# Accelerated Galerkin BEM for 3D Potential Theory

Sylvain Nintcheu Fata

***Abstract*— This paper is concerned with the development of a fast spectral method for solving boundary integral equations in three-dimensional potential theory. Upon discretizing the underlying boundary integrals via a Galerkin approximation, the proposed method overlays the problem domain with a regular Cartesian grid that serves as an auxiliary platform for computation. With the aid of the Fast Fourier Transform, the necessary influence matrices of the discretized problem are rapidly evaluated on the regular grid in a sparse manner. Unlike traditional techniques dealing with boundary integrals, the sparse representation of the featured coefficient matrices results in a significant reduction in computer memory requirements. The computational cost associated with the sparse approximation of influence matrices is asymptotically lower than that of conventional methods. For a numerical solution of the resulting linear system, a Krylov-subspace (e.g. BiCGSTAB) iterative method is further employed wherein the sparse influences are used to rapidly compute the matrix-vector products involved at each iteration. Several key features of the formulation, including the mapping of density distributions onto the regular Cartesian grid, are highlighted. Numerical experiments are presented to illustrate the performance of the spectral method. The proposed approach will find application in areas involving large simulations with complex and moving boundaries.**

***Index Terms*— Boundary integral method, regular grid method, fast Fourier transform, fast algorithm, potential theory.**

## I. Introduction

THE mathematical modeling of many engineering problems often involves the solution of boundary integral equations. A numerical technique for producing discretized boundary integral equations is known as the Boundary Element Method (BEM) [1], [2]. Boundary element techniques are highly accurate and, in comparison to domain methods, the inherent surface-only discretization can be very advantageous. For instance, the Boundary Integral Equation (BIE) approach is the preferred modeling choice for practical problems involving unbounded media, cracks, and moving or unknown boundaries. However, BIE methods typically produce linear systems that contain fully-populated influence matrices. For practical problems demanding a fine discretization to cope with the details of surface structures, the solution time and storage requirement of the dense linear systems become prohibitive.

Over the past decade, several sparsification techniques, Fast Multipole Method (FMM) [3], [4], fast Fourier transform methods [5]–[7], Regular Grid Method (RGM) [8], wavelet-based discretization method [9], [10], have been developed to accelerate the solution of BIE for large scale problems. These acceleration methods are based on some type of fast summation technique which consists of (i) grouping boundary influences into near-field and far-field, and (ii) using special techniques to treat short and long range influences separately. In this decoupled process, long-range influences are typically approximated by means of a suitable, less expensive approach rather than calculated directly.

A fast spectral algorithm presented herein follows the lines of the Precorrected-FFT (PFFT) developed in [5]. It revolves around the use of an auxiliary regular Cartesian grid containing the discretized boundary, and the Fast Fourier Transform (FFT) to rapidly approximate far-field influences. The calculation via the homogeneous grid incorporates inaccurate near-field contributions that are removed and replaced (i.e., precorrected) by explicit pre-computed short range influences. This procedure leads to a sparse representation of boundary influence matrices which, in turn, results in a significant reduction in memory requirements.

Although most numerical treatments of boundary integrals by the PFFT method deal with single-layer potentials [5], [11] and piecewise constant surface interpolations [12], a study utilizing a double-layer potential has been presented in [13] wherein the discretized solution could not be obtained at nodal points on the surface mesh but only at integration points on boundary elements. Moreover, so far, there have not been any attempts to systematically extend the PFFT methodology to the solution of direct BIEs.

Aimed at bridging this gap, a generalization of the PFFT for solving direct BIEs, i.e., involving single- and double-layer potentials, and arbitrarily varying local interpolation functions is the focus of this study. To this end, a Galerkin approach [14] is employed to discretize the boundary integral equations. By means of a well-defined homogeneous Cartesian grid, the fast Fourier transform and local surface-to-grid interpolations, the necessary influence matrices of the discretized problem are rapidly computed in a sparse manner. In the decoupling process, a computational savings is gained through the use of the FFT algorithm, and a substantially less memory requirement is also a consequence of the sparsification. To efficiently resolve the linear system, the sparse representation of influences is used in a Krylov-subspace iterative solver such as GMRES [15], [16] or BiCGSTAB [17] to compute the matrix-vector multiplications featured at every iteration of the solution process.

Details of the computational treatment, including the generation of the sparse interpolation operators which map boundary data onto the regular grid and back to the surface, are elucidated. Numerical examples are included to illustrate the performance of the proposed method.

The U.S. Government retains a nonexclusive royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

S. Nintcheu Fata is in Computer Science and Mathematics Division, Oak Ridge National Laboratory, P.O. Box 2008, MS 6367, Oak Ridge, TN 37831-6367, USA. Email: nintcheufats@ornl.gov

## II. Problem formulation

Of interest in this study is the numerical treatment of a three-dimensional boundary integral equation for the Laplace equation in a domain $D \subset \mathbb{R}^3$ with boundary $S$ via a Galerkin approximation. With reference to a Cartesian frame $\{\mathbf{0}; x_1, x_2, x_3\}$, consider the boundary-value problem for the potential function $u(\boldsymbol{x})$ $(\boldsymbol{x} \in D)$ satisfying

$$\nabla^2 u = 0 \tag{1}$$

in the domain $D$. It is further assumed that $u$ satisfies either a Dirichlet, Neumann or Mixed boundary condition on $S$. By use of Green's theorem [2], [18], it can be shown that a solution $u$ to (1) admits the representation

$$\int_S G(\boldsymbol{x}, \boldsymbol{y})\, t(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} - \int_S \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, u(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} = \begin{cases} u(\boldsymbol{x}), & \boldsymbol{x} \in D \\ 0, & \boldsymbol{x} \in \mathbb{R}^3 \setminus \bar{D} \end{cases}, \tag{2}$$

where $G$ is the free space fundamental solution of the Laplace equation expressed as

$$G(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{4\pi \|\boldsymbol{x} - \boldsymbol{y}\|}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3, \quad \boldsymbol{x} \neq \boldsymbol{y}. \tag{3}$$

In addition, $t = \boldsymbol{n} \cdot \nabla u$ denotes the flux associated with the potential $u$, and $\boldsymbol{H}$ is the gradient of $G$ given by

$$\boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{4\pi} \frac{\boldsymbol{x} - \boldsymbol{y}}{\|\boldsymbol{x} - \boldsymbol{y}\|^3}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3, \quad \boldsymbol{x} \neq \boldsymbol{y}, \tag{4}$$

with $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{y})$ denoting the unit normal to $S$ directed towards the exterior of $D$. It is important to mention that the numerical analysis presented below is applicable regardless of the types of boundary conditions, i.e., Dirichlet, Neumann or Mixed boundary conditions.

Herein, the boundary integral equation to be solved is understood in the sense of a *limit to the boundary* ( [14], [19]). This approach enables writing the same equation for points either inside/outside the domain $D$ or on the boundary $S$. In what follows, let $\mathbb{R}^3 \setminus \bar{D} \ni \boldsymbol{x}_\varepsilon = \boldsymbol{x} + \varepsilon \boldsymbol{n}(\boldsymbol{x})$, $\boldsymbol{x} \in S$, $\varepsilon > 0$, $\boldsymbol{n}(\boldsymbol{x})$ is the unit outward normal to $S$ at $\boldsymbol{x}$. The singular BIE

$$\lim_{\varepsilon \to 0} \left( \int_S G(\boldsymbol{x}_\varepsilon, \boldsymbol{y})\, t(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} - \int_S \boldsymbol{H}(\boldsymbol{x}_\varepsilon, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, u(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \right) = 0, \tag{5}$$

is to be solved with the singular integrals calculated for $\boldsymbol{x}_\varepsilon$ approaching the boundary $S$ from outside the domain $D$.

### A. Galerkin approximation

To accomplish the numerical solution of (5) associated with a boundary value problem, it is common practice to (i) partition $S$ into non-overlapping surface patches called boundary elements, each of which is characterized by its nodes, and (ii) to approximate the boundary potential $u(\boldsymbol{y})$ and flux $t(\boldsymbol{y})$ $(\boldsymbol{y} \in S)$ in terms of respective nodal values and basis shape functions $\psi_j$ at discrete points $\boldsymbol{y}^j$ on $S$ as

$$u(\boldsymbol{y}) = \sum_j u(\boldsymbol{y}^j)\, \psi_j(\boldsymbol{y}), \quad t(\boldsymbol{y}) = \sum_j t(\boldsymbol{y}^j)\, \psi_j(\boldsymbol{y}). \tag{6}$$

In practice, boundary elements on $S$ are often constructed as triangular or quadrilateral surface patches (see Fig. 1), and shape functions are selected as constant, linear or higher order polynomials over a boundary element.

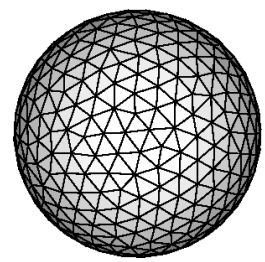

Fig. 1. Triangulation of a unit sphere featuring 796 triangles with 400 nodes.

With the above definition, the Galerkin approach for solving (5) rests on a weighted-residual statement, wherein the interpolators $\psi_i$ serve as the weighting functions in an error argument as

$$\lim_{\varepsilon \to 0} \int_S \psi_i(\boldsymbol{x}) \left\{ \int_S G(\boldsymbol{x}_\varepsilon, \boldsymbol{y})\, t(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} - \int_S \boldsymbol{H}(\boldsymbol{x}_\varepsilon, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, u(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \right\} \mathrm{d}s_{\boldsymbol{x}} = 0. \tag{7}$$

The use of the interpolated approximations (6) in (7) leads to a dense linear system of algebraic equations for boundary potential $u$ and flux $t$

$$\mathbf{G}\{t\} = \mathbf{H}\{u\}, \tag{8}$$

where $\{u\}$ and $\{t\}$ are vectors containing nodal potentials $u(\boldsymbol{y}^j)$ and fluxes $t(\boldsymbol{y}^j)$ respectively; components of influence matrices $\mathbf{G}$ and $\mathbf{H}$ take the form

$$\mathrm{G}_{ij} = \lim_{\varepsilon \to 0} \int_S \int_S \psi_i(\boldsymbol{x})\, G(\boldsymbol{x}_\varepsilon, \boldsymbol{y})\, \psi_j(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \mathrm{d}s_{\boldsymbol{x}},$$

$$\mathrm{H}_{ij} = \lim_{\varepsilon \to 0} \int_S \int_S \psi_i(\boldsymbol{x})\, \boldsymbol{H}(\boldsymbol{x}_\varepsilon, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, \psi_j(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \mathrm{d}s_{\boldsymbol{x}}. \tag{9}$$

Following the standard approach for solving (8), the double integrations in (9) are explicitly evaluated. In doing so, appropriate techniques are used to treat singular integrals [14]. As a result, the influence matrices $\mathbf{G}$ and $\mathbf{H}$ are formed and stored in the computer memory. Upon specifying the boundary conditions of the boundary-value problem dealing with (1), the linear system (8) can be recast as

$$\mathbf{A}\{z\} = \{b\}, \tag{10}$$

where all unknown quantities on $S$ have been collected in $\{z\}$, and $\{b\}$ is a vector whose entries are obtained from known boundary data.

The dense linear system (10) can be solved in a conventional manner by the LU decomposition scheme. Unfortunately, the LU or Gaussian elimination method becomes prohibitive when the order of the matrix $\mathbf{A}$, $N$, is "large" since it requires $O(N^3)$ arithmetic operations and $O(N^2)$ memory storage. In this situation, Krylov-subspace iterative solvers such as GMRES [15], [16] or BiCGSTAB [17] are preferable as they

demand only $O(N^2)$ operations per iteration. This latter number of operations directly stemming from the dense matrix-vector multiplications in each iteration is still rather expensive for large scale problems. The goal of this study is to reduce the number of operations per iteration and the computer memory needed to evaluate the matrix-vector products featured in (10) when $N$ is very large. To this end, a homogeneous grid method for computing a matrix-vector product called the *precorrected-FFT* is generalized.

## III. Precorrected-FFT for singular BIE

Originally introduced in [5] for the 3D analysis of the Laplace equation employing only single-layer potentials, the Precorrected-FFT (PFFT) technique is an algorithm for rapid computation of a dense matrix-vector multiplication associated with discretized integral equations. Related work in 2D elasticity, utilizing an FFT and multipole type Green's function expansions, was first considered in [6].

The underlying idea for acceleration in the PFFT revolves around the fact that (i) integrals featured in (9), when evaluated over boundary elements, can be decomposed into near-field and far-field parts, and (ii) the far-field part can be approximated accurately and efficiently on a regular grid by use of the fast Fourier transform. On the other hand, the near-field part of (9) that includes not only all singular and weakly-singular integrals is computed in a conventional manner, i.e., by numerically performing the double integrations in which suitable limiting strategies are carried out to deal with singular integrals (see [14]). Unlike the popular fast multipole method [4], the PFFT algorithm can be easily implemented for all kernels of convolution type, i.e., kernels that depend only on the relative position $(\boldsymbol{x}-\boldsymbol{y})$ between the source point $\boldsymbol{x}$ and the receiver point $\boldsymbol{y}$.

In this study, a new version of the PFFT method capable of dealing with (i) single-layer and double-layer potentials, and (ii) arbitrary shape functions, will be described in detail. In addition, a general mathematical framework for defining the interpolation operators to/from the regular grid is also presented. The new fast spectral method, also referred to as precorrected-FFT technique, will be used to rapidly compute the matrix-vector multiplications $\mathbf{G}\{t\}$ and $\mathbf{H}\{u\}$ involved in the iterative solution of (10) without explicitly generating $\mathbf{G}$ and $\mathbf{H}$. For large problems, this procedure is highly desirable as a direct formation of the coefficient matrix $\mathbf{A}$ is avoided.

To effectively deal with Galerkin boundary integrals such as those featured in (9), it is important to employ an auxiliary parallelepiped containing the discretized boundary $S$ of the domain $D$ as shown in Fig. 2. With reference to the figure, the computational box is partitioned into $m \times m \times m$ cubes termed cells, where $m$ is the number cells in each coordinate direction. With this subdivision, it is assumed that every cell is formed with $p \times p \times p$ grid points, where $p$ is the number of points per cell in each coordinate direction. This cell-to-cell discretization creates a uniform grid throughout the computational parallelepiped. Next, boundary elements on $S$ are sorted (without repetition) into the computational cells. A cell containing boundary elements is called non-empty. Now, define by $\mathcal{M}$ and $M$, respectively, the total number of cells and non-empty cells in the computational box. To facilitate the ensuing analysis, let $S_k$ be the union of all boundary elements in a non-empty cell $k$. With this definition, it is clear that $S_k$ may protrude out of the $k$-th cell. Also, denote by $N_{E_k}$ the number of boundary elements on $S_k$, and let $N_k$ be the number of boundary nodes on $S_k$.

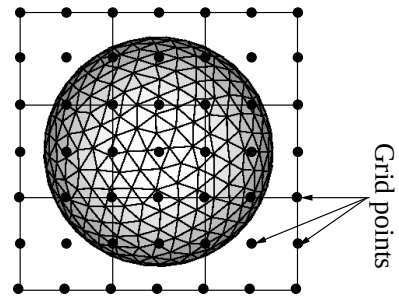


Fig. 2. Side view of a uniform Cartesian grid in the computational box with 27 cells ($m = 3$) and 27 grid points ($p = 3$) per cell.

### A. Regular grid approximation

To describe the grid approximation of the *far-field* part of the integrals in (9), mostly involving interaction with *distant* boundary elements, one can assume without loss of generality that the convolution-type kernels $G$ and $\boldsymbol{H}$ are non-singular, well-behaved and bounded in the auxiliary computational cube. In view of these assumptions, $\boldsymbol{x}_\varepsilon$ can be safely replaced by $\boldsymbol{x}$ in the limit in (9). With reference to (9b), one can write an approximation to $\mathrm{H}_{ij}$ as

$$\widetilde{\mathrm{H}}_{ij} = \sum_{k=1}^{\mathcal{M}} \sum_{l=1}^{\mathcal{M}} \widetilde{\mathrm{H}}_{ij}^{kl}, \tag{11}$$

where $\widetilde{\mathrm{H}}_{ij}^{kl}$ is the contribution to $\widetilde{\mathrm{H}}_{ij}$ from the $k$-th cell when it interacts with the $l$-th cell. In (11), it is assumed that $\widetilde{\mathrm{H}}_{ij}^{kl} = 0$ if cell $k$ is empty or cell $l$ is empty. For a pair of non-empty cells $k$ and $l$,

$$\widetilde{\mathrm{H}}_{ij}^{kl} = \sum_{s=1}^{N_{E_k}} \sum_{t=1}^{N_{E_l}} \widetilde{\mathrm{H}}_{ij}^{kl(s,t)}, \tag{12}$$

where $\widetilde{\mathrm{H}}_{ij}^{kl(s,t)}$ is the contribution to $\widetilde{\mathrm{H}}_{ij}^{kl}$ from an element couple $(E_s, E_t) \in S_k \times S_l$, and it is given by

$$\widetilde{\mathrm{H}}_{ij}^{kl(s,t)} = \int_{E_s} \int_{E_t} \psi_i(\boldsymbol{x})\, \boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, \psi_j(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \mathrm{d}s_{\boldsymbol{x}}. \tag{13}$$

By use of a suitable integration scheme, e.g., Gaussian quadrature rules [20] on triangular boundary elements, $\widetilde{\mathrm{H}}_{ij}^{kl(s,t)}$ can be approximated as

$$\widetilde{\mathrm{H}}_{ij}^{kl(s,t)} = \sum_{f=1}^{3} \sum_{\beta=1}^{N_G} \sum_{\alpha=1}^{N_G} C_i^{s\beta} C_j^{t\alpha} n_f^{t\alpha} H_f(\boldsymbol{x}^{s\beta}, \boldsymbol{y}^{t\alpha}), \tag{14}$$

where

$$C_i^{s\beta} = w^{s\beta} J(\boldsymbol{x}^{s\beta}) \psi_i(\boldsymbol{x}^{s\beta}),$$
$$C_j^{t\alpha} = w^{t\alpha} J(\boldsymbol{y}^{t\alpha}) \psi_j(\boldsymbol{y}^{t\alpha}), \quad n_f^{t\alpha} = n_f(\boldsymbol{y}^{t\alpha}). \tag{15}$$

In (15), $\boldsymbol{x}^{s\beta}$ and $w^{s\beta}$ are Gauss points and weights on element $E_s$, and $J(\boldsymbol{x}^{s\beta})$ is the jacobian of the mapping of a parent triangle (i.e. parameter space) into $E_s \in S_k$; $\boldsymbol{y}^{t\alpha}$, $w^{t\alpha}$ and $J(\boldsymbol{y}^{t\alpha})$ are respectively Gauss points, weights, and jacobian on $E_t \in S_l$; $N_G$ is the number of Gauss points on a boundary element.

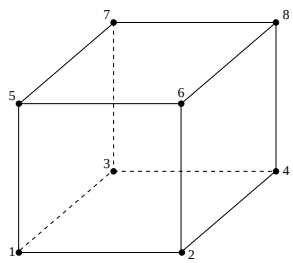


Fig. 3. Generic computational cell with $p = 2$.

Now, for a non-singular and sufficiently smooth kernel $\boldsymbol{H}$ in the computational box, one can postulate the decomposition

$$H_f(\boldsymbol{x}, \boldsymbol{y}) = \sum_{r=1}^{p^3} \sum_{q=1}^{p^3} d_r(\boldsymbol{x})\, d_q(\boldsymbol{y})\, H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q), \tag{16}$$

where $\hat{\boldsymbol{x}}^r$, $\hat{\boldsymbol{y}}^q$ represent computational grid points in the $k$-th and $l$-th cells respectively; $d_r(\boldsymbol{x})$ can be given, for instance, by the Lagrange interpolation polynomials constructed for the $k$-th cell such that

$$d_r(\hat{\boldsymbol{x}}^r) = 1, \quad d_r(\hat{\boldsymbol{x}}^s) = 0, \quad s \neq r. \tag{17}$$

In the above settings, the parameter $p$ thus corresponds to the polynomial order in each coordinate direction. With reference to Fig. 3, an example of polynomial interpolation functions $d_r(\boldsymbol{x})$, with $p = 2$, for a generic cube $[-1,1]\times[-1,1]\times[-1,1]$ can be expressed as

$$\begin{aligned}
&\zeta_1(x) = \frac{1}{2}(1-x), \quad \zeta_2(x) = \frac{1}{2}(1+x), \quad x \in [-1,1] \\
&d_1(\boldsymbol{x}) = \zeta_1(x_1)\zeta_1(x_2)\zeta_1(x_3),\ d_2(\boldsymbol{x}) = \zeta_2(x_1)\zeta_1(x_2)\zeta_1(x_3) \\
&d_3(\boldsymbol{x}) = \zeta_1(x_1)\zeta_2(x_2)\zeta_1(x_3),\ d_4(\boldsymbol{x}) = \zeta_2(x_1)\zeta_2(x_2)\zeta_1(x_3) \\
&d_5(\boldsymbol{x}) = \zeta_1(x_1)\zeta_1(x_2)\zeta_2(x_3),\ d_6(\boldsymbol{x}) = \zeta_2(x_1)\zeta_1(x_2)\zeta_2(x_3) \\
&d_7(\boldsymbol{x}) = \zeta_1(x_1)\zeta_2(x_2)\zeta_2(x_3), \\
&d_8(\boldsymbol{x}) = \zeta_2(x_1)\zeta_2(x_2)\zeta_2(x_3),\ \boldsymbol{x} = (x_1, x_2, x_3).
\end{aligned} \tag{18}$$

In practice, the parameter $p$ is usually a small integer. Typically, $p = 2, 3, 4$. It now follows from (16) that there exist in the auxiliary computational parallelepiped coefficients $d_r^{s\beta}$, $d_q^{t\alpha}$ such that

$$H_f(\boldsymbol{x}^{s\beta}, \boldsymbol{y}^{t\alpha}) = \sum_{r=1}^{p^3} \sum_{q=1}^{p^3} d_r^{s\beta}\, d_q^{t\alpha}\, H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q), \tag{19}$$

where $d_r^{s\beta} = d_r(\boldsymbol{x}^{s\beta})$, $d_q^{t\alpha} = d_q(\boldsymbol{y}^{t\alpha})$.

By use of (19) in (14), one can write

$$\widetilde{\mathrm{H}}_{ij}^{kl(s,t)} = \sum_{f=1}^{3} \sum_{r=1}^{p^3} \sum_{q=1}^{p^3} W_{ri}^{k,s} H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q) V_{qj}^{l,tf}, \tag{20}$$

where

$$W_{ri}^{k,s} = \sum_{\beta=1}^{N_G} d_r^{s\beta}\, C_i^{s\beta}, \quad V_{qj}^{l,tf} = \sum_{\alpha=1}^{N_G} d_q^{t\alpha}\, C_j^{t\alpha}\, n_f^{t\alpha}. \tag{21}$$

The superscripts $k$ and $l$ in (20) and (21) are used to indicate that the coefficients $W_{ri}^{k,s}$ and $V_{qj}^{l,tf}$ pertain to the $k$-th and $l$-th non-empty cells respectively. With the aid of (20) in (12), the contribution to $\widetilde{\mathrm{H}}_{ij}$ from two interacting non-empty cells, $k$ and $l$, can be expressed as

$$\widetilde{\mathrm{H}}_{ij}^{kl} = \sum_{f=1}^{3} \sum_{r=1}^{p^3} \sum_{q=1}^{p^3} W_{ri}^{k}\, H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q) V_{qj}^{l,f}. \tag{22}$$

In deriving (22), (21) was employed to write the interpolation coefficients as

$$W_{ri}^{k} = \sum_{s=1}^{N_{E_k}} \sum_{\beta=1}^{N_G} d_r^{s\beta}\, C_i^{s\beta}, \quad V_{qj}^{l,f} = \sum_{t=1}^{N_{E_l}} \sum_{\alpha=1}^{N_G} d_q^{t\alpha}\, C_j^{t\alpha}\, n_f^{t\alpha}. \tag{23}$$

To compactly express (22) for all boundary nodes $i, j$ in the interacting cells $k, l$, denote by $\mathbf{W}^k \in \mathbb{R}^{p^3 \times N_k}$ the matrix whose entries, $W_{ri}^k$, are given via (23a), and let $\mathbf{V}^{l,f} \in \mathbb{R}^{p^3 \times N_l}$ be the matrix with components, $V_{qj}^{l,f}$, as defined in (23b). Also, let $\widehat{\mathbf{H}}_f^{kl} \in \mathbb{R}^{p^3 \times p^3}$ be the matrix whose entries are $H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q)$, where $\hat{\boldsymbol{x}}^r$ is a grid point that lies in the $k$-th cell and $\hat{\boldsymbol{y}}^q$ is a grid point that resides in the $l$-th cell. With the above definitions, (22) can be written in matrix form as

$$\widetilde{\mathbf{H}}^{kl} = \sum_{f=1}^{3} \mathbf{W}^{k^{\mathrm{T}}} \widehat{\mathbf{H}}_f^{kl}\, \mathbf{V}^{l,f}, \quad \widetilde{\mathbf{H}}^{kl} \in \mathbb{R}^{N_k \times N_l}, \tag{24}$$

where $\mathbf{W}^{k^{\mathrm{T}}}$ is the transposed or adjoint of the interpolation matrix $\mathbf{W}^k$. A multiplication of $\mathbf{W}^{k^{\mathrm{T}}}$ by a vector defined in the $k$-th cell will be called *adjoint interpolation* or simply *anterpolation* as in [21]. With the aid of (11), (24) and the fact that $\widetilde{\mathrm{H}}_{ij}^{kl} = 0$ if at least cell $k$ or cell $l$ is empty, one can write the far-field approximation of $\mathbf{H}$ as

$$\widetilde{\mathbf{H}} = \sum_{f=1}^{3} \sum_{k=1}^{M} \sum_{l=1}^{M} \mathbf{W}^{k^{\mathrm{T}}} \widehat{\mathbf{H}}_f^{kl}\, \mathbf{V}^{l,f}, \tag{25}$$

where $M$ is the total number of non-empty cells in the computational parallelepiped. In case of cell self-interaction, i.e., when $k = l$, the diagonal components of $\widehat{\mathbf{H}}_f^{kk}$ can be set to any constant value. It is typically set to zero,.i.e., $H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{x}}^r) = 0$. The cell self-interaction contribution will be handled with care in the subsequent development. Also, note that an individual entry of (25) can be obtained by summing up the contributions $\widetilde{\mathrm{H}}_{ij}^{kl}$ over all cells $k$ and $l$, and grouping similar terms at corresponding grid points to yield

$$\widetilde{\mathrm{H}}_{ij} = \sum_{f=1}^{3} \sum_{r=1}^{Q} \sum_{q=1}^{Q} W_{ri}\, H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q) V_{qj}^{f}, \tag{26}$$

where $Q = (m(p-1)+1)^3$ is the total number of grid points in the computational box, and the interpolation coefficients $W_{ri}$, and $V_{qj}^f$ are set to zero for all grid points $r$ and $q$ pertaining to empty cells. In view of (26), denote by $\widehat{\mathbf{H}}_f \in \mathbb{R}^{Q \times Q}$ the matrix whose entries are $H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q)$ with $H_f(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{x}}^r) = 0$,

i.e., $H_f$ evaluated on the entire grid in the computational box. Next, define $\mathbf{W} \in \mathbb{R}^{Q\times N}$ and $\mathbf{V}^f \in \mathbb{R}^{Q\times N}$ matrices with components $W_{ri}$ and $V^f_{qj}$ respectively. With such definitions, (26) can formally be written in matrix form as

$$\widetilde{\mathbf{H}} = \sum_{f=1}^{3} \mathbf{W}^{\mathrm{T}} \widehat{\mathbf{H}}_f \mathbf{V}^f, \tag{27}$$

where the superscript "T" stands for the matrix transpose. In fact, (27) is an alternative representation of (25) over the entire computational grid. It simply illustrates the factorization of the far-field approximation of $\mathbf{H}$ in terms of global operators defined on the whole computational grid. An efficient implementation of the PFFT technique should never generate $\mathbf{W}$, $\mathbf{V}^i$ and $\widehat{\mathbf{H}}_i$ $(i = 1, 2, 3)$ explicitly.

Owing to the fact that $H_i$ $(i = 1, 2, 3)$ is a convolution-type kernel, the grid-to-grid mapping characterized by $\widehat{\mathbf{H}}_i$ corresponds to a *discrete convolution* on a regular grid. As a result, an operation consisting of multiplying $\widehat{\mathbf{H}}_i$ by a vector on the grid can be effected by the FFT algorithm (e.g. [22]) over the entire computational grid. The FFT scheme requires on the order of $O(Q \ln Q)$ arithmetic operations. However, for grid and surface discretizations that are such that $m^3$ scales as $O(N)$ and the fact that $p$ is a small integer, the cost of the FFT in the fast spectral method will reduce to $O(N \ln N)$. Such is the case for the so-called homogeneous distribution of boundary elements into computational cells (see [5]).

Following the procedures elaborated above, it can be shown that the far-field approximation of $\mathbf{G}$ in (9a), involving the single-layer potential kernel $G$, admits the representation

$$\widetilde{\mathbf{G}} = \mathbf{W}^{\mathrm{T}} \widehat{\mathbf{G}} \mathbf{W}, \tag{28}$$

where $\widehat{\mathbf{G}} \in \mathbb{R}^{Q\times Q}$ is a matrix with components $G(\hat{\boldsymbol{x}}^r, \hat{\boldsymbol{y}}^q)$ and $G(\hat{\boldsymbol{y}}^q, \hat{\boldsymbol{y}}^q) = 0$; $\mathbf{W}$ is the interpolation matrix featured in (27). Note that the symmetry of the Galerkin integral for the kernel $G$ is preserved. The contribution to $\widetilde{\mathbf{G}}$ from a pair of interacting non-empty cells, $k$ and $l$, can be expressed as

$$\widetilde{\mathbf{G}}^{kl} = \mathbf{W}^{k^{\mathrm{T}}} \widehat{\mathbf{G}}^{kl} \mathbf{W}^l, \quad \widetilde{\mathbf{G}}^{kl} \in \mathbb{R}^{N_k \times N_l}, \tag{29}$$

where $\mathbf{W}^k \in \mathbb{R}^{p^3 \times N_k}$ is the interpolation matrix for the $k$-th cell as in (24), and $\widehat{\mathbf{G}}^{kl} \in \mathbb{R}^{p^3\times p^3}$ is the portion of the convolution matrix $\widehat{\mathbf{G}}$ associated with the interaction between cell $k$ and cell $l$.

In view of (27) and (28), one can infer that interpolation matrices $\mathbf{W}$ and $\mathbf{V}^i$ $(i = 1, 2, 3)$ map boundary data prescribed on $S$ onto the regular grid. Furthermore, $\mathbf{W}^{\mathrm{T}}$ maps grid quantities back onto the boundary $S$. Owing to the cell-by-cell construction of the global interpolation matrices via $\mathbf{W}^k$ and $\mathbf{V}^{k,i}$ $(i = 1, 2, 3)$, $\mathbf{W}$ and $\mathbf{V}^i$ are sparse. They contain $O(N)$ non-zero entries. Indeed, the number of operations needed to generate, e.g., $\mathbf{W}$ is roughly equal to $Mp^3N_k$, where $p^3N_k$ is the cost of $\mathbf{W}^k$. Since $p$ is small and $N_k \leq N$, the operation count needed to generate $\mathbf{W}^k$ is $O(N)$. Moreover, for large $N$, the number of non-empty cells $M$ is much smaller than $N$. With these results, it follows that the computational cost of $\mathbf{W}$ scales as $O(N)$. Consequently, the multiplication of $\mathbf{W}$ or $\mathbf{V}^i$ by a vector will require $O(N)$ arithmetic operations and memory storage.

### B. Precorrection

The rapid far-field approximation of the products $\mathbf{G}\{t\}$ and $\mathbf{H}\{u\}$, carried out over the entire computational grid via the FFT technique, will inaccurately represent the near-field contribution to $\mathbf{G}$ and $\mathbf{H}$. For example, the cell self-interaction terms such as $\mathbf{W}^{k^{\mathrm{T}}} \widehat{\mathbf{G}}^{kk} \mathbf{W}^k$ and $\mathbf{W}^{k^{\mathrm{T}}} \widehat{\mathbf{H}}_i^{kk} \mathbf{V}^{k,i}$, which are embedded in the regular grid evaluations via the FFT, represent unsatisfactory approximations of the actual double integrals featured in (9) from within the considered cell. Also, from the smoothness assumption on the kernels $G$ and $\boldsymbol{H}$, the regular grid approximation cannot accurately handle singular or weakly-singular boundary integrals. To alleviate these impediments, a correction of near-field contributions to the designated influences is performed for every non-empty cell and their set of near-neighboring non-empty cells. To this end, let $M_k$ be the number of near-neighboring cells of a given non-empty cell $k$ in the computational cube.

Taking (9b) again as a point of departure, assume that some boundary trial data $\{u\}$ are prescribed on $S$, and one is to compute the product $\mathbf{H}\{u\}$. To facilitate the ensuing development, let $\mathbf{H}^{kl} \in \mathbb{R}^{N_k\times N_l}$ denote the block of $\mathbf{H}$ expressed in (9b) associated with boundary nodes in the interaction between cell $k$ and its near-neighbor cell $l$, where $N_k$ is the number of boundary nodes in the $k$-th cell. The entries of $\mathbf{H}^{kl}$ are computed by explicitly evaluating the integrals

$$\mathrm{H}^{kl}_{ij} = \lim_{\varepsilon\to 0} \int_{S_k}\int_{S_l} \psi_i(\boldsymbol{x})\, \boldsymbol{H}(\boldsymbol{x}_\varepsilon, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, \psi_j(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \mathrm{d}s_{\boldsymbol{x}}, \tag{30}$$

$(i = 1, 2, \ldots, N_k,\ j = 1, 2, \ldots, N_l)$, for all boundary element pairs $(E_s, E_t) \in S_k \times S_l$. In the computation of (30), suitable limiting techniques are used the tackle the singular behavior of coincident, vertex-adjacent and edge-adjacent element pairs (see [14]). Also, let $\{u\}^k$ be the vector composed with the entries of $\{u\}$ associated with the boundary nodes in the $k$-th cell, and denote by $\{\underline{u}\}^k$ the regular grid approximation (i.e., the far-field approximation obtained globally via the FFT) of $\mathbf{H}\{u\}$ in the $k$-th cell.

With theses definitions,

$$\{\underline{u}\}^{(k/l)} = \sum_{i=1}^{3} \mathbf{W}^{k^{\mathrm{T}}}\, \widehat{\mathbf{H}}_i^{kl}\, \mathbf{V}^{l,i}\, \{u\}^l \tag{31}$$

represents the inaccurate contribution to $\{\underline{u}\}^k$ from the grid approximation due to surface density $\{u\}^l$ in the neighboring $l$-th cell. A complete approximation of $\mathbf{H}\{u\}$ in the $k$-th cell due to surface density $\{u\}^l$ from the $l$-th cell can be obtained as

$$\{\mathbf{H}\{u\}\}^{(k/l)} = \{\underline{u}\}^k - \{\underline{u}\}^{(k/l)} + \mathbf{H}^{kl}\{u\}^l, \tag{32}$$

i.e., subtracting off the inaccurate contribution from the grid and adding in the correct near-field contribution. In view of (31), (32) can be expressed as

$$\{\mathbf{H}\{u\}\}^{(k/l)} = \{\underline{u}\}^k + \mathbf{P}^{kl}\, \{u\}^l\,, \tag{33}$$

where $\mathbf{P}^{kl} \in \mathbb{R}^{N_k\times N_l}$ is a precorrection matrix for the $k$-th cell in interaction with the $l$-th near-neighboring cell given by

$$\mathbf{P}^{kl} = \mathbf{H}^{kl} - \sum_{i=1}^{3} \mathbf{W}^{k^{\mathrm{T}}}\, \widehat{\mathbf{H}}_i^{kl}\, \mathbf{V}^{l,i}. \tag{34}$$

With (33) and (34), a satisfactory approximation of $\mathbf{H}\{u\}$ (from the $k$-th cell) is obtained by adding up corrections from all near-neighboring cells $l$ as

$$\{\mathbf{H}\{u\}\}^k = \{\underline{u}\}^k + \sum_{l=1}^{M_k} \mathbf{P}^{kl}\,\{u\}^l\,, \tag{35}$$

where $M_k$ is the number of computational cells in the near-field region of the $k$-th non-empty cell. By construction, it is assumed that $\mathbf{P}^{kl} = \mathbf{0}$ if at least cell $k$ or cell $l$ is empty. With the above settings, local precorrection matrices $\mathbf{P}^{kl}$ can be accumulated for all cells $k$ and all near-neighboring cells $l$ to form the whole precorrection matrix $\mathbf{P}$. Since interactions involving non-neighboring cells are approximated globally via the regular grid, the whole precorrection matrix $\mathbf{P}$ is sparse. With this analysis and (27), it follows that an approximation of the influence matrix $\mathbf{H}$ by the precorrected-FFT method results in the decomposition

$$\mathbf{H} = \mathbf{P} + \sum_{i=1}^{3} \mathbf{W}^{\mathrm{T}} \widehat{\mathbf{H}}_i \mathbf{V}^i, \tag{36}$$

where $\mathbf{P}$, $\mathbf{V}^i$ and $\mathbf{W}$ are sparse matrices.

Similarly, for the treatment of (9a), one can also introduce a precorrection matrix for the $k$-th cell in interaction with a near-neighboring $l$-th cell as

$$\mathbf{R}^{kl} = \mathbf{G}^{kl} - \mathbf{W}^{k^{\mathrm{T}}}\,\widehat{\mathbf{G}}^{kl}\,\mathbf{W}^l, \tag{37}$$

where $\widehat{\mathbf{G}}^{kl} \in \mathbb{R}^{p^3\times p^3}$ is the block of the convolution matrix $\widehat{\mathbf{G}}$ in the interaction between cell $k$ and its near-neighbor cell $l$; $\mathbf{G}^{kl} \in \mathbb{R}^{N_k\times N_l}$ denotes the portion of $\mathbf{G}$ expressed in (9a) when cell $k$ interacts with its near-neighbor cell $l$. The components of $\mathbf{G}^{kl}$ are obtained via direct calculation of the integrals

$$\mathrm{G}^{kl}_{ij} = \lim_{\varepsilon\to 0} \int_{S_k}\int_{S_l} \psi_i(\boldsymbol{x})\, G(\boldsymbol{x}_\varepsilon, \boldsymbol{y})\, \psi_j(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}} \mathrm{d}s_{\boldsymbol{x}}, \tag{38}$$

($i = 1, 2, \ldots, N_k,\ j = 1, 2, \ldots, N_l$), for all boundary element pairs $(E_s, E_t) \in S_k \times S_l$. In the computation of (38), appropriate limiting procedures are also used to deal with weakly-singular integrals. On the basis of (37), an approximation of $\mathbf{G}\{t\}$ in the $k$-th cell can be conveniently written as

$$\{\mathbf{G}\{t\}\}^k = \{\underline{t}\}^k + \sum_{l=1}^{M_k} \mathbf{R}^{kl}\,\{t\}^l\,, \tag{39}$$

where $\{\underline{t}\}^k$ is the regular grid approximation of $\mathbf{G}\{t\}$ in the $k$-th cell obtained via interpolation to the grid, convolution on the grid, and anterpolation to the surface portion in the $k$-th cell; $\{t\}^l$ is the flux density in the near-neighboring cell $l$. Similar to the situation involving $\mathbf{H}$, the precorrected-FFT approximation of $\mathbf{G}$ yields the decomposition

$$\mathbf{G} = \mathbf{R} + \mathbf{W}^{\mathrm{T}} \widehat{\mathbf{G}} \mathbf{W}, \tag{40}$$

where $\mathbf{R}$ and $\mathbf{W}$ are sparse matrices. A decomposition of the form (40) was first proposed in [5] for the study of the Laplace equation using only single-layer potential representation and piece-wise constant approximations.

The interpolation operators, $\mathbf{W}$ and $\mathbf{V}^i$ ($i = 1, 2, 3$), featured in the decomposition (36) and (40) play a key role in the PFFT method. Specifically, these operators are used to map data to/from the auxiliary computational grid in the so-called interpolation/anterpolation process of the PFFT scheme. With the assumption that the near-field correction is exact, the accuracy of the PFFT method is entirely determined by the precision of the interpolation operators. By construction, this precision is strongly affected by the method parameter $p$ which characterizes the order of interpolations in each coordinate direction.

## IV. Potential matching method

To establish a direct connection with the PFFT method proposed in [5], it is useful to introduce an alternative procedure to generate the interpolation operators $\mathbf{W}$ and $\mathbf{V}^i$ ($i = 1, 2, 3$). To this end, a particular form of $\mathbf{W}$ and $\mathbf{V}^i$ can be constructed on a cell-by-cell basis via a potential matching method. In this approach, potentials given on $S_k$, a portion of the surface $S$ in the $k$-th non-empty cell, are replaced by "equivalent" potentials generated by a set of point sources in the designated cell. To illustrate this procedure, consider, e.g., the single-layer potential

$$v^{SL}(\boldsymbol{x}) = \int_\Gamma G(\boldsymbol{x}, \boldsymbol{y})\, \zeta(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \Gamma \tag{41}$$

with potential density $\zeta$ that is piecewise continuous on a surface $\Gamma$. The far-field behavior of $v^{SL}$ can be expressed as

$$v^{SL}(\boldsymbol{x}) = G(\boldsymbol{x}, \mathbf{0})\, q + O\left(\frac{1}{\|\boldsymbol{x}\|^2}\right), \quad \text{as} \quad \|\boldsymbol{x}\| \to \infty, \tag{42}$$

where $\mathbf{0} = (0, 0, 0)^{\mathrm{T}}$ is the origin of the coordinate system, and $q = \int_\Gamma \zeta(\boldsymbol{y}) \mathrm{d}s_{\boldsymbol{y}}$ is termed the monopole moment. It is seen from (42) that the monopole term $G(\boldsymbol{x}, \mathbf{0})\, q$ can be used as a first order approximation of the single-layer potential outside any ball enclosing the origin $\mathbf{0}$ and $\Gamma$.

Now, define the double-layer potential

$$v^{DL}(\boldsymbol{x}) = \int_\Gamma \boldsymbol{H}(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{n}(\boldsymbol{y})\, \chi(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \Gamma \tag{43}$$

with piecewise continuous density $\chi$ on $\Gamma$. The far-field contribution to $v^{DL}$ can be shown to take the form

$$v^{DL}(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{x}, \mathbf{0}) \cdot \boldsymbol{d} + O\left(\frac{1}{\|\boldsymbol{x}\|^3}\right), \quad \text{as} \quad \|\boldsymbol{x}\| \to \infty, \tag{44}$$

where $\boldsymbol{d} = \int_\Gamma \boldsymbol{n}(\boldsymbol{y}) \chi(\boldsymbol{y}) \mathrm{d}s_{\boldsymbol{y}}$ is called the dipole moment. The result given by (44) suggests that the dipole term $\boldsymbol{H}(\boldsymbol{x}, \mathbf{0}) \cdot \boldsymbol{d}$ can be used to approximate the double-layer potential outside any ball enclosing the origin $\mathbf{0}$ and $\Gamma$.

With these observations, it will be shown in the sequel that alternative local interpolation matrices, $\mathbf{Y}^k$ and $\mathbf{X}^{k,i}$ ($i = 1, 2, 3$), can be generated by representing the single-layer potential with monopoles and the double-layer potential with dipoles at grid points in the $k$-th non-empty cell respectively.

### A. Monopole representation

To construct the linear operator $\mathbf{Y}$ that maps boundary data onto a regular grid in the computational parallelepiped, consider the single-layer potential

$$v^{SL}(\boldsymbol{x}) = \int_{S_k} G(\boldsymbol{x},\boldsymbol{y})\,\zeta^k(\boldsymbol{y})\,\mathrm{d}s_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3\setminus S_k, \tag{45}$$

where $S_k$ is the union of all boundary elements in the $k$-th non-empty cell and $\zeta^k$ is a piecewise continuous density defined on $S_k$. Similar to the decomposition (6), the potential density $\zeta^k$ in the $k$-th computational cell can be approximated as

$$\zeta^k(\boldsymbol{y}) = \sum_{j=1}^{N_k} \zeta^{kj}\,\psi_j^k(\boldsymbol{y}), \quad \zeta^{kj} = \zeta^k(\boldsymbol{y}^j), \tag{46}$$

where $\boldsymbol{y}^j$ is the $j$-th boundary node on $S_k$, and $\psi_j^k$ is the shape function associated with $\boldsymbol{y}^j$ in the $k$-th cell. On the basis of (46), the single-layer potential expressed in (45) reduces to

$$v^{SL}(\boldsymbol{x}) = \sum_{j=1}^{N_k} \zeta^{kj} \int_{S_k} G(\boldsymbol{x},\boldsymbol{y})\,\psi_j^k(\boldsymbol{y})\,\mathrm{d}s_{\boldsymbol{y}}, \quad \boldsymbol{x}\in\mathbb{R}^3\setminus S_k. \tag{47}$$

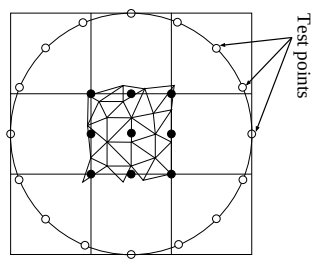


Fig. 4. Test surface centered at a generic non-empty computational cell.

To represent the surface potential $v^{SL}$ outside the $k$-th cell, one can introduce the grid potential

$$v^G(\boldsymbol{x}) = \sum_{j=1}^{p^3} G(\boldsymbol{x},\hat{\boldsymbol{z}}^{kj})\,\gamma^{kj}, \quad \boldsymbol{x}\in\mathbb{R}^3\setminus\{\hat{\boldsymbol{z}}^{kj}\}_{j=1}^{p^3} \tag{48}$$

as a sum of monopoles $G(\boldsymbol{x},\hat{\boldsymbol{z}}^{kj})\,\gamma^{kj}$ with moments $\gamma^{kj}$ located at grid points $\{\hat{\boldsymbol{z}}^{kj}\}_{j=1}^{p^3}$ in the $k$-th cell.

Now, suppose that boundary data $\zeta^{kl}$ ($l=1,2,\ldots,N_k$) are prescribed on $S_k$. With this assumption, one can evaluate the source intensities $\gamma^{kj}$ ($j=1,2,\ldots,p^3$) by requiring that

$$v^G(\tilde{\boldsymbol{x}}^i) = v^{SL}(\tilde{\boldsymbol{x}}^i), \quad i=1,2,\ldots,N_t \tag{49}$$

at selected test points $\tilde{\boldsymbol{x}}^i$ resting on a sphere containing the $k$-th cell (see Fig. 4). In practice, the number of test points $N_t$ is specified so that $N_t \geq p^3$. The equality expressed in (49) makes sense as it is in accordance with the far-field pattern given by (42). By virtue of (47) and (48), equality (49) yields

$$\begin{aligned} &\check{\mathbf{G}}^k\{\gamma\}^k = \mathbf{D}^k\{\zeta\}^k, \\ &\{\gamma\}^k = \{\gamma^{kj}\}_{j=1}^{p^3}, \quad \{\zeta\}^k = \{\zeta^{kl}\}_{l=1}^{N_k}, \end{aligned} \tag{50}$$

where the components of $\check{\mathbf{G}}^k \in \mathbb{R}^{N_t\times p^3}$ are given as

$$\check{G}_{ij}^k = G(\tilde{\boldsymbol{x}}^i,\hat{\boldsymbol{z}}^{kj}), \quad i=1,2,\ldots,N_t,\ j=1,2,\ldots,p^3, \tag{51}$$

and the entries of $\mathbf{D}^k \in \mathbb{R}^{N_t\times N_k}$ are expressed as

$$D_{il}^k = \int_{S_k} G(\tilde{\boldsymbol{x}}^i,\boldsymbol{y})\,\psi_l^k(\boldsymbol{y})\,\mathrm{d}s_{\boldsymbol{y}}, \quad l=1,2,\ldots,N_k. \tag{52}$$

For an efficient computation of the mapping, the test sphere is selected so that it is centered at the $k$-th cell. With this choice, the relative position of the test points and the grid points, $(\tilde{\boldsymbol{x}}^i - \hat{\boldsymbol{z}}^{kj})$, does not depend on the cell number $k$. As the full-space Green's function $G$ is of convolution-type, it follows from (51) that $\check{\mathbf{G}}^k$ is also independent of the cell number $k$ and remains the same for all cells in the computational parallelepiped. As a result, $\check{\mathbf{G}}^k$ will be denoted simply as $\check{\mathbf{G}}$. By use of the singular value decomposition [23], the linear system (50) can be inverted for $\{\gamma\}^k$ and one can write

$$\{\gamma\}^k = \check{\mathbf{G}}^\dagger\mathbf{D}^k\{\zeta\}^k, \tag{53}$$

where $\check{\mathbf{G}}^\dagger$ is the Moore-Penrose pseudo-inverse of $\check{\mathbf{G}}$. On the basis of (53), one can now introduce the interpolation matrix $\mathbf{Y}^k \in \mathbb{R}^{p^3\times N_k}$ for the $k$-th cell as

$$\mathbf{Y}^k = \check{\mathbf{G}}^\dagger\mathbf{D}^k. \tag{54}$$

By construction, $\mathbf{Y}^k = \mathbf{0}$ whenever $N_k = 0$. For any given cell $k$, $\mathbf{Y}^k$ maps the prescribed boundary data onto the grid data in the designated cell.

To formally generate the global interpolation matrix $\mathbf{Y}$, it is important to note that the grid density $\gamma^{kj}$ at a grid point $\hat{\boldsymbol{z}}^{kj}$ shared by multiple cells is obtained as a sum of the contribution from all cells having $\hat{\boldsymbol{z}}^{kj}$ as a common node. With the presence of empty cells in the computational box, the whole interpolation matrix $\mathbf{Y}$ is sparse. Moreover, the operation count necessary to create $\mathbf{Y}$ reduces to $O(N)$. Indeed, the computational cost of constructing $\mathbf{Y}$ is proportional to $Mp^3N_k$, where the cost of the local linear map $\mathbf{Y}^k$ is $p^3N_k$. On recalling that $N_k \leq N$, $p$ is small (usually $p=2,3,4$) and that, for large $N$, the number of non-empty cells $M$ is much smaller than the boundary mesh $N$, it follows that the generation of $\mathbf{Y}$ requires $O(N)$ operations and memory storage.

To establish a link with the polynomial-based interpolation method presented in §III, it is useful to employ the same numerical integration scheme as in (14) and write an approximation to (52) as

$$D_{lj}^k = \sum_{s=1}^{N_{E_k}} \sum_{\beta=1}^{N_G} C_j^{s\beta} G(\tilde{\boldsymbol{x}}^l,\boldsymbol{y}^{s\beta}), \tag{55}$$

($l=1,2,\ldots,N_t,\ j=1,2,\ldots,N_k$), where $C_j^{s\beta}$ is given by (15). From (55), the entries of (54) can be expressed as

$$Y_{rj}^k = \sum_{s=1}^{N_{E_k}} \sum_{\beta=1}^{N_G} d_r^{s\beta}\,C_j^{s\beta}, \tag{56}$$

($r=1,2,\ldots,p^3,\ j=1,2,\ldots,N_k$), where $d_r^{s\beta}$ is the value of the function

$$d_r(\boldsymbol{y}) = \sum_{i=1}^{N_t} \check{G}_{ri}^\dagger G(\tilde{\boldsymbol{x}}^i,\boldsymbol{y}) \tag{57}$$

at an integration point $\boldsymbol{y}^{s\beta}$ resting on $S_k$, i.e. that $d_r^{s\beta} = d_r(\boldsymbol{y}^{s\beta})$. In (57), $\check{G}_{ri}^{\dagger}$ is an individual component of $\check{\mathbf{G}}^{\dagger}$. It is important to note that $Y_{rj}^k$ in (56) is formally expressed by the same formula as $W_{rj}^k$ in (23a) except that the coefficients $d_r^{s\beta}$ are characterized by the Green's functions (57). One also has to recall that $W_{rj}^k$ was derived using grid polynomials as, for example, in (18). With these observations, the monopole representation is indeed an interpolation method generated by the Green's functions $G(\tilde{\boldsymbol{x}}^i, \cdot)$ acting at test points $\tilde{\boldsymbol{x}}^i$ ($i = 1, 2, \ldots, N_t$). By specifying the shape function $\psi_l^k$ to be constant on the $l$-th boundary element of $S_k$, the local operator $\mathbf{Y}^k$ expressed in (54) corresponds to the mapping given in [5]. The interpolation operator $\mathbf{Y}$ can be used in place of $\mathbf{W}$ in the PFFT method to map the potential density on the Fourier grid.

### B. Dipole representation

To compute the entries of $\mathbf{X}^i$ ($i = 1, 2, 3$), the linear operator that maps the flux density onto the regular grid, consider the potential

$$v_i^{DL}(\boldsymbol{x}) = \int_{S_k} H_i(\boldsymbol{x}, \boldsymbol{y})\, n_i(\boldsymbol{y})\, \zeta^k(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus S_k, \quad (58)$$

(no sum on $i$), where $\zeta^k$ now denotes the flux density that is confined to the $k$-th cell according to a decomposition similar to (46). In view of (44), $v_i^{DL}$ can be approximated outside cell $k$ by use of a set of dipoles $H_i(\boldsymbol{x}, \hat{\boldsymbol{z}}^{kj})\, \gamma^{kj}$ with intensities $\gamma^{kj}$ located at grid points $\{\hat{\boldsymbol{z}}^{kj}\}_{j=1}^{p^3}$ in the $k$-th cell as

$$v_i^H(\boldsymbol{x}) = \sum_{j=1}^{p^3} H_i(\boldsymbol{x}, \hat{\boldsymbol{z}}^{kj})\, \gamma^{kj}, \quad \boldsymbol{x} \in \mathbb{R}^3 \setminus \{\hat{\boldsymbol{z}}^{kj}\}_{j=1}^{p^3}. \quad (59)$$

With reference to Fig. 4 and the specified boundary data $\zeta^{kl}$ ($l = 1, 2, \ldots, N_k$) on $S_k$, one can determine the coefficients $\gamma^{kj}$ ($j = 1, 2, \ldots, p^3$) by requiring that

$$v_i^H(\tilde{\boldsymbol{x}}^q) = v_i^{DL}(\tilde{\boldsymbol{x}}^q), \quad q = 1, 2, \ldots, N_t, \quad N_t \geq p^3 \quad (60)$$

at selected test points $\tilde{\boldsymbol{x}}^q$ resting on a sphere centered at, and enclosing, the $k$-th cell. By use of (46), (58) through (60), one can write the linear system

$$\check{\mathbf{H}}^i \{\gamma\}^k = \mathbf{E}^{k,i} \{\zeta\}^k,$$
$$\{\gamma\}^k = \{\gamma^{kj}\}_{j=1}^{p^3}, \quad \{\zeta\}^k = \{\zeta^{kl}\}_{l=1}^{N_k}, \quad (61)$$

where the components of $\check{\mathbf{H}}^i \in \mathbb{R}^{N_t \times p^3}$ specified as

$$\check{H}_{qj}^i = H_i(\tilde{\boldsymbol{x}}^q, \hat{\boldsymbol{z}}^{kj}), \; q = 1, 2, \ldots, N_t, \; j = 1, 2, \ldots, p^3, \quad (62)$$

are independent of the cell number $k$ owing to the choice of the test sphere; the entries of $\mathbf{E}^{k,i} \in \mathbb{R}^{N_t \times N_k}$ are given by

$$E_{ql}^{k,i} = \int_{S_k} H_i(\tilde{\boldsymbol{x}}^q, \boldsymbol{y})\, n_i(\boldsymbol{y})\, \psi_l^k(\boldsymbol{y})\, \mathrm{d}s_{\boldsymbol{y}}, \quad (63)$$

($l = 1, 2, \ldots, N_k$). On employing the singular value decomposition, (61) can be solved for $\{\gamma\}^k$ and one can write

$$\{\gamma\}^k = \check{\mathbf{H}}^{i\dagger} \mathbf{E}^{k,i} \{\zeta\}^k, \quad (64)$$

where $\check{\mathbf{H}}^{i\dagger}$ is the generalized inverse of $\check{\mathbf{H}}^i$. By virtue of (64), one can define the interpolation matrix $\mathbf{X}^{k,i} \in \mathbb{R}^{p^3 \times N_k}$ for the $k$-th cell as

$$\mathbf{X}^{k,i} = \check{\mathbf{H}}^{i\dagger} \mathbf{E}^{k,i}. \quad (65)$$

Also, the assumption that $N_k = 0$ implies that $\mathbf{X}^{k,i} = \mathbf{0}$.

The global matrix $\mathbf{X}^i$ ($i = 1, 2, 3$) can be formally assembled from local contributions $\mathbf{X}^{k,i}$ for all non-empty cells. As in the case of the single-layer potential, $\mathbf{X}^i$ is sparse. Moreover, it can be shown that the operation count necessary to generate the global matrix $\mathbf{X}^i$ is $O(N)$, where $N$ is the total number of boundary nodes.

The connection of the foregoing procedure with the polynomial-based interpolation method examined in §III can be established provided that the same numerical integration scheme as in (14) is employed to approximate (63). With such an integration scheme, approximation of (63) can be given as

$$E_{lj}^{k,i} = \sum_{s=1}^{N_{E_k}} \sum_{\beta=1}^{N_G} C_j^{s\beta} H_i(\tilde{\boldsymbol{x}}^l, \boldsymbol{y}^{s\beta}) n_i^{s\beta}, \quad (66)$$

($l = 1, 2, \ldots, N_t, \; j = 1, 2, \ldots, N_k$), where $C_j^{s\beta}$ is, as before, given by (15) and $n_i^{s\beta} = n_i(\boldsymbol{y}^{s\beta})$. With the aid of (66) in (65), the components of $\mathbf{X}^{k,i}$ can be written as

$$X_{rj}^{k,i} = \sum_{s=1}^{N_{E_k}} \sum_{\beta=1}^{N_G} d_r^{s\beta}\, C_j^{s\beta} n_i^{s\beta}, \quad (67)$$

($r = 1, 2, \ldots, p^3, \; j = 1, 2, \ldots, N_k$), where $d_r^{s\beta}$ is the value of the function

$$d_r(\boldsymbol{y}) = \sum_{l=1}^{N_t} \check{H}_{rl}^{i\dagger} H_i(\tilde{\boldsymbol{x}}^l, \boldsymbol{y}), \quad \text{(no sum on } i) \quad (68)$$

at an integration point $\boldsymbol{y}^{s\beta}$ resting on $S_k$, i.e. that $d_r^{s\beta} = d_r(\boldsymbol{y}^{s\beta})$. In (68) $\check{H}_{rl}^{i\dagger}$ is an entry of $\check{\mathbf{H}}^{i\dagger}$. It is again clear from (67) that $X_{rj}^{k,i}$ is formally expressed via the same formula as $V_{rj}^{k,i}$ in (23b) except that the multipliers $d_r^{s\beta}$ are specified by (68). It is also instructive to note that $V_{rj}^{k,i}$ was obtained by virtue of local grid polynomials (see, e.g., (18)). These remarks reveal that the dipole representation corresponds to an interpolation method generated by functions $H_i(\tilde{\boldsymbol{x}}^l, \cdot)$ acting at test points $\tilde{\boldsymbol{x}}^l$ ($l = 1, 2, \ldots, N_t$). In the PFFT scheme, the interpolation operator $\mathbf{X}^i$ can be used as a replacement of $\mathbf{V}^i$ to map the flux density on the regular grid.

Compared to the polynomial-based interpolation procedure, the potential matching method introduces two additional parameters to the PFFT technique. More precisely, $R_t$ which is the radius of the test sphere and $N_t$ which is the number of test points resting on the test sphere. By construction, these parameters can affect the accuracy of the mappings $\mathbf{Y}$ and $\mathbf{X}^i$ which determine the precision of the PFFT method. Moreover, the overall performance of the PFFT algorithm can also be affected, as the generation of these mappings involves the use of the singular value decomposition.

## V. Numerical treatment

The precorrected-FFT method is an algorithm to rapidly compute the matrix-vector products $\mathbf{G}\{t\}$ and $\mathbf{H}\{u\}$ involved in an iterative solution of discretized BIEs such as (10), where $\{t\}$ and $\{u\}$ are some trial data prescribed on the boundary $S$. More precisely, the grid-based treatment of a matrix-vector product can be achieved via (i) a mapping of boundary data onto a regular grid, (ii) a convolution on the grid, (iii) an adjoint interpolation (anterpolation) to boundary nodes, and (iv) a precorrection of near-field influences that are not accurately approximated on the uniform grid.

To practically describe these operations, one can use the mappings $\mathbf{W}$ and $\mathbf{V}^i$ introduced in §III, to construct the grid data $\{\hat{t}\} = \mathbf{W}\{t\}$ and $\{\hat{u}^i\} = \mathbf{V}^i\{u\}$. With this grid data, the FFT algorithm can be employed to rapidly compute the discrete convolutions, $\{\underline{\hat{t}}\} = \widehat{\mathbf{G}}\,\{\hat{t}\}$ and $\{\underline{\hat{u}}\} = \sum_{i=1}^{3} \widehat{\mathbf{H}}_i\{\hat{u}^i\}$, featured in (27) and (28) respectively. Next, $\mathbf{W}$ can again be used to obtain the far-field approximations, $\{\underline{t}\} = \mathbf{W}^{\mathrm{T}}\,\{\underline{\hat{t}}\}$ and $\{\underline{u}\} = \mathbf{W}^{\mathrm{T}}\{\underline{\hat{u}}\}$, of $\mathbf{G}\{t\}$ and $\mathbf{H}\{u\}$ respectively. In view of (36) and (40), the far-field approximations $\{\underline{t}\}$ and $\{\underline{u}\}$ are combined with corresponding corrections $\mathbf{R}\{t\}$ and $\mathbf{P}\{u\}$ to complete the matrix-vector procedure.

For an efficient design of the PFFT scheme, steps (i), (iii) and (iv) are implemented on a cell-by-cell basis for all non-empty cells, i.e., cells containing boundary elements. This methodology results in a significant reduction in memory requirements. Step (ii) is performed over the entire computational grid and requires the FFTs of the single-layer potential kernel $G$ ($\widehat{\mathcal{G}} = \mathrm{FFT}(\widehat{\mathbf{G}})$), and double-layer potential kernel $\boldsymbol{H}$ ($\widehat{\mathcal{H}}_i = \mathrm{FFT}(\widehat{\mathbf{H}}_i)$) on the grid, where $\widehat{\mathbf{G}}$ and $\widehat{\mathbf{H}}_i$ are matrices representing $G$ and $H_i$ over the computational grid (see §III). In the implementation of the PFFT method, it is important to note that the generation of interpolation matrices $\mathbf{W}^k$ and $\mathbf{V}^{k,i}$ $(i = 1, 2, 3)$ for all non-empty cells $k = 1, 2, \ldots, M$, the FFTs of the featured kernels $G$ and $\boldsymbol{H}$, and the formation of precorrection matrices $\mathbf{R}^{kl}$ and $\mathbf{P}^{kl}$ $(l = 1, 2, \ldots, M_k)$ do not depend on the trial boundary data $\{t\}$ and $\{u\}$. Therefore, these laborious procedures are performed only once in the so-called pre-processing phase of the PFFT algorithm.

With the assumption that the pre-processing part of the PFFT scheme is already computed, the following pseudo-code constitutes a complete algorithm for the rapid evaluation of the matrix-vector products $\mathbf{G}\{t\}$ and $\mathbf{H}\{u\}$.

*Algorithm 1 (PFFT):*

//Interpolation

Set $\{\hat{t}\} = 0$ and $\{\hat{u}^i\} = 0$

For $k = 1, \ldots, M$ //Loop over all non-empty cells

$\{\hat{t}\}^k = \{\hat{t}\}^k + \mathbf{W}^k\{t\}^k$

$\{\hat{u}^i\}^k = \{\hat{u}^i\}^k + \mathbf{V}^{k,i}\{u\}^k$

end (For)

//Convolution

$\widehat{\mathcal{T}} = \mathrm{FFT}(\{\hat{t}\})$,

$\widehat{\mathcal{U}}^i = \mathrm{FFT}(\{\hat{u}^i\})$ //FFTs of grid data

$\underline{\widehat{\mathcal{T}}} = \widehat{\mathcal{G}}\,\widehat{\mathcal{T}}$, $\underline{\widehat{\mathcal{U}}^i} = \widehat{\mathcal{H}}_i\widehat{\mathcal{U}}^i$ //Convolution

$\{\underline{\hat{t}}\} = \mathrm{FFT}^{-1}\left(\underline{\widehat{\mathcal{T}}}\right)$,

$\{\underline{\hat{u}}^i\} = \mathrm{FFT}^{-1}\left(\underline{\widehat{\mathcal{U}}^i}\right)$ //Inverse FFTs

//Anterpolation

Set $\{\underline{t}\} = 0$ and $\{\underline{u}^i\} = 0$

For $k = 1, \ldots, M$ //Loop over all non-empty cells

$\{\underline{t}\}^k = \{\underline{t}\}^k + \mathbf{W}^{k\,\mathrm{T}}\,\{\underline{\hat{t}}\}^k$

$\{\underline{u}^i\}^k = \{\underline{u}^i\}^k + \mathbf{W}^{k\,\mathrm{T}}\,\{\underline{\hat{u}}^i\}^k$

end (For)

Set $\{\underline{u}\} = 0$

For $i = 1, 2, 3$ //Loop over all components

$\{\underline{u}\} = \{\underline{u}\} + \{\underline{u}^i\}$

end (For)

//Precorrection

For $k = 1, \ldots, M$ //Loop over all non-empty cells

For $l = 1, \ldots, M_k$ //Loop over all near-
//neighboring cells to the $k$-th cell

$\{\underline{t}\}^k = \{\underline{t}\}^k + \mathbf{R}^{kl}\{t\}^l$

$\{\underline{u}\}^k = \{\underline{u}\}^k + \mathbf{P}^{kl}\{u\}^l$

end (For)

end (For).

An complete application of the above routine will yield $\mathbf{G}\{t\} \approx \{\underline{t}\}$ and $\mathbf{H}\{u\} \approx \{\underline{u}\}$. In the routine, one can respectively replace the interpolation operators $\mathbf{W}$ and $\mathbf{V}^i$ by the alternative mappings $\mathbf{Y}$ and $\mathbf{X}^i$ introduced in §IV to obtain the PFFT method with potential matching technique.

## VI. Results

To present the numerical experiments more efficiently, it is important to recall that the auxiliary computational box containing the discretized boundary of the problem domain is formed with uniform cubes termed cells (see §III). On denoting by $d_c$ the diameter of a computational cell and assuming that $\mathbf{W}$ and $\mathbf{V}^i$ $(i = 1, 2, 3)$ are used as interpolation matrices, the PFFT method is mainly governed by three parameters, $m$, $p$, and $q$, that affect the accuracy, the memory requirements and the computational time of the algorithm. The parameter $m$ characterizes the number of computational cells in each coordinate direction and it is mostly responsible for the overall performance of the PFFT algorithm. An illustration of the behavior of $m$ will be exposed in the sequel. Further, $p$ represents the number of grid points per cell in each coordinate direction and significantly affects the accuracy of the PFFT

technique via the far-field approximation. In fact, it was shown in §III that $p$ corresponds to the degree of the interpolation functions used on the grid in each coordinate direction. The parameter $q$ can be introduced via the relation $R_c = q(d_c/2)$, where $R_c$ is the radius of the sphere defining the near-neighboring region in the precorrection process of the fast spectral algorithm. Namely, the near-neighboring region of a given non-empty cell in the PFFT method can be specified, in practice, by all non-empty cells contained in the closure of a ball with radius $R_c$ constructed from the center of the considered cell. For example, if $q = 1$, then the near-neighboring region of a non-empty cell is composed solely with the cell itself. If $q = 3$, then the near-neighboring region of a non-empty cell is formed not only with the considered cell itself but also with all non-empty cells that have a common vertex with the designated cell. This situation is sometimes referred to as the first nearest neighbors (see [24]). With these definitions, $q$ simply defines the near-neighboring region in the precorrection process of the PFFT algorithm and affects the memory and accuracy of the PFFT method via the conventional near-field calculation. In view of the method parameters, the polynomial-based precorrected-FFT algorithm will be denoted as PFFT($m, p, q$).

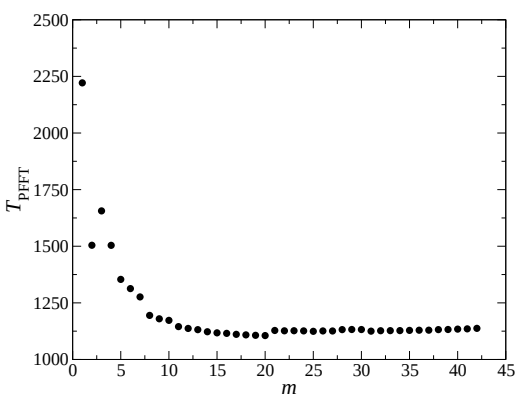


Fig. 5. Pre-processing time (in seconds) of the PFFT($m, 4, 3$).

In situations where the interpolation operators $\mathbf{Y}$ and $\mathbf{X}^i$ ($i = 1, 2, 3$) are employed in the precorrected-FFT algorithm, the fast spectral scheme will be denoted as PFFT($m, p, q, r_t, N_t$), where $r_t$ is the dimensionless radius of the test sphere defined via $R_t = r_t(d_c/2)$ and $N_t$ is the number of test points resting on the test sphere.

To illustrate the performance of the PFFT algorithm, a Dirichlet problem for the Laplace equation (1) has been solved on a unit sphere that is centered at the origin of a reference Cartesian frame. Namely, the following interior Dirichlet problem is considered everywhere in this study: solve the Laplace equation (1) in a unit ball $D = \{(x_1, x_2, x_3) \in \mathbb{R}^3 : x_1^2 + x_2^2 + x_3^2 < 1\}$ with boundary condition on a unit sphere $S$ specified as $u|_S = x_1^2 + x_2^2 - 2x_3^2$. Obviously, the solution everywhere in $D$ is $u(x_1, x_2, x_3) = x_1^2 + x_2^2 - 2x_3^2$, and the sought boundary flux is $t|_S = 2x_1^2 + 2x_2^2 - 4x_3^2$. Also, the Galerkin BEM of this study employs linear shape functions.

On employing the BiCGSTAB [17] method to solve iteratively the discretized BIE (10), the total solution time can be written as $T_{\mathrm{MAT}} + T_{\mathrm{IT}} + T_{\mathrm{RHS}}$ and $T_{\mathrm{PFFT}} + T_{\mathrm{IT}} + T_{\mathrm{RHS}}$ for the conventional approach and the PFFT technique respectively. Here $T_{\mathrm{MAT}}$ is the time needed to generate $\mathbf{G}$ and $\mathbf{H}$ influence matrices; $T_{\mathrm{PFFT}}$ is the pre-processing time by the PFFT method; $T_{\mathrm{IT}} = T_{\mathrm{BiCGSTAB(3)}}$ is the time consumed by the biconjugate gradient stabilized method, and $T_{\mathrm{RHS}}$ is the time necessary to compute the right-hand side of the linear system. Also, the BiCGSTAB(3) iterative solver is used without preconditioner in all numerical examples presented in this study. The three-dimensional FFT algorithm employed in this communication is an adaptation of the power of 2 algorithm from [22]. These calculations were all performed on a single Intel Xeon(EM64T) processor running at 3.2GHz with 1MB L2 cache of a dual CPU workstation with a total of 4GB DDR2 memory.

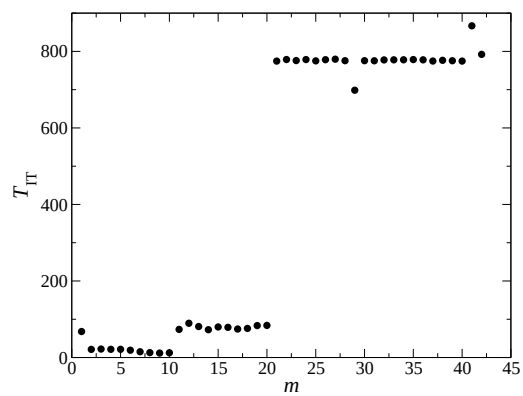


Fig. 6. Iteration time (in seconds) of the PFFT($m, 4, 3$) for various $m$.

To facilitate the presentation of the computational results, it is useful to monitor the behavior of the PFFT($m, 4, 3$) in terms of the running time as a function of the method parameter $m$. To this end, one is to solve the interior Dirichlet problem specified above. The unit sphere is discretized with 8188 triangles utilizing 4096 boundary nodes.

On the basis of the problem parameters set as $p = 4$ and $q = 3$, Fig. 5 shows the change of $T_{\mathrm{PFFT}}$ with respect to the method parameter $m$ that characterizes the discretization of the computational box. As can be seen from the figure, $T_{\mathrm{PFFT}}$ levels up to an asymptotic value as $m$ evolves. In contrast, Fig. 6 depicts the time used by the BiCGSTAB(3) iterative solver as $m$ gradually increases. It is seen from the figure that $T_{\mathrm{IT}}$ experiences a finite jump at certain values of $m$ that correspond to sharp increase in computer memory. This behavior of $T_{\mathrm{IT}}$ is due to the fact that as $m$ rises with $p$ fixed ($p = 4$), the total number of grid points, $Q = (m(p-1)+1)^3$, in the auxiliary computational box also rises. As a result, the computer memory needed to store the FFTs of $G$ and $\boldsymbol{H}$ kernels at all grid points suddenly increases. On Fig. 5 and Fig. 6, one could not go beyond $m = 42$ because of the computer memory limitation of 4GB of total RAM.

In what follows, Fig. 7 displays the total running time of the PFFT($m, 4, 3$) as a function of $m$. From this generic behavior, one can see that there exists $m$ (in this case $m = 10$) for which $T_{\mathrm{Total}}$ is minimized. However, for a range of $m$, $T_{\mathrm{Total}}$ remains around its minimal value and only the computer

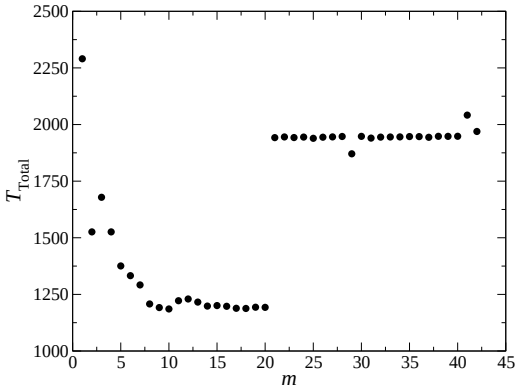


Fig. 7. Total time (in seconds) of the PFFT$(m, 4, 3)$ for various $m$.

memory is affected. With this latter fact, a useful $m$ can always be selected to benefit the most from the fast spectral method.

TABLE I
INTERIOR DIRICHLET PROBLEM ON A UNIT SPHERE (TIME IN SECONDS, MEMORY IN MEGABYTES).
Conventional approach

| $Nodes$ | $T_{\text{MAT}}$ | $T_{it}$ | $Total$ | $\text{L}_2$-$error$ | $Mem$ |
|---|---|---|---|---|---|
| 400 | 107 | 0 | 107 | $4.750\times10^{-3}$ | 3.87 |
| 1024 | 291 | 0 | 291 | $3.668\times10^{-3}$ | 17 |
| 2500 | 780 | 1 | 781 | $2.469\times10^{-3}$ | 97 |
| 4096 | 1411 | 3 | 1415 | $2.309\times10^{-3}$ | 258 |
| 6400 | 2565 | 8 | 2577 | $2.471\times10^{-3}$ | 628 |
| 8281 | 3483 | 13 | 3503 | $2.912\times10^{-3}$ | 1024 |
| 10000 | 4522 | 21 | 4554 | $2.795\times10^{-3}$ | 1536 |
| 16384 | 9237 | 58 | 9328 | $3.997\times10^{-3}$ | 4100.91 |
| 36864 | 33893 | 304 | 34377 | $--$ | 20794.76 |

TABLE II
INTERIOR DIRICHLET PROBLEM ON A UNIT SPHERE (TIME IN SECONDS, MEMORY IN MEGABYTES).
PFFT$(m, 4, 3)$

| $Nodes$ | $T_{\text{PFFT}}$ | $T_{it}$ | $Total$ | $\text{L}_2$-$error$ | $Mem$ | $m$ |
|---|---|---|---|---|---|---|
| 400 | 107 | 0 | 107 | $4.750\times10^{-3}$ | 7.07 | 2 |
| 1024 | 290 | 1 | 291 | $3.667\times10^{-3}$ | 28 | 2 |
| 2500 | 693 | 10 | 703 | $2.875\times10^{-3}$ | 73 | 10 |
| 4096 | 1173 | 12 | 1186 | $2.770\times10^{-3}$ | 142 | 10 |
| 6400 | 1922 | 21 | 1943 | $2.997\times10^{-3}$ | 283 | 10 |
| 8281 | 2309 | 88 | 2400 | $3.995\times10^{-3}$ | 332 | 17 |
| 10000 | 2778 | 104 | 2886 | $4.517\times10^{-3}$ | 332 | 20 |
| 16384 | 4700 | 125 | 4829 | $5.476\times10^{-3}$ | 618 | 20 |
| 36864 | 11729 | 254 | 11987 | $7.031\times10^{-3}$ | 2048 | 20 |

### A. PFFT method with polynomial-based interpolations

To contrast the traditional approach and the PFFT method, Table I and Table II show results for the interior Dirichlet problem with different discretizations. The last column of Table II represents values of $m$ for which the total computational time is minimized. It is seen from the tables that $T_{\text{PFFT}}$ is never greater than $T_{\text{MAT}}$ for all discretizations. In fact, the PFFT$(m, p, q)$ is designed to completely replace the traditional approach for solving the Laplace equation. Indeed, the PFFT$(1, p, q)$ corresponds to the conventional approach with a small overhead associated with the computation and storage of $\mathbf{W}$ and $\mathbf{V}^i$, and the FFTs of $G$ and $\boldsymbol{H}$ kernels at all grid points. One can also notice that the efficiency of the PFFT$(m, 4, 3)$ is achieved with the same level of accuracy as compared to the conventional approach. This accuracy is defined by the Euclidean norm of the discrepancy from an exact solution at all boundary points via $\text{L}_2$-$error = \frac{\|\boldsymbol{t}-\boldsymbol{t}_e\|}{\|\boldsymbol{t}_e\|}$, where $\boldsymbol{t}_e$ is a vector with entries generated from the known analytic flux $t|_S = 2x_1^2+2x_2^2-4x_3^2$, and the vector $\boldsymbol{t}$ is the numerical flux at all boundary nodes either by the conventional approach or by the PFFT method. From the columns displaying the $\text{L}_2$-$error$, one might conclude that both algorithms do not converge as discretization increases. This behavior is caused by the *near-singular* or *quasi-singular* integrals (integrals over pairs of triangular elements that are "very close") in the traditional Galerkin BEM. The PFFT simply mimics that behavior in the precorrection step. Analytic integration techniques are under development to deal with these quasi-singular integrals. The last two rows of Table I correspond to cases where the memory required for the calculations has exceeded the 4GB computer limit. In these situations, the running times and memories shown in the table were estimated via regression curves $c_1N^2 + c_2N$ and $a_1N^2 + a_2N$ respectively. Here $N$ is the number of boundary unknowns. The coefficients $c_1, c_2$ and $a_1, a_2$ were obtained via least squares approximations using data from Table I for boundary points up to 10000 nodes. Another regression curve $c_3N\ln N$ was constructed with all running times of Table II to compare the actual timings with the estimated curve. As can be seen from Fig. 8, the PFFT$(m, 4, 3)$ agrees extremely well with the $O(N\ln N)$ asymptotic curve. In view of the problem size which goes up to 36864 unknowns in this study, Fig. 8 also indicates that the PFFT$(m, 4, 3)$ agrees very well with a linear estimate $c_4N$, where $c_4$ is computed via a least squares technique. However, one should keep in mind that these estimate curves describe the asymptotic behavior (i.e., "large" $N$) of the fast spectral method and can be strongly affected by the geometry and topology of the domain under consideration. Nonetheless, these examples on a simple spherical geometry illustrate the potential behavior of the fast spectral method. On Fig. 9, it is seen that the PFFT method indeed scales well with the linear complexity curve $a_3N$, where $a_3$ is also obtained using a least squares approximation with Dirichlet data taken from Table II.

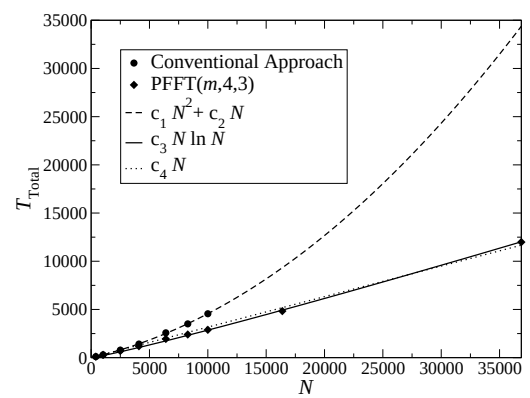


Fig. 8. Total running time (in seconds) for interior Dirichlet problems and estimate curves.

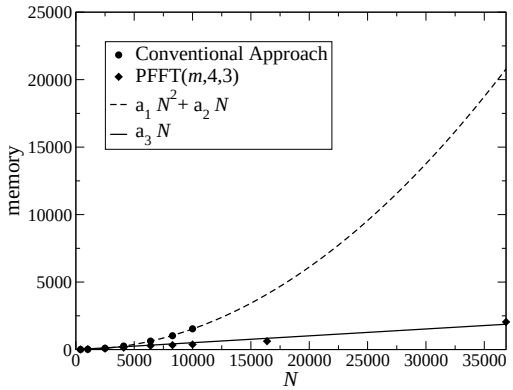


Fig. 9. Memory (in megabytes) used by the traditional and the PFFT methods and their regression curves for the interior Dirichlet problem.

By setting $p = 3$, Table III depicts computational results for the same Dirichlet problem, again employing only the first nearest neighbors. In the table, $m$ is selected so as to minimize the total running time. A comparison of Table I, Table II and Table III reveals a quick drop in accuracy of an order of magnitude. It follows from these observations that quadratic polynomials in every coordinate direction ($p = 2$) are still not sufficient to accurately capture the decaying kernels $G$ and $\boldsymbol{H}$ in the far-field. Table III also shows a great reduction in memory requirements wherein the Dirichlet problem with 36864 nodes can be solved using only 1.1GB of memory as compared to an estimated 20.3GB by the conventional approach.

TABLE III

INTERIOR DIRICHLET PROBLEM ON A UNIT SPHERE (TIME IN SECONDS, MEMORY IN MEGABYTES).

PFFT$(m, 3, 3)$

| $Nodes$ | $T_{\mathrm{PFFT}}$ | $T_{it}$ | $Total$ | $\mathrm{L}_2$-$error$ | $Mem$ | $m$ |
|---|---|---|---|---|---|---|
| 400 | 107 | 0 | 107 | $4.750\times10^{-3}$ | 6.47 | 2 |
| 1024 | 289 | 1 | 290 | $3.667\times10^{-3}$ | 26 | 2 |
| 2500 | 670 | 9 | 679 | $1.721\times10^{-2}$ | 48 | 15 |
| 4096 | 1113 | 11 | 1124 | $1.735\times10^{-2}$ | 83 | 15 |
| 6400 | 1770 | 15 | 1785 | $1.767\times10^{-2}$ | 150 | 15 |
| 8281 | 2314 | 18 | 2332 | $1.790\times10^{-2}$ | 222 | 15 |
| 10000 | 2849 | 22 | 2872 | $1.794\times10^{-2}$ | 303 | 15 |
| 16384 | 4502 | 120 | 4625 | $2.541\times10^{-2}$ | 421 | 31 |
| 36864 | 10619 | 184 | 10807 | $2.757\times10^{-2}$ | 1126.4 | 31 |

### B. PFFT method with potential matching interpolations

To provide further insight into the efficacy of the foregoing fast spectral method, it is useful to re-examine the numerical solution of the Dirichlet problem for the Laplace equation on the unit sphere in the context of the PFFT$(m, p, q, r_t, N_t)$. To this end, it is therefore important to rerun previous computational experiments and subsequently compare results with the conventional approach and the polynomial-based PFFT$(m, p, q)$ in terms of accuracy, memory and elapsed time. The dimensionless radius of the test sphere and the number of test points are specified respectively as $r_t = 3$ and $N_t = 100$.

TABLE IV

INTERIOR DIRICHLET PROBLEM ON A UNIT SPHERE (TIME IN SECONDS, MEMORY IN MEGABYTES).

PFFT$(m, 4, 3, 3, 100)$

| $Nodes$ | $T_{\mathrm{PFFT}}$ | $T_{it}$ | $Total$ | $\mathrm{L}_2$-$error$ | $Mem$ | $m$ |
|---|---|---|---|---|---|---|
| 400 | 107 | 0 | 107 | $4.750\times10^{-3}$ | 7.51 | 2 |
| 1024 | 290 | 1 | 291 | $3.669\times10^{-3}$ | 28 | 2 |
| 2500 | 691 | 9 | 700 | $2.511\times10^{-3}$ | 73 | 10 |
| 4096 | 1169 | 13 | 1183 | $2.361\times10^{-3}$ | 143 | 10 |
| 6400 | 1917 | 21 | 1939 | $2.535\times10^{-3}$ | 290 | 10 |
| 8281 | 2303 | 91 | 2397 | $3.036\times10^{-3}$ | 336 | 17 |
| 10000 | 2769 | 109 | 2882 | $2.944\times10^{-3}$ | 375 | 20 |
| 16384 | 4683 | 124 | 4811 | $4.085\times10^{-3}$ | 619 | 20 |
| 36864 | 11682 | 237 | 11924 | $5.821\times10^{-3}$ | 2048 | 20 |

The results are displayed on Table IV, where the last column contains values of $m$ for which the total computational time is minimized. A comparison of Table I, Table II and Table IV reveals that solutions obtained by the PFFT$(m, 4, 3, 3, 100)$ are more accurate than that of the PFFT$(m, 4, 3)$. Also, the discrepancy in running time between the two PFFT methods is not significant. Moreover, both PFFTs require about the same amount of memory. For problem sizes considered in this study, one can conclude that the computational cost and memory required by the singular value decomposition have a minimal effect on the overall efficiency of the PFFT$(m, 4, 3, 3, 100)$.

TABLE V

INTERIOR DIRICHLET PROBLEM ON A UNIT SPHERE (TIME IN SECONDS, MEMORY IN MEGABYTES).

PFFT$(m, 3, 3, 3, 100)$

| $Nodes$ | $T_{\mathrm{PFFT}}$ | $T_{it}$ | $Total$ | $\mathrm{L}_2$-$error$ | $Mem$ | $m$ |
|---|---|---|---|---|---|---|
| 400 | 107 | 0 | 107 | $4.750\times10^{-3}$ | 6.77 | 2 |
| 1024 | 289 | 1 | 290 | $3.667\times10^{-3}$ | 26 | 2 |
| 2500 | 670 | 7 | 677 | $7.473\times10^{-3}$ | 50 | 15 |
| 4096 | 1112 | 9 | 1121 | $8.653\times10^{-3}$ | 83 | 15 |
| 6400 | 1770 | 14 | 1784 | $1.027\times10^{-2}$ | 152 | 15 |
| 8281 | 2325 | 17 | 2343 | $1.101\times10^{-2}$ | 222 | 15 |
| 10000 | 2847 | 20 | 2867 | $1.161\times10^{-2}$ | 302 | 15 |
| 16384 | 4499 | 107 | 4609 | $1.685\times10^{-2}$ | 419 | 31 |
| 36864 | 10614 | 168 | 10786 | $2.079\times10^{-2}$ | 1126.4 | 31 |

The situation with $p=3$ is shown in Table V. A comparison with Table III again shows that the PFFT$(m, 3, 3, 3, 100)$ provides more accurate results than the PFFT$(m, 3, 3)$.

Numerical experiments have also been performed for a exterior Neumann problem on the unit sphere. The problem was simulated via a point source solution at the origin. In all aspects of the study, similar conclusions were obtained as in the case of the Dirichlet problem.

## VII. CONCLUSIONS

In this study, a fast spectral method to expedite the solution of singular BIEs is investigated within the framework of the precorrected-FFT technique rooted in capacitance extraction problems. To this end, the proposed method employs a regular Cartesian grid, the fast Fourier transform and boundary-to-grid interpolations to rapidly generate surface influences of the discretized problem in a sparse manner. The sparse representation of influences results in a significant memory reduction.

Further, the sparse influences are utilized in the BiCGSTAB iterative solver to quickly compute the solution of the problem.

The new algorithm, also referred to as precorrected-FFT method, is a systematic generalization of the capacitance extraction counterpart to deal with boundary value problems in the context of the direct and indirect boundary integral formulations. The proposed method can handle potential problems involving not only the single-layer potential kernel but also the double-layer kernel. It is founded on a more general mathematical framework and employs an improved interpolation scheme in the far-field treatment of boundary influences.

In contrast to the popular fast multipole method, the PFFT technique is relatively easy to implement and it is restricted only to problems featuring integral kernels that are associated with a convolution-type tensor. These integral kernels are defined as multilinear form of some vectors with coefficients given by components of the associated convolution-type tensor. In particular, the foregoing methodology is applicable to the hypersingular flux equation of potential theory wherein the hypersingular kernel is a bilinear form of unit normal vectors and it is associated to a symmetric rank 2 tensor.

Numerical experiments have demonstrated that the precorrected-FFT indeed accelerates the solution of singular BIEs while preserving the level of accuracy of the traditional approach. It was shown that the proposed method performs at best when the method parameter $m$, describing the number of computational cells in every coordinate direction, is selected so as to minimize the total computational time of the considered problem. At the moment, one does not have a tool to identify this optimum parameter a priori. However, it was also shown that for a wide range of $m$, the overall running time remains around its minimal value. This latter fact makes possible a selection of $m$ that is useful to efficiently resolve the problem. Computational details have additionally indicated that the PFFT method with local interpolation operators generated by the potential matching technique yields a solution that is more accurate than that of the simple polynomial-based PFFT algorithm. The memory requirements of the PFFT method was shown to scale linearly with the number of boundary unknowns.

## ACKNOWLEDGMENT

This document describes activities performed under contract number DE-AC0500OR22750 between the U.S. Department of Energy and Oak Ridge Associated Universities. All opinions expressed in this report are the authors' and do not necessarily reflect policies and views of the U.S. Department of Energy or the Oak Ridge Institute for Science and Education.

## REFERENCES

[1] P. K. Banerjee, *The Boundary Element Methods in Engineering*. London: McGraw-Hill, 1994.

[2] M. Bonnet, *Boundary Integral Equation Methods for Solids and Fluids*. New York: Wiley & Sons, 1995.

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

[4] N. Nishimura, "Fast multipole accelerate boundary integral equation methods," *ASME Appl.Mech. Rev.*, vol. 55, no. 4, pp. 299–324, 2002.

[5] J. R. Phillips and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 16, no. 10, pp. 1059–1072, 1997.

[6] A. P. Pierce and J. A. L. Napier, "A spectral multipole method for efficient solution of large-scale boundary element models in elastostatics," *Int. J. Numer. Meth. eng.*, vol. 38, no. 23, pp. 4009–4034, 1995.

[7] J. D. Richardson, L. J. Gray, T. Kaplan, and J. A. L. Napier, "Regularized spectral multipole BEM for plane elasticity," *Eng. Anal. with Boundary Elements*, vol. 25, pp. 297–311, 2001.

[8] A. N. Bespalov, "On the use of a regular grid for implementation of boundary integral methods for wave problems," *Russ. J. Numer. Anal. Math. Modelling*, vol. 15, no. 6, pp. 469–488, 2000.

[9] C. Lage, C.and Schwab, "Wavelet Galerkin algorithms for boundary integral equations," *SIAM J. Scient. Stat. Computing*, vol. 20, pp. 2195–2222, 1998.

[10] J. Tausch, "Sparse BEM for potential theory and Stokes flow using variable order wavelets," *Comp. Mech.*, vol. 32, no. 4-6, pp. 312–318, 2003.

[11] X.-C. Nie, L.-W. Li, and N. Yuan, "Precorrected-FFT algorithm for solving combined field integral equations in electromagnetic scattering," *J. of Electromagn. Waves and Appl.*, vol. 16, no. 8, pp. 1171–1187, 2002.

[12] N. Masters and W. Ye, "Fast BEM solution for coupled 3D electrostatic and linear elastic problems," *Eng. Anal. with Boundary Elements*, vol. 28, pp. 1175–1186, 2004.

[13] S. Tissari and J. Rahola, "A precorrected-FFT method to accelerate the solution of the forward problem in magnetoencephalography," *Phys. Med. Biol.*, vol. 48, pp. 523–541, 2003.

[14] L. J. Gray, J. M. Glaeser, and T. Kaplan, "Direct evaluation of hypersingular Galerkin surface integrals," *SIAM J. Sci. Comput.*, vol. 25, no. 5, pp. 1534–1556, 2004.

[15] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM J. Sci. Comput.*, vol. 14, no. 2, pp. 461–469, 1993.

[16] C. Van der Vorst, H. A.and Vuik, "GMRESR: A family of nested GMRES methods," *Num. lin. Alg. Appl.*, vol. 14, pp. 369–386, 1994.

[17] G. L. G. Sleijpen and D. R. Fokkema, "BiCGSTAB(*l*) for linear equations involving unsymmetric matrices with complex spectrum," *ETNA*, vol. 1, pp. 11–32, 1993.

[18] F. París and J. Canãs, *Boundary Element Method: Fundamentals and Applications*. New York: Oxford University Press, 1997.

[19] A. D. Lutz and L. J. Gray, "Exact evaluation of singular boundary integrals without CPV," *Comm. Num. Meth. Engrg.*, vol. 9, pp. 909–915, 1993.

[20] D. A. Dunavant, "High degree efficient symmetrical Gaussian quadrature rules for the triangle," *Int. J. Num. Meth. Eng.*, vol. 21, pp. 1129–1148, 1985.

[21] A. Brandt, "Multilevel computations of integral transforms and particle interactions with oscillatory kernels," *Comput. Phys. Commun.*, vol. 65, pp. 24–38, 1991.

[22] W. H. Press, S. A. Teukolsky, W. T. Vertterling, and B. P. Flannery, *Numerical Recipes: The art of scientific computing*. Cambridge: Cambridge University Press, 1986.

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimmore: John Hopkins, 1996.

[24] H. Hu, D. T. Blaauw, V. Zolotov, K. Gala, M. Zhao, R. Panda, and S. Sapatnekar, "Fast on-chip inductance simulation using a precorrected-FFT method," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 22, no. 1, pp. 49–66, 2003.