Towards a fast single domain – subdomain BEM algorithm for 3D incompressible fluid flow

J. Ravnik, L. Škerget & Z. Žunič
University of Maribor, Faculty of Mechanical Engineering, Smetanova ulica 17, SI-2000 Maribor, Slovenia

Abstract

Work towards acceleration and computer memory reduction of an algorithm for the simulation of laminar viscous flows by solving of 3D velocity-vorticity formulation of the Navier–Stokes equations is presented. The algorithm employs a combination of a subdomain boundary element method (BEM) and fast single domain BEM. The single domain BEM on a Poisson type equation is employed to calculate boundary vorticity values. After discretization, the single domain BEM algorithm yields a fully populated system of linear equations. The non-homogenous part of the Poisson equation yields a fully populated matrix of domain integrals. In order to lower the computational demand, a fast multipole expansion algorithm is used on the domain matrix. The fundamental solution is expanded in terms of spherical harmonics. The computational domain and its boundary are recursively cut up forming a tree of clusters of boundary elements and domain cells. Compression is achieved in parts of the matrix, which correspond to boundary-domain cluster pairs that are admissible for expansion. Significant reduction of the complexity is achieved. The paper presents results of initial testing of the FMM algorithm.
Keywords: fast multipole method, boundary element method, Poisson equation.

1 Introduction

Our research group works on the application of the Boundary Element Method to fluid flow problems. Recently Ravnik *et al.* [1] developed a 3D subdomain - single domain BEM numerical scheme for solving incompressible velocity-vorticity formulation of Navier–Stokes equations. A crucial part of this algorithm is the calculation of boundary vorticity values. They are obtained by solving a Poisson type partial differential equation using single domain BEM. The domain integral, which arises from the non-homogenous part of the Poisson equation, requires the

WIT Transactions on Modelling and Simulation, Vol 47, © 2008 WIT Press
www.witpress.com, ISSN 1743-355X (on-line)
doi:10.2495/BE080011

discretization of the domain and the calculation of a large number of domain matrix elements, thus limiting the maximal mesh size. In this work, we are presenting the application of the Fast Multipole Method (FMM) on the Poisson type equation for a data sparse representation of the non-homogenous part.

The origins of FMM can be found in a fast algorithm for particle simulations developed by Greengard and Rokhlin [2]. The algorithm decreases the complexity of interacting particles from (n^2) to linear complexity. The method was since then used by many authors for a wide variety of problems using different expansion strategies. Recently Bui *et al.* [3] combined FMM with the Fourier transform to study multiple bubbles dynamics. Gumerov and Duraiswami [4] applied the FMM for the biharmonic equation in three dimensions. The boundary integral Laplace equation was accelerated with FMM by Popov and Power [5] and Popov *et al.* [6].

2 BEM for Poisson equation

The Poisson equation is a partial differential equation including a diffusion operator and a non-homogenous right-hand side,

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

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 of such problems 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 (Wrobel [7]):

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

where $\vec{\xi}$ is the collocation point on the boundary, $\vec{n}$ is the unit normal 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 domain Φ, boundary φ and boundary flux ϕ shape functions. In this paper domain cells are hexahedra and boundary elements are parallelepipeds. In each domain cell 27 nodes are used to achieve quadratic interpolation of function. Nine continuous nodes are used in boundary elements for quadratic interpolation of function and four discontinuous nodes are used to interpolate fluxes linearly. Considering these approximations in equation (2) we have:

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

The integrals are traditionally named as

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

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

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

Since all integral kernels are non zero, the matrices of equation (5) are fully populated. Considering the boundary conditions, the system of equations (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.

The size of the boundary matrices $[H]$ and $[G]$ scale as number of boundary nodes squared. This are small compared to the domain matrix $[B]$, which scales as the number of boundary nodes times the number of domain nodes. Considering a cube with N nodes per edge, we can estimate the number of domain nodes as $n_d = N^3$, and the number of boundary nodes as $n_b \approx N^2$, thus the complexity of the domain integral matrix is $\mathcal{O}(N^5)$. Since, clearly, the domain contribution takes up most of the CPU time and storage space, this paper presents an application fast multipole method to obtain a data sparse representation of the domain matrix in such a way that the number of non-zero elements scales linearly with the number of nodes $\mathcal{O}(n_d)$.

3 Fast multipole method for the domain matrix

Let us consider the domain integral in equation (4). Since for each collocation point $\vec{\xi}$ integrals for all domain cells c must be evaluated, we are obviously faced by a problem of quadratic complexity. This is analogous to the problem of interaction of n particles (Barnes and Hut [8]), where the origins of the FMM can be found.

The method is based on the fact that it is possible to expand integral kernel, i.e. the fundamental solution, into a series and by doing so, separate the variables – the collocation point $\vec{\xi}$ and the domain integration point $\vec{r}$. In this work we will use spherical harmonics to expand the integral kernel into a series. Other expansions are also possible, such as Taylor series, Lagrangian polynomials, etc. In polar coordinate system $\vec{r} = (r, \varphi_r, \theta_r)$ and $\vec{\xi} = (\xi, \varphi_\xi, \theta_\xi)$ the integral kernel may be

expanded into such series

$$\frac{1}{4\pi|\vec{r}-\vec{\xi}|}=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{(-1)^m}{2l+1}\underbrace{\frac{1}{\xi^{l+1}}Y_l^{-m}(\theta_\xi,\varphi_\xi)}_{f(\vec{\xi})}\underbrace{r^l Y_l^m(\theta_r,\varphi_r)}_{g(\vec{r})}, \tag{6}$$

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

$$Y_l^m(\theta,\varphi)=\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)e^{im\varphi}. \tag{7}$$

The associated Legendre polynomials are evaluated using recurrence relations as described in Press *et al.* [9]. The domain integral of equation (4) may now be written as

$$\beta_i^{c,\vec{\xi}}=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}F_l^m(\vec{\xi})\int_{\Omega_c}G_l^{m,i}(\vec{r})d\Omega. \tag{8}$$

We are now able to calculate each entry in the domain matrix with the above sum. The advantage of this becomes obvious when we consider a cluster of n_r nearby collocation nodes and a cluster of n_c nearby domain cells. These correspond to a $n_r \times n_c$ matrix block. Since the variables are separated, it is possible to evaluate two smaller matrix blocks ($n_r \times n_{\text{exp}}$) and ($n_{\text{exp}} \times n_c$), where n_{exp} is the number of expansion terms. Multiplication of the two smaller matrix block 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. In order for this technique to yield a data sparse representation of the full matrix block, the number of terms in the two smaller matrices should be smaller than the number of terms in the full matrix block, i.e.

$$2(n_r n_{\text{exp}}+n_c n_{\text{exp}})<n_r n_c; \tag{9}$$

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, 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 kernel is singular. Such cluster pairs are called inadmissible and the corresponding matrix block is evaluated in full, without compression.

The collocation points on the boundary must be divided into clusters and coupled with clusters of domain cells. We constructed a tree of clusters of collocation points and a tree of clusters of domain cells. The trees were constructed in a recursive hierarchical manner. The problem domain is enclosed by a parallelepiped. All collocation points and all of the domain cells are within this root parallelepiped.

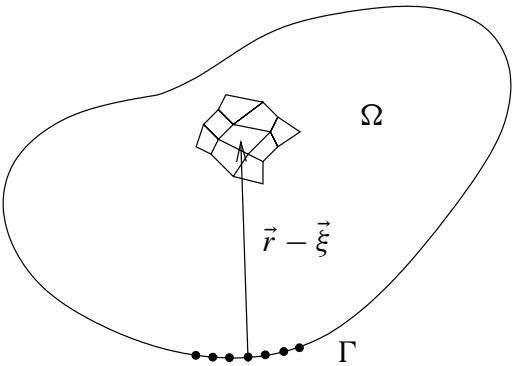

Figure 1: A problem domain shown with a cluster of collocation points $\vec{\xi}$ and a cluster of domain cells.

They make up root clusters of both trees. The parallelepiped is cut in half by a plane splitting 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.

With both cluster trees is place, the next step is to pair them, so a list of admissible and inadmissible matrix blocks can be formed. The admissible and inadmissible matrix blocks correspond to leaves on the tree of pairs of clusters. 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 forming branches on the tree of pairs of clusters. For each pair a decision is taken weather it is possible to do compression (i.e. the cluster pair is admissible) or lower level pairs must be considered. If admissibility is not reached until the last level, such pairs are inadmissible and will be calculated in full without compression.

The admissibility criterion is devised as follows. We are considering a cluster of collocation points and a cluster of domain cells. Firstly we try to find a origin of the coordinate system in nodes corresponding to domain cells in the cluster. We choose such origin that the ratio r/ξ is minimal for all pairs of collocation nodes and domain cells. If the minimal ratio is above one, series expansion is not possible for this pair of clusters, thus this pair is not admissible. Secondly, when the r/ξ ratio is below one, we calculate the number of expansion terms needed to have the error of calculation of the integral kernel less than a user prescribed criteria ϵ. If the number of expansion terms low enough, so that compression is achieved (equation (9)), this cluster pair is admissible. At this point the tree of pairs of clusters gets a leaf - no further branching is necessary.

Considering a cubic mesh of 16^3 cells with 33^3 nodes the matrix structure showing admissible and inadmissible blocks is shown on Figure 2.

The described FMM based algorithm was implemented into the BEM Poisson solver code. It is capable of constructing a data sparse approximation $[B']$ of the full domain matrix $[B]$ and use it to evaluate the right-hand side of the system of equations.

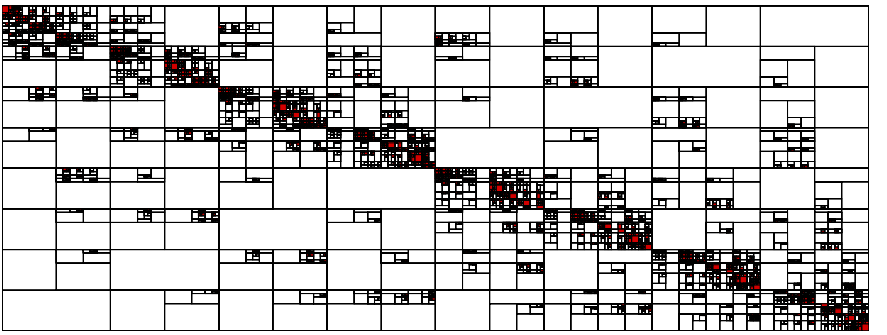

Figure 2: Matrix structure of a cubic mesh (16^3 cells, 33^3 nodes). Filled areas show inadmissible matrix blocks, white areas are admissible leafs obtained using an admissibility criteria of $\epsilon = 10^{-5}$. The corresponding tree of pairs of clusters had 19 levels.

4 Numerical tests

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 in Figure 3 where the storage requirements of the full matrix $[B]$ and the FMM compressed matrix

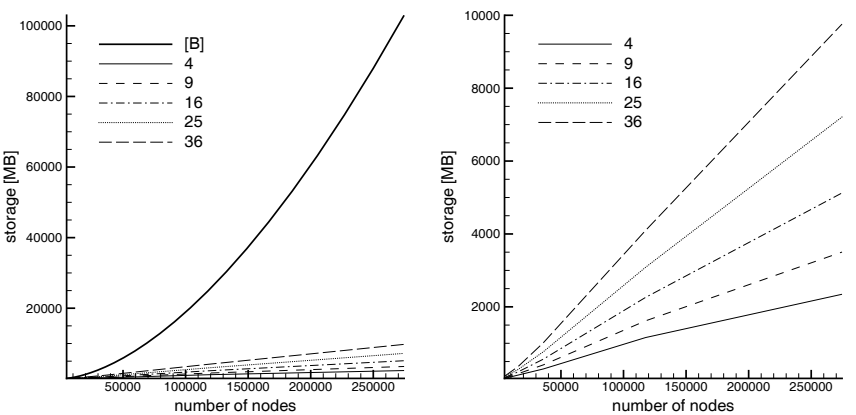

Figure 3: The graphs show a comparison of the memory required to store a full matrix $[B]$ (thick solid line) and FMM compressed matrices $[B']$ with different number of expansion terms. We observe the linear dependence of storage requirements for $[B']$ on the number of nodes regardless of the number of expansion terms.

$[B']$ are compared for different meshes. 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 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 linear dependence of $\mathcal{O}(n_d)$.

The accuracy of carrying out the calculation using the FMM domain matrix was analysed by solving the Poisson equations with known analytical solutions. The numerical solution was obtained with the use of the full matrix and by using the FMM data sparse domain matrix $[B']$. The problem geometry was a unit cube. Table 1 summarizes the equations and boundary conditions.

The uniform norms of solutions obtained using the full domain matrix $[B]$ are presented in Table 2. The same problems were solved using FMM $[B']$ matrices with different number of expansion terms. Let us define the data ratio $\mathcal{D}$ as the amount of data required to store the $[B']$ matrix divided by the amount of data in the $[B]$ matrix. A larger number of expansion terms results in larger $[B']$ matrix and thus a larger data ratio. Figures 4, 5 and 6 present uniforms norms for problems *a*), *b*) and *c*) respectively. We observe the norms increasing with decreasing data ratio. Comparing the results of the three problems, we see that the accuracy is best with problem *a*) and worst with problem *c*). This can be explained by the fact that the domain vector is a steeper function in problem *c*) than in problem *a*). At the same time we observe that the accuracy of solution converges with increasing data ratio to the accuracy full domain matrix solution (Table 2).

Table 1: 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 2: Solutions of Poisson equations in Table 1. Uniform norms for the full domain matrix $[B]$ solution against the analytical solution are presented.

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

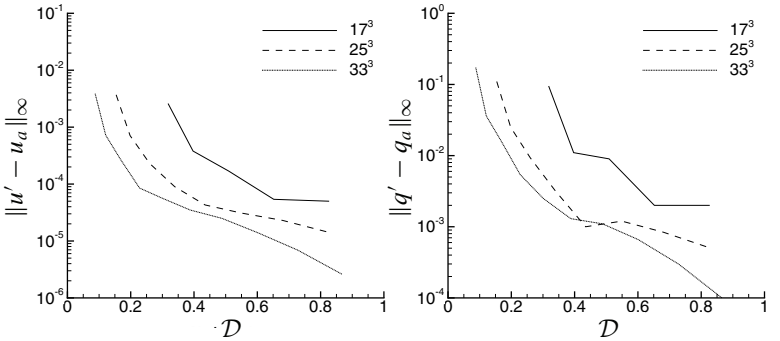

Figure 4: The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem *a*) in Table 1 using FMM matrices $[B']$.

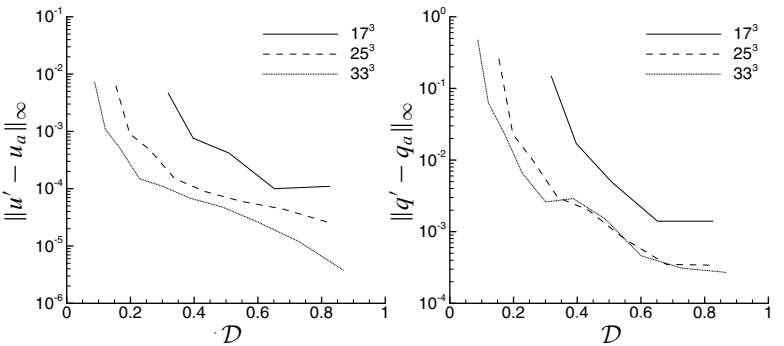

Figure 5: The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem *b*) in Table 1 using FMM matrices $[B']$.

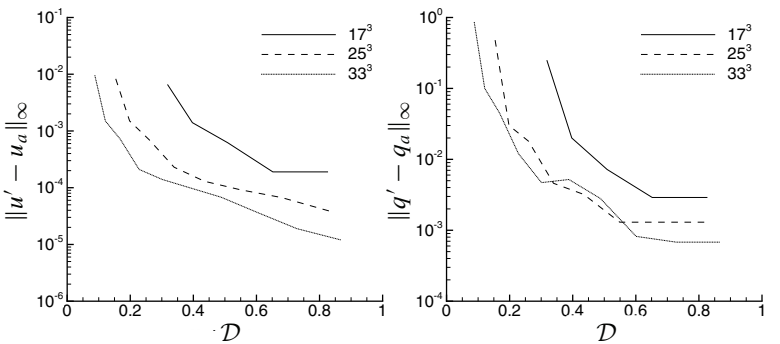

Figure 6: The graphs present uniform norms versus the data ratio $\mathcal{D}$ of solutions of problem *c*) in Table 1 using FMM matrices $[B']$.

5 Conclusions

We presented a Fast Multipole Method for data sparse representation and fast evaluation of the domain matrix of the BEM integral Poisson equation. The method uses a spherical harmonic series expansion of the integral kernel. The complexity of the calculation was decreased from $\mathcal{O}(n_d \cdot n_b)$ to linear dependence of $\mathcal{O}(n_d)$, where n_d is the number of nodes in the domain and n_b is the number of nodes on the boundary.

The method was tested on Poisson equations with known analytical solutions. It was shown, that for a given solution accuracy, the method enables better data ratios on denser meshes.

The main advantages of the method are the decreased storage requirements and the acceleration of matrix vector multiplication. Since only a small portion of the matrix must be evaluated, we observed a considerable acceleration of the time needed to set up the matrix as well.

In the near future, the method will be used to accelerate the solution of the kinematics equation, which is also of the Poisson type. The solution of this equation is an important part of the algorithm that solves fluid flow the velocity-vorticity formulation of incompressible Navier–Stokes equations.

References

[1] Ravnik, J., Škerget, L. & Žunič, Z., Combined single domain and subdomain BEM for 3D laminar viscous flow. *Eng Anal Bound Elem*, **submitted**, pp. xx–xx, 2008.

[2] Greengard, L. & Rokhlin, V., A fast algorithm for particle simulations. *J Comput Phys*, **73**, pp. 325–348, 1987.

[3] Bui, T.T., Ong, E.T., Khoo, B.C., Klaseboer, E. & Hung, K.C., A fast algorithm for modeling multiple bubbles dynamics. *J Comput Phys*, **216**, pp.430–453, 2006.

[4] Gumerov, N.A. & Duraiswami, R., Fast multipole method for the biharmonic equation in three dimensions. *J Comput Phys*, **215**, pp. 363–383, 2006.

[5] Popov, V. & Power, H., An O(N) Taylor series multipole boundary element method for three-dimensional elasticity problems. *Eng Anal Boud Elem*, **25**, pp. 7–18, 2001.

[6] Popov, V., Power, H. & Walker, S.P., Numerical comparison between two possible multipole alternatives for the BEM solution of 3D elasticity problems based upon Taylor series expansions. *Eng Anal Boun Elem*, **27**, pp. 521–531, 2003.

[7] Wrobel, L.C., *The Boundary Element Method*. John Willey & Sons Ltd, 2002.

[8] Barnes, J. & Hut, P., A hierarchical O(N log N) force calculation algorithm. *Nature*, **324**, pp. 446–449, 1986.

[9] Press, W.H., Teukolsky, S.A., Vetterling, W.T. & Flannery, B.P., *Numerical Recipes - The Art of Scientific computing, Second Edition*. Cambridge University Press, 1997.