INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2017; **00**:1–26
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme

Accelerating isogeometric boundary element analysis for three-dimensional elastostatics problems through black-box fast multipole method with proper generalized decomposition

S. Li[1], J. Trevelyan[2*†], W. Zhang[1], D. Wang[1]

[1] *College of Aeronautics and Astronautics, National University of Defense Technology, 109 Deya Road, Changsha 410073, China*
[2] *Department of Engineering, Durham University, South Road, Durham DH1 3LE, UK*

SUMMARY

The isogeometric approach to computational engineering analysis makes use of Non-Uniform Rational B-splines (NURBS) to discretise both the geometry and the analysis field variables, giving a higher fidelity geometric description and leading to improved convergence properties of the solution over conventional piecewise polynomial descriptions. Because of its boundary-only modelling, with no requirement for a volumetric NURBS geometric definition, the boundary element method is an ideal choice for isogeometric analysis of solids in 3-D. An isogeometric boundary element analysis (IGABEM) algorithm is presented for the solution of such problems in elasticity, and is accelerated using the black-box Fast Multipole Method (bbFMM). The bbFMM scheme is of $O(n)$ complexity, giving a general kernel-independent separation that can be easily integrated into existing, conventional IGABEM codes with little modification. In the bbFMM scheme, an important process of obtaining a low rank approximation of M2L operators has been hitherto based on Singular Value Decomposition (SVD), which can be very time consuming for large 3-D problems, and this motivates the present work. We introduce the Proper Generalized Decomposition (PGD) method as an alternative approach, and this is demonstrated to enhance efficiency in comparison with schemes that rely on the SVD. In the worst case a factor of approximately 2 performance gain is achieved. Numerical examples show the performance gains that are achievable in comparison to standard IGABEM solutions, and demonstrate that solution accuracy is not affected. The results illustrate the potential of this numerical technique for solving arbitrary large scale elastostatics problems directly from CAD models. Copyright © 2017 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Isogeometric analysis; NURBS; Boundary element method; Black-box fast multipole method; Proper generalized decomposition

1. INTRODUCTION

In the traditional structural design process, mesh generation is an essential task in applying numerical stress analysis tools to geometric models built within a Computer Aided Design (CAD) environment. This can be a very time-consuming process that dominates the whole design lifecycle. To alleviate this burden, the idea of Isogeometric Analysis (IGA) [1] has received much attention in recent years as it offers precise and efficient geometric modelling, refinement without re-meshing the CAD model and control over the smoothness of the basis functions. This provides a uniform representation for the design and analysis models, significantly reducing the overall analysis time including the creation and refinement of the analysis model. Rapidly, the concept has expanded

*Correspondence to: J. Trevelyan, Department of Engineering, Durham University, South Road, Durham DH1 3LE, UK.
†E-mail: jon.trevelyan@durham.ac.uk

from elasticity [1, 2] to various applications such as fluid dynamics [3, 4], electromagnetics [5], vibration [6] and acoustics [7–9]. Early work on IGA was initially based on NURBS [1, 10], and then extended to T-splines [11] and PHT-splines [12–14].

The Boundary Element Method (BEM) [15] is a domain discretisation technique for solving partial differential equations, which takes advantage of boundary integral equations to decrease the dimensionality. The fact that both CAD and BEM require only a boundary representation suggests that there is much more scope for further development in linking BEM to CAD geometries than the Finite Element Method (FEM). BEM analysis of 3-D solids proceeds directly from a boundary representation, the most common geometric representation in CAD. Conversely, an isogeometric finite element analysis of a 3-D solid would require the user first to generate a model comprised of NURBS volumes, which is far from straightforward. This natural connection motivated many researchers to consider some early attempts to include CAD representations in a BEM framework. Arnold [16, 17] proposed a spline collocation method which is used to develop convergence estimates for BEM in two and three dimensions. A BEM formulation based on cubic splines [18, 19] was developed to solve groundwater flow problems and the Laplace equation. Turco [20] presented an approach by which the elasticity problem could be solved by B-spline elements. Rivas [21] was the first to put forward a combination of BEM with non-uniform rational B-splines (NURBS) in the context of the method of moments.

Importantly, IGA offers a considerable potential advantage over the conventional BEM in that it can circumvent the meshing procedure and eliminate geometry representation error. The idea of a true integration between CAD and analysis was explored in various areas including elastostatics [22–24], shape optimisation [25, 26], acoustics [7, 8, 27] and fracture mechanics [28]. However, a well-known drawback of both BEM and IGABEM remains, i.e. the dense (and, for the collocation form of the BEM, nonsymmetric) matrix for which $O(n^2)$ operations are needed to compute the matrix coefficients and another $O(n^3)$ operations to solve the system using direct solvers (n being the number of degrees of freedom (DOF)). We note that symmetric Galerkin formulations [29] are more amenable to mathematical analysis and optimal preconditioner strategies (i.e. operator-preconditioning [30]) are available. This presents great challenges in computational complexity for 3-D problems if they have a large number of DOF. Hence, for dealing with large scale engineering problems, some matrix compression techniques should be used to reduce the complexity to $O(n \log n)$ or, better still, $O(n)$. Some popular techniques have been successfully applied in order to accelerate BEM schemes, including the Fast Multipole Method (FMM) [31–33], Adaptive Cross Approximation (ACA) method [34–36] and pre-corrected Fast Fourier Transformation (pFFT) method [37–39]. Similar to most acceleration schemes, these methods approximate the far-field kernel interactions through efficient hierarchical data structures, which allow for use of matrix-vector multiplication with almost linear complexity. The IGABEM or BEM can be accelerated with the FMM, ACA, or pFFT to solve large scale problems on a personal computer. From another point of view, these methods perform the matrix compression in a purely algebraic manner, or based on an analytical expansion. In order to obtain the kernel with numerical approximation [9, 40–42], the black-box FMM [43] was developed by using Chebyshev interpolation to provide a more general, kernel-independent implementation for the matrix compression with a fast $O(n)$ algorithm. The current paper is based on this technique.

The current work contains two key elements of novelty:

1. In both the conventional and black-box FMM variants, the construction of the M2L operators is the most numerically intensive stage. Some methods have been proposed to ameliorate the cost of this stage, and have typically been constructed through a low rank operation calculated by wavelet decomposition [44, 45], Singular Value Decomposition (SVD) [41, 46], QR decomposition [47], "skeletonization" technique [48, 49], QR factorization with Adaptive Cross Approximation (ACA) [50, 51] and subsampling strategy [52]. As a canonical method to extract the optimal approximation basis, the computational cost of SVD is $O(mn^2)$ for an $m \times n$ matrix, and is a significant part of the whole bbFMM calculation. The Proper Generalized Decomposition (PGD) method [53, 54] provides a quasi-optimal basis based on an iterative procedure with a low computational effort ($O(2mn)$), rather than directly

DOI: 10.1002/nme

solving eigenfunctions as in SVD. In this paper, we introduce PGD into the M2L operators, which alleviates the expense of pre-computing the low rank decomposition and computing the matrix-vector products while maintaining a desired accuracy. Although some of the alternative acceleration techniques listed above have been shown to outperform SVD, our comparison in this paper is limited to SVD and PGD since our research focuses on problems of elastostatics, to which the above techniques have not yet been applied.

2. From the implementation perspective, our paper provides all the required details for solving 3-D elastostatics problems based on bbFMM and IGABEM which have not reported before. Some special attention will be given to illustrate unique features of this kind of problem. The PGD acceleration of bbFMM, which is the main novelty in this paper, can be applied in either a conventional BEM or an IGABEM context, but IGABEM is chosen for its well-known benefits in accuracy of its geometric description and its good convergence properties. Thus, with a view to practical application in industry, an accelerated IGABEM is more appealing and promising than the conventional BEM.

The paper is organised as follows: a brief introduction is given to the NURBS-based IGABEM discretisation procedure and the corresponding linear system of equations; the bbFMM algorithm combined with PGD is described and, finally, numerical examples are given to verify the scheme and compare the time and memory saving with the conventional IGABEM for the solution of 3-D elastostatics problems.

2. IGABEM DISCRETISATION

2.1. Boundary integral equation

For a 3-D linear elastic problem, the structure occupies a continuous physical domain, $\Omega \subset \mathbb{R}^3$, with boundary Γ. The boundary integral equation (BIE), in the absence of body forces, can be written as follows:

$$\mathbf{C}(\mathbf{s})\mathbf{u}(\mathbf{s}) + \int\!\!\!\!\!\!-_{\Gamma} \mathbf{T}(\mathbf{s},\mathbf{x})\mathbf{u}(\mathbf{x})\mathrm{d}\Gamma(\mathbf{x}) = \int_{\Gamma} \mathbf{U}(\mathbf{s},\mathbf{x})\mathbf{t}(\mathbf{x})\mathrm{d}\Gamma(\mathbf{x}), \tag{1}$$

$$u_i = \bar{u}_i \quad \text{on } \Gamma_{\bar{u}_i} \subset \Gamma, \tag{2}$$

$$t_i = \bar{t}_i \quad \text{on } \Gamma_{\bar{t}_i} \subset \Gamma, \tag{3}$$

where $\mathbf{s} \in \Gamma$ denotes the source point and $\mathbf{x} \in \Gamma$ the field point, $\mathbf{u} \in \mathbb{R}^3$ the displacement field, $\mathbf{t} \in \mathbb{R}^3$ the traction field, $\mathbf{U}(\mathbf{s},\mathbf{x}) = [U_{ij}]$ the displacement fundamental solutions kernel, $\mathbf{T}(\mathbf{s},\mathbf{x}) = [T_{ij}]$ the traction fundamental solutions kernel, $\mathbf{C}(\mathbf{s}) = [C_{ij}]$ the jump term, $\bar{u}_i$ and $\bar{t}_i$ the prescribed displacements and tractions, $\Gamma_{\bar{u}_i}$ and $\Gamma_{\bar{t}_i}$ the parts of Γ over which displacements and tractions are prescribed in each specific direction with $\Gamma_{u_i} \cup \Gamma_{t_j} = \Gamma$, $\Gamma_{u_i} \cap \Gamma_{t_j} = \emptyset$, $i \neq j$, i and j the indices running from 1 to 3 in three dimensions to denote the x-,y- and z-direction and $\int\!\!\!\!\!\!-$ denotes an integral taken in the Cauchy Principal Value (CPV) sense. The integrals in Eq. (1) contain a weak singularity and a strong singularity, and their evaluation will be discussed in the following sections.

The displacement and traction fundamental solutions are given as:

$$U_{ij}(\mathbf{s},\mathbf{x}) = \frac{1}{16\pi\mu(1-\nu)r}[(3-4\nu)\delta_{ij} + r_{,i}r_{,j}], \tag{4}$$

$$T_{ij}(\mathbf{s},\mathbf{x}) = -\frac{1}{8\pi(1-\nu)r^2}\left\{\frac{\partial r}{\partial \mathbf{n}}[(1-2\nu)\delta_{ij} + 3r_{,i}r_{,j}] + (1-2\nu)(r_{,j}n_i - r_{,i}n_j)\right\}, \tag{5}$$

where $r = r(\mathbf{s},\mathbf{x}) = \|\mathbf{s}-\mathbf{x}\|$ is the distance between source point and field point, n_i the ith component of unit outward normal vector $\mathbf{n}$, $r_{,i} = \frac{\partial r}{\partial x_i}$, μ the shear modulus and ν the Poisson's ratio.

DOI: 10.1002/nme

2.2. *B-splines and NURBS basis*

This section provides a brief overview of B-splines and Non-Uniform Rational B-splines (NURBS). These are standard bases for geometry representation, and the interested reader is directed to [10] for a full description. B-splines are geometric entities that are constructed from a group of piecewise polynomials which are defined by a knot vector:

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}, \quad \xi_a \in \mathbb{R}, \tag{6}$$

which is a set of non-decreasing real numbers in the parametric space. Here, a denotes the knot index, p the curve degree, and n the number of basis functions or control points. Each real number ξ_a is called a knot. The number of knots in a valid knot vector is always $n + p + 1$. The half open interval $[\xi_i, \xi_{i+1})$ is called a knot span. The basis functions $N_{a,p}$ may be defined using the Cox-de Boor recursion formula [55, 56]; starting with $p = 0$:

$$N_{a,0}(\xi) = \begin{cases} 1 & \text{if } \xi_a \leqslant \xi < \xi_{a+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

and for $p = 1, 2, 3, \ldots$:

$$N_{a,p}(\xi) = \frac{\xi - \xi_a}{\xi_{a+p} - \xi_a} N_{a,p-1}(\xi) + \frac{\xi_{a+p+1} - \xi}{\xi_{a+p+1} - \xi_{a+1}} N_{a+1,p-1}(\xi). \tag{8}$$

The B-spline geometry is found through a mapping from the parametric space, ξ, to physical space (x, y, z) through a linear combination of the basis functions $N_{a,p}$, and the corresponding coefficients may be considered to be the geometric coordinates of a set of control points located in physical space. The B-spline curve can thus be expressed as:

$$\mathbf{x}(\xi) = \sum_{a=1}^{n_a} N_{a,p}(\xi) \mathbf{P}_a, \tag{9}$$

where $\mathbf{x}(\xi)$ is the location of the physical curve, ξ the spatial coordinate in parametric space, $N_{a,p}$ the B-spline basis function of order p and $\mathbf{P}_a$ the control point related to each basis function.

Multivariate basis functions are defined by tensor products of univariate basis functions of parametric direction, given by:

$$N_a(\boldsymbol{\xi}|\boldsymbol{\Xi}_a) \equiv \prod_{i=1}^{d_p} N_a^i(\xi_a^i|\Xi_a^i), \tag{10}$$

where i denotes the direction index and d_p the number of dimensions.

NURBS are developed from B-splines and can offer significant advantages due to their ability to represent a wide variety of geometric entities. Importantly for engineering applications, they have the ability to describe circular arcs and other conic sections exactly, whereas the traditional piecewise polynomial description cannot. The expression defining a NURBS curve is similar to that with B-splines,

$$\mathbf{x}(\xi) = \sum_{a=1}^{n_a} R_{a,p}(\xi) \mathbf{P}_a, \tag{11}$$

where $\mathbf{P}_a$ is the set of control point coordinates, and $R_{a,p}$ the NURBS basis function which is given by:

$$R_{a,p}(\xi) = \frac{w_a N_{a,p}(\xi)}{W(\xi)}, \tag{12}$$

DOI: 10.1002/nme

with

$$W(\xi) = \sum_{a=1}^{n_a} w_a N_{a,p}(\xi), \tag{13}$$

where a denotes the control points index, p the curve order, n_a the number of control points, $N_{a,p}$ the B-spline basis function and w_a a weight associated to each basis function or control point. The weight can influence the distance between the associated control point and the NURBS geometry; a larger weight w_a will cause the curve to be drawn closer to the control point $\mathbf{P}_a$. If all of the weights are equal to 1, the NURBS curve will degenerate into a B-spline curve.

2.3. IGABEM implementation

The implementation of IGABEM is similar to the conventional BEM with the concept of isoparametric elements. We use NURBS to discretise both the solution variables (traction and displacement) and the geometry using the same shape functions. For practical application, if the geometry is sufficiently simple (without conic sections such as circular arcs) one might use B-spline shape functions instead to enhance performance. In the general case, though, a NURBS expansion is used and typically transformed into Bézier elements [57], so that the displacement and traction fields around the boundary are expressed:

$$\mathbf{u}(\xi_u, \xi_v) = \sum_{a=1}^{n_a} R_a(\xi_u, \xi_v)\tilde{\mathbf{u}}_a, \tag{14}$$

$$\mathbf{t}(\xi_u, \xi_v) = \sum_{a=1}^{n_a} R_a(\xi_u, \xi_v)\tilde{\mathbf{t}}_a, \tag{15}$$

where $\tilde{\mathbf{u}}_a$ and $\tilde{\mathbf{t}}_a$ are the nodal displacement and traction parameters associated with the control point with index a, and $\tilde{\boldsymbol{\xi}} = (\xi_u, \xi_v)$ the intrinsic coordinates (i.e. in parametric space) of the field point in a specific patch or element. It should be noted that $\tilde{\mathbf{u}}_a$ and $\tilde{\mathbf{t}}_a$ should not be interpreted as the displacements and tractions at control points. Indeed, the control points may lie outside the geometry. They are simply coefficients using which the displacements and tractions can be recovered using Eq. (14) and Eq. (15). We use a continuous IGABEM formulation so control points are shared between adjacent elements, and the degrees of freedom $\tilde{\mathbf{u}}_a$ and $\tilde{\mathbf{t}}_a$ apply to all elements to which the control point with index a belongs. With this NURBS expansion, the BIE (Eq. (1)) can be written as a discretised form:

$$\begin{aligned}\mathbf{C}(\tilde{\boldsymbol{\zeta}}_c) \sum_{a_0=1}^{n_{a_0}} R_{e_0 a_0}(\tilde{\boldsymbol{\zeta}}_c)\tilde{\mathbf{u}}_{e_0 a_0} + \sum_{e=1}^{n_e} \int\!\!\!\!\!\!- _{\Gamma_e} \mathbf{T}(\tilde{\boldsymbol{\zeta}}_c, \tilde{\boldsymbol{\xi}}) \sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}})\tilde{\mathbf{u}}_{ea} J_e(\tilde{\boldsymbol{\xi}}) \mathrm{d}\tilde{\boldsymbol{\xi}} \\ = \sum_{e=1}^{n_e} \int_{\Gamma_e} \mathbf{U}(\tilde{\boldsymbol{\zeta}}_c, \tilde{\boldsymbol{\xi}}) \sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}})\tilde{\mathbf{t}}_{ea} J_e(\tilde{\boldsymbol{\xi}}) \mathrm{d}\tilde{\boldsymbol{\xi}}.\end{aligned} \tag{16}$$

We use this BIE in a collocation scheme, so that $\tilde{\boldsymbol{\zeta}}_c = (\zeta_u, \zeta_v)$ indicates the intrinsic coordinate of the collocation point, c the collocation point index, e_0 the element in which the collocation point is located, and a_0 is the local index of the collocation point in element e_0. $\tilde{\boldsymbol{\xi}}$ denotes the intrinsic coordinates of field point in parent element, e the element index, a the local index of the node in element e, R_{ea} the shape function, J_e the Jacobian and Γ_e the portion of boundary Γ represented by element e.

There are singular cases in which the collocation point lies in Γ_e; the evaluation of these integrals is discusssed in sections 2.5 and 2.6. With the exception of these singular cases, the above integrals are regular and can be directly evaluated by Gauss-Legendre quadrature. By considering Eq. (16) at a sufficient number of collocation points, a system of equations can be assembled into a matrix form:

$$\mathbf{H}\tilde{\mathbf{u}} = \mathbf{G}\tilde{\mathbf{t}}, \tag{17}$$

DOI: 10.1002/nme

where matrix $\mathbf{H}$ is a square matrix containing a combination of the integrals of the $\mathbf{T}$ kernel and the jump terms, $\mathbf{G}$ a rectangular matrix of $\mathbf{U}$ kernel integrals, $\tilde{\mathbf{u}}$ contains the nodal displacement coefficients and $\tilde{\mathbf{t}}$ the nodal traction coefficients. Both $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{t}}$ include unknown values and the values prescribed by boundary conditions. Since the NURBS basis functions lack the Kronecker delta property, prescription of non-uniform boundary conditions introduces an added complexity. In this case, the traction and displacement coefficients describing these conditions can be calculated by collocating at a sufficient number of points on the physical surface. Application of the boundary conditions in the usual BEM fashion then yields the final form of the linear system:

$$\mathbf{A}\boldsymbol{\lambda} = \mathbf{f}, \tag{18}$$

where matrix $\mathbf{A}$ contains the entries of kernel coefficients associated with the unknown displacements and tractions, $\boldsymbol{\lambda}$ includes all the unknown displacement and traction coefficients and $\mathbf{f}$ the column vector of all the known values. This linear system can now be solved using any solver capable of dealing with a dense, non-symmetric matrix.

In addition, in this paper, the Greville abscissae [58] are adopted to provide a set of fixed collocation points, since the control points may lie off the physical boundary Γ and therefore cannot generally be chosen for collocation in IGABEM, a polar transformation [59] is used to evaluate the weakly singular integrals and the regularized BIE form [60] is adopted to preclude the requirement to evaluate strongly singular integrals. It should further be noted that the computation of the NURBS basis functions incurs somewhat more computational cost than the conventional polynomial shape function. This is an important consideration because, as will be shown, quadrature dominates the run time. The balance between the additional costs of quadrature in IGABEM and the reduced costs from reduction in the required model size will be an interesting research topic.

3. FAST MULTIPOLE EXPANSION OF IGABEM

It has been seen that the BEM offers significant advantages in its boundary-only description, giving it a natural advantage over volumetric analysis methods in exploiting the benefits of the isogeometric concept. Attention is now turned to alleviating the computational burden for large problems. The basic idea of the Fast Multipole Method (FMM) is to apply an iterative solver (e.g. GMRES) to solve Eq. (18) and use multipole expansions to accelerate the matrix-vector multiplication ($\mathbf{A}\boldsymbol{\lambda}$) in each iteration step without forming the explicit matrix $\mathbf{A}$.

3.1. Integral redefinition

Before executing the multipole expansion process, a key step in the FMM is to find a separation of variables, i.e. to separate the effect of the source and field points in the expression of the kernel functions. Eq. (18) may be rewritten:

$$[\widetilde{\mathbf{A}}_{IJ}]\{\tilde{\boldsymbol{\lambda}}_J\} = \{\tilde{\mathbf{f}}_I\}, \tag{19}$$

where $\widetilde{\mathbf{A}} = [\widetilde{A}_{ij}]$ is the submatrix generated from the kernel $\mathbf{T}$ or $\mathbf{U}$, $\tilde{\boldsymbol{\lambda}} = \{\tilde{\lambda}_j\}$, $\tilde{\mathbf{f}} = \{\tilde{f}_i\}$, I the index of collocation points and J the index of control points. Further, we can reformulate (19) as:

$$\{\boldsymbol{\mathcal{H}}_I\} = \{\tilde{\mathbf{f}}_I\}, \tag{20}$$

where the left side includes the integral of unknown displacement and traction which can be defined respectively as:

$$\boldsymbol{\mathcal{H}}^u(\mathbf{s}) = \int_{\Gamma_u} \mathbf{T}(\mathbf{s},\mathbf{x})(\mathbf{u}(\mathbf{x}) - \mathbf{u}(\mathbf{s}))\mathrm{d}\Gamma_u(\mathbf{x}), \tag{21}$$

$$\boldsymbol{\mathcal{H}}^t(\mathbf{s}) = \int_{\Gamma_t} \mathbf{U}(\mathbf{s},\mathbf{x})\mathbf{t}(\mathbf{x})\mathrm{d}\Gamma_t(\mathbf{x}), \tag{22}$$

DOI: 10.1002/nme

with

$$\boldsymbol{\mathcal{H}}(\mathbf{s}) = \boldsymbol{\mathcal{H}}^u(\mathbf{s}) - \boldsymbol{\mathcal{H}}^t(\mathbf{s}), \tag{23}$$

where Γ_u and Γ_t denote the portions of boundary Γ over which the displacements and tractions are unknown.

The right hand side includes the integrals of kernels multiplied by known displacement and traction, and can similarly be defined:

$$\tilde{\mathbf{f}}^u(\mathbf{s}) = \int_{\Gamma_{\bar{u}}} \mathbf{T}(\mathbf{s},\mathbf{x})(\bar{\mathbf{u}}(\mathbf{x}) - \bar{\mathbf{u}}(\mathbf{s}))\mathrm{d}\Gamma_{\bar{u}}(\mathbf{x}), \tag{24}$$

$$\tilde{\mathbf{f}}^t(\mathbf{s}) = \int_{\Gamma_{\bar{t}}} \mathbf{U}(\mathbf{s},\mathbf{x})\bar{\mathbf{t}}(\mathbf{x})\mathrm{d}\Gamma_{\bar{t}}(\mathbf{x}), \tag{25}$$

with

$$\tilde{\mathbf{f}}(\mathbf{s}) = -\tilde{\mathbf{f}}^u(\mathbf{s}) + \tilde{\mathbf{f}}^t(\mathbf{s}), \tag{26}$$

where $\Gamma_{\bar{u}}$ and $\Gamma_{\bar{t}}$ denote the portions of boundary Γ over which displacement and traction boundary conditions are prescribed.

For any source point $\mathbf{s}$, the singularity is cancelled by expressing the displacment at $\mathbf{s}$ *either* as $\mathbf{u}(\mathbf{s})$ in Eq. (21) *or* as $\bar{\mathbf{u}}(\mathbf{s})$ in Eq. (24), depending on whether the source point has a displacement or traction boundary condition. Further, in Eq. (21), (22), (24) and (25), each one can be divided into three parts: the far-field, near-field and singular components. For example, Eq. (21) can be separated as:

$$\boldsymbol{\mathcal{H}}^u(\mathbf{s}) = \boldsymbol{\mathcal{H}}^{u,f}(\mathbf{s}) + \boldsymbol{\mathcal{H}}^{u,n}(\mathbf{s}) + \boldsymbol{\mathcal{H}}^{u,s}(\mathbf{s}), \tag{27}$$

Only the far field part can be considered using the fast multipole expansion; the near-field and singular part will be evaluated directly. Each of the four integrals above contains products of kernels and the displacements or tractions at the field point. This relation between the source point and field point can be expressed in a discretized form:

$$\boldsymbol{\mathcal{H}}^u(\mathbf{s}_k) = \sum_{l=1}^{n_g} \boldsymbol{\mathcal{K}}^t(\mathbf{s}_k,\mathbf{x}_l)\boldsymbol{\sigma}^u(\mathbf{x}_l), \tag{28}$$

$$\boldsymbol{\mathcal{H}}^t(\mathbf{s}_k) = \sum_{l=1}^{n_g} \boldsymbol{\mathcal{K}}^u(\mathbf{s}_k,\mathbf{x}_l)\boldsymbol{\sigma}^t(\mathbf{x}_l), \tag{29}$$

where $k = 1,\ldots,n_c$ is the index of the source points (collocation points), $l = 1,\ldots,n_g$ the index of the field points (Gauss points), $\boldsymbol{\mathcal{K}}^t$ the equivalent kernel of traction, $\boldsymbol{\mathcal{K}}^u$ the equivalent kernel of displacement, $\boldsymbol{\sigma}^u$ the equivalent displacement at each field point and $\boldsymbol{\sigma}^t$ the equivalent traction. For brevity, we use $\boldsymbol{\mathcal{K}}$ or $\boldsymbol{\sigma}$ to denote both of them as required.

In order to develop the FMM in a separated form, the equivalent displacement kernel $\boldsymbol{\mathcal{K}}^u$ has the same form as $\mathbf{U}$, and the the equivalent traction kernel $\boldsymbol{\mathcal{K}}^t$ should have the same form as $\mathbf{T}$ but we remove the effect of the unit outward normal vector $\mathbf{n}$ at the field point, leaving the new traction kernel written as:

$$\mathcal{K}^t_{ijk} = -\frac{1}{8\pi(1-\nu)r^2}\left[3r_{,i}r_{,j}r_{,k} + (1-2\nu)\Theta_{ijk}\right], \tag{30}$$

with

$$\Theta_{ijk} = \begin{cases} r_{,k} & \text{if } i = j, \\ r_{,j} & \text{if } i \neq j \text{ and } i = k, \\ -r_{,i} & \text{if } i \neq j \text{ and } j = k, \\ 0 & \text{if } i \neq j \neq k. \end{cases} \tag{31}$$

DOI: 10.1002/nme

where i, j, k denote the index from 1 to 3 representing the x-, y- and z-directions. Expressions for $\boldsymbol{\sigma}^u$ and $\boldsymbol{\sigma}^t$ are given as follows:

$$\boldsymbol{\sigma}^u(\mathbf{x}_l) = \boldsymbol{\sigma}^u(\mathbf{x}_{eg}) = J_e(\tilde{\boldsymbol{\xi}}) w_{eg} \mathbf{n}_{eg}^{\intercal} \left[\sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}}) \tilde{\mathbf{u}}_{ea} - \sum_{a_0=1}^{n_{a_0}} R_{e_0 a_0}(\tilde{\boldsymbol{\zeta}}_c) \tilde{\mathbf{u}}_{e_0 a_0} \right], \tag{32}$$

$$\boldsymbol{\sigma}^t(\mathbf{x}_l) = \boldsymbol{\sigma}^t(\mathbf{x}_{eg}) = J_e(\tilde{\boldsymbol{\xi}}) w_{eg} \sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}}) \tilde{\mathbf{t}}_{ea}, \tag{33}$$

where w_{eg} is the weight associated with each field point, and e, a, e_0, a_0 denote the same meaning as Eq. (16). The term $\mathbf{n}_{eg}^{\intercal}$ is the transpose of the unit outward normal vector at the field point, and is included here following its removal from the traction kernel.

3.2. Decomposition of the kernel function

After rewriting the integrals, the source points and field points should be separated in the kernel. We seek to approximate the kernel $\boldsymbol{\mathcal{K}}$ as the following form:

$$\boldsymbol{\mathcal{K}}(\mathbf{s}, \mathbf{x}) \approx \sum_{m=1}^{n_m} \sum_{n=1}^{n_n} \boldsymbol{\mathcal{K}}(\bar{\mathbf{s}}_m, \bar{\mathbf{x}}_n) R(\bar{\mathbf{s}}_m, \mathbf{s}) R(\bar{\mathbf{x}}_n, \mathbf{x}), \tag{34}$$

where

$$R(\bar{\mathbf{s}}_m, \mathbf{s}) = \prod_{i=1}^{3} S(\bar{s}_m^i, s^i), \tag{35}$$

in three dimensions, with a directly corresponding expression for $R(\bar{\mathbf{x}}_n, \mathbf{x})$. The above expression can be approximated by any suitable polynomial interpolation functions. Chebyshev polynomials are used in the current work which is based on [43] and reproduced here. So m is the index of Chebyshev nodes associated with source points and n the Chebyshev nodes index for field points.

The Chebyshev nodes are located at:

$$\bar{s}_m = \cos\left(\frac{2m-1}{2n_m} \pi \right), \; m = 1, \ldots, n_m. \tag{36}$$

The mapping function $S(\bar{s}_m, s)$ has the following form:

$$S(\bar{s}_m, s) = \frac{1}{n_m} + \frac{2}{n_m} \sum_{l=1}^{n_m - 1} T_l(s) T_l(\bar{s}_m), \tag{37}$$

where

$$T_l(s) = \cos(l \cos^{-1}(s)), \tag{38}$$

with the same form used for $T_l(\bar{s}_m)$ and $S(\bar{x}_n, x)$.

3.3. Far field approximation

In order to fully separate the source points and field points, for a 2-D problem, a quadtree structure should be built and, for 3-D, an octree structure. Fig. 1 shows a quadtree example. A square (2-D) or cube (3-D) is called a cell, and the cells exist at different levels. The level l_0 cell should cover the whole problem domain. A higher level cell at level l_q is derived from the subdivision of its parent cell at level l_{q-1}. Cells are divided into increasingly higher levels until a suitable level $l_{\max}$ is reached, when the number of source and field points contained in the cell is smaller than a prespecified number, and this highest level cell is called a leaf. Large differences in size of adjacent elements

DOI: 10.1002/nme

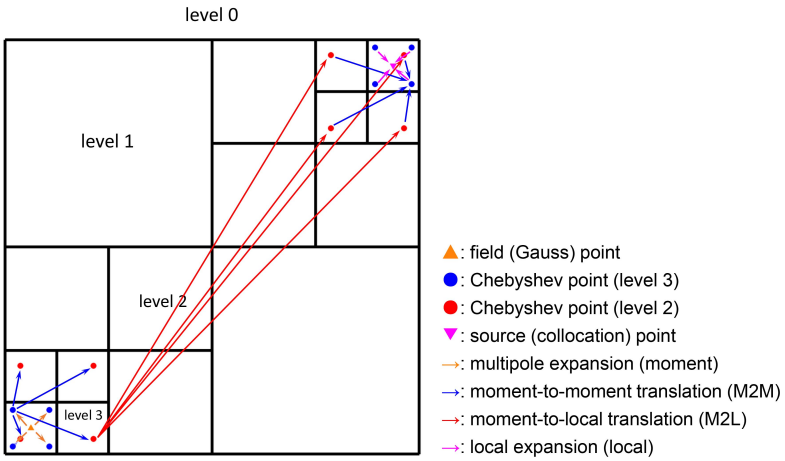


Figure 1. A quadtree based bbFMM scheme

should be avoided, not only to limit the near-field area but also to keep computational stability. The source points and field points are allocated to their corresponding leaves. The definition for the far-field is that two cells are nonadjacent but their parent cells are adjacent, and the near-field definition is that the two cells are adjacent. Similar to the conventional FMM, the process of far field approximation through bbFMM can be described as the following step-by-step procedure:

Step 1. Multipole expansion (moment)

In Fig. 1, the orange point denotes an arbitrary field point (Gauss point) existing in a leaf at level l_3. More generally, for any field point in a leaf $J_{\max}$ at level $l_{\max}$, the weights in Eq. (34) associated with the Chebyshev nodes $\bar{\mathbf{x}}_n^{J_{\max}}$ can be calculated by:

$$\mathbf{m}^{J_{\max}} = [R(\bar{\mathbf{x}}_n^{J_{\max}}, \mathbf{x}_l)], \tag{39}$$

where $\mathbf{m}^{J_{\max}}$ denotes a vector formed by all the Chebyshev nodes in the leaf $J_{\max}$, $\mathbf{x}_l$ the field point in the leaf and l the index of this field point.

Step 2. Moment-to-moment translation (M2M)

The M2M process (represented by the blue arrows in Fig. 1) is used to generate a transfer matrix from a child cell J_{q+1} at level l_{q+1} into its parent cell J_q at level l_q. The transfer matrix is given as:

$$\mathbf{M}^{J_q, J_{q+1}} = [R(\bar{\mathbf{x}}_n^{J_q}, \bar{\mathbf{x}}_{n'}^{J_{q+1}})], \tag{40}$$

where n' is the index of Chebyshev nodes in the child cell J_{q+1}. The same transfer matrix can be formed at each level and stopped at level l_2. The transfer matrices can be stacked as:

$$\mathbf{M}^{J_2,\ldots,J_{\max}} = \prod_{l_q=l_{\max}-1}^{l_2} [R(\bar{\mathbf{x}}_n^{J_q}, \bar{\mathbf{x}}_{n'}^{J_{q+1}})]. \tag{41}$$

Step 3. Moment-to-Local translation (M2L)

In order to avoid the singularity, the lowest level to be considered for the FMM approximation of the far-field interactions will be level l_2. For the same reason, the cells should not be adjacent, i.e. there is no common vertex shared by two cells. Thus, a list will be formed for each pair (I_q, J_q) of source point and field point. In the list, only the pairs lying in nonadjacent cells (interaction list) will be considered by the bbFMM scheme, and others will be evaluated directly. Now, the coefficient matrix associated with two sets of Chebyshev nodes in the interaction list can be written as:

$$\mathbf{B}^{I_q, J_q} = [\boldsymbol{\mathcal{K}}(\bar{\mathbf{s}}_m^{I_q}, \bar{\mathbf{x}}_n^{J_q})], \tag{42}$$

DOI: 10.1002/nme

where I_q and J_q denote the different cells of two sets of Chebyshev nodes located at level l_q. The M2L stage is the most numerically intensive. Each cell containing the source point will interact with up to $6^3 - 3^3 = 189$ other cells at the same level, each of which is described by a multiplicity of Chebyshev nodes. There will therefore be a serious detrimental effect on performance if no action is taken to accelerate the computation of these operators. We note that the M2L computation is also the most expensive in classical FMM schemes, though in bbFMM we can expect the additional cost of computing Chebyshev interpolations instead of a few terms in a Taylor series in the multipole expansions.

Step 4. Local-to-local translation (L2L)

By reversing the process of M2M, the transfer matrices from the lowest level to the highest level can be written as:

$$\mathbf{W}^{I_{\max},\ldots,I_2} = \prod_{l_q=l_2}^{l_{\max}-1} [R(\bar{\mathbf{s}}_{m'}^{I_{q+1}}, \bar{\mathbf{s}}_{m}^{I_q})], \tag{43}$$

where cell I_q is the parent of cell I_{q+1} and m' the index of Chebyshev nodes in the child cell.

Step 5. Local expansion (local)

A source point located in a leaf $I_{\max}$ of level $l_{\max}$ can be considered following the same procedure as the moment:

$$\mathbf{w}^{I_{\max}} = [R(\bar{\mathbf{s}}_{m}^{I_{\max}}, \mathbf{s}_k)], \tag{44}$$

where $\mathbf{w}$ is a vector spanned by all the Chebyshev nodes in the leaf $I_{\max}$ and k the index of the source point.

Step 6. Assembly

For a specific source point, the effect at a field point can now be written as:

$$\boldsymbol{\mathcal{H}}^f(\mathbf{s}_k, \mathbf{x}_l) = (\mathbf{w}^{I_{\max}})^\intercal \mathbf{W}^{I_{\max},\ldots,I_q} \mathbf{B}^{I_q,J_q} \mathbf{M}^{J_q,\ldots,J_{\max}} \mathbf{m}^{J_{\max}} \boldsymbol{\sigma}(\mathbf{x}_l). \tag{45}$$

Here, q can identify any level which depends on the interaction list. A summation form now can be written to evaluate all the far-field contributions:

$$\begin{aligned} \boldsymbol{\mathcal{H}}^f(\mathbf{s}_k) = (\mathbf{w}^{I_{\max}})^\intercal \sum_{\mathbf{x}_l \in (I_q,J_q)} \mathbf{W}^{I_{\max},\ldots,I_q} \sum_{\mathbf{x}_l \in (I_q,J_q)} \mathbf{B}^{I_q,J_q} \\ \sum_{\mathbf{x}_l \in (I_q,J_q)} \mathbf{M}^{J_q,\ldots,J_{\max}} \sum_{\mathbf{x}_l \in J_{\max}} \mathbf{m}^{J_{\max}} \boldsymbol{\sigma}(\mathbf{x}_l). \end{aligned} \tag{46}$$

The far field part on the right hand side of Eq. (20) follows the procedure as above. However, in order to execute the last step efficiently, some optimization for the M2L operation is needed to reduce the time required to compute the matrix-vector products. A low rank approximation based on the Proper Generalized Decomposition (PGD) for matrix compression will be presented in the next section.

3.4. Optimization of the M2L operation by PGD

The PGD approach is based on a natural phenomenon that any data can be represented by the combination of the information in each of their directions. In other words, the kernel function $\boldsymbol{\mathcal{K}}$ may be written in a separated form:

$$\boldsymbol{\mathcal{K}}(s_1, s_2, s_3, x_1, x_2, x_3) = \sum_{i=1}^{n} \mathbf{S}_i^1(s_1) \circ \mathbf{S}_i^2(s_2) \circ \mathbf{S}_i^3(s_3) \circ \mathbf{X}_i^1(x_1) \circ \mathbf{X}_i^2(x_2) \circ \mathbf{X}_i^3(x_3), \tag{47}$$

where the operator $\circ$ denotes a Hadamard product. For the sake of brevity, we write the above equation as:

$$\boldsymbol{\mathcal{K}}(\phi_1, \ldots, \phi_6) = \sum_{i=1}^{n} \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_i^d(\phi_d). \tag{48}$$

DOI: 10.1002/nme

The PGD approximation is shown above as a summation of n terms comprising functional products. Instead of attempting to approximate all the matrices describing the kernel behaviour between Chebyshev points in two cells in the interaction list at some level in the bbFMM, an important development here is that we seek to use PGD to approximate the behaviour of the kernel over the entire domain of far-field interactions. Recalling the summation process (Eq. (46)), for each source cell, there are at most 189 far-field cells which will interact and a leaf cell. Hence, executing the summation process will incur memory and time cost, especially for the 3-D problem. Writing the kernel function as above has some remarkable advantages: 1) acceleration of the matrix-vector products in Eq. (46), 2) reduction in memory required to store the kernel information.

For the acceleration of the M2L process, if we seek a low rank approximation $\widetilde{\boldsymbol{\mathcal{K}}}(\phi_1,\ldots,\phi_6)$ to the kernel function $\boldsymbol{\mathcal{K}}(\phi_1,\ldots,\phi_6)$, we may consider this as the solution of a simple governing equation:

$$\widetilde{\boldsymbol{\mathcal{K}}}(\phi_1,\ldots,\phi_6)=\boldsymbol{\mathcal{K}}(\phi_1,\ldots,\phi_6),\ (\phi_1,\ldots,\phi_6)\in\Omega=\Omega_1\times\cdots\times\Omega_6. \tag{49}$$

We might use a weighted residual method to seek this solution, and the corresponding weighted residual form will be

$$\int_\Omega \boldsymbol{\mathcal{K}}^*\circ\left(\widetilde{\boldsymbol{\mathcal{K}}}-\boldsymbol{\mathcal{K}}\right)\mathrm{d}\Omega=0,\ \forall\boldsymbol{\mathcal{K}}^*, \tag{50}$$

where $\boldsymbol{\mathcal{K}}^*$ is an arbitrary weight function.

An initial guess can always be made to generate the first approximation, and can be written as:

$$\widetilde{\boldsymbol{\mathcal{K}}}^1(\phi_1,\ldots,\phi_6)=\prod_{d=1}^{6}\boldsymbol{\mathcal{F}}_1^d(\phi_d). \tag{51}$$

Here, the initial guess could use some simple functions (e.g. $\boldsymbol{\mathcal{F}}_1^d(\phi_d)=[1]$) or the modes of a similar problem previously solved. We further note that $\boldsymbol{\mathcal{K}}^t$ includes 27 coefficients, which can be written as a $3\times3\times3$ matrix or a 3×9 matrix as required, and a 3×3 matrix is used to approximate the displacement kernel $\boldsymbol{\mathcal{K}}^u$ for a 3-D problem.

In order to reach a sufficiently accurate low-rank approximation, an enrichment process will be carried out as follows:

$$\widetilde{\boldsymbol{\mathcal{K}}}^n(\phi_1,\ldots,\phi_6)=\widetilde{\boldsymbol{\mathcal{K}}}^{n-1}(\phi_1,\ldots,\phi_6)+\prod_{d=1}^{6}\boldsymbol{\mathcal{F}}_n^d(\phi_d)=\sum_{i=1}^{n-1}\prod_{d=1}^{6}\boldsymbol{\mathcal{F}}_i^d(\phi_d)+\prod_{d=1}^{6}\boldsymbol{\mathcal{F}}_n^d(\phi_d), \tag{52}$$

where the functions $\boldsymbol{\mathcal{F}}_n^1(\phi_1),\cdots,\boldsymbol{\mathcal{F}}_n^6(\phi_6)$ are the only unknowns. In order to solve this non-linear problem, an alternating direction scheme is used. At any iteration step p, the approximated function is:

$$\widetilde{\boldsymbol{\mathcal{K}}}^n(\phi_1,\ldots,\phi_6)=\sum_{i=1}^{n-1}\prod_{d=1}^{6}\boldsymbol{\mathcal{F}}_i^d(\phi_d)+\prod_{\substack{d=1\\d\neq k}}^{6}\boldsymbol{\mathcal{F}}_n^{d,p-1}(\phi_d)\circ\boldsymbol{\mathcal{F}}_n^{k,p}(\phi_k), \tag{53}$$

where $\boldsymbol{\mathcal{F}}_n^{d,1}(\phi_d)$ can be taken from the previous modes, e.g. $\boldsymbol{\mathcal{F}}_1^d(\phi_d)$, and it remains to find $\boldsymbol{\mathcal{F}}_n^{k,p}(\phi_k)$. The weight function $\boldsymbol{\mathcal{K}}^*$ is given the form:

$$\widetilde{\boldsymbol{\mathcal{K}}}^*(\phi_1,\ldots,\phi_6)=\prod_{\substack{d=1\\d\neq k}}^{6}\boldsymbol{\mathcal{F}}_n^{d,p-1}(\phi_d)\circ\boldsymbol{\mathcal{F}}_n^{k,*}(\phi_k), \tag{54}$$

DOI: 10.1002/nme

Substituting Eq. (53) and (54) into Eq. (50) yields:

$$\int_{\Omega} \prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \circ \boldsymbol{\mathcal{F}}_n^{k,*} \circ \left(\prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \circ \boldsymbol{\mathcal{F}}_n^{k,p} \right) \mathrm{d}\Omega$$
$$= \int_{\Omega} \prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \circ \boldsymbol{\mathcal{F}}_n^{k,*} \circ \left(\boldsymbol{\mathcal{K}} - \sum_{i=1}^{n-1} \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_i^{d} \right) \mathrm{d}\Omega. \tag{55}$$

The ϕ_k-direction integrals are then evaluated over Ω_k to yield:

$$\int_{\Omega_k} \boldsymbol{\alpha} \circ \boldsymbol{\mathcal{F}}_n^{k,*} \circ \boldsymbol{\mathcal{F}}_n^{k,p} \mathrm{d}\Omega_k = \int_{\Omega_k} \boldsymbol{\mathcal{F}}_n^{k,*} \circ \left(\boldsymbol{\beta} - \sum_{i}^{n-1} \boldsymbol{\gamma}_i \circ \boldsymbol{\mathcal{F}}_i^{k} \right) \mathrm{d}\Omega_k, \tag{56}$$

where

$$\boldsymbol{\alpha} = \int_{\Omega \setminus \Omega_k} \left(\prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \right)^{\circ 2} \mathrm{d}\Omega, \tag{57}$$

$$\boldsymbol{\beta}(\phi_k) = \int_{\Omega \setminus \Omega_k} \left(\prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \right) \circ \boldsymbol{\mathcal{K}} \mathrm{d}\Omega, \tag{58}$$

$$\boldsymbol{\gamma}_i = \int_{\Omega \setminus \Omega_k} \left(\prod_{\substack{d=1 \\ d \neq k}}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1} \circ \boldsymbol{\mathcal{F}}_i^{d} \right) \mathrm{d}\Omega. \tag{59}$$

The corresponding strong form is given as:

$$\boldsymbol{\alpha} \circ \boldsymbol{\mathcal{F}}_n^{k,p} = \boldsymbol{\beta} - \sum_{i}^{n-1} \boldsymbol{\gamma}_i \circ \boldsymbol{\mathcal{F}}_i^{k}, \tag{60}$$

from which $\boldsymbol{\mathcal{F}}_n^{k,p}(\phi_k)$ may readily be obtained.

By increasing p, we can continue to alternate between progressively improving forms of $\boldsymbol{\mathcal{F}}_n^{d}(\phi_d)$ until convergence is achieved. A suitable stopping criterion for the iteration process is given as:

$$\tilde{\epsilon}(p) > \frac{\left\| \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_n^{d,p}(\phi_d) - \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1}(\phi_d) \right\|}{\left\| \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_n^{d,p-1}(\phi_d) \right\|}, \tag{61}$$

where the $\|\cdot\|$ is the L^2-norm and $\tilde{\epsilon}(p)$ is a suitable threshold value.

After some enrichment steps, the whole approximation process will be stopped once the following criterion is satisfied:

$$\epsilon(n) > \frac{\left\| \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_n^{d}(\phi_d) \right\|}{\left\| \prod_{d=1}^{6} \boldsymbol{\mathcal{F}}_1^{d}(\phi_d) \right\|}, \tag{62}$$

DOI: 10.1002/nme

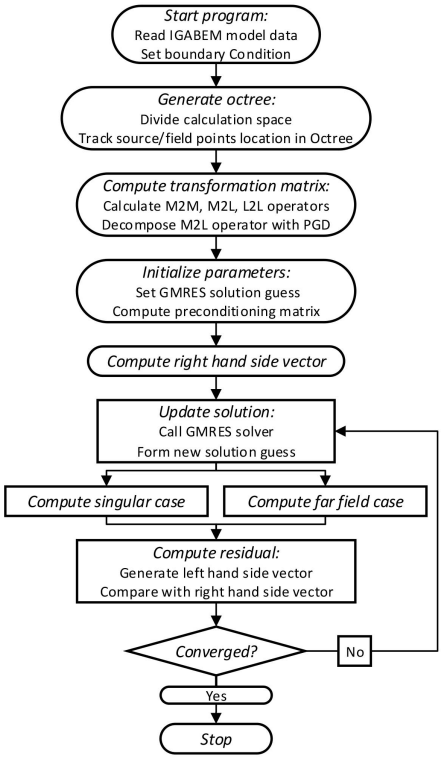


Figure 2. Algorithm of the bbFMM solver

with an appropriate threshold value $\epsilon(n)$.

Now, back to Eq. (46), all the matrices $\mathbf{B}^{I_q,J_q}$ could be grouped as a single matrix:

$$\mathbf{B}^{I_q} = [\mathbf{B}^{I_q,J_q^1}, \ldots, \mathbf{B}^{I_q,J_q}, \ldots, \mathbf{B}^{I_q,J_q^{\max}}], \tag{63}$$

substitute the related Chebyshev nodes into Eq. (47) and group the first and last 3 terms as:

$$\mathcal{K}(\mathbf{s},\mathbf{x}) = \sum_{i=1}^{n} \mathbf{S}_i(\mathbf{s}) \circ \mathbf{X}_i(\mathbf{x}), \tag{64}$$

by using Eq. (64), Eq. (63) can be written as:

$$\mathbf{B}^{I_q} = \boldsymbol{\mathcal{S}}^{I_q}[\boldsymbol{\mathcal{X}}^{I_q,J_q^1}, \ldots, \boldsymbol{\mathcal{X}}^{I_q,J_q}, \ldots, \boldsymbol{\mathcal{X}}^{I_q,J_q^{\max}}], \tag{65}$$

here, the kernel function has been represented as a separated form with small size.

3.5. Algorithm

The full implementation of bbFMM with IGABEM is shown in Fig. 2. It may be noticed that the steps except for the iterative part can be precomputed, and the iterative part will be calculated by GMRES until convergence. Where we refer to a preconditioning matrix we use a block diagonal preconditioner formed by directly evaluating the singular integrals, though other preconditioners are available. In addition, the implementation of bbFMM with conventional BEM follows the above steps, the only differences being in the shape functions used and integral interval ($[0,1] \rightarrow [-1,1]$).

DOI: 10.1002/nme

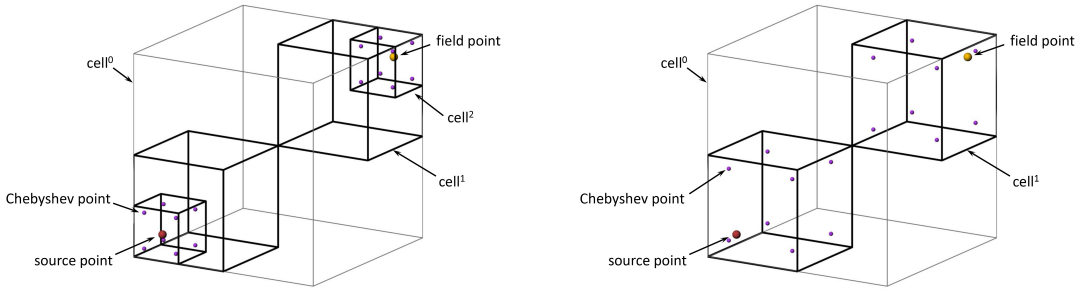
4. NUMERICAL RESULTS

4.1. Kernel approximation by Chebyshev polynomial

Before presenting the data compression, it is valuable to consider the influence of the number of Chebyshev nodes. It is clear that the accuracy of interpolation will be improved by increasing the number of Chebyshev nodes, but this comes at some computational cost. The relative error can be defined as:

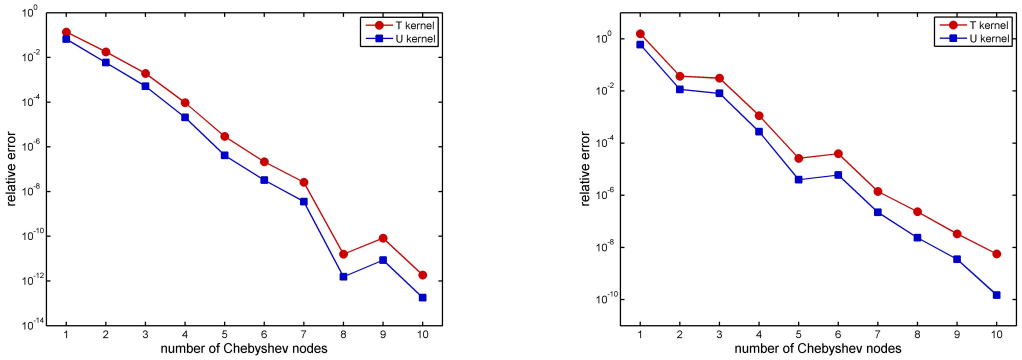
$$E_c = \frac{\|\boldsymbol{\mathcal{K}}^c - \boldsymbol{\mathcal{K}}^d\|}{\|\boldsymbol{\mathcal{K}}^d\|} \times 100\%, \tag{66}$$

where $\boldsymbol{\mathcal{K}}^c$ denotes the kernel interpolated by Chebyshev nodes, $\boldsymbol{\mathcal{K}}^d$ is the kernel evaluated directly and $\|\bullet\|$ the L^2-norm (similarly hereinafter). The test cases are depicted in Fig. 3; the case (a) shows two cells which would lie in an interaction list, and are amenable to FMM acceleration, while the case (b) shows two adjacent cells. The bounding box (cell^0) can be expressed as $[0, 10] \times [0, 10] \times [0, 10]$, and in both cases we consider the source point coordinates $(1, 1, 1)$ and a field point located at $(9, 9, 9)$. The relative errors in the traction and displacement kernels, approximated by using different numbers of Chebyshev nodes, are illustrated in Fig. 4. It can be seen that with an increasing number of nodes, the accuracy is consistently improved; once 5 nodes are used in each direction the accuracy will become acceptable for the nonadjacent case, but approximating through two adjacent cells will lead to some relatively large errors even though the source and field points are unchanged.



(a) Cells in interaction list (b) Adjacent cells

Figure 3. Geometry of the test cases



(a) Cells in interaction list (b) Adjacent cells

Figure 4. Relative error with the number of Chebyshev nodes

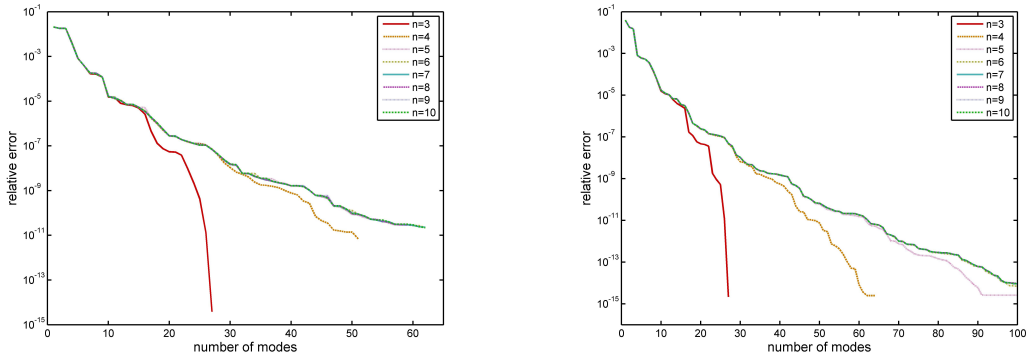
DOI: 10.1002/nme

4.2. *Data compression by PGD*

While the number of Chebyshev nodes used determines the accuracy of the scheme, the number of PGD modes (i.e. enrichment steps) determines the amount of compression that can be achieved during the M2L process. We introduce a relative error defined as:

$$E_p = \frac{\|\widetilde{\boldsymbol{\mathcal{K}}} - \boldsymbol{\mathcal{K}}^c\|}{\|\boldsymbol{\mathcal{K}}^c\|} \times 100\%, \tag{67}$$

where $\widetilde{\boldsymbol{\mathcal{K}}}$ is the approximated kernel, which can be calculated by PGD or SVD. For convenience, we only compare a single pair of cells appearing in the interaction list which can be referred to Fig. 3a. Fig. 5a shows the relative error E_p obtained for the traction kernel using different numbers of PGD modes and Chebyshev nodes, denoted n. It can be seen that an increase in the number of Chebyshev nodes brings a requirement for more modes in order to satisfy the stopping criterion (Eq. (62)). After 5 Chebyshev nodes, the number of modes required will be almost fixed with the same relative error. Fig. 5b shows the SVD computation to have the same tendency as the PGD result. By comparing these two figures, it is evident that the PGD solutions are not able to reach a lower relative error than the SVD solutions; this is mainly because the coefficients $\boldsymbol{\alpha}$ in the PGD enrichment process will become larger after several iterations, eventually exceeding the range of values represented by double precision (i.e. from 1.79769×10^{308} to 2.22507×10^{-308}). However, both PGD and SVD yield a reduction in the size of the data structure, and in a separated form.



(a) PGD (b) SVD

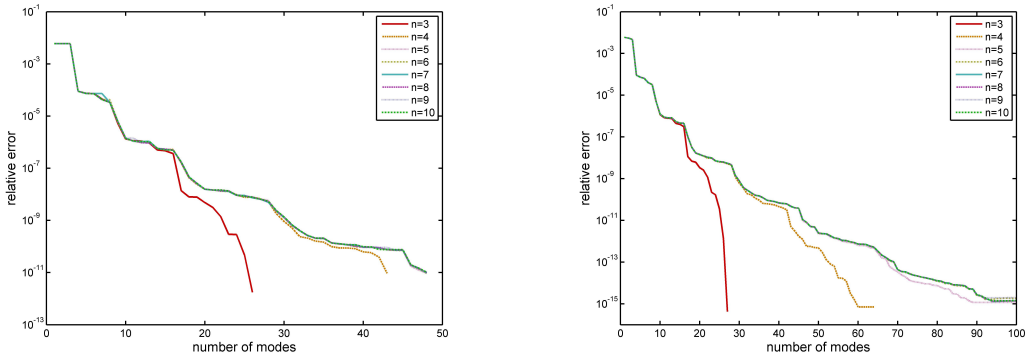
Figure 5. Relative error of $\boldsymbol{\mathcal{K}}^t$ kernel with the number of modes

Fig. 6a and Fig. 6b show the corresponding analysis for the $\boldsymbol{\mathcal{K}}^u$ kernel. The behaviour is similar to that for the $\boldsymbol{\mathcal{K}}^t$ kernel.

There are two variables which can control the final number of modes generated by the PGD algorithm. One variable $\epsilon(n)$, which can be treated as a global threshold value, decides the precision of the final approximate result. Another variable $\tilde{\epsilon}(P)$, is a local threshold value which can influence both the computational speed and the final number of modes. Fig. 7 shows the relationship between the local threshold value $\tilde{\epsilon}(P)$ and the final number of modes required to generate the PGD. It can be seen that, in each enrichment step, by decreasing the local threshold value, a smaller size of data structure will result; in other words, this offers a higher compression ratio, but at the expense of additional computational effort.

The PGD algorithm presented in section 3.4 proceeds in a very different way to the SVD compression algorithms in the literature. While SVD has been shown to provide a low-rank approximation to a matrix of kernel functions relating Chebyshev nodes in one cell to those in another cell at the same level, the PGD algorithm approximates the kernel across the problem domain (though it applies only in the regions of far-field interaction). In spite of this fundamental difference, the computational complexity of PGD and SVD may be further revealed by a test in

DOI: 10.1002/nme



(a) PGD (b) SVD

Figure 6. Relative error of $\mathcal{K}^u$ kernel with the number of modes

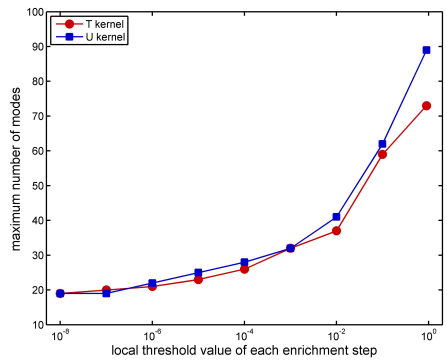


Figure 7. Maximum number of modes generated by different threshold values $\tilde{\epsilon}(P)$

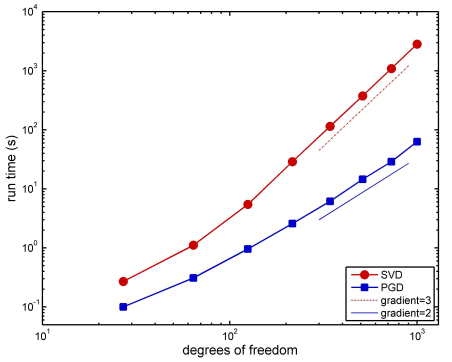
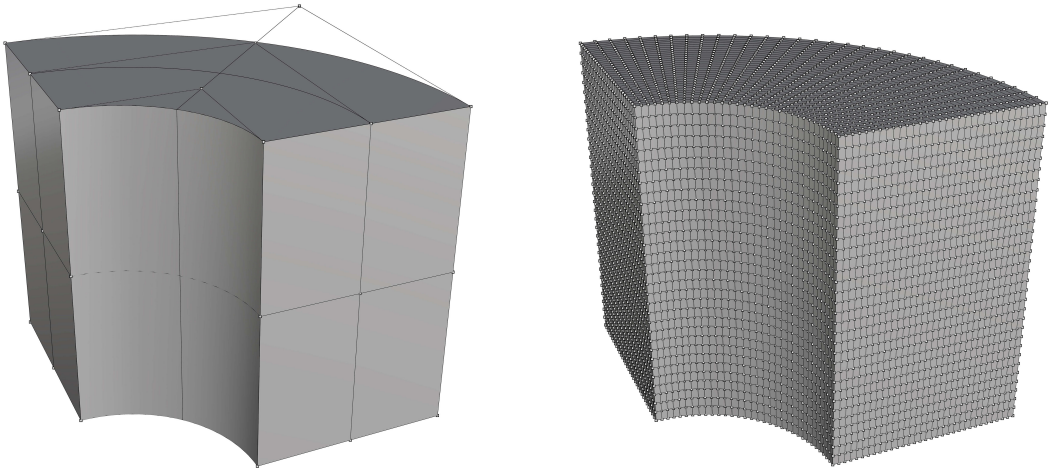


Figure 8. Comparison of SVD and PGD run-time for M2L operation

which we limit our PGD to the scope of the SVD algorithms, i.e. approximating the kernel behaviour between the Chebyshev nodes contained in two cells. We present in Fig. 8 the run time required by both schemes to complete this operation. By applying different numbers of Chebyshev nodes, the size of the data structure will grow rapidly. Here, as the Chebyshev interpolation is built in 3-D, the size of the kernel matrix being approximated will be related to the cube of the number of Chebyshev nodes in each direction, and this is shown as "degrees of freedom" on the horizontal axis of Fig. 8. It can be seen that, once around 100 DOF is reached, the PGD exhibits $O(n^2)$ complexity while the SVD exhibits $O(n^3)$ for the computation of the M2L process. This significant improvement found in approximating the kernel only between adjacent cells will be magnified further by using PGD to approximate the kernel over the whole domain of far-field interactions, not just a single cell. It will be shown in section 4.3 that this new approach accelerates the solution of problems across the whole range of problem sizes tested.

From the discussion above, the advantages of using PGD compared with SVD can be summarised as: 1) run time will be clearly decreased; 2) PGD offers a flexible choice for the number of modes generated; a loose compression will be obtained quickly and a tight compression will further accelerate the final solving process; 3) without precompressing [43], any suitable quadrature rule can be directly employed in the coefficient integrals (Eq. (57), (58) and (59)) contained in of the enrichment process. This will further enhance the efficiency of computing matrix-vector products.

DOI: 10.1002/nme



(a) Mesh 0 (78 DOF) (b) Mesh 5 (28806 DOF)

Figure 9. Quarter cylinder geometry with NURBS control points

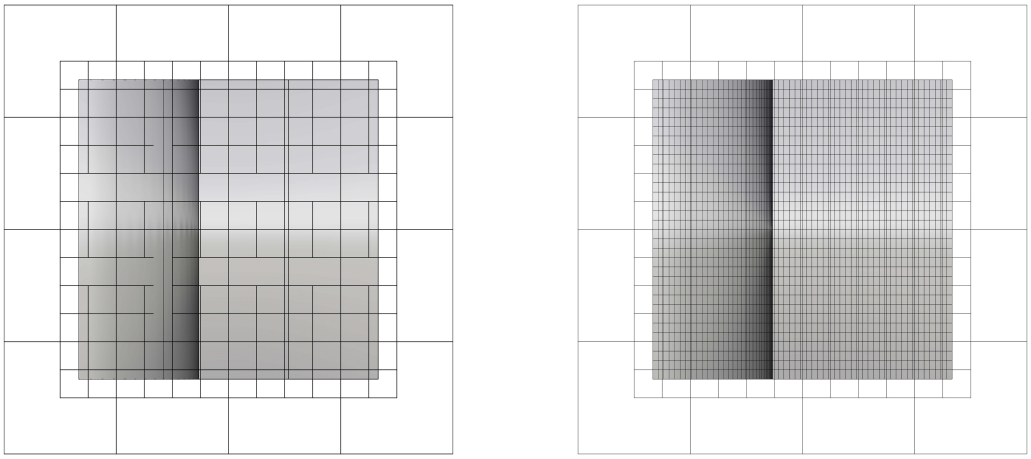
4.3. Quarter cylinder example

The first example studied is a single quarter cylinder, shown in Fig. 9, under an internal pressure for which the analytical solution gives a maximum displacement of $0.341215m$. The right, left and bottom surfaces have prescribed displacement constraints in the direction normal to each surface. The height is 1m, inner radius 0.4m and outer radius 1m. The control points and geometry are displayed in the figure. An octree structure is presented in Fig. 10 and 11. In Fig. 10a and 11a, as we adopt a 10×10 Gaussian quadrature scheme in each patch, the octree structure is generated depending on the location of source and field points, so there will exist some parts uncovered by the leaves. With the patch size decreasing, the leaves could fully contain all the patches (Fig. 10b and 11b). Note that some cells appear as rectangles or other shapes that are not normally seen in octree subdivisions, this optical effect simply results from the fact that we draw only the cells that contain source and field points. Figs. 12a and 12b show that both a coarse and fine model are able to approximate well the analytical deformation magnitude.

In order to compare fully the efficiency between the bbFMM IGABEM and the conventional IGABEM, a knot insertion algorithm [10] is introduced to increase the DOF. During the computation, the Gaussian elimination method was adopted for solving the final matrix generated by conventional IGABEM and a GMRES iterative solver was introduced for the bbFMM IGABEM approach, with a residual tolerance of 10^{-5}. For both schemes the same Gaussian quadrature rule was used for the integrals ($n_g = 10$). The interpolation in the bbFMM approach was constructed using 5 Chebyshev nodes in each direction. All the codes were executed on a 2.80GHz quadcore processor with 8 parallel threads, but in order to provide a true comparison of the algorithms, the core is limited to only one with a single thread.

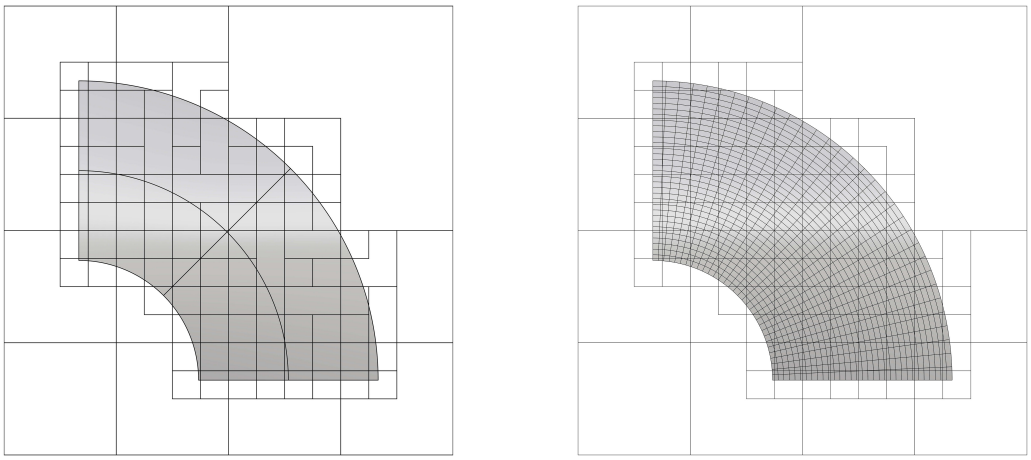
The runtime cost is shown in Fig. 13. Compared with the conventional IGABEM, the time cost of the bbFMM IGABEM for solving a large problem will be clearly decreased. The $O(n^2)$ behaviour for conventional IGABEM and $O(n)$ for bbFMM IGABEM are obtained. This suggests that the computational complexity $O(n^3)$ for solving the conventional IGABEM linear system by using Gaussian elimination does not play a dominant role in the overall computation. On the contrary, the matrix assembly will dominate. On the other hand, the bbFMM algorithm combined with PGD outperforms the use of SVD. When PGD is used in preference to SVD performance gains are seen across the range of model sizes tested, and in the worst case this gain is approximately a factor of 2. It is interesting to note the improvement for small problem sizes, effectively offsetting the overheads involved in using FMM for these problems. Although the time cost for the M2L operation is almost fixed, depending only on the depth of the octree structure, use of PGD will provide significant

DOI: 10.1002/nme



(a) Mesh 0 (b) Mesh 5

Figure 10. Octree structure of the quarter cylinder model (Front view)

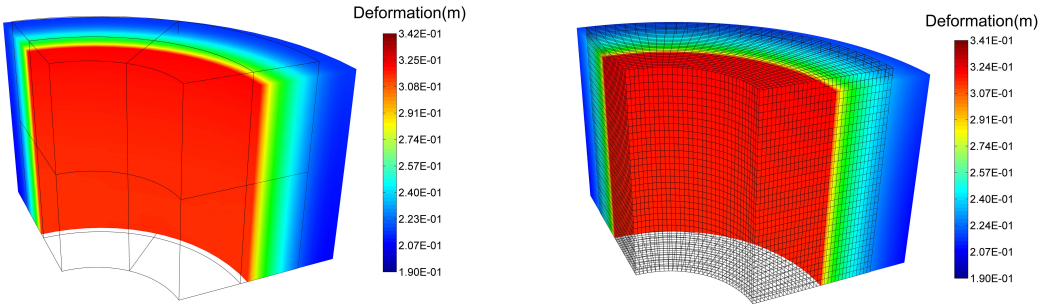


(a) Mesh 0 (b) Mesh 5

Figure 11. Octree structure of the quarter cylinder model (Top view)

improvements for calculating the matrix-vector products. Furthermore, the pre-computational time can be saved. Fig. 14 shows the maximum memory requirement for the two methods. Although the bbFMM incurs a lower memory cost than conventional IGABEM, both methods exhibit $O(n)$ memory requirement. While readers might expect the memory requirement to be $O(n^2)$ if we are storing a matrix, the $O(n)$ behaviour arises because the main source of memory consumption is that, in order to improve computational efficiency, we store the basis function values, quadrature points and normal vectors. For the range of model sizes tested this is seen to dominate over the memory requirement to store the system matrix terms. Meanwhile, an analysis of the relative error compared with the conventional IGABEM, shown in Tab. I, shows that the accuracy is almost unaffected by the use of this PGD accelerated bbFMM algorithm. In this table, the L^2 error is a relative displacement

DOI: 10.1002/nme



(a) Mesh 0 (b) Mesh 5

Figure 12. Deformation solution with undeformed patches of the quarter cylinder model

Table I. Relative error of the quarter cylinder problem, bbFMM vs. conventional IGABEM

Mesh	DOF	L^2-norm	
		bbFMM	conventional
0	78	2.34E-3	2.43E-3
1	294	7.49E-4	7.80E-4
2	654	9.10E-5	8.38E-5
3	1806	6.04E-5	7.12E-5
4	7206	8.80E-5	6.58E-5
5	28806	9.07E-5	9.52E-5

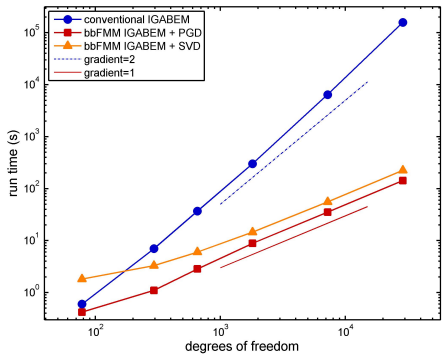


Figure 13. Time comparison

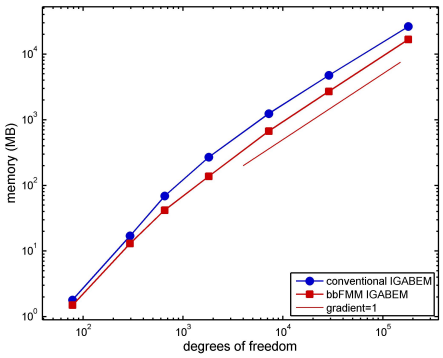


Figure 14. Memory comparison

error defined as:

$$E_r = \frac{\sum_{e=1}^{n_e} \int_{\Gamma_e} \sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}}) ||\tilde{\mathbf{u}}_{ea} - \tilde{\mathbf{u}}_{ea}^{ex}|| J_e(\tilde{\boldsymbol{\xi}}) \mathrm{d}\tilde{\boldsymbol{\xi}}}{\sum_{e=1}^{n_e} \int_{\Gamma_e} \sum_{a=1}^{n_a} R_{ea}(\tilde{\boldsymbol{\xi}}) ||\tilde{\mathbf{u}}_{ea}^{ex}|| J_e(\tilde{\boldsymbol{\xi}}) \mathrm{d}\tilde{\boldsymbol{\xi}}} \times 100\%, \tag{68}$$

where $\tilde{\mathbf{u}}_{ea}$ and $\tilde{\mathbf{u}}_{ea}^{ex}$ are the coefficients that allow us to recover the approximate and exact displacements, respectively, in a NURBS basis.

DOI: 10.1002/nme

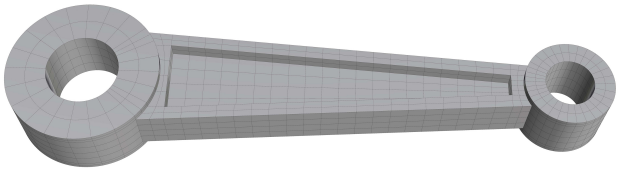


Figure 15. Torque arm geometry (control points omitted)

4.4. *Torque arm example*

A typical mechanical part in engineering, the torque arm, is selected as the second case study. The geometry is shown in Fig. 15, in which, for clarity, the control points have been omitted. The torque arm is subjected to a pressure of 100MPa prescribed on the surface of the circular hole on the right side, and the component is fixed at the hole on the left. Young's modulus and Poisson's ratio are 200 GPa and 0.3 respectively. The octree structure is generated from a 0.5m initial cube.

The model consists of 4794 DOF and 400 Bézier elements, and requires a run time for conventional IGABEM and bbFMM IGABEM of 2453s and 24s respectively. Fig. 16 shows contours of the deformation results.

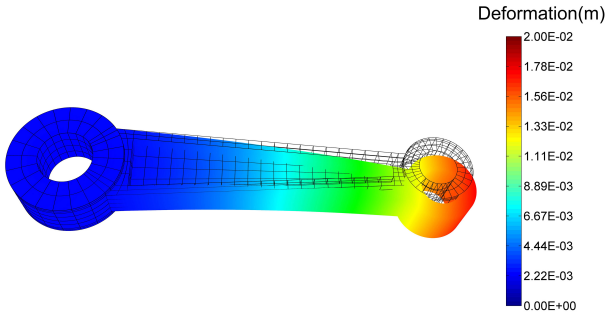


Figure 16. Deformation solution with undeformed patches of the torque arm model

4.5. *Propeller example*

The third model is defined as a propeller (Fig 17) to illustrate the ability of this method to handle complex engineering geometries. The entire geometry consists of 9972 control points and 2493 Bézier elements, giving 29916 DOF. A load of 100MPa is applied to all blade surfaces, and the interior cylindrical surface is constrained against displacement. The octree structure is generated from a 0.3m initial cube. Fig 18 shows the deformation result with undeformed patches. The solution times for the conventional IGABEM and bbFMM IGABEM simulations are 163641s and 151s respectively.

From the studies presented, it is evident that by using the bbFMM scheme, the IGABEM analysis can be considerably accelerated. Furthermore, the PGD algorithm has addressed the computational bottleneck in data compression that is seen with bbFMM schemes that rely upon SVD, and this makes the M2L operators and matrix-vector products much faster to compute than before.

5. PARAMETRIC STUDY

In order to evaluate the influence of the parameters used in this paper, a parametric study is performed by using the quarter cylinder geometry in Sec. 4.3 with the same boundary conditions.

DOI: 10.1002/nme

Figure 17. Propeller geometry (control points omitted)

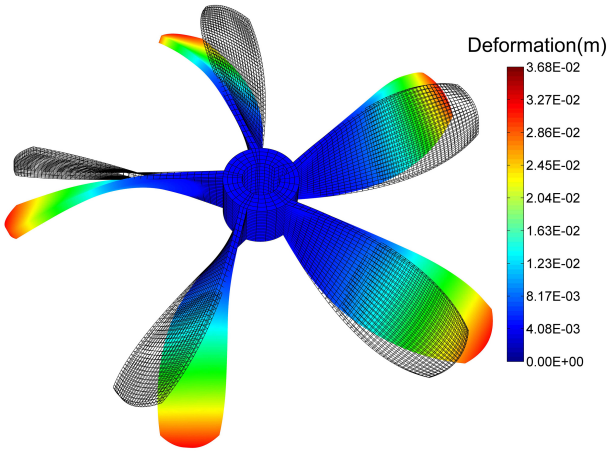


Figure 18. Deformation solution with undeformed patches of the propeller model

The study includes four main parameters: computational complexity with knot insertion (h), NURBS degree (p), number of Chebyshev points (n_c) and precision of PGD approximation ($\epsilon(n)$).

The first study is related to the scaling of elements of the run time under an h-refinement scheme. In an IGABEM context this is done by knot insertion, this was shown in Sec. 4.3 but here we show some extra details. The knot vector used for the initial geometry (Mesh 0) is $\Xi = \{0, 0, 0, 1, 1, 1\}$ for each dimension in parametric space. We then insert knots h times with p repetitions to arrive at $\Xi = \{0, 0, 0, \xi_1, \xi_1, \ldots, \xi_h, \xi_h, 1, 1, 1\}$, so that each surface is converted into $(h+1) \times (h+1)$ individual Bézier elements for calculation. This can be seen as an isogeometric form of h-refinement and the details are shown in Table II, where n_e is the number of Bézier elements, E_r the relative error computed by using Eq. (68), t_f the time cost of the far-field integration during the whole calculation process, t_d the time cost of the direct integration (which includes both the near-field and singular parts), t_p the PGD computational time during the pre-computing process, t_g the time cost of GMRES solver without matrix-vector products, t the total run time and n_g the number of GMRES iterations. From the table it is evident that the cost of the far-field and direct parts both exhibit a

DOI: 10.1002/nme

linear complexity, the time cost of PGD computation will be $O(1)$ as the source and field points occupy some relatively stable octree cells, the time cost of GMRES mainly includes two parts: the least squares problem ($O(n_g^2)$) and Arnoldi iteration ($O(n_{dof} \times n_g)$) which also exhibits an almost linear computational complexity with little increase in the number of iterations.

Table II. Parameter study: h-refinement

DOF	n_e	p	h	n_c	$\epsilon(n)$	E_r	t_f (s)	t_d (s)	t_p (s)	t_g (s)	t (s)	n_g
78	6	2	0	5	1.00E-5	2.34E-3	9.32E-2	1.43E-1	1.62E-1	3.41E-4	4.22E-1	11
294	24	2	1	5	1.00E-5	7.49E-4	3.66E-1	4.37E-1	3.15E-1	1.25E-3	1.17	11
654	54	2	2	5	1.00E-5	9.10E-5	9.99E-1	1.20	5.88E-1	3.86E-3	2.84	12
1806	150	2	4	5	1.00E-5	6.04E-5	3.53	3.90	1.07	1.28E-2	8.84	13
7206	600	2	9	5	1.00E-5	8.80E-5	1.51E1	1.48E1	0.99	5.85E-2	3.52E1	13
28806	2400	2	19	5	1.00E-5	9.07E-5	6.81E1	6.67E1	1.12	2.69E-1	1.42E2	13

The second study relates to degree elevation (the isogeometric form of p-refinement). The surface degree in each dimension is increased from 2 to 7 with the knot vector expressed as $\Xi = \{\overbrace{0,\ldots,0}^{p+1},\overbrace{1,\ldots,1}^{p+1}\}$. Table III shows the comparisons between different degrees, and it can be seen that the relative error will increase for large p and cost additional time during the local expansion, so the surface degrees $p = 2, 3, 4$ are recommended depending on the complexity of the geometry.

Table III. Parameter study: p-refinement

DOF	n_e	p	h	n_c	$\epsilon(n)$	E_r	t_f (s)	t_d (s)	t_p (s)	t_g (s)	t (s)	n_g
78	6	2	0	5	1.00E-5	2.34E-3	9.32E-2	1.43E-1	1.62E-1	3.41E-4	4.22E-1	11
168	6	3	0	5	1.00E-5	7.95E-4	1.53E-1	1.63E-1	1.89E-1	6.80E-4	5.31E-1	11
294	6	4	0	5	1.00E-5	1.41E-3	2.44E-1	2.48E-1	2.11E-1	1.13E-3	7.44E-1	11
456	6	5	0	5	1.00E-5	6.33E-3	3.52E-1	3.46E-1	2.31E-1	1.82E-3	9.71E-1	12
654	6	6	0	5	1.00E-5	6.42E-3	4.83E-1	5.13E-1	2.29E-1	2.75E-3	1.33	12
888	6	7	0	5	1.00E-5	9.62E-3	6.88E-1	7.24E-1	2.63E-1	4.13E-3	1.98	12

The third study investigates the influence of the number of Chebyshev nodes, n_c, used (Table IV). One can find that use of an insufficient number of Chebyshev nodes will lead to a degradation in accuracy in the final result, although it will incur a reduced time cost in the far-field, direct and PGD parts. The convergence speed will also degrade. For more Chebyshev nodes, the computation will obtain a good convergence property but cost more time so that use of $n_c = 5, 6$ is recommended.

The last study relates to the influence of the precision of the PGD (Tab. V). A low level compression will make the far-field computation fast for each iteration but with poor precision and convergence properties, and a high level compression will cause more time to be spent during each matrix-vector calculation. The results suggest a tolerance $\epsilon(n) = 10^{-5} \sim 10^{-6}$ is suitable.

6. CONCLUSION

Because of its boundary-only modelling, with no requirement for a volumetric NURBS geometric definition, the boundary element method is an ideal choice for isogeometric analysis of solids in

DOI: 10.1002/nme

Table IV. Parameter study: number of Chebyshev nodes

DOF	n_e	p	h	n_c	$\epsilon(n)$	E_r	t_f (s)	t_d (s)	t_p (s)	t_g (s)	t (s)	n_g
78	6	2	0	2	1.00E-5	1.01E-2	1.51E-1	2.39E-1	3.73E-3	5.77E-4	4.18E-1	18
78	6	2	0	3	1.00E-5	5.41E-3	1.09E-1	1.72E-1	1.38E-2	4.44E-4	3.16E-1	13
78	6	2	0	4	1.00E-5	3.14E-3	9.27E-2	1.44E-1	5.33E-2	3.43E-4	3.14E-1	11
78	6	2	0	5	1.00E-5	2.34E-3	9.32E-2	1.43E-1	1.62E-1	3.41E-4	4.22E-1	11
78	6	2	0	6	1.00E-5	2.31E-3	1.11E-1	1.31E-1	3.28E-1	3.12E-4	5.92E-1	10
78	6	2	0	7	1.00E-5	2.31E-3	1.39E-1	1.33E-1	7.25E-1	3.14E-4	1.03	10

Table V. Parameter study: precision of PGD

DOF	n_e	p	h	n_c	$\epsilon(n)$	E_r	t_f (s)	t_d (s)	t_p (s)	t_g (s)	t (s)	n_g
78	6	2	0	5	1.00E-2	8.11E-2	1.13E-1	2.06E-1	9.72E-2	4.94E-4	4.39E-1	16
78	6	2	0	5	1.00E-3	1.57E-2	1.19E-1	1.94E-1	1.16E-1	4.67E-4	4.53E-1	15
78	6	2	0	5	1.00E-4	5.19E-3	9.51E-2	1.57E-1	1.41E-1	3.70E-4	4.16E-1	12
78	6	2	0	5	1.00E-5	2.34E-3	9.32E-2	1.43E-1	1.62E-1	3.41E-4	4.22E-1	11
78	6	2	0	5	1.00E-6	2.14E-3	9.72E-2	1.42E-1	3.23E-1	3.42E-4	5.85E-1	11
78	6	2	0	5	1.00E-7	2.26E-3	9.55E-2	1.28E-1	6.77E-1	3.10E-4	9.21E-1	10

3-D. In this paper, we have presented a black-box fast multipole isogeometric boundary element analysis for the analysis of elasticity in 3-D solid components. The algorithm is combined with a model reduction approach based on the use of proper generalized decomposition to accelerate the computation of the M2L operators and matrix-vector products. When PGD is used in preference to SVD performance gains are seen across the range of model sizes tested, and in the worst case this gain is approximately a factor of 2. Together, these strategies provide a promising computational approach for the fast analysis of large-scale elastostatics problems. The computational resources required are decreased without significant loss of accuracy. As this method can be easily integrated into conventional IGABEM codes, it will be a good choice for future application in commercial software for wide dissemination to the industrial community.

REFERENCES

1. Hughes T, Cottrell J, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.* oct 2005; **194**(39-41):4135–4195, doi:10.1016/j.cma.2004.10.008. URL http://dx.doi.org/10.1016/j.cma.2004.10.008.
2. Schillinger D, Dedè L, Scott MA, Evans JA, Borden MJ, Rank E, Hughes TJ. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and t-spline CAD surfaces. *Comput. Methods Appl. Mech. Engrg.* dec 2012; **249-252**:116–150, doi:10.1016/j.cma.2012.03.017. URL http://dx.doi.org/10.1016/j.cma.2012.03.017.
3. Bazilevs Y, Calo VM, Hughes TJR, Zhang Y. Isogeometric fluid-structure interaction: theory, algorithms, and computations. *Comput. Mech.* aug 2008; **43**(1):3–37, doi:10.1007/s00466-008-0315-x. URL http://dx.doi.org/10.1007/s00466-008-0315-x.
4. Bazilevs Y, Hsu MC, Scott M. Isogeometric fluid–structure interaction analysis with emphasis on non-matching discretizations, and with application to wind turbines. *Comput. Methods Appl. Mech. Engrg.* dec 2012; **249-252**:28–41, doi:10.1016/j.cma.2012.03.028. URL http://dx.doi.org/10.1016/j.cma.2012.03.028.
5. Buffa A, Sangalli G, Vázquez R. Isogeometric analysis in electromagnetics: B-splines approximation. *Comput. Methods Appl. Mech. Engrg.* mar 2010; **199**(17-20):1143–1152, doi:10.1016/j.cma.2009.12.002. URL http://dx.doi.org/10.1016/j.cma.2009.12.002.

6. Cottrell J, Reali A, Bazilevs Y, Hughes T. Isogeometric analysis of structural vibrations. *Comput. Methods Appl. Mech. Engrg.* aug 2006; **195**(41-43):5257–5296, doi:10.1016/j.cma.2005.09.027. URL http://dx.doi.org/10.1016/j.cma.2005.09.027.
7. Simpson R, Scott M, Taus M, Thomas D, Lian H. Acoustic isogeometric boundary element analysis. *Comput. Methods Appl. Mech. Engrg.* feb 2014; **269**:265–290, doi:10.1016/j.cma.2013.10.026. URL http://dx.doi.org/10.1016/j.cma.2013.10.026.
8. Peake M, Trevelyan J, Coates G. Extended isogeometric boundary element method (XIBEM) for three-dimensional medium-wave acoustic scattering problems. *Comput. Methods Appl. Mech. Engrg.* feb 2015; **284**:762–780, doi:10.1016/j.cma.2014.10.039. URL http://dx.doi.org/10.1016/j.cma.2014.10.039.
9. Simpson R, Liu Z. Acceleration of isogeometric boundary element analysis through a black-box fast multipole method. *Eng. Anal. Boundary Elem.* may 2016; **66**:168–182, doi:10.1016/j.enganabound.2016.03.004. URL http://dx.doi.org/10.1016/j.enganabound.2016.03.004.
10. Piegl L, Tiller W. *The NURBS Book*. Springer Berlin Heidelberg, 1995, doi:10.1007/978-3-642-97385-7. URL http://dx.doi.org/10.1007/978-3-642-97385-7.
11. Bazilevs Y, Calo V, Cottrell J, Evans J, Hughes T, Lipton S, Scott M, Sederberg T. Isogeometric analysis using t-splines. *Comput. Methods Appl. Mech. Engrg.* jan 2010; **199**(5-8):229–263, doi:10.1016/j.cma.2009.02.036. URL http://dx.doi.org/10.1016/j.cma.2009.02.036.
12. Nguyen-Thanh N, Kiendl J, Nguyen-Xuan H, Wchner R, Bletzinger K, Bazilevs Y, Rabczuk T. Rotation free isogeometric thin shell analysis using PHT-splines. *Comput. Methods Appl. Mech. Engrg.* nov 2011; **200**(47-48):3410–3424, doi:10.1016/j.cma.2011.08.014. URL http://dx.doi.org/10.1016/j.cma.2011.08.014.
13. Nguyen-Thanh N, Nguyen-Xuan H, Bordas S, Rabczuk T. Isogeometric analysis using polynomial splines over hierarchical t-meshes for two-dimensional elastic solids. *Comput. Methods Appl. Mech. Engrg.* may 2011; **200**(21-22):1892–1908, doi:10.1016/j.cma.2011.01.018. URL http://dx.doi.org/10.1016/j.cma.2011.01.018.
14. Wang P, Xu J, Deng J, Chen F. Adaptive isogeometric analysis using rational PHT-splines. *Comput.-Aided Des.* nov 2011; **43**(11):1438–1448, doi:10.1016/j.cad.2011.08.026. URL http://dx.doi.org/10.1016/j.cad.2011.08.026.
15. Cruse T. Numerical solutions in three dimensional elastostatics. *Int. J. Solids Struct.* dec 1969; **5**(12):1259–1274, doi:10.1016/0020-7683(69)90071-7. URL http://dx.doi.org/10.1016/0020-7683(69)90071-7.
16. Arnold DN, Wendland WL. On the asymptotic convergence of collocation methods. *Math. Comput.* 1983; **41**(164):349–381. URL https://www.ima.umn.edu/~arnold/papers/collocation.pdf.
17. Arnold DN, Saranen J. On the asymptotic convergence of spline collocation methods for partial differential equations. *SIAM J. Numer. Anal.* jun 1984; **21**(3):459–472, doi:10.1137/0721034. URL http://dx.doi.org/10.1137/0721034.
18. Ligget JA, Salmon JR. Cubic spline boundary elements. *Int. J. Numer. Methods Eng.* apr 1981; **17**(4):543–556, doi:10.1002/nme.1620170405. URL http://dx.doi.org/10.1002/nme.1620170405.
19. Yu M, Kuffel E. Spline element for boundary element method. *IEEE Trans. Magn.* 1994; **30**(5):2905–2907, doi:10.1109/20.312544. URL http://dx.doi.org/10.1109/20.312544.
20. Turco E, Aristodemo M. A three-dimensional b-spline boundary element. *Comput. Methods Appl. Mech. Engrg.* mar 1998; **155**(1-2):119–128, doi:10.1016/S0045-7825(97)00147-3. URL http://dx.doi.org/10.1016/S0045-7825(97)00147-3.
21. Rivas F, Valle L, Cátedra M. A moment method formulation for the analysis of wire antennas attached to arbitrary conducting bodies defined by parametric surfaces. *Appl. Comput. Electrom.* 1996; **11**:32–39. URL http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA311663#page=35.
22. Simpson R, Bordas S, Trevelyan J, Rabczuk T. A two-dimensional isogeometric boundary element method for elastostatic analysis. *Comput. Methods Appl. Mech. Engrg.* feb 2012; **209-212**:87–100, doi:10.1016/j.cma.2011.08.008. URL http://dx.doi.org/10.1016/j.cma.2011.08.008.
23. Scott M, Simpson R, Evans J, Lipton S, Bordas S, Hughes T, Sederberg T. Isogeometric boundary element analysis using unstructured t-splines. *Comput. Methods Appl. Mech. Engrg.* feb 2013; **254**:197–221, doi:10.1016/j.cma.2012.11.001. URL http://dx.doi.org/10.1016/j.cma.2012.11.001.
24. Wang Y, Benson D. Multi-patch nonsingular isogeometric boundary element analysis in 3D. *Comput. Methods Appl. Mech. Engrg.* aug 2015; **293**:71–91, doi:10.1016/j.cma.2015.03.016. URL http://dx.doi.org/10.1016/j.cma.2015.03.016.
25. Li K, Qian X. Isogeometric analysis and shape optimization via boundary integral. *Comput.-Aided Des.* nov 2011; **43**(11):1427–1437, doi:10.1016/j.cad.2011.08.031. URL http://dx.doi.org/10.1016/j.cad.2011.08.031.
26. Lian H, Kerfriden P, Bordas SPA. Implementation of regularized isogeometric boundary element methods for gradient-based shape optimization in two-dimensional linear elasticity. *Int. J. Numer. Methods Eng.* apr 2016; **106**(12):972–1017, doi:10.1002/nme.5149. URL http://dx.doi.org/10.1002/nme.5149.
27. Peake M, Trevelyan J, Coates G. Extended isogeometric boundary element method (XIBEM) for two-dimensional helmholtz problems. *Comput. Methods Appl. Mech. Engrg.* jun 2013; **259**:93–102, doi:10.1016/j.cma.2013.03.016. URL http://dx.doi.org/10.1016/j.cma.2013.03.016.
28. Peng X, Atroshchenko E, Kerfriden P, Bordas S. Isogeometric boundary element methods for three dimensional static fracture and fatigue crack growth. *Comput. Methods Appl. Mech. Engrg.* jun 2016; doi:10.1016/j.cma.2016.05.038. URL http://dx.doi.org/10.1016/j.cma.2016.05.038.
29. Bonnet M, Maier G, Polizzotto C. Symmetric galerkin boundary element methods. *Applied Mechanics Reviews* 1998; **51**(11):669, doi:10.1115/1.3098983.
30. Hiptmair R. Operator preconditioning. *Computers & Mathematics with Applications* sep 2006; **52**(5):699–706, doi:10.1016/j.camwa.2006.10.008.

31. Rokhlin V. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.* sep 1985; **60**(2):187–207, doi:10.1016/0021-9991(85)90002-6. URL http://dx.doi.org/10.1016/0021-9991(85)90002-6.
32. Peirce AP, Napier JAL. A spectral multipole method for efficient solution of large-scale boundary element models in elastostatics. *Int. J. Numer. Methods Eng.* dec 1995; **38**(23):4009–4034, doi:10.1002/nme.1620382307. URL http://dx.doi.org/10.1002/nme.1620382307.
33. Liu Y, Nishimura N. The fast multipole boundary element method for potential problems: A tutorial. *Eng. Anal. Boundary Elem.* may 2006; **30**(5):371–381, doi:10.1016/j.enganabound.2005.11.006. URL http://dx.doi.org/10.1016/j.enganabound.2005.11.006.
34. Hackbusch W. A sparse matrix arithmetic based on $\cal h$ -matrices. part i: Introduction to ${\cal h}$ -matrices. *Comput.* apr 1999; **62**(2):89–108, doi:10.1007/s006070050015. URL http://dx.doi.org/10.1007/s006070050015.
35. Bebendorf M, Kriemann R. Fast parallel solution of boundary integral equations and related problems. *Comput. Visual Sci.* nov 2005; **8**(3-4):121–135, doi:10.1007/s00791-005-0001-x. URL http://dx.doi.org/10.1007/s00791-005-0001-x.
36. Marussig B, Zechner J, Beer G, Fries TP. Fast isogeometric boundary element method based on independent field approximation. *Comput. Methods Appl. Mech. Engrg.* feb 2015; **284**:458–488, doi:10.1016/j.cma.2014.09.035. URL http://dx.doi.org/10.1016/j.cma.2014.09.035.
37. Phillips J, White J. A precorrected-FFT method for electrostatic analysis of complicated 3-D structures. *IEEE Trans. Comput.-Aided Des.* 1997; **16**(10):1059–1072, doi:10.1109/43.662670. URL http://dx.doi.org/10.1109/43.662670.
38. Fata SN. Fast galerkin BEM for 3D-potential theory. *Comput. Mech.* feb 2008; **42**(3):417–429, doi:10.1007/s00466-008-0251-9. URL http://dx.doi.org/10.1007/s00466-008-0251-9.
39. Yan Z, Zhang J, Ye W. Rapid solution of 3-D oscillatory elastodynamics using the pFFT accelerated BEM. *Eng. Anal. Boundary Elem.* nov 2010; **34**(11):956–962, doi:10.1016/j.enganabound.2010.06.008. URL http://dx.doi.org/10.1016/j.enganabound.2010.06.008.
40. Dutt A, Gu M, Rokhlin V. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM Journal on Numerical Analysis* oct 1996; **33**(5):1689–1711, doi:10.1137/0733082.
41. Gimbutas Z, Rokhlin V. A generalized fast multipole method for nonoscillatory kernels. *SIAM Journal on Scientific Computing* jan 2003; **24**(3):796–817, doi:10.1137/S1064827500381148.
42. Ying L, Biros G, Zorin D. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics* may 2004; **196**(2):591–626, doi:10.1016/j.jcp.2003.11.021.
43. Fong W, Darve E. The black-box fast multipole method. *J. Comput. Phys.* dec 2009; **228**(23):8712–8725, doi:10.1016/j.jcp.2009.08.031. URL http://dx.doi.org/10.1016/j.jcp.2009.08.031.
44. Alpert B, Beylkin G, Coifman R, Rokhlin V. Wavelet-like bases for the fast solution of second-kind integral equations. *SIAM Journal on Scientific Computing* jan 1993; **14**(1):159–184, doi:10.1137/0914010.
45. Dahmen W, Harbrecht H, Schneider R. Compression techniques for boundary integral equations—asymptotically optimal complexity estimates. *SIAM Journal on Numerical Analysis* jan 2006; **43**(6):2251–2271, doi:10.1137/s0036142903428852.
46. Rodriguez JL, Taboada JM, Araujo MG, Basteiro FO, Landesa L, Garcia-Tunon I. On the use of the singular value decomposition in the fast multipole method. *IEEE Transactions on Antennas and Propagation* aug 2008; **56**(8):2325–2334, doi:10.1109/tap.2008.926761.
47. Suda R, Kuriyama S. Another preprocessing algorithm for generalized one-dimensional fast multipole method. *Journal of Computational Physics* apr 2004; **195**(2):790–803, doi:10.1016/j.jcp.2003.10.018.
48. Cheng H, Gimbutas Z, Martinsson PG, Rokhlin V. On the compression of low rank matrices. *SIAM Journal on Scientific Computing* jan 2005; **26**(4):1389–1404, doi:10.1137/030602678.
49. Martinsson PG, Rokhlin V. An accelerated kernel-independent fast multipole method in one dimension. *SIAM Journal on Scientific Computing* jan 2007; **29**(3):1160–1178, doi:10.1137/060662253.
50. Messner M, Schanz M, Darve E. Fast directional multilevel summation for oscillatory kernels based on chebyshev interpolation. *Journal of Computational Physics* feb 2012; **231**(4):1175–1196, doi:10.1016/j.jcp.2011.09.027.
51. Messner M, Bramas B, Coulaud O, Darve E. Optimized M2L kernels for the chebyshev interpolation based fast multipole method. *CoRR* 2012; **abs/1210.7292**. URL http://arxiv.org/abs/1210.7292.
52. March WB, Biros G. Far-field compression for fast kernel summation methods in high dimensions. *Applied and Computational Harmonic Analysis* jul 2017; **43**(1):39–75, doi:10.1016/j.acha.2015.09.007.
53. Chinesta F, Ammar A, Leygue A, Keunings R. An overview of the proper generalized decomposition with applications in computational rheology. *J. Non-Newtonian Fluid Mech.* jun 2011; **166**(11):578–592, doi:10.1016/j.jnnfm.2010.12.012. URL http://dx.doi.org/10.1016/j.jnnfm.2010.12.012.
54. Chinesta F, Keunings R, Leygue A. *The Proper Generalized Decomposition for Advanced Numerical Simulations*. Springer International Publishing, 2014, doi:10.1007/978-3-319-02865-1. URL http://dx.doi.org/10.1007/978-3-319-02865-1.
55. COX MG. The numerical evaluation of b -splines. *IMA J. Appl. Math.* 1972; **10**(2):134–149, doi:10.1093/imamat/10.2.134. URL http://dx.doi.org/10.1093/imamat/10.2.134.
56. de Boor C. On calculating with b-splines. *J. Approx. Theory* jul 1972; **6**(1):50–62, doi:10.1016/0021-9045(72)90080-9. URL http://dx.doi.org/10.1016/0021-9045(72)90080-9.
57. Borden MJ, Scott MA, Evans JA, Hughes TJR. Isogeometric finite element data structures based on bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering* aug 2010; **87**(1-5):15–47, doi:10.1002/nme.2968.
58. Greville TNE. Numerical procedures for interpolation by spline functions. *SIAM J. Numer. Anal.* jan 1964; **1**(1):53–68, doi:10.1137/0701005. URL http://dx.doi.org/10.1137/0701005.
59. Hayami K. *A Projection Transformation Method for Nearly Singular Surface Boundary Element Integrals*. Springer Berlin Heidelberg, 1992, doi:10.1007/978-3-642-84698-4. URL http://dx.doi.org/10.1007/

978-3-642-84698-4.
60. Liu YJ, Rudolphi TJ. New identities for fundamental solutions and their applications to non-singular boundary element formulations. *Comput. Mech.* oct 1999; **24**(4):286–292, doi:10.1007/s004660050517. URL http://dx.doi.org/10.1007/s004660050517.