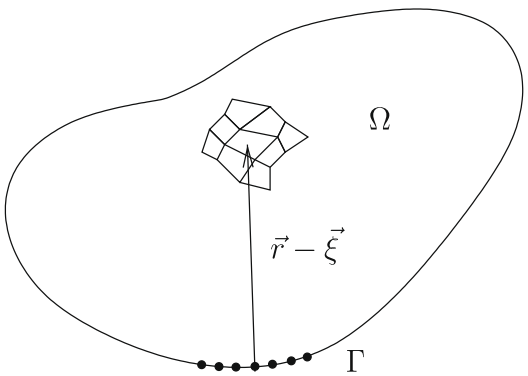

Fig. 3. A problem domain shown with a cluster of collocation points $\vec{\xi}$ and a cluster of domain cells.

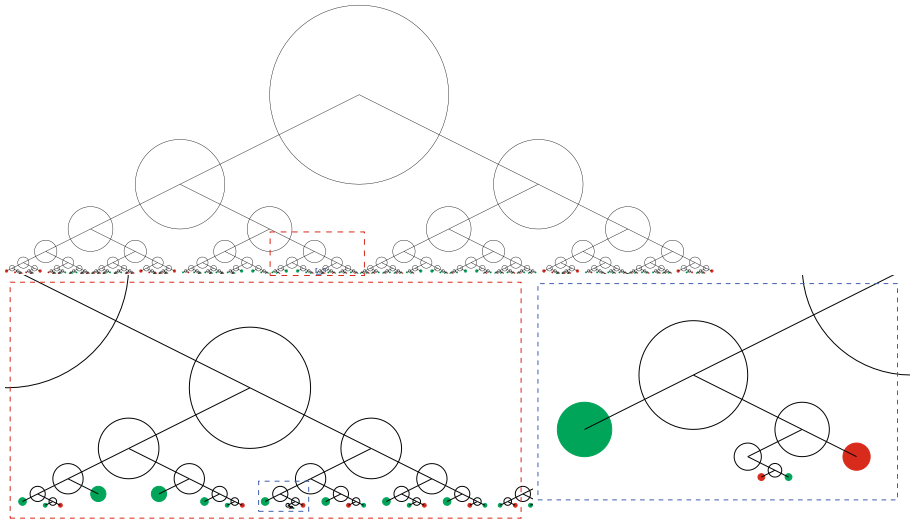

Fig. 4. The tree of pairs of clusters of a very small mesh ($4 \times 4 \times 4$) is shown along with two subsequent zooms. Each circle represents a branch in the tree of pairs of clusters with a cluster of collocation points paired with a cluster of domain cells. Green (light colour) circles represent admissible leaves; red (dark colour) circles represent inadmissible leaves. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

In order to be able to build a sparse approximation of the whole domain matrix, we must divide the collocation points and domain cells into clusters. We constructed a tree of collocation point clusters and a tree of clusters of domain cells. The trees were constructed in a recursive hierarchical manner. The problem domain was enclosed by a parallelepiped. All collocation points and all of the domain cells are within this root parallelepiped. They make up root clusters of both trees. The parallelepiped is cut in half by a plane, breaking the root clusters into two. The cutting process is repeated recursively, so the clusters on each level have less and less collocation points and domain cells. Each branch in the tree of clusters has two child branches corresponding to the cluster's domain being cut in half. The cutting planes are parallel to the coordinate system axes, a sequence of $x-y$, $x-z$ and $y-z$ is used. Thus three cuts are needed to cut a cube into eight equal parts. The cutting sequence is stopped, when the number of collocation nodes and domain cells in the cluster is so small, that the condition (21) can no longer be satisfied.

With both cluster trees in place, the next step is to pair them, so a tree of pairs of clusters can be formed. Each branch of the collocation tree is paired with each branch of the domain cells tree on the same level and with each branch of the domain cells tree on the next level thus forming branches on the tree of pairs of clusters. For each pair a decision is taken based on the admissibility criterion whether a sparse approximation for this cluster pair is possible or not. If the pair is admissible, the branch on the tree becomes an admissible leaf, where the two low-order matrices will be calculated. If admissibility criterion is not reached until the last level of the tree, such cluster pairs are inadmissible and will be calculated in full and not with the sparse approximation.

4.3. Admissibility criterion

The admissibility criterion is devised as follows. Let us consider one branch of the tree of pairs of clusters, which has a cluster of collocation points and a cluster of domain cells. Firstly, we try to find an origin of the coordinate system in nodes within the domain cells of the cluster. We choose such origin that the ratio r/ξ is minimal for all pairs of collocation nodes and domain cells so the series will converge as fast as possible. If the minimal ratio is above one, series expansion for this pair of clusters is not possible. Thus this pair is not admissible. Secondly, if the r/ξ ratio is below one, we calculate the number of expansion terms needed to have the accuracy of calculation of the integral kernel less than user's prescribed criteria ϵ. If the number of expansion terms is low enough, so that condition (21) is fulfilled, this cluster pair is admissible. At this point the tree of pairs of clusters gets a leaf - no further branching is necessary.

To illustrate the algorithm, a cubic domain is considered meshed by 4^3 domain cells having in total 9^3 nodes. The tree of cluster pairs for this mesh is shown in Fig. 4. The corresponding matrix block structure is shown in Fig. 5. As the mesh density increases, the ratio between admissible and inadmissible blocks turns in favour of admissible blocks, yielding more saved memory. This can readily be seen by examining the matrix structure of a 16^3 cells mesh with 33^3 nodes on Fig. 6.

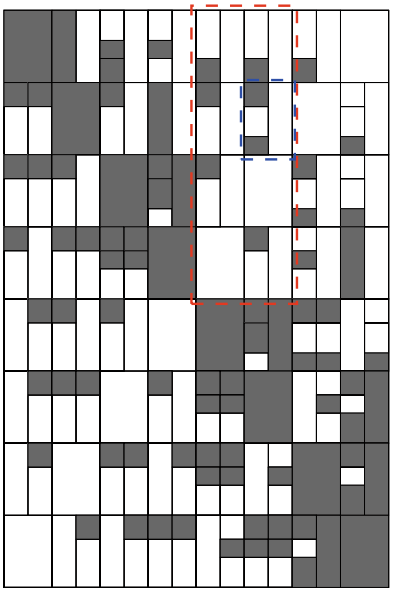

Fig. 5. Matrix structure of a cubic mesh (4^3 cells, 9^3 nodes). The depicted dashed rectangles correspond to tree parts in Fig. 4. Dark areas show inadmissible matrix blocks, white areas are admissible matrix blocks.

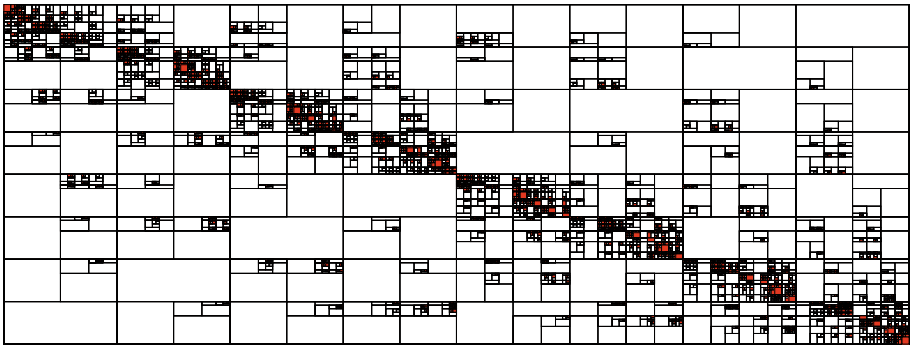

Fig. 6. Matrix structure of a cubic mesh (16^3 cells, 33^3 nodes). Filled areas show inadmissible matrix blocks, white areas are admissible matrix blocks obtained using an admissibility criteria of $\epsilon = 10^{-5}$. The corresponding tree of pairs of clusters has 19 levels.

4.4. Implementation

A set of routines was written to form a hierarchical tree structure with evaluated matrices on each of the admissible or inadmissible leaves. Let $[B']$ denote such a tree and represent the FMM data sparse approximation of the domain matrix.

The described FMM based algorithm was implemented into the BEM Poisson and kinematics equation solver codes. It is capable of constructing an approximation of the domain matrix $[B']$, which can be used to evaluate the right hand side of the system of equations. The advantages of using $[B']$ instead of the fully populated $[B]$ are summarized in the following points.

- Since we have taken careful care that the amount of data required to store matrices in all admissible leaves is smaller than data storage of their fully populated counterparts, we know that the memory required to store the FMM sparse approximation of the domain matrix $[B']$ will be less than the memory required to store $[B]$.
- The computer memory requirements needed to store the fully populated matrix scale as $\mathcal{O}(n_d \cdot n_b)$. By using FMM we were able to decrease the memory requirements to a linear dependence of the number of nodes, i.e. they scale as $\mathcal{O}(n_d)$. Proof of this is given in the numerical tests section below.
- Evaluation of the two lower order matrices for each admissible leaf is computationally less expensive than the computation of their full matrix block counterpart. For one, there are less matrix elements to evaluate; secondly one of the low-order matrices holds values only (see Eq. (20)) and not integrals. The elements of the second low-order matrix are integrals of slowly varying functions, thus less effort is needed to evaluate them.
- There are three domain matrices of Eq. (11), each holding integrals of one component of the gradient of the fundamental solution. Since one of the low-order matrices in admissible leaves holds values independent of $\vec{r}$, they are also independent of the gradient direction. Thus only one set of these matrices, which is common for all three directions, needs to be stored. This saves additional data storage space.

5. Wavelet transform for the domain matrix

Let $[B]$ be a domain matrix. In order to solve the Poisson or kinematics equations we must multiply the domain matrix with a vector to calculate the right hand side of the system. In this section we explain how to use wavelet transform to obtain a sparse approximation of the domain matrix, thus saving storage space and CPU time needed to make matrix vector multiplication.

Let W be a wavelet matrix, which if multiplied by a vector, transforms the vector into a vector of wavelet coefficients. It is set up using the Haar wavelet transform for vectors of arbitrary length and is capable of transforming matrices of arbitrary size. The W transform is still in its essence the Haar wavelet transform. Before the Haar transformation, the vector is modified in such manner, that just the right number of wavelet coefficient end up zero. Not storing zeros makes it possible to apply the W transform to a vector with an arbitrary number of components and store only the same number of wavelet coefficients. The matrix W is never stored in memory. Its structure and layout of non-zero elements are known, so the matrix is set up on the fly, during matrix vector multiplication. Algorithms for performing the wavelet matrix time vector product were developed by Ravnik et al. [19].

The transpose of the wavelet matrix is equal to its inverse, $W^T = W^{-1}$. If $\{b\}$ is a vector, we may write domain matrix times a vector product as

$$[B]\{b\} = W^T\left(W[B]W^T\right)W\{b\}. \tag{22}$$

The product $W[B]$ is the wavelet transform of all columns in $[B]$, while $(W[B])W^T$ transforms all rows in the product $W[B]$. Thus the majority of information is written in large elements of $[B_w] = W[B]W^T$, while the redundant information of $[B]$ is represented in small elements in $[B_w]$. Small elements of $[B_w]$ may be set to zero without greatly diminishing the accuracy of the matrix–vector product (22). This operation is called *thresholding* [4]. Elements in the matrix, whose absolute value is less than the thresholding limit, are set to zero. The zeros in the wavelet approximation of the domain matrix $[B_w]$ are not stored, thus saving storage space. Compressed row storage matrix format is used.

By setting the thresholding limit we can arbitrarily choose the amount of data in the wavelet approximation of the domain matrix $[B_w]$. In contrast to FMM, where it is not possible to arbitrarily choose the number of expansion terms and through them the size of the FMM approximation of the domain matrix $[B']$. The number of expansion terms is always a square of a natural number. This property is inherited from the spherical harmonic series expansion.

In Ravnik et al. [19], a fixed thresholding limit was used to zero out small elements in the wavelet matrix. It was shown by Dahmen et al. [5], Harbrecht and Schneider [13] and von Petersdorff and Schwab [25] that the thresholding limit should rather be wavelet level dependent. In the numerical tests we compared the fixed and three variable thresholding limits:

$$\epsilon = \begin{cases} \kappa\overline{m} & \dots \quad FT \\ 2^{J-\frac{j+j'}{2}}\kappa\overline{m}; \quad 0 \leqslant j, j' \leqslant J & \dots \quad VT2 \\ \sqrt{2}^{J-\frac{j+j'}{2}}\kappa\overline{m}; \quad 0 \leqslant j, j' \leqslant J & \dots \quad VT2^{0.5} \\ 0.5^{J-\frac{j+j'}{2}}\kappa\overline{m}; \quad 0 \leqslant j, j' \leqslant J & \dots \quad VT0.5 \end{cases} \tag{23}$$

Table 1
Comparison of memory requirements between the full matrix $[B]$ and the FMM data sparse matrix $[B']$ in megabytes [MB].

No. of nodes	$[B]$	Number of expansion terms of $[B']$				
		4	9	16	25	36
17^3	115	37	46	59	75	95
25^3	824	127	163	214	279	358
33^3	3369	296	408	565	768	1015
49^3	24,818	1157	1617	2260	3086	4097
65^3	102,988	2345	3503	5125	7210	9758

where $\overline{m}$ is the average matrix element value, κ is the user prescribed value, J is the maximal wavelet level and j and j' are the wavelet levels of the matrix element.

Since the wavelet transform method is purely algebraic, it can work on any matrix without any change to the algorithm. This is an advantage over the FMM, which is based on expansion of the integral kernel and thus requires major changes for each integral kernel. On the other hand, in order to construct the data sparse wavelet matrix, we require the calculation of the full matrix, while the FMM never requires the calculation of the full matrix.

6. Numerical tests

We developed two methods, which enable construction of data sparse approximations of domain integral matrices: the fast multipole expansion method was described in Section 4 and the wavelet transform method in Section 5. In this section we present several tests for the scalar Poisson equation (1) and vector kinematics Eq. (6), which were devised to compare the methods.

We made several tests in order to check the accuracy of using $[B']$ and $[B_w]$ as data sparse approximations of fully populated matrix $[B]$. By using coarse meshes (up to 33^3 nodes), we were able to compare results obtained by all three domain matrices and compare the accuracy of the results. The differences between results are presented in terms of the uniform norm and *RMS* value. For given vectors of nodal values $\{f\}$ and $\{g\}$ we define:

$$\|\{f\}-\{g\}\|_\infty = \max\{|f_1-g_1|,\ldots,|f_n-g_n|\},$$
$$RMS = \sqrt{\frac{\sum(f_i-g_i)^2}{\sum f_i^2}}. \tag{24}$$

In Section 6.1, we are testing domain matrices, which were obtained by integration of the Laplace fundamental solution in 3D. Section 6.2 deals with domain matrices, whose integral kernel is the gradient of the Laplace fundamental solution in 3D.

Table 2
Relationship between the data ratio $\mathcal{D}$ and the number of multipole expansion terms for different meshes is linear.

No. of terms	$\mathcal{D}$		
	17^3	25^3	33^3
4	0.317	0.154	0.088
9	0.397	0.198	0.121
16	0.509	0.260	0.168
25	0.652	0.339	0.228
36	0.827	0.435	0.301
49		0.549	0.388
64		0.681	0.488
81		0.830	0.601
100			0.728
121			0.868

6.1. The Poisson equation

Let the problem domain be a unit cube. A structured mesh with equally spaced hexahedral cells fills the cube. Let N be the number of nodes in each direction. The number of nodes in the domain is N^3 and the number of nodes on the boundary scales as $\mathcal{O}(N^2)$. The number of elements in the fully populated right hand side matrix $[B]$ thus scales as $\mathcal{O}(N^5)$. This fact is confirmed by the data of Table 1, where the storage requirements of $[B]$ and $[B']$ are compared for different meshes and different numbers of expansion terms. Naturally the storage requirements for $[B']$ increase, when the number of expansion terms is increased. However, looking at the storage requirement at a chosen number of expansion terms, we observe (see Fig. 7), that it increases linearly with the number of nodes, i.e. scales as $\mathcal{O}(N^3)$. This relationship holds regardless of the number of expansion terms. Thus, by employing the FMM, we were able to decrease the storage requirements from $\mathcal{O}(n_d \cdot n_b)$ to a linear dependence of $\mathcal{O}(n_d)$.

6.1.1. Matrix–vector multiplication

We chose one hundred random vectors $\{b\}$ and compared the results of multiplication with the fully populated matrix $\{f\} = [B]\{b\}$, the FMM data sparse matrix $\{g'\} = [B']\{b\}$ and the wavelet data sparse matrix $\{g_w\} = [B_w]\{b\}$. We calculated uniform norms and *RMS* values for FMM sparse approximations of matrices obtained with different number of expansion terms and wavelet

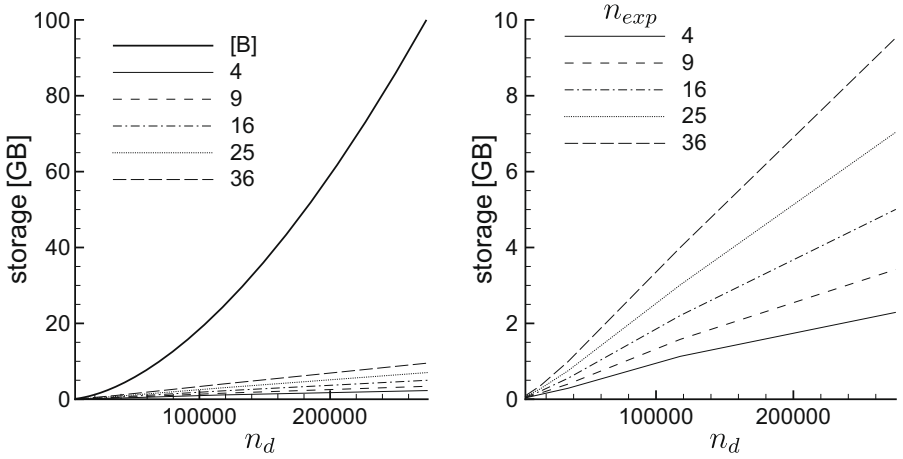

Fig. 7. The graphs show a comparison of the memory required to store a full matrix $[B]$ (thick solid line) and FMM data sparse matrices $[B']$ with different number of expansion terms. We observe the linear dependence of storage requirements with the number of domain nodes n_d regardless of the number of expansion terms.

matrices with different thresholding limits. The geometry was a unit cube. Three meshes with equal sized elements were considered: the first had 8^3 elements with 17^3 nodes, the second 12^3 elements with 25^3 nodes and the last 16^3 elements with 33^3 nodes.

In order to be able to compare all of the results, we decided to plot them against the data ratio. The data ratio $\mathcal{D}$ is defined as the amount of data required to store the sparse approximation of a matrix ($[B']$ or $[B_w]$) divided by the amount of data in the full $[B]$ matrix. Since the choice of the thresholding limit, which is used to zero out elements in the wavelet approximation of the matrix, is arbitrary, we can choose any data ratio. But, in the FMM, we are limited by the number of terms in the expansion, which can only be chosen as a square of a natural number. Table 2 shows the relationship between the data ratio and number of terms in the expansion for the three above mentioned meshes. The relationship is linear because of Eq. (21). The slope of the line gets steeper with increasing mesh density. The denser is the mesh the lower is data ratio at the same number of expansion terms. The FMM admissibility condition was set to $\epsilon = 10^{-3}$.

In Fig. 8, we compare the fixed and three variable wavelet thresholding approaches of Eq. (23). We are able to conclude that, especially for the low data ratios, the variable thresholding limit

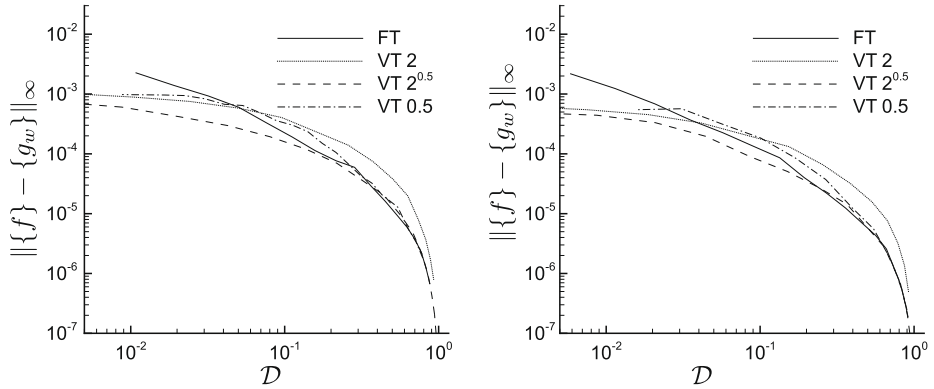

Fig. 8. The graphs present uniform norms of random vector multiplication with the full and wavelet matrices for three thresholding strategies of Eq. (23). Results of the 25^3 mesh are shown on the left and mesh 33^3 is shown on the right.

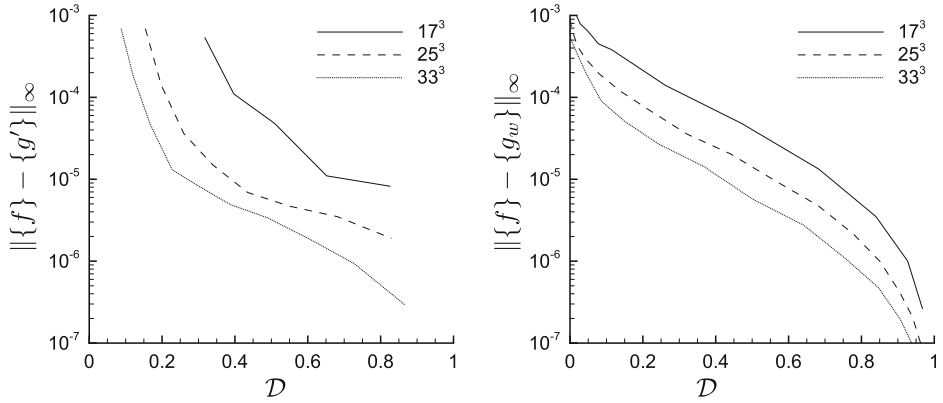

Fig. 9. The graphs present uniform norms of random vector multiplication with the full and FMM matrices (left) and full and wavelet matrix (right). For a given norm the data ratio of a dense mesh is smaller than the data ratio of a coarse mesh.

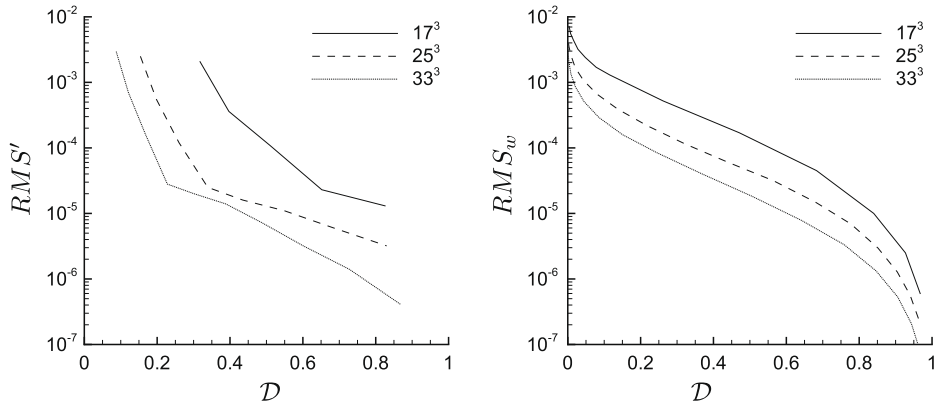

Fig. 10. The graphs present *RMS* values of random vector multiplication with the full and FMM matrices (left) and full and wavelet matrix (right).

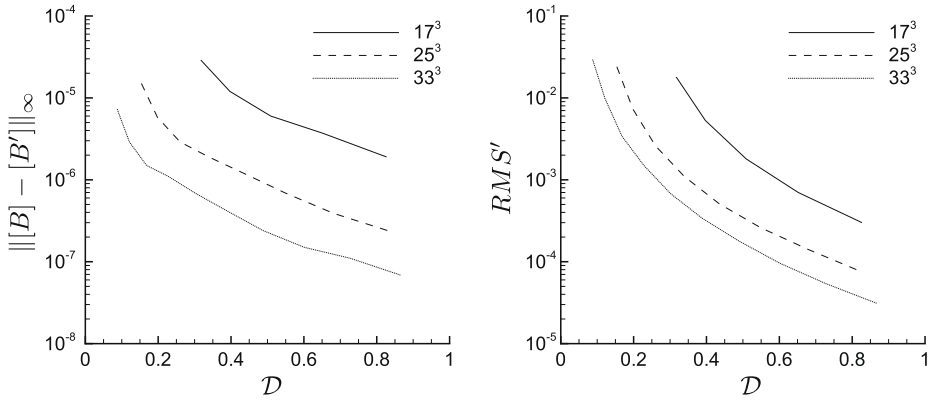

Fig. 11. The graphs present uniform norm and *RMS* values of comparison of FMM and full matrices.

approach give roughly an order of magnitude more accurate results than the fixed thresholding approach. Results show that $VT2^{0.5}$ approach works best, thus we employed it for all subsequent calculations.

Graphs in Fig. 9 show uniform norms versus data ratio of the matrix vector multiplication test case. The largest norm of the one hundred random vectors is shown. FMM approximation of the domain matrix was used to obtain the results in the left graph while wavelet approximation was used for the right graph. Similarly, the *RMS* values are shown in Fig. 10. We observe that the data ratio, which corresponds to a certain norm, is lower in case of denser meshes. In other words, as the mesh density increases lower data ratios will yield the same accuracy.

Comparing the FMM and wavelet methods, we observe that the decrease of the uniform norm and *RMS* value with increasing data ratio is approximately exponential for the wavelet method and even faster for the FMM. For low data ratios ($\mathcal{D} < 0.2$), and for high data ratios ($\mathcal{D} > 0.8$) the wavelet method gives more accurate results than the FMM. For moderate data ratios, the FMM gives slightly better results.

Table 3
Poisson equations with analytical solutions. The geometry was a unit cube. The boundary conditions in all cases were $u(x=0)=0$, $u(x=1)=1$, $q(y=0,y=1,z=0,z=1)=0$.

	Equation	Analytical solution
(a)	$\nabla^2 u = 2$	$u_a = x^2$, $q_{a,x=0}=0$, $q_{a,x=1}=2$
(b)	$\nabla^2 u = 6x$	$u_a = x^3$, $q_{a,x=0}=0$, $q_{a,x=1}=3$
(c)	$\nabla^2 u = 12x^2$	$u_a = x^4$, $q_{a,x=0}=0$, $q_{a,x=1}=4$

Table 4
Solutions of Poisson equations in Table 3. Uniform norms for the full domain matrix $[B]$ solution against the analytical solution are presented.

Problem		17^3	25^3	33^3
(a)	$\|u-u_a\|_\infty$	6.2×10^{-6}	2.8×10^{-6}	1.6×10^{-6}
(a)	$\|q-q_a\|_\infty$	6.4×10^{-5}	4.0×10^{-5}	3.9×10^{-5}
(b)	$\|u-u_a\|_\infty$	2.4×10^{-5}	7.4×10^{-6}	3.1×10^{-6}
(b)	$\|q-q_a\|_\infty$	7.7×10^{-4}	3.4×10^{-4}	1.9×10^{-4}
(c)	$\|u-u_a\|_\infty$	9.0×10^{-5}	2.8×10^{-5}	1.2×10^{-5}
(c)	$\|q-q_a\|_\infty$	2.9×10^{-3}	1.3×10^{-3}	6.8×10^{-4}

6.1.2. Comparison of FMM and full matrices

The FMM sparse domain matrix approximation stores the matrix as a tree of admissible and inadmissible leaves. Inadmissible leaves store full matrices, while in admissible leaves, two lower order complex matrices of expansion terms are stored. If we multiply the two lower order matrices in admissible leaves together, we obtain an approximation of the full matrix. This technique was used to calculate back the full matrix out of the FMM sparse approximation. The difference between such full matrix and the original full matrix was measured in terms of the uniform norm and *RMS* value. The results are shown on a graph in Fig. 11. We observe similar behaviour as in the matrix times vector test. Higher mesh density enables lower data ratio for the same accuracy. For a given data ratio, the accuracy gets better as the mesh density increases.

6.1.3. Solving the Poisson equation

The third series of tests was preformed by solving the Poisson equation with known analytical solutions. The numerical solution was obtained with the use of the full domain matrix and by using

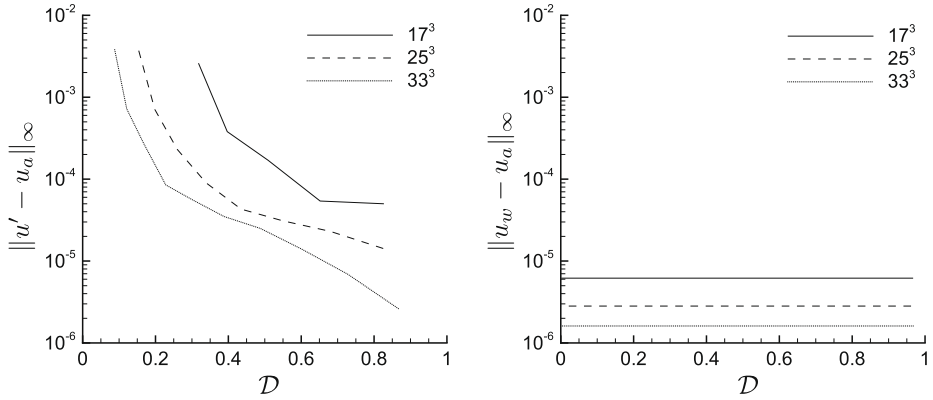

Fig. 12. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem (a) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

its FMM and wavelet approximations. The problem geometry was a unit cube. Table 3 summarizes the equations and boundary conditions. The equations were chosen in such a way, that the solution is a polynomial with an increasing leading exponent; it is 2 in equation (a) and 4 in equation (c). The function was known on two opposite sides of the cube, while flux was known on the other four. We solved the problem for function on four sides and for flux on two sides.

Similarly to the tests in previous sections, this problem was also solved on three meshes that still enable computation with the full domain matrix. Let u_a, q_a be the analytical solution and let u, q represent the solution obtained using the full matrix $[B]$. Uniform norms of the full matrix solution against the analytical solution are given in Table 4. As the mesh density increases we observe a decrease of the norms. Since the domain cells size of 17^3 mesh is exactly twice as large as at 33^3, we were able to use the Richardson extrapolation to estimate the order of our method. The order is the log of the uniform norm at 17^3 mesh over the uniform norm at 33^3 mesh divided by the log of 2. The average order obtained using values in Table 4 was 2.1.

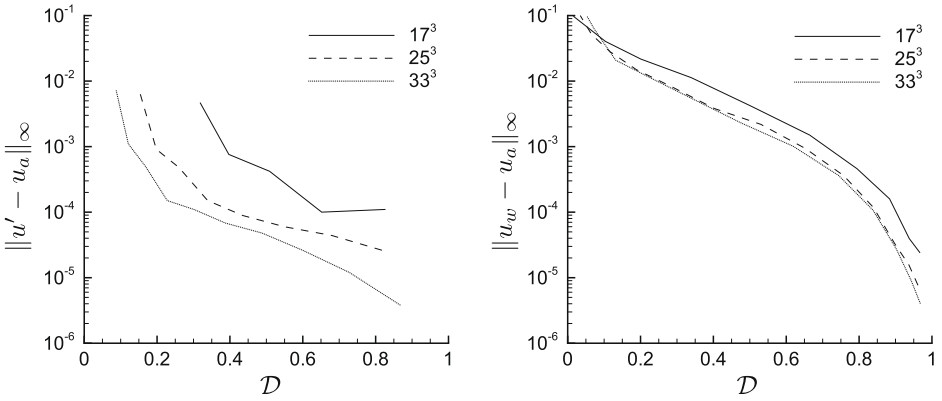

Fig. 13. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem (b) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

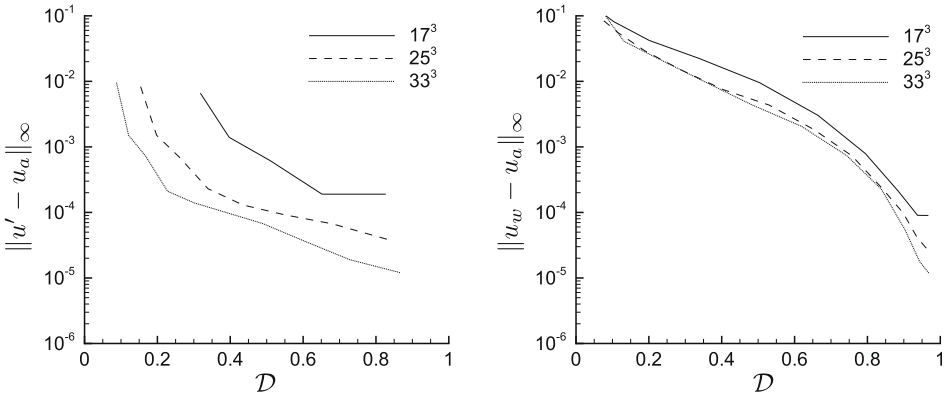

Fig. 14. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem (c) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

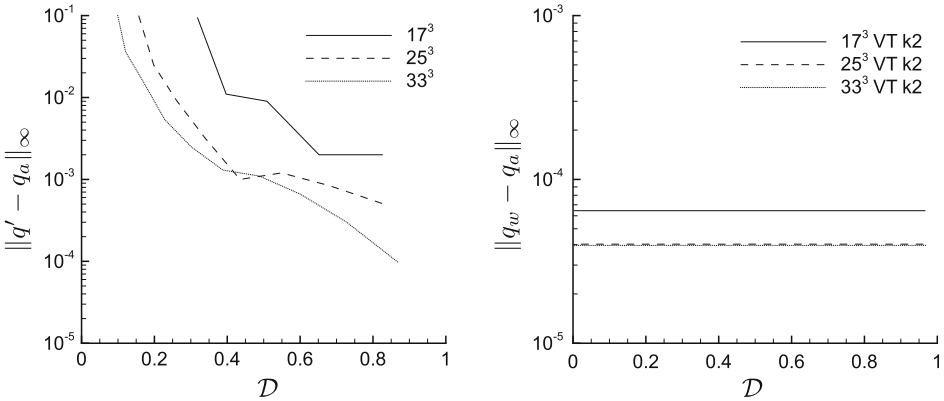

Fig. 15. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of flux solutions of problem (a) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

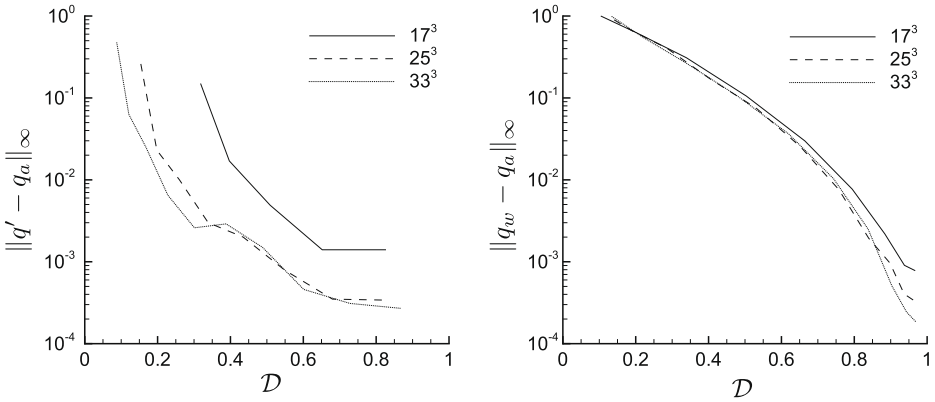

Fig. 16. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of flux solutions of problem (b) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

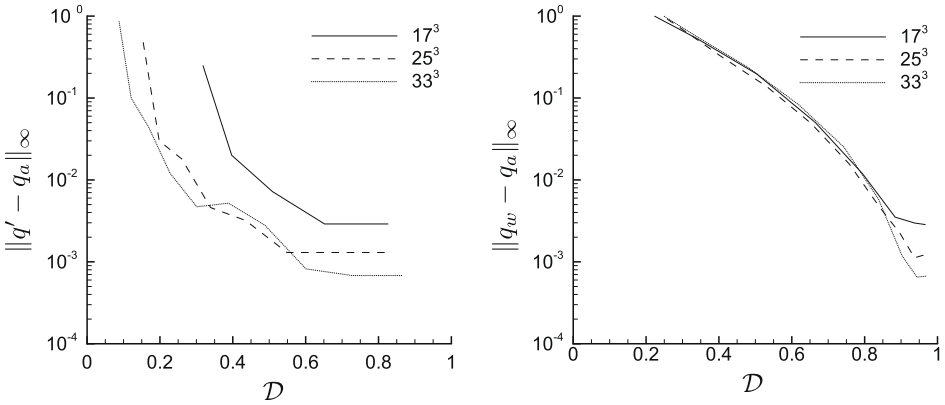

Fig. 17. The graphs present uniform norms versus the data ratio $\mathcal{D}$ of flux solutions of problem (c) in Table 3 using FMM matrices $[B']$ (left) and wavelet matrices $[B_w]$ (right). Results for three mesh densities are presented.

Let u', q' denote the solution obtained using the FMM sparse approximation of the domain matrix, $[B']$. The results obtained using the wavelet approximation of the domain matrix $[B_w]$ are denoted by u_w, q_w. The equations were solved for different numbers of expansion terms and different thresholding limits. Figs. 12–14 show uniform norms of the function solution of our equations using FMM and wavelet approximations of domain matrix for equations (a), (b) and (c), respectively. Norms of calculated fluxes are shown in the same manner in Figs. 15–17.

Comparing the graphs, we are again able to observe that a denser mesh requires a lower data ratio for the same accuracy of solution. Comparison of the FMM and wavelet method reveals better accuracy of the wavelet method for low and high data ratios, while FMM gives more accurate results for moderate data ratios. The first

Table 5
Velocity and vorticity fields that are the solution of the kinematics equation. The geometry is a unit cube. All fields except boundary values of ω_y are known.

	Velocity, $\vec{v}$	Vorticity, $\vec{\omega}$
(d)	$(y^2z, y, -z)$	$(0, y^3, -3y^2z)$
(e)	$(\sin(z), \sin(x)\sin(y))$	$(\cos(y), \cos(z), \cos(x))$
(f)	(e^z, e^x, e^y)	(e^y, e^z, e^x)

Table 6
Solutions of kinematics equations in Table 5. *RMS* values of full domain matrix solution against the analytical solution are presented.

Mesh	(d)	(e)	(f)
25^3	1.2×10^{-5}	2.1×10^{-4}	2.0×10^{-4}
33^3	5.1×10^{-6}	1.2×10^{-4}	1.1×10^{-4}

Table 7
Solutions of kinematics equations in Table 5 on the 25^3 mesh. *RMS* values of FMM domain matrix solution against the analytical solution are presented.

Terms	$\mathcal{D}$	(d)	(e)	(f)
4	0.250	1.9×10^{-1}	4.5×10^{-2}	6.9×10^{-2}
9	0.326	9.3×10^{-2}	1.7×10^{-2}	2.5×10^{-2}
16	0.433	2.2×10^{-2}	5.4×10^{-3}	8.2×10^{-3}
25	0.571	6.6×10^{-3}	1.5×10^{-3}	2.2×10^{-3}
36	0.739	2.2×10^{-3}	5.5×10^{-4}	8.9×10^{-4}
49	0.938	1.1×10^{-3}	3.1×10^{-4}	4.0×10^{-4}

Table 8
Solutions of kinematics equations in Table 5 on the 25^3 mesh. *RMS* values of the solution obtained by wavelet approximation of the domain matrix against the analytical solution are presented.

$\mathcal{D}$	(d)	(e)	(f)
0.063	8.9×10^{-1}	1.7×10^{-1}	2.8×10^{-1}
0.111	5.3×10^{-1}	1.0×10^{-1}	1.6×10^{-1}
0.158	3.6×10^{-1}	6.8×10^{-2}	1.1×10^{-1}
0.243	2.0×10^{-1}	3.8×10^{-2}	6.0×10^{-2}
0.304	1.4×10^{-1}	2.6×10^{-2}	4.1×10^{-2}
0.419	6.8×10^{-2}	1.3×10^{-2}	2.0×10^{-2}
0.629	1.6×10^{-2}	2.9×10^{-3}	4.6×10^{-3}
0.907	4.0×10^{-4}	2.2×10^{-4}	2.2×10^{-4}

equation (a) is an exception of this conclusions, since using the wavelet transform does not introduce any additional error and the problem is solved with the accuracy of the full matrix. This can be explained by the fact, that the right hand side of equation (a) is constant. The wavelet transform uses Haar wavelets, which require only one non-zero coefficient when transforming constant functions.

Comparing the rate of decrease of the norms with increasing data ratio, we observe, that they decrease approximately exponentially for the wavelet solutions, while they decrease even faster for the FMM solutions. This fact is especially prominent for the low data ratios, which are of practical importance.

6.2. The kinematics equation

Consider the three velocity and vorticity fields given in Table 5. They are all solenoidal and solve the kinematics equation. The fields were chosen so, that they are analytical and that they ensure variations are present in all three spatial dimensions. Such fields enable a thorough test of the algorithm and are the solution of the kinematics equation. However, they are not physical, since they do not satisfy the Navier–Stokes equation.

The geometry for three test cases was a unit cube. We chose the y component of vorticity as the unknown. The kinematics equation was solved for ω_y, with the right hand side evaluated by full domain matrices and by their data sparse approximations. Table 6 presents the *RMS* values of the solution obtained by the full domain matrices using the 25^3 and 33^3 meshes. The errors obtained, using data sparse approximations of domain matrices, are larger than the errors of the full domain matrix solution. They depend heavily on the number of terms used in the expansion and on the thresholding limit. Tables 7 and 9 present the *RMS* value obtained using the FMM data sparse approximation of domain matrices with 4 to 64 expansion terms. The *RMS* value obtained by the wavelet data sparse approximation with different thresholding limits are presented in Tables 8 and 10. All tables show an increase in accuracy when the data ratio is increased. The error decreases towards the error obtained by the full matrix solution.

The *RMS* values of problems (e) and (f) are shown graphically in Fig. 18. Several conclusions can readily be drawn. Firstly, using a denser mesh yields more accurate results and it enables better data ratio for the same error. Secondly, at moderate data ratios, the FMM method yields lower error than the wavelet method. At high data ratios ($\mathcal{D} > 0.8$) the wavelets yield better results. This is due to the fact that as the data ratio increases towards one, the wavelet data sparse approximation of a matrix converges towards the full matrix. Without any thresholding (at $\mathcal{D} = 1$) the wavelet method yields identical results to the full matrix. On the other hand, only an infinite number of terms in the FMM expansion would give identical results. At low data ratios ($\mathcal{D} < 0.2$) the wavelets also present better accuracy than the FMM.

Comparing the results between (e) and (f) test cases, we observe that the sparse matrix approximation *RMS* curves are approximately similar, even tough the full matrix solution yields more than an order of magnitude more accurate solution in the (e) test case. This means that the domain matrix approximation is the main source of error. The error of solution will be approximately equal regardless of the problem we are solving and regardless of the accuracy of the full domain matrix solution.

Table 9
Solutions of kinematics equations in Table 5 on the 33^3 mesh. *RMS* values of FMM domain matrix solution against the analytical solution are presented.

Terms	$\mathcal{D}$	(d)	(e)	(f)
4	0.135	2.4×10^{-1}	5.7×10^{-2}	8.8×10^{-2}
9	0.194	4.2×10^{-1}	3.6×10^{-2}	8.7×10^{-2}
16	0.275	4.5×10^{-2}	1.2×10^{-2}	1.8×10^{-2}
25	0.380	1.2×10^{-2}	2.7×10^{-3}	4.2×10^{-3}
36	0.510	3.2×10^{-3}	7.5×10^{-4}	1.2×10^{-3}
49	0.660	1.2×10^{-3}	3.6×10^{-4}	5.5×10^{-4}
64	0.835	9.7×10^{-4}	2.2×10^{-4}	3.1×10^{-4}

Table 10
Solutions of kinematics equations in Table 5 on the 25^3 mesh. *RMS* values of the solution obtained by wavelet approximation of the domain matrix against the analytical solution are presented.

$\mathcal{D}$	(d)	(e)	(f)
0.097	6.1×10^{-1}	1.3×10^{-1}	1.8×10^{-1}
0.138	4.2×10^{-1}	7.9×10^{-2}	1.2×10^{-1}
0.213	2.4×10^{-1}	4.4×10^{-2}	7.0×10^{-2}
0.267	1.6×10^{-1}	3.0×10^{-2}	4.8×10^{-2}
0.374	8.2×10^{-2}	1.5×10^{-2}	2.4×10^{-2}
0.580	1.9×10^{-2}	3.6×10^{-3}	5.8×10^{-3}
0.880	5.5×10^{-4}	1.7×10^{-4}	2.0×10^{-4}

7. Concluding remarks

The boundary-domain integral formulation of a non-homogenous Poisson type equation includes a domain integral. After discretization it yields a fully populated domain matrix. We presented the usage of the fast multipole method and the wavelet transform method for obtaining a sparse approximation of the domain

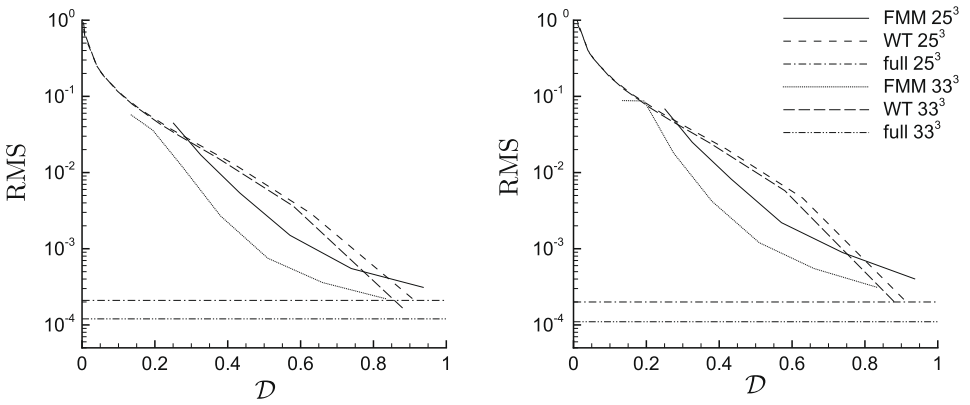

Fig. 18. The graphs present *RMS* values versus the data ratio $\mathcal{D}$ of solutions of problems (e) (left) and (f) (right) of Table 5. FMM and wavelet data sparse approximations were used. The full matrix errors are presented with horizontal lines. Results for two mesh densities are presented.

matrix. We reduced the storage requirements for the domain matrix from $\mathcal{O}(n_d \cdot n_b)$ to $\mathcal{O}(n_d)$, where n_d is the number of nodes in the domain and n_b is the number of nodes on the boundary.

The methods were tested on scalar Poisson equations and on velocity–vorticity vector kinematics equations in 3D. We found that the FMM yielded results of better accuracy for moderate data ratios ($0.2 < \mathcal{D} < 0.8$), while the wavelet method gives better results for high and low data ratios. The FMM does not require the calculation of the full matrix, whereas the wavelet method needs the full matrix in order to make the transform. Since the wavelet method is purely algebraic, it can be applied without modification on any matrix. The FMM is based on an expansion of the integral kernel, thus requiring modification before it can be applied on a different integral matrix. In this work we used spherical harmonic expansion of the Laplace fundamental solution and its gradient; however, other expansions may be used as well.

This work is continued by incorporating the developed FMM vector kinematics equation solver into a 3D BEM based fluid flow solver [22].

References

[1] J. Barnes, P. Hut, A hierarchical $O(N \log N)$ force calculation algorithm, Nature 324 (1986) 446–449.
[2] M. Bebendorf, S. Rjasanow, Adaptive low rank approximation of collocation matrices, Computing 70 (2003) 1–24.
[3] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms, Commun. Pure Appl. Math. 44 (1991) 141–183.
[4] H.F. Bucher, L.C. Wrobel, W.J. Mansur, C. Magluta, A novel approach to applying fast wavelet transforms in boundary element method, Elem. J. Bound. Elem. BETEQ 2001 (2) (2002) 187–195.
[5] W. Dahmen, H. Harbrecht, R. Schneider, Compression techniques for boundary integral equations – optimal complexity estimates, SIAM J. Numer. Anal. 43 (2006) 2251–2271.
[6] I. Daubechies, Orthonormal bases of compactly supported wavelets, Commun. Pure Appl. Math. 41 (1988) 909–996.
[7] M. Dehghan, D. Mirzaei, The dual reciprocity boundary element method (DRBEM) for two-dimensional sine-Gordon equation, Comput. Methods Appl. Mech. Engrg. 197 (2008) 476–486.
[8] J. Englund, A higher order scheme for two-dimensional quasi-static crack growth simulations, Comput. Methods Appl. Mech. Engrg. 196 (2007) 2527–2538.
[9] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 73 (1987) 325–348.
[10] W. Hackbusch, A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: Introduction to $\mathscr{H}$-matrices, Computing 62 (1999) 89–108.
[11] W. Hackbusch, B. Khoromskij, S. Sauter, On $\mathscr{H}^2$-matrices, Lect. Appl. Math. (2002) 9–29.
[12] W. Hackbusch, Z.P. Nowak, On the fast multiplication in the boundary element method by panel clustering, Numer. Math. 54 (1989) 463–491.
[13] H. Harbrecht, R. Schneider, Wavelet Galerkin schemes for boundary integral equations – implementation and quadrature, SIAM J. Sci. Comput. 27 (2006) 1347–1370.
[14] H.-Y. Hu, Z.-C. Li, Collocation methods for Poisson's equation, Comput. Methods Appl. Mech. Engrg. 195 (2006) 4139–4160.
[15] P. Partridge, C. Brebbia, L. Wrobel, The Dual Reciprocity Boundary Element Method, Computational Mechanics Publications, Southampton, UK, Boston, London, New York, 1992.
[16] V. Popov, H. Power, S.P. Walker, Numerical comparison between two possible multipole alternatives for the BEM solution of 3D elasticity problems based upon Taylor series expansions, Engrg. Anal. Bound. Elem. 27 (2003) 521–531.
[17] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes – The Art of Scientific Computing, second ed., Cambridge University Press, 1997.
[18] A. Rathsfeld, A wavelet algorithm for the boundary element solution of a geodetic boundary value problem, Comput. Methods Appl. Mech. Engrg. 157 (1998) 267–287.
[19] J. Ravnik, L. Škerget, M. Hriberšek, The wavelet transform for BEM computational fluid dynamics, Engrg. Anal. Bound. Elem. 28 (2004) 1303–1314.
[20] J. Ravnik, L. Škerget, M. Hriberšek, 2D velocity vorticity based LES for the solution of natural convection in a differentially heated enclosure by wavelet transform based BEM and FEM, Engrg. Anal. Bound. Elem. 30 (2006) 671–686.
[21] J. Ravnik, L. Škerget, M. Hriberšek, Z. Žunič, Numerical simulation of dilute particle laden flows by wavelet BEM–FEM, Comput. Methods Appl. Mech. Engrg. 197 (6–8) (2008) 789–805.
[22] J. Ravnik, L. Škerget, Z. Žunič, Fast single domain–subdomain BEM algorithm for 3D incompressible fluid flow and heat transfer, Int. J. Numer. Methods Engrg., doi:10.1002/nme.2467.
[23] J. Ravnik, L. Škerget, Z. Žunič, Velocity–vorticity formulation for 3D natural convection in an inclined enclosure by BEM, Int. J. Heat Mass Transfer 51 (2008) 4517–4527.
[24] C. Schwab, R.A. Todor, Karhunen–Loève approximation of random fields by generalized fast multipole methods, J. Comput. Phys. 217 (2006) 100–122.
[25] T. von Petersdorff, C. Schwab, Multiscale Wavelet Methods for PDEs, Chap. Fully Discrete Multiscale Galerkin BEM, Academic Press, 1997.
[26] L.C. Wrobel, The Boundary Element Method, John Wiley & Sons Ltd., 2002.
[27] Z. Žunič, M. Hriberšek, L. Škerget, J. Ravnik, 3-D boundary element-finite element method for velocity–vorticity formulation of the Navier–Stokes equations, Engrg. Anal. Bound. Elem. 31 (2007) 259–266.