The accurate and efficient evaluation of Newtonian potentials over general 2-D domains is a subject of great importance for the numerical solution of Poisson's equation and volume integral equations. Complicated domains typically require discretization by unstructured meshes, over which the direct evaluation of the potential by quadrature becomes costly. In this paper, we present a simple and effective algorithm for computing Newtonian potentials, based on the use of Green's third identity for transforming the volume integral into a collection of boundary integrals, which can then be easily handled by the Helsing-Ojala method. As a result, the time cost of the classically expensive near field and self-interaction computations over an unstructured mesh becomes roughly the same as the time cost of the FMM-based far field interaction computation. One of the key components of our algorithm is the high-order 2-D monomial expansion approximation of a function over a mesh element, which is often regarded as an ill-conditioned problem, since it involves the solution of a Vandermonde linear system. In fact, it has long been observed that, when the function is sufficiently smooth, and when an appropriate linear system solver is used, the resulting monomial expansion can approximate the function uniformly to high accuracy. We rigorously formalize this observation in this paper. The performance of our algorithm is illustrated through several numerical experiments.

Keywords: Newtonian potential; Poisson's equation; Green's third identity; Vandermonde matrix; Monomial interpolation

Rapid evaluation of Newtonian potentials on planar domains

Zewen Shen[†⋆] and Kirill Serkh[‡⋄]

University of Toronto NA Technical Report
August 22, 2022

⋄ This author's work was supported in part by the NSERC Discovery Grants RGPIN-2020-06022 and DGECR-2020-00356.

† Dept. of Computer Science, University of Toronto, Toronto, ON M5S 2E4
‡ Dept. of Math. and Computer Science, University of Toronto, Toronto, ON M5S 2E4

⋆ Corresponding author

Contents

1 Introduction

The accurate and efficient discretization of the Newtonian potential integral operator

$$\mathcal{N}_\Omega[f](x) := \frac{1}{2\pi} \iint_\Omega \log(\|x-y\|) f(y)\, \mathrm{d}A_y \tag{1}$$

for a complicated 2-D domain Ω is of great importance for the numerical solution of Poisson's equation and volume integral equations. However, its numerical evaluation poses three main difficulties. Firstly, the integrand is weakly-singular, and thus, special-purpose quadrature rules are required. Secondly, a complicated domain Ω typically requires a discretization by an unstructured mesh, over which the direct evaluation of the potential by quadrature becomes costly, since the function spaces of near field and self-interactions are non-compact. Finally, the algorithm for evaluation should have linear time complexity with small constants. We refer the readers to [26] for a thorough discussion of these difficulties in the scenario in which the domain is discretized by an unstructured mesh.

When solving Poisson's equation, the Newtonian potential is used as a particular solution to the equation. This particular solution can be obtained by evaluating the volume integral in (1) directly (see, for example, [26, 1, 32, 24]), or, alternatively, be obtained by computing the Newtonian potential over a regular domain Ω^+ that contains the irregular domain Ω, which allows for efficient precomputations for accelerating the potential evaluation [10, 3]. Following the latter approach, the order of convergence depends on the smoothness of the density function f over the new domain Ω^+. Therefore, when f is only given over the original domain Ω, it is necessary to construct a smooth extension of f over Ω^+, in order to reach high accuracy within a reasonable computational budget. We refer the readers to, for example, [12, 3, 11, 8], for a series of work along this line.

When solving volume integral equations, the aforementioned function extension method is no longer applicable, as the computation does not require a particular solution to Poisson's equation, but rather, an accurate discretization of the operator (1). However, as is shown in [26, 1], difficulties arise when the domain is discretized by an unstructured mesh, and a quadrature-based method is used. Firstly, the Newtonian potential generated by a mesh element at a target location close to that element is costly to compute, as the integrand is nearly-singular, and thus, expensive adaptive integration is generally required. Furthermore, one cannot efficiently precompute these near interactions as is done in [10], since the relative position of the target location and the nearby mesh elements is arbitrary when an unstructured mesh is used. Secondly, efficient self-interaction computations (i.e., when the target location is inside the mesh element generating the Newtonian potential) generally require a large number of precomputed generalized Gaussian quadrature rules [6, 7], which could be nontrivial to construct.

There are several previously proposed methods [2, 25, 9, 28] which avoid these issues, by not directly evaluating the volume integral. One such method is the dual reciprocity method (DRM) [25], which first constructs a global approximation of the anti-Laplacian of the density function over the domain, and then reduces the evaluation of the Newtonian potential over the domain to the evaluation of layer potentials over the boundary of the domain by Green's third identity. As the 1-D layer potential evaluation problem has been studied extensively, such a reduction is favorable. Furthermore, the method does not require the domain to be meshed, and thus, is particularly suitable for use in the boundary integral equation method [30]. However, approximating the anti-Laplacian of the density function globally over the domain using, for example, radial basis functions, is challenging, and the method is often inefficient when high accuracy is required.

In this paper, we present a simple and effective algorithm that unifies the far, near and self-interaction computations, and resolves all of the aforementioned problems. As in the DRM, we use the anti-Laplacian to reduce the volume integral to a collection of boundary integrals. However, unlike the DRM, we approximate the anti-Laplacian locally over each mesh element, and then reduce the Newtonian potential to layer potentials over the boundaries of the individual mesh elements. We efficiently evaluate the resulting layer potentials to machine precision using the Helsing-Ojala method. As a result, we are able to rapidly evaluate the Newtonian potential generated by each mesh element at any target location to machine accuracy, with the speed of the evaluation independent of the target location. In particular, the speeds of close and self-evaluations for a single mesh element are almost the same as the speed of evaluating a layer potential over the

element boundary by naive quadrature. Furthermore, the use of Green's third identity reduces the number of quadrature nodes (for the far field interaction computation) over a single mesh element from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. When our algorithm is used to compute the Newtonian potential to machine precision over a general domain, the total time spent on calculating the near and self-interactions is roughly the same as the time spent on the far field interaction computation (i.e., the FMM computation). Finally, we note that the precomputation required by our algorithm is essentially negligible.

One may observe that our algorithm resembles the method proposed in Chapter 5 of [9]. There are two main differences between our algorithm and the method in [9]. One of the key differences is that [9] limits the order of their monomial expansions to 6th order, due to the ill-conditioning of the Vandermonde matrix, while our method allows for extremely high-order approximations. While the computation of the expansion coefficients involves the solution of a Vandermonde linear system, it has long been observed that, when the function being approximated is sufficiently smooth, and when an appropriate linear system solver is used, the resulting monomial expansion can approximate the function uniformly to high accuracy (see, for example, [20, 18]). In this paper, we formalize this observation by first characterizing the space of functions that can be efficiently approximated by monomial expansions, and then, by characterizing the properties that the linear system solver must be equipped with. Another key difference between our method and the one in [9] is the choice of discretization of a general 2-D domain Ω. In our method, we discretize the domain solely by (possibly curved) triangles, while in [9], the domain is discretized by the Cartesian cut cell method, where the potentials generated over the interior boxes are computed by the box code [10], and the ones generated over the cut cells are computed via Green's third identity. There are four advantages to solely using an unstructured mesh. Firstly, it substantially simplifies the implementation of the algorithm, as the box code is no longer required, and the evaluation of potentials generated over triangles and over curved triangles can be treated in exactly the same way in our algorithm. On the other hand, in [9], special care is required to handle various types of cut cells. Secondly, unstructured meshes allow for great flexibility in domain discretization, which is favorable, especially when the domain is complicated. Thirdly, with our algorithm, there are only $\mathcal{O}(N)$ quadrature nodes over the boundary of each mesh element for the far field interaction computation, while with the use of the box code, there are $\mathcal{O}(N^2)$ quadrature nodes inside each mesh element. Additionally, the use of a level restricted quadtree in the box code increases the number of boxes that are intrinsically unnecessary for the discretization. Finally, in our algorithm, the near field and self-interaction computations over an unstructured mesh are roughly as fast as the far field interaction computation, which is a nearly-optimal computational speed.

2 Mathematical and numerical preliminaries

2.1 Newtonian potential

Definition 2.1. The infinite-space Green's function for Poisson's equation is

$$G(x, y) = \frac{1}{2\pi} \log \|x - y\|, \tag{2}$$

where $x, y \in \mathbb{R}^2$.

It is well-known that the function G satisfies

$$\nabla_x^2 G(x,y) = \delta(x-y), \tag{3}$$

where δ denotes the Dirac delta function.

Definition 2.2. Given a domain Ω and an integrable function $f : \Omega \to \mathbb{R}$, the Newtonian potential with density f is defined to be

$$u(x) = \iint_\Omega G(x,y) f(y)\,\mathrm{d}A_y = \frac{1}{2\pi}\iint_\Omega \log(\|x-y\|) f(y)\,\mathrm{d}A_y. \tag{4}$$

It follows immediately from (3) that the Newtonian potential $u(x)$ satisfies $\nabla^2 u = f$ in Ω.

2.2 Numerical solution of ill-conditioned linear systems

When the truncated singular value decomposition (SVD) is used to solve a linear system, the norm of the solution and the size of the residual can be bounded independently of the condition number of the matrix. Below, we first give a definition of the truncated SVD that is used in our paper, and then describe these bounds.

Definition 2.3. Suppose that $A = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^*$ is the full SVD of an arbitrary matrix $A \in \mathbb{C}^{N\times N}$, where $\widetilde{\Sigma} = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_N)$. Then the truncated SVD of A is defined as

$$A = U\Sigma V^*, \tag{5}$$

where U, V equals the first n columns of $\widetilde{U}, \widetilde{V}$, $\Sigma = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_n)$, and n denotes the index of the smallest singular value of A that is larger than $3\varepsilon\|A\|_2$.

Theorem 2.1. *Suppose that $x \in \mathbb{C}^N$ satisfies $Ax = b$ for some $A \in \mathbb{C}^{N\times N}, b \in \mathbb{C}^N$. The solution $\widehat{x}$ computed by the truncated SVD in floating-point arithmetic satisfies*

$$\|\widehat{x}\|_2 < \frac{7}{3}\|x\|_2(1+\varepsilon), \tag{6}$$

where ε denotes machine precision.

Proof. Let $A = U\Sigma V^*$ and $\widehat{A} = \widehat{U}\widehat{\Sigma}\widehat{V}^*$ denote the truncated SVD of A in exact arithmetic and floating-point arithmetic, respectively, where the truncated SVD discards the singular vectors corresponding to singular values smaller than $3\varepsilon\|A\|_2$ from the factorization. It follows that $\widehat{U}$ and $\widehat{V}$ are matrices of size $N \times n$, and $\widehat{\Sigma} = \mathrm{diag}\,(\widehat{\sigma}_1, \widehat{\sigma}_2, \ldots, \widehat{\sigma}_n)$, where n denotes the index of the smallest singular value of A that is larger than $3\varepsilon\|A\|_2$. As $\widehat{V}$ and $\widehat{U}^*$ are orthogonal projection matrices up to machine precision, we have that

$$\|\widehat{x}\|_2 = \|\widehat{V}^*\widehat{x}\|_2 = \|\widehat{\Sigma}^{-1}\widehat{U}^*\widehat{b}\|_2 = \left\|\widehat{\Sigma}^{-1}(U^*b + \delta b)\right\|_2 \le \left\|\widehat{\Sigma}^{-1}U^*b\right\|_2 + \left\|\widehat{\Sigma}^{-1}\delta b\right\|_2, \tag{7}$$

for some $\delta b \in \mathbb{C}^n$ that satisfies $\|\delta b\|_2 \approx \varepsilon\|b\|_2$. Given the exact solution x, we can expand x in terms of the (full) right singular vectors of A

$$x = \sum_{i=1}^{N} c_i v_i, \tag{8}$$

for some $\{c_i\}_{i=1,2,\ldots,N}$ that satisfies $\|c\|_2 = \|x\|_2$, from which it follows that

$$(U^*b)_{(i)} = (\Sigma V^* x)_{(i)} = \sigma_i c_i, \tag{9}$$

for $i = 1, 2, \ldots, n$, where the subscript (i) denotes the ith entry of a vector. By the backward stability of the truncated SVD, we have that

$$\widehat{\Sigma}^{-1} = (\Sigma + \delta\Sigma)^{-1}, \tag{10}$$

for some diagonal matrix $\delta\Sigma = \mathrm{diag}\,(\delta\sigma_1, \delta\sigma_2, \ldots, \delta\sigma_n)$ that satisfies

$$|\delta\sigma_i| \approx \varepsilon \|\Sigma\|_2 = \varepsilon\sigma_1, \tag{11}$$

for $i = 1, \ldots, n$. Additionally, since the singular values smaller than $3\varepsilon\|S\|_2$ are discarded, we have that

$$\sigma_i + \delta\sigma_i > 3\varepsilon\|S\|_2 = 3\varepsilon\sigma_1, \tag{12}$$

and it follows from (11) that

$$\sigma_i > 2\varepsilon\sigma_1(1+\varepsilon), \tag{13}$$

for $i = 1, 2, \ldots, n$. Therefore, by (11), (13), the first term in the right hand side of (7) becomes

$$(\widehat{\Sigma}^{-1}U^*b)_{(i)} = \frac{\sigma_i}{\sigma_i + \delta\sigma_i}c_i = \frac{1}{1 + \delta\sigma_i/\sigma_i}c_i \le 2c_i(1+\varepsilon), \tag{14}$$

for $i = 1, 2, \ldots, n$, from which it follows that

$$\left\|\widehat{\Sigma}^{-1}U^*b\right\|_2 \le 2\|c\|_2 = 2\|x\|_2(1+\varepsilon). \tag{15}$$

By (12), the second term in the right hand side of (7) becomes

$$\|\widehat{\Sigma}^{-1}\delta b\|_2 \le \|\widehat{\Sigma}^{-1}\|_2\|\delta b\|_2 \le \frac{1}{3\varepsilon\sigma_1}\cdot\varepsilon\|b\|_2(1+\varepsilon) \le \frac{\|A\|_2}{3\sigma_1}\|x\|_2(1+\varepsilon) \le \frac{1}{3}\|x\|_2(1+\varepsilon). \tag{16}$$

By combining (7), (15) and (16), the theorem is proved. ■

Theorem 2.2. *Suppose that $A \in \mathbb{C}^{N\times N}$ and $b \in \mathbb{C}^N$ for some positive integer N. The solution $\widehat{x}$ to $Ax = b$ computed by a backward stable solver in floating-point arithmetic satisfies*

$$\|A\widehat{x} - b\|_2 < \varepsilon\|A\|_2\|\widehat{x}\|_2, \tag{17}$$

where ε is the machine precision.

Proof. By backward stability, we have that $\widehat{x}$ satisfies

$$(A + \delta A)\widehat{x} = b, \tag{18}$$

for some $\delta A \in \mathbb{C}^{N\times N}$ in exact arithmetic, where δA satisfies

$$\frac{\|\delta A\|_2}{\|A\|_2} \approx \varepsilon. \tag{19}$$

Therefore, we have that

$$\|A\widehat{x} - b\|_2 = \|\delta A\cdot\widehat{x}\|_2 \le \varepsilon\|A\|_2\|\widehat{x}\|_2. \tag{20}$$

■

2.3 Green's third identity

In this section, we describe Green's third identity, and its proof can be found in, for example, [9].

Theorem 2.3. *Let Ω be a 2-D planar domain and f be an integrable function on Ω. Suppose that $\varphi : \Omega \to \mathbb{R}$ satisfies $\nabla^2 \varphi = f$. Then,*

$$\iint_{\Omega} G(x,y) f(y) \, \mathrm{d}A_y = \varphi(x) \mathbb{1}_{\Omega}(x) + \oint_{\partial\Omega} \Big(G(x,y) \frac{\partial \varphi}{\partial n_y}(y) - \frac{\partial G(x,y)}{\partial n_y} \varphi(y) \Big) \, \mathrm{d}\ell_y, \quad (21)$$

for $x \in \mathbb{R}^2 \setminus \partial\Omega$, where $\mathbb{1}_{\Omega}$ denotes the indicator function for the domain Ω, and n_y denotes the outward pointing unit normal vector at the point y.

2.4 The Helsing-Ojala method for evaluating 1-D layer potentials

In this section, we discuss the evaluation of the 1-D single- and double-layer potentials

$$\int_{\Gamma} G(x,y) \frac{\partial \varphi}{\partial n_y}(y) \, \mathrm{d}\ell_y \quad (22)$$

and

$$\int_{\Gamma} \frac{\partial G(x,y)}{\partial n_y} \varphi(y) \, \mathrm{d}\ell_y, \quad (23)$$

where Γ is a curve in $\mathbb{R}^2$, and $x \in \mathbb{R}^2$. When x is far away from Γ, these layer potentials can be efficiently integrated by the Gauss-Legendre rule to high accuracy. When x is close to Γ, however, the integrand becomes nearly-singular and, thus, the standard quadrature rules cannot be used to compute the integral efficiently. Below, we present the Helsing-Ojala method [20] for the efficient and accurate close evaluation of layer potentials.

Firstly, observe that

$$\int_{\Gamma} G(x,y) \frac{\partial \varphi}{\partial n_y}(y) \, \mathrm{d}\ell_y = \frac{1}{2\pi} \operatorname{Re} \int_{\Gamma} \log(z - x) \cdot \frac{\partial \varphi}{\partial n_y}(z) \cdot \frac{\mathrm{d}\ell_y}{\mathrm{d}z} \cdot \mathrm{d}z, \quad (24)$$

and

$$\int_{\Gamma} \frac{\partial G(x,y)}{\partial n_y} \varphi(y) \, \mathrm{d}\ell_y = \operatorname{Re} \frac{1}{2\pi i} \int_{\Gamma} \frac{\varphi(z)}{z - x} \, \mathrm{d}z, \quad (25)$$

where, in a slight abuse of notation, we equate $\mathbb{R}^2$ with $\mathbb{C}$. One can derive a recurrence relation such that the values

$$\Big\{ \int_{\Gamma} z^k \log(z - x) \, \mathrm{d}z \Big\}_{k=0,1,\ldots,n} \quad \text{and} \quad \Big\{ \int_{\Gamma} \frac{z^k}{z - x} \, \mathrm{d}z \Big\}_{k=0,1,\ldots,n} \quad (26)$$

can be computed to high accuracy in $\mathcal{O}(n)$ operations for x arbitrarily close to Γ, where the base case $k = 0$ are given by analytical formulas. Then, provided that the complex density functions $\frac{\partial \varphi}{\partial n_y}(z) \cdot \frac{\mathrm{d}\ell_y}{\mathrm{d}z}$ and $\varphi(z)$ are expressed in terms of 1-D complex monomial expansions, the single- and double-layer potentials (22) and (23) can be readily evaluated

via (24) and (25), respectively. To approximate the density functions $\frac{\partial \varphi}{\partial n_y}(z) \cdot \frac{\mathrm{d}\ell_y}{\mathrm{d}z}$ and $\varphi(z)$ by a monomial expansion, one collocates at the Gauss-Legendre nodes over Γ, and then solves the resulting linear system with a backward stable solver. When the density functions are sufficiently smooth, the residual of the linear system is around machine epsilon and the magnitudes of the monomial coefficients are of order one. In Section 3, we present a rigorous justification for the use of the monomial expansion approximation.

We refer the readers to [18, 29, 20] for a detailed discussion and implementation of the algorithm.

Remark 2.1. One could store the factorization of the Vandermonde matrix with the Gauss-Legendre collocation nodes on $[-1, 1]$. This factorization can be used to accelerate the cost of constructing 1-D monomial expansion from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ when Γ is a straight line segment.

3 Monomial expansion approximation of functions

In this section, we discuss the feasibility of approximating functions by monomial expansions. For simplicity, we restrict our discussion to the 1-D case, although the results can be trivially extended to higher dimensions.

Given a function $F : [-1, 1] \to \mathbb{R}$, we are interested in computing its Nth order Taylor expansion approximation

$$F(x) \approx \sum_{i=0}^{N} a_i x^i, \tag{27}$$

where "$\approx$" denotes the equality in the L_∞ norm within an error of size machine epsilon. The collocation method is one of the most natural ways for approaching this problem. More specifically, given a set of $(N + 1)$ collocation points over $[-1, 1]$

$$\{x_j\}_{j=0,1,\ldots,N}, \tag{28}$$

we solve for the solution to the system of equations

$$\sum_{i=0}^{N} a_i x_j^i = F(x_j), \tag{29}$$

for $j = 0, 1, \ldots, N$. In matrix form, the system of equations is written as

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^N \\ 1 & x_1 & x_1^2 & \cdots & x_1^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^N \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} F(x_0) \\ F(x_1) \\ \vdots \\ F(x_N) \end{pmatrix}, \tag{30}$$

and we denote the matrix by S, the right hand side vector by f, and the solution by a. When floating-point arithmetic is used, we use a hat to denote the numerically computed quantity. It is easy to observe that S is the Vandermonde matrix, and thus, for any set of real collocation points, its condition number grows exponentially in the

order of the monomial basis. Using exact arithmetic, the computed Nth order Taylor expansion matches the polynomial interpolant of degree N for the set of collocation points $\{x_i\}_{i=0,1,\ldots,N}$. The large condition number of S would seem to imply that it is infeasible to accurately compute the solution $\widehat{a}$. However, it is still possible to obtain an accurate numerical approximation of the function F, if F is sufficiently smooth and an appropriate linear solver is used. Below, we rigorously characterize this argument.

Theorem 3.1. *Suppose that S is a Vandermonde matrix of size $(N+1)\times(N+1)$ with the set of collocation nodes $X := \{x_0, x_1, \ldots, x_N\}$ over $[-1,1]$. Let Λ_N denote the Lebesgue constant for the Nth order polynomial interpolation in the set of nodes X. Given an arbitrary function $F : [-1,1] \to \mathbb{R}$,*

$$\begin{aligned}\left\|F_N - F\right\|_{L^\infty([-1,1])} &\leq (\Lambda_N + 1)\|f_N - f\|_\infty + \left\|P_N - F\right\|_{L^\infty([-1,1])} &(31)\\ &< \varepsilon(\Lambda_N+1)\|S\|_2\|\widehat{a}\|_2 + \left\|P_N - F\right\|_{L^\infty([-1,1])} &(32)\\ &< \frac{7}{3}\varepsilon(\Lambda_N+1)\|S\|_2\|a\|_2(1+\varepsilon) + \left\|P_N - F\right\|_{L^\infty([-1,1])}, &(33)\end{aligned}$$

where ε denotes machine precision, $f := (F(x_0), F(x_1), \ldots, F(x_N))$, $F_N(x) := \sum_{i=0}^N \widehat{a}_i x^i$ is the monomial expansion computed by solving the Vandermonde system with the truncated SVD in floating-point arithmetic, $f_N := (F_N(x_0), F_N(x_1), \ldots, F_N(x_N))$, P_N denotes the Nth order polynomial interpolant of the function F with interpolation nodes $x_0, x_1, \ldots, x_N$ in exact arithmetic, and a denotes the monomial coefficients of P_N.

Proof. Since the truncated SVD is backward stable, by Theorem 2.2,

$$\|f_N - f\|_2 = \|S\widehat{a} - f\|_2 < \varepsilon\|S\|_2\|\widehat{a}\|_2. \tag{34}$$

Therefore, by definition of the Lebesgue constant Λ_N, we have that

$$\|F_N - P_N\|_{L^\infty([-1,1])} \leq (\Lambda_N+1)\|f_N - f\|_\infty < \varepsilon(\Lambda_N+1)\|S\|_2\|\widehat{a}\|_2. \tag{35}$$

Finally, by Theorem 2.1 and the triangle inequality, we have that

$$\begin{aligned}\left\|F_N - F\right\|_{L^\infty([-1,1])} &\leq \left\|F_N - P_N\right\|_{L^\infty([-1,1])} + \left\|P_N - F\right\|_{L^\infty([-1,1])}\\ &< \varepsilon(\Lambda_N+1)\|S\|_2\|\widehat{a}\|_2 + \left\|P_N - F\right\|_{L^\infty([-1,1])}\\ &< \frac{7}{3}\varepsilon(\Lambda_N+1)\|S\|_2\|a\|_2(1+\varepsilon) + \left\|P_N - F\right\|_{L^\infty([-1,1])}, &(36)\end{aligned}$$

where $a \in \mathbb{R}^{N+1}$ denotes the monomial coefficients of the interpolant P_N. ■

Remark 3.1. The inequalities (31) and (32) in Theorem 3.1 can both be used as a posteriori error estimates of the L^∞ error of the monomial expansion approximation. The former one gives a tight bound, but requires the computation of the residual, while the latter one gives a looser bound that is essentially free to compute. We note that when the collocation nodes are chosen to be, for example, the Chebyshev nodes or the Gauss-Legendre nodes, the terms $\|S\|_2$ and Λ_N in (31) and (32) can be easily estimated.

The following corollary shows that, when the collocation nodes are chosen to be Chebyshev nodes, the error of the monomial expansion approximation is bounded by an expression involving the Chebyshev interpolation error and the norm of monomial coefficients.

Corollary 3.2. *Given an arbitrary function $F : [-1, 1] \to \mathbb{R}$, we have that*

$$\big\|F_N - F\big\|_{L^\infty([-1,1])} < \varepsilon \|a\|_2 \cdot 6\big(1 + \frac{1}{\pi}\log(N+1)\big)\sqrt{N+1} + \big\|P_N - F\big\|_{L^\infty([-1,1])}, \tag{37}$$

where ε denotes machine precision, $F_N(x) := \sum_{i=0}^{N} \widehat{a}_i x^i$ is the monomial expansion computed by solving the Vandermonde system with $N+1$ Chebyshev collocation nodes using the truncated SVD in floating-point arithmetic, P_N denotes the Nth order Chebyshev interpolant of the function F in exact arithmetic, and a denotes the monomial coefficients of P_N.

Proof. The proof directly follows from Theorem 3.1, combined with the additional fact that, when the Chebyshev nodes are used as the interpolation nodes,

$$\Lambda_N \leq \frac{2}{\pi}\log(N+1) + 1, \tag{38}$$

and

$$\|S\|_2 \approx \sqrt{N+1}. \tag{39}$$

We note that the estimate is deliberately loosened to simplify the expression. ■

Observation 3.2. In all of the previous theorems in this section, we assume that the linear system is solved by the truncated SVD, which is known to be costly to compute. It has long been observed that the truncated pivoted QR factorization manifests similar properties as the truncated SVD, and its computational cost is lower. Alternatively, one could pollute the Vandermonde matrix by adding to it a random matrix with 2-norm ε, and use a pivoted LU factorization to solve the linear system. The singular values of the perturbed matrix are modified by a perturbation bounded by ε. Based on extensive numerical experiments, we find that the last approach leads to a good monomial expansion approximation whenever such an approximation is possible, despite that there is no theoretical guarantee.

Observation 3.3. Based on extensive numerical experiments, we make the following conjecture. Given $F : [-1, 1] \to \mathbb{R}$, suppose that $F(x) := \sum_{i=0}^{N} c_i P_i(x)$, where P_i denotes the ith order Legendre polynomial with unit L_2 norm. If

$$|c_i| < 2^{-i}, \tag{40}$$

for $i = 0, 1, \ldots, N$, then the 2-norm of the coefficients in the monomial expansion of of F grows slowly. Combined with the inequality (37) in Corollary 3.2, this means that the convergence of the monomial expansion approximation is essentially as fast as the convergence of the Legendre polynomial approximation.

Remark 3.4. Analogs of Theorem 3.1 and Corollary 3.2 similarly hold for 2-D monomial interpolation.

Observation 3.5. The restriction of a 2-D monomial expansion of order N centered at the origin to a straight line segment results in a 1-D monomial expansion of the same order N. However, when the distance between the origin and the line segment is large, the resulting 1-D monomial expansion could have large coefficients. The restriction of a 2-D monomial expansion of order N centered at the origin to a sufficiently smooth curve likewise results in a 1-D monomial expansion of order slightly larger than N.

Remark 3.6. It is well known that the Björck-Pereyra method [5] solves the 1-D Vandermonde linear system in $\mathcal{O}(N^2)$ operations. In [22], the author shows that under certain assumptions, this algorithm computes the monomial expansion coefficients of a given function to high accuracy by using the Björck-Pereyra method. However, the assumptions made in [22] do not hold in the context of this paper. Furthermore, the proof cannot be directly generalized to higher dimensions.

4 Numerical algorithm

In this section, we introduce an algorithm for computing the Newtonian potential (4) when Ω is a (possibly curved) triangle. Then, we describe how to apply the algorithm to compute the Newtonian potential over a general integration domain. Firstly, we give an overview of the method.

Given a density function f over a (possibly curved) triangle, we first compute its anti-Laplacian. Then, by Green's third identity, we reduce the evaluation of the Newtonian potential with density f to the evaluation of 1-D layer potentials over the edges of the triangle, with density functions given by the traces of the anti-Laplacian of f, and its normal derivatives. Then, we evaluate the 1-D layer potentials using the Helsing-Ojala method [20]. We note that the precomputation required by the algorithm is fast, and, as a result, the near field and self-interactions generated over a triangle can be evaluated almost instantaneously.

Given an arbitrary 2-D domain, one can discretize it into triangles and curved triangles, and compute the Newtonian potential generated over that domain efficiently and accurately by combining the fast multipole method [17] with the algorithm presented in this Section (see [26, 1] for examples of volume potential evaluation frameworks).

In the rest of the section, we assume that $\overline{\Delta}$ is a triangle whose bounding circle is the unit circle centered at the origin of $\mathbb{R}^2$.

4.1 Construction of the anti-Laplacian

Given a density function $f : \overline{\Delta} \to \mathbb{R}$, we present an algorithm for computing the anti-Laplacian of f.

Step I: Precompute the anti-Laplacian mapping in the 2-D monomial basis. With a slight abuse of notation, we denote the anti-Laplacian by ∇^{-2}. It is easy to verify

that

$$\nabla^{-2}[x^m y^n] = \frac{x^{m+2}y^n}{(m+2)(m+1)} - \frac{n(n-1)}{(m+2)(m+1)}\nabla^{-2}[x^{m+2}y^{n-2}] \tag{41}$$

$$= \frac{x^m y^{n+2}}{(n+2)(n+1)} - \frac{m(m-1)}{(n+2)(n+1)}\nabla^{-2}[x^{m-2}y^{n+2}], \tag{42}$$

for nonnegative integers m, n where $m > 1$ or $n > 1$. We note that $\nabla^{-2}[x^m y^n]$ for $m, n \in \{0, 1\}$ is trivial to compute. Clearly, the identity (41) is preferable to use when $m \geq n$, and vice versa for (42). Thus, one can use the recurrence to precompute a mapping between the coefficients of a 2-D Nth order monomial expansion and the 2-D $(N+2)$th order monomial expansion of its anti-Laplacian. We note that this mapping only has $\mathcal{O}(N^2)$ non-zero entries, and should be stored in a sparse format.

Step II: Compute the 2-D monomial expansion of f. To represent the density function f in terms of a 2-D monomial expansion, it is important to choose a set of well-conditioned collocation points. When the domain is a triangle, the set of Vioreanu-Rokhlin nodes [31] is often a good choice. Let

$$\{(x_i, y_i)\}_{i=1,2,\ldots,M} \tag{43}$$

denote the Vioreanu-Rokhlin nodes of order N with length M over the triangle $\overline{\Delta}$. We compute the 2-D monomial expansion coefficients of f by collocation at these nodes. This involves solving a 2-D Vandermonde matrix system, which is famous for its exponentially growing condition number in the order of the monomial basis. In fact, it has long been observed that, when the function is sufficiently smooth, and when an appropriate linear system solver is used, the resulting monomial expansion can approximate the function uniformly to high accuracy [20, 18]. We provide a rigorous justification for this observation in Section 3.

Step III: Compute $\nabla^{-2}[f]$ using the 2-D monomial expansion of f. In Step II, we compute the expansion coefficients $\{a_{mn}\}$ such that

$$f(x, y) \approx \sum_{m=0}^{N} \sum_{n=0}^{m} a_{m,n} x^{m-n} y^n. \tag{44}$$

Then, $\nabla^{-2}[f]$ can be computed by applying the precomputed anti-Laplacian mapping to the expansion coefficients. Computing the anti-Laplacian requires only linear time complexity in the number of expansion terms, due to the sparsity of the mapping.

4.2 Close evaluation of 1-D layer potentials

In the previous section, we compute the 2-D monomial expansion of the anti-Laplacian φ of the density function f. Recall from Theorem 2.3 that the Newtonian potential can be written as

$$\iint_{\overline{\Delta}} G(x, y) f(y) \, \mathrm{d}A_y = \varphi(x) \mathbb{1}_{\overline{\Delta}}(x) + \oint_{\partial\overline{\Delta}} \Big(G(x, y) \frac{\partial \varphi}{\partial n_y}(y) - \frac{\partial G(x, y)}{\partial n_y} \varphi(y) \Big) \, \mathrm{d}S_y. \tag{45}$$

When φ is represented as a 2-D monomial expansion, its gradient can be computed analytically, from which it follows that its normal derivative $\frac{\partial \varphi}{\partial n_y}$ can be evaluated easily.

Remark 4.1. Evaluating a 2-D monomial expansion of order N naively requires $\mathcal{O}(N^3)$ multiplications. However, provided that one reuses the previously computed values of x^m, y^n, the evaluation only requires $\mathcal{O}(N^2)$ multiplications. We also note that Horner's method is the optimal algorithm for evaluating monomial expansions.

Thus, it remains to compute the layer potentials

$$\int_{L_i} G(x,y)\frac{\partial \varphi}{\partial n_y}(y)\,\mathrm{d}\ell_y \tag{46}$$

and

$$\int_{L_i} \frac{\partial G(x,y)}{\partial n_y}\varphi(y)\,\mathrm{d}\ell_y, \tag{47}$$

where L_1, L_2, L_3 denote the edges of $\overline{\Delta}$. When x is close to L_i, we use the Helsing-Ojala method (see Section 2.4) to handle the nearly-singular integrands.

Remark 4.2. By default, the branch cut of the potential computed by the Helsing-Ojala method is the straight line segment connecting the two endpoints of the integration domain L_i. Therefore, in the case of a triangle domain, the computed branch cut matches the true branch cut exactly.

We note that, when the density function f is an Nth order 2-D monomial expansion, the restriction of its anti-Laplacian to the edge L_i, $\varphi|_{L_i}$, is an $(N+2)$th order 1-D monomial expansion. It follows that, when we compute that 1-D monomial expansion, it is sufficient for us to set the order of the expansion to $N+2$ by collocating it at $N+3$ Gauss-Legendre nodes on the edge. We also note that the Helsing-Ojala method is only suitable for evaluating the layer potential near the edge, as the recurrence relations for computing (26) become increasingly unstable when x gets further away from a bounding circle of L_i centered at its midpoint. In our implementation, when x is inside the bounding circle with a diameter $1.3|L_i|$, we use the Helsing-Ojala method to evaluate the layer potentials at x; otherwise, we use the Gauss-Legendre rule of order $2(N+3)$ to evaluate the integrals (such that the values of φ and $\frac{\partial \varphi}{\partial n_y}$ at the $N+3$ Gauss-Legendre nodes are reused).

We now discuss the changes to be introduced when the integration domain of the Newtonian potential is an arbitrary triangle Δ. Since the 2-D monomial basis is the most well-conditioned inside a unit circle centered at the origin, we introduce the transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$,

$$T(x) := (x - C(\Delta))/R(\Delta), \tag{48}$$

where $C(\Delta)$ and $R(\Delta)$ denote the center and the radius of the bounding circle of Δ, respectively. Furthermore, we define $\overline{\Delta} := T(\Delta)$, and it is clear that the bounding circle of $\overline{\Delta}$ is the unit circle centered at the origin. It is also clear that $T^{-1}(\widetilde{x}) = R(\Delta)\widetilde{x} + C(\Delta)$. To express the Newtonian potential over Δ in terms of a Newtonian potential over $\overline{\Delta}$, we

proceed as follows. Define $\widetilde{x} := T(x)$. Then, by change of variables, we have that

$$\begin{aligned}
&\iint_{\Delta} G(x,y) f(y)\,\mathrm{d}A_y = \frac{1}{2\pi}\iint_{\Delta} \log\|x-y\| f(y)\,\mathrm{d}A_y \\
&= \frac{R(\Delta)^2}{2\pi}\iint_{\overline{\Delta}} \log\|T^{-1}(\widetilde{x}) - T^{-1}(\widetilde{y})\| \cdot f\circ T^{-1}(\widetilde{y})\,\mathrm{d}A_{\widetilde{y}} \\
&= \frac{R(\Delta)^2}{2\pi}\iint_{\overline{\Delta}} \log\|R(\Delta)\cdot(\widetilde{x}-\widetilde{y})\| \cdot f\circ T^{-1}(\widetilde{y})\,\mathrm{d}A_{\widetilde{y}} \\
&= \frac{\log\|R(\Delta)\|}{2\pi}\iint_{\Delta} f(y)\,\mathrm{d}A_y + R(\Delta)^2\iint_{\overline{\Delta}} G(\widetilde{x},\widetilde{y})\cdot f\circ T^{-1}(\widetilde{y})\,\mathrm{d}A_{\widetilde{y}} \\
&= R(\Delta)^2\cdot\Big(\frac{\log\|R(\Delta)\|}{2\pi}\iint_{\partial\overline{\Delta}} \frac{\partial\varphi(\widetilde{y})}{\partial n_{\widetilde{y}}}\,\mathrm{d}S_{\widetilde{y}} + \iint_{\overline{\Delta}} G(\widetilde{x},\widetilde{y})\cdot f\circ T^{-1}(\widetilde{y})\,\mathrm{d}A_{\widetilde{y}}\Big),
\end{aligned} \tag{49}$$

where the last equality follows from the 2-D divergence theorem. The Newtonian potential in the second term of (49) can be evaluated by the algorithm presented at the beginning of this section, as the integration domain $\overline{\Delta}$ is now a triangle whose bounding circle is a unit circle centered at the origin. Thus, one can rapidly evaluate the Newtonian potential generated over a triangle Δ accurately everywhere in $\mathbb{R}^2 \setminus \partial\Delta$.

4.3 Generalization to a curved triangle domain

In this Section, we discuss the generalization of the algorithm to the case where the integration domain is a curved triangle. We focus on the case where the curved triangle has only one curved side.

Given an arbitrary curved triangle $\widetilde{\Delta}$ with only one curved side, let $\gamma : [0, L] \to \mathbb{R}^2$ be the parametrization of the curved side of $\widetilde{\Delta}$, and let $O := (x_0, y_0)$ denote the vertex opposite to the curved side. Without loss of generality, we assume that γ is contained in the unit disk centered at the origin. We use the blending function method [13] to construct a well-conditioned mapping $\rho : \Delta^0 \to \widetilde{\Delta}$, where

$$\Delta^0 = \{(x,y) \in \mathbb{R}^2 : 0 \le x \le 1, 0 \le y \le 1-x\}, \tag{50}$$

and

$$\begin{aligned}
\rho(\xi,\eta) := {} & (1-\xi-\eta)\cdot\gamma(L) + \xi\cdot\gamma(0) + \eta\cdot O \\
& + \frac{1-\xi-\eta}{1-\xi}\Big(\gamma\big(L(1-\xi)\big) - (1-\xi)\cdot\gamma(L) - \xi\cdot\gamma(0)\Big).
\end{aligned} \tag{51}$$

4.3.1 Construction of the anti-Laplacian

Clearly, Steps I and III in Section 4.1 are oblivious to the geometry of the domain. In this Section, we discuss the modification to Step II when the domain is a curved triangle. To obtain a set of collocation points over a curved triangle $\widetilde{\Delta}$, we map the Vioreanu-Rokhlin nodes over Δ^0 to $\widetilde{\Delta}$ via (51), and since both the mapping and the set of Vioreanu-Rokhlin nodes are well-conditioned over Δ^0, this results in a set of well-conditioned collocation points over $\widetilde{\Delta}$.

4.3.2 Close evaluation of 1-D layer potentials

To compute the Newtonian potential on a curved triangle, one needs to evaluate the layer potentials over a curved edge γ

$$\oint_{\gamma} \Big(G(x,y) \frac{\partial \varphi}{\partial n_y}(y) - \frac{\partial G(x,y)}{\partial n_y} \varphi(y) \Big) \, \mathrm{d}S_y. \tag{52}$$

The Helsing-Ojala method is capable of handling the curved edge case when the curvature is small [18]. All that remains is to put the branch cut of the layer potential in the correct location (see Remark 4.2). In the original paper [20], the authors propose a method that makes a correction to the branch cut when the integration domain γ is a curve, and is based on a simple computational geometric algorithm. However, such an approach could be unstable when the target location lies exactly (up to machine precision) on the reference branch cut, due to round-off error. In Section 3.3.1 of [29], the authors proposed an effective technique for rotating the branch cut, such that it is pushed "behind" the curve. When the target location is assumed to be inside the domain, this technique guarantees that the correct branch cut is always used during self-interaction computations. When the target location is outside of the curved triangle and close to the curved edge, the branch cut should be rotated in the opposite direction.

In addition, as is noted in Observation 3.5, one should slightly increase the number of the Gauss-Legendre nodes required to evaluate the layer potential in the far field, as the restriction of a 2-D monomial expansion over a smooth curve often results in a 1-D monomial expansion with a higher order. In practice, we find that an increase in the order by three is often sufficient.

4.4 Generalization to a general domain

In this section, we discuss the generalization of the algorithm to the case where the domain is a general 2-D geometry, with an emphasis on the distinct features of our algorithm that the generalization should account for.

Given an arbitrary 2-D planar domain Ω, we first discretize it into triangles and curved triangles (see, for example, Appendix A in [26]). Then, we construct the anti-Laplacian of the density function in the form of a 2-D monomial expansion for each mesh element. We also compute the 1-D monomial expansion coefficients for the restriction of the anti-Laplacian and its normal derivatives to the edges of each element. At this stage, all of the required precomputations are done.

Then, we use the point-based fast multipole method to compute the far field interactions. Typically, one uses 2-D quadrature rules over triangles to compute the far field interactions generated over mesh elements (see, for example, [26, 1]), and thus, the number of quadrature nodes over each element for computing far field interactions is of order $\mathcal{O}(N^2)$, where N is the order of the 2-D monomial expansion for the density function. We note, however, that in the far field, the layer potentials in Green's third identity (21) can be computed efficiently and accurately by Gauss-Legendre rules. It follows that, in our algorithm, the number of quadrature nodes over each element is of order $\mathcal{O}(N)$, since the nodes are now located on the edges, rather than in the interiors. Finally, we compute the near and self-interactions using the algorithm presented in Section 4.2.

Remark 4.3. Compared to the volume-based far field interaction computation, the edge-based one reduces the dimensionality of the problem, at the cost of making the accuracy of the far field interactions dependent on the approximation error of the anti-Laplacian of the density function. We note that such a trade-off is desirable, because an accurate approximation of the anti-Laplacian is essential for accurate and efficient near and self-interaction computations in our algorithm.

Remark 4.4. By Green's third identity (21), it would appear that the Newtonian potential evaluation over a general domain Ω can be reduced to the evaluation of two layer potentials over the boundary of the domain $\partial\Omega$, plus the evaluation of the anti-Laplacian of the density function at the target location. This is only true provided that the anti-Laplacian is twice-continuously differentiable over the whole domain Ω. However, the anti-Laplacian constructed using our method is only twice-continuously differentiable inside each mesh element, from which it follows that we must compute the contribution from each mesh element separately.

Remark 4.5. Given two adjacent mesh elements, their collocation nodes over the common edge coincide, and thus, one could merge the nodes in the FMM computation to reduce the number of sources by half. Similarly, one could merge the expansion coefficients of the two 1-D monomial expansions over the common edge of two elements, and this reduces the near interaction computational cost by half.

Remark 4.6. It is observed in [16] that it is most efficient to compute the far field interaction using the point-based FMM combined with the "subtract-and-add" technique, in the context of computing surface potential with quadrature-based techniques. In [26], we apply this idea to the quadrature-based 2-D volume potential computation, and observe that the overhead incurred from the "subtract-and-add" technique is negligible compared to the costly near field and self-interaction computations. However, with the algorithm presented in this paper, the near field and self-interaction computations are so fast that the overhead becomes non-negligible. Therefore, it is preferable to disable the computation of the local interactions in the FMM, and compute them directly with our algorithm later on.

Remark 4.7. In the context of solving Poisson's equation, it is important to compute the Newtonian potential on the boundary of the domain. In previous quadrature-based work (see, for example, [26, 1]), this evaluation is done by interpolation over curved triangles, since the direct evaluation of the potential on the boundary is costly under these quadrature-based frameworks. However, in the region close to a very curved or nonsmooth boundary, the potential typically loses some of its smoothness, from which it follows that the interpolation causes some loss of accuracy. On the contrary, by using our method, the speed of potential evaluation is both fast and independent of the target location. Thus, the Newtonian potential at the boundary of the domain can be rapidly and accurately computed without interpolation.

Remark 4.8. Quadrature-based methods for computing volume potentials typically require the the mesh elements to be star-shaped (see, for example, [1, 24]). When the domain is regular, such a requirement can be trivially satisfied. However, when the domain is complicated, the curved elements can easily violate this requirement, if the mesh near the boundary is not sufficiently refined. In our algorithm, we make no such

assumptions, except when constructing a set of well-conditioned collocation points over a curved triangle by mapping the Vioreanu-Rokhlin nodes over a triangle to the curved element by the blending function (51), which requires the mesh element to be star-shaped. We note that this is not the only way to construct well-conditioned collocation nodes, and thus, our algorithm relaxes this requirement on the mesh generation process.

4.5 Time complexity analysis

In this section, we present the time complexity of our algorithm. Suppose that we approximate the density function by a Nth order 2-D monomial expansion over each element. It follows that there are $(N+1)(N+2)/2 = \mathcal{O}(N^2)$ terms in the expansion. We estimate the various costs as follows.

Precomputation that only has to be executed once:

1. The construction of the anti-Laplacian mapping takes $\mathcal{O}(N^2)$ operations.
2. The pivoted LU factorization of the 1-D Vandermonde matrix with Gauss-Legendre collocation nodes of order $\mathcal{O}(N)$ over $[-1, 1]$ takes $\mathcal{O}(N^3)$ operations.

Precomputation for a mesh element:

1. The computation of the 2-D monomial expansion coefficients of the density function takes $\mathcal{O}(N^6)$ operations, since the cost is dominated by the factorization of a 2-D Vandermonde matrix of size $\mathcal{O}(N^2) \times \mathcal{O}(N^2)$.
2. The computation of the anti-Laplacian takes $\mathcal{O}(N^2)$ operations.
3. The computation of the 1-D monomial expansions of the anti-Laplacian and its normal derivatives takes $\mathcal{O}(N^3)$ operations, since the cost is dominated by the evaluation of the anti-Laplacian (a 2-D monomial expansion) at the $\mathcal{O}(N)$ collocation nodes. To solve the 1-D Vandermonde system, when the mesh element is a triangle, one can reuse the previously factorized 1-D Vandermonde matrix with Gauss-Legendre collocation nodes over $[-1, 1]$. We emphasize that the precomputed factorization only works for the straight edges, and one has to compute the factorization of the 1-D Vandermonde matrix for the curved edges on the fly, which costs $\mathcal{O}(N^3)$ operations each time. We note that the number of curved edges is generally much fewer than the number of straight edges.

Near and self-evaluation of the Newtonian potential:

1. It takes $\mathcal{O}(N)$ operations to evaluate the right hand side of Green's third identity (21), either by the Gauss-Legendre rule or the Helsing-Ojala method.

Below, we consider the number of operations required to evaluate the Newtonian potential over all of the discretization nodes over the domain Ω. Suppose that Ω is discretized into m mesh elements. Then, the number of discretization nodes N_{tot} is of order $\mathcal{O}(mN^2)$. First, the precomputation takes $\mathcal{O}(mN^6) = \mathcal{O}(N_{\text{tot}}N^4)$ operations. Second, the far field interaction costs (i.e., the FMM cost) are of order $\mathcal{O}(mN)$. Third, the

near interaction computation takes $\mathcal{O}(N_{\text{tot}}N)$ operations, as each discretization node is inside the near fields of a constant number of mesh elements. Finally, the self-interaction computation takes $\mathcal{O}(N_{\text{tot}}N^2)$ operations, since the evaluation of the anti-Laplacian takes $\mathcal{O}(N^2)$ operations. Since N is generally a small constant ($N \leq 20$), our algorithm has linear time complexity. Furthermore, we note that the constant associated with the cost is small (see Table 4). Moreover, our algorithm is particularly suitable for the case where the number of targets is large, as the near and self-interaction computations are instantaneous after the precomputations have been performed.

5 Numerical experiments

In this section, we illustrate the performance of the algorithm with several numerical examples. We implemented our algorithm in Fortran 77 and Fortran 90, and compiled it using the Intel Fortran Compiler, version 2021.6.0, with the `-Ofast` flag. We conducted all experiments on a ThinkPad laptop, with 16 GB of RAM and an Intel Core i7-10510U CPU.

We use the table of Vioreanu-Rokhlin rules [31] that are publicly available in [15]. We use the FMM library published in [14] in our implementation. We also make use of high-performance linear algebra libraries, e.g., BLAS, LAPACK, etc, to accelerate the precomputation step in Section 4.1. In particular, we use subroutines `dgetrf` and `dgetrs` (i.e., the pivoted LU factorization) from LAPACK as our linear system solver for the Vandermonde system with pollution (See Observation 3.2). We make no use of parallelization. While we comment in Remarks 4.5, 4.6 that one should loop through edges instead of triangles, and that one should disable the self-interaction computation in the FMM, these features are not implemented in our code, for the sake of simplicity.

We list the notations that appear in this section below.

- S_{exps}: the number of targets at which the Newtonian potential can be evaluated, per second, using our algorithm, after the precomputation.
- S_{adap}: the number of targets at which the Newtonian potential can be evaluated, per second, using adaptive integration.
- E_{exps}: the absolute error of the potential evaluation computed using our algorithm.
- E_{adap}: the absolute error of the potential evaluation computed using adaptive integration.
- h_0: the mesh element size.
- N_{tri}: the total number of triangles.
- N_{ord}: the order of the 2-D monomial expansion approximation to the density function over a mesh element.
- $N_{\text{tot}}^{\text{tgt}}$: the total number of targets.
- $N_{\text{tot}}^{\text{src}}$: the total number of sources (i.e., quadrature nodes).

- T_{geom}: The time spent on the geometric algorithms: quadtree constructions, nearby elements queries, etc. The mesh creation time is not counted.
- T_{init}: The time spent on the precomputations required by our algorithm.
- $T_{\mathcal{F}}$: The time spent on far field interaction computation (i.e., the FMM computation).
- $T_{\mathcal{N}}$: The time spent on near field interaction computation, including the subtraction of spurious contributions from the far field interaction computation (see [16, 26]).
- $T_{\mathcal{S}}$: The time spent on self-interaction computation, including the subtraction of spurious contributions from the far field interaction computation (see [16, 26]).
- T_{tot}: The total time for the evaluation of the volume potential at all of the discretization nodes.
- $\frac{\#\text{tgt}}{\text{sec}}$: the number of targets that the algorithm can evaluate the volume potential at, per second.
- $\widetilde{E}_{\text{abs}}$: the largest absolute error of the solution to Poisson's equation at all of the target nodes.

5.1 Monomial approximation of functions

The accurate 2-D monomial approximation of the density function over a mesh element is an essential part of our algorithm. In this section, we demonstrate the approximation power of the 2-D monomial expansions. Firstly, we consider the case where the domain is a standard simplex

$$\Delta := \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq 1, 0 \leq y \leq 1 - x\}. \tag{53}$$

We choose its bounding circle to be

$$\Big\{(x, y) \in \mathbb{R}^2 : \big(x - \frac{1}{3}\big)^2 + \big(y - \frac{1}{3}\big)^2 = \frac{5}{9}\Big\}. \tag{54}$$

Given a function $f : \Delta \to \mathbb{R}$, we approximate it by an Nth order Koornwinder polynomial expansion

$$f(x, y) \approx \sum_{m=0}^{N} \sum_{n=0}^{m} b_{m-n,n} K_{m-n,n}(x, y), \tag{55}$$

and also by an Nth order 2-D monomial expansion

$$f(x, y) \approx \sum_{m=0}^{N} \sum_{n=0}^{m} a_{m-n,n} \widetilde{x}^{m-n} \widetilde{y}^{n}, \tag{56}$$

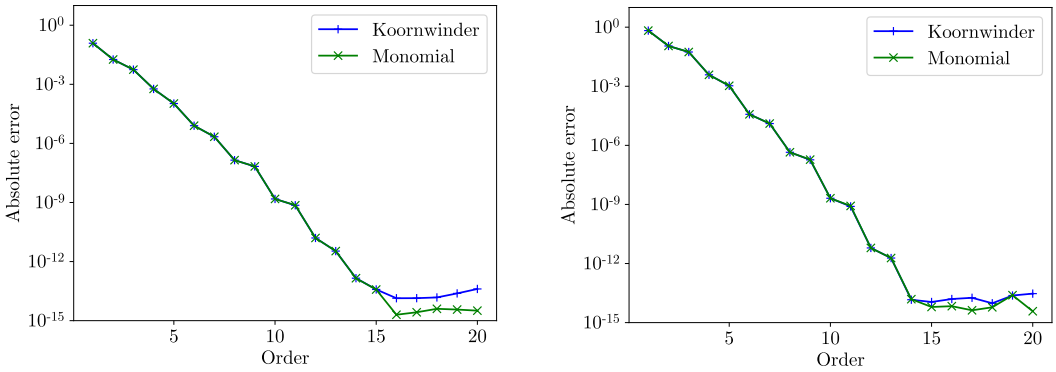
where $\widetilde{x} = \frac{3}{\sqrt{5}}(x - \frac{1}{3})$, $\widetilde{y} = \frac{3}{\sqrt{5}}(y - \frac{1}{3})$. We note that the Koornwinder expansion coefficients are computed by standard techniques (see, for example, [26]), and the monomial expansion approximation is computed by our algorithm. In Figure 1, for various functions f and

various orders N, we report the L_∞ error of the approximations by uniformly sampling 20000 points over Δ (including $\partial\Delta$), and comparing the values of the two approximations with the true value of the function. One can observe from Figures 1a and 1b that, when the function f is sufficiently smooth, the approximation by monomial expansions converges as fast as the approximation by Koornwinder polynomial expansions. This is because that the function admits a Koornwinder polynomial expansion whose expansion coefficients decay quickly enough (see Figure 2a), from which it follows that the function can be represented by a monomial expansion with expansion coefficients of size one (as is shown in Figure 2b). Thus, with a backward stable linear system solver, approximating a sufficiently smooth function by a monomial expansion is well-conditioned.

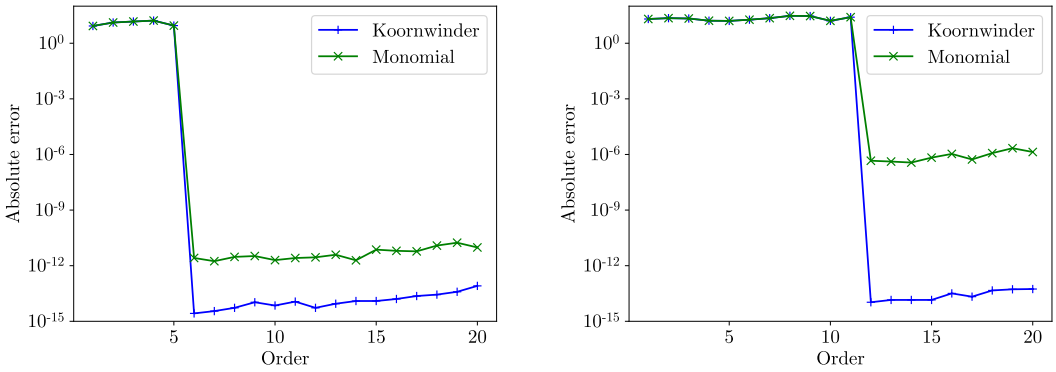
Then, we consider the case where the domain is a curved triangle

$$\widetilde{\Delta} := \{(r\cos(\theta), r\sin(\theta)) \in \mathbb{R}^2 : 0 \leq r \leq \sqrt{\frac{1}{2\pi}}, 0 \leq \theta \leq \pi/2\}. \tag{57}$$

It is easy to verify that $\text{Area}(\widetilde{\Delta}) = \text{Area}(\Delta)$. In Figure 3, for various functions f and various orders N, we report the L_∞ error of the monomial expansion approximation by uniformly sampling 20000 points over $\widetilde{\Delta}$ (including $\partial\widetilde{\Delta}$), and the magnitude of the expansion coefficients. One can observe that the performance of the monomial expansion approximation over a curved triangle is almost identical to the triangle domain case (see Figure 1).

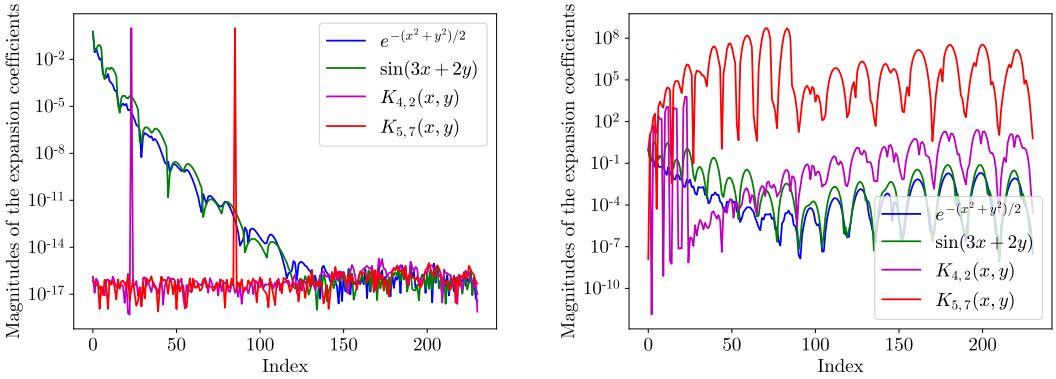


(a) $f(x, y) = e^{-x^2-y^2}$ (b) $f(x, y) = \sin(5x + 6y)$



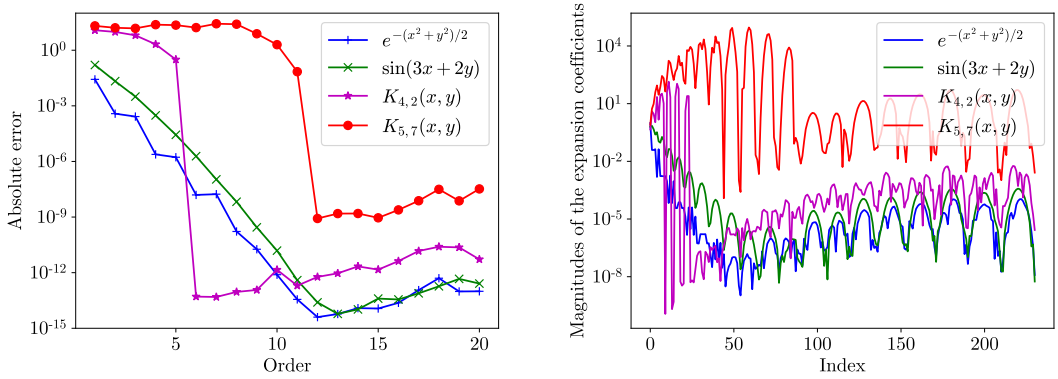
(c) $f(x, y) = K_{4,2}(x, y)$ (d) $f(x, y) = K_{5,7}(x, y)$

Figure 1: **Comparison between the monomial expansion approximations and the Koornwinder polynomial expansion approximations.**



(a) 20th order Koornwinder expansion (b) 20th order Taylor expansion

Figure 2: **Magnitudes of the coefficients of the Koornwinder polynomial expansions and the monomial expansions.**



(a) L_∞ error (b) 20th order Taylor expansion coefficients

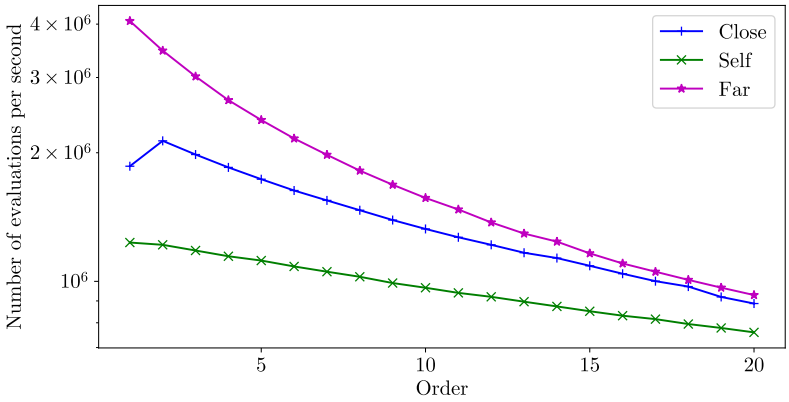
Figure 3: **The errors of the approximation and magnitudes of coefficients of the monomial expansion when the domain is a curved element**.

5.2 Newtonian potential generated over a mesh element

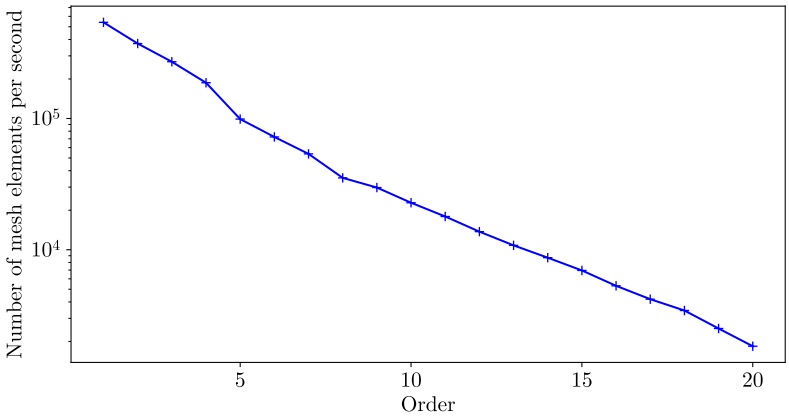
In this section, we illustrate the performance of our algorithm. In Table 1, we report the speed-up of the close evaluation of the Newtonian potential (after the precomputation) compared to the conventional adaptive integration approach. In Figure 4, we report the speeds for the close and self-evaluations, and of the precomputations, for various orders of approximations. In Figure 5, we show the plots of the computational errors of the Newtonian potential.

h	S_{exps}	S_{adap}	$\frac{S_{\text{exps}}}{S_{\text{adap}}}$	E_{exps}	E_{adap}
5×10^{-1}	9.57×10^{5}	1.44×10^{5}	6.64	1.49×10^{-15}	2.08×10^{-17}
5×10^{-2}	1.11×10^{6}	1.47×10^{4}	75.3	7.49×10^{-16}	6.38×10^{-16}
5×10^{-3}	1.16×10^{6}	9.15×10^{3}	126	2.80×10^{-15}	1.62×10^{-15}
5×10^{-4}	1.16×10^{6}	5.86×10^{3}	197	3.13×10^{-15}	1.91×10^{-15}
5×10^{-5}	1.16×10^{6}	4.83×10^{3}	239	3.19×10^{-15}	2.23×10^{-15}
5×10^{-6}	1.16×10^{6}	3.71×10^{3}	312	3.14×10^{-15}	1.92×10^{-15}

Table 1: **The speed-up of close evaluation of Newtonian potentials**. In this example, the domain Δ is defined in (53), and the density function is given by $f(x, y) = e^{-x^2-y^2}$. The target location is set to be $(0.5, -h)$. In the adaptive integration computation, we use the 12th order Vioreanu-Rokhlin rule, equipped with the error control technique introduced in [26]. To make the adaptive integration speed independent of the complexity of the density function, we exclude the time spent on the density function evaluations on the first level, but include the time spent on interpolating the density function values to the next level (see [6] for the interpolation technique). Furthermore, we accelerate the application of the interpolation matrix by LAPACK, and fine-tuned the baseline to make the comparisons fair. In our algorithm, we approximate the density function by a 12th order 2-D monomial expansion, and its trace by a 14th order 1-D monomial expansion. The reference solutions are computed by extremely high-order adaptive integration.

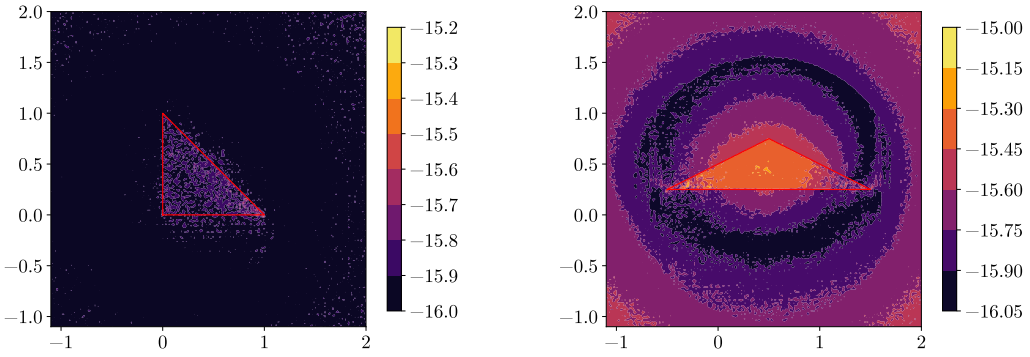


(a) The speed of far, close and self-potential evaluations



(b) The speed of precomputations

Figure 4: **The speed of the algorithm**. We set the order of the 1-D monomial expansion used on the traces of the mesh elements to be the order of the 2-D monomial approximation to the density function (reported on the x-axis) plus two. We note that the speed of our algorithm is independent of the target location's proximity to the triangle, and only depends on the order of the approximation. It follows that parameters other than the order do not need to be reported. We also note that the label "Far" denotes the evaluation of the Newtonian potential via (21) by a Gauss-Legendre rule over the trace of the mesh element, with length equal to the order of the 2-D monomial approximation (reported on the x-axis) plus three.



(a) (b)

Figure 5: **Error contours**. We set the density function to be $f(x, y) = e^{-x^2-y^2}$, and the orders of the 2-D and 1-D monomial expansion to be 20 and 24, respectively. One can observe that the results are accurate even when the domain is a stretched triangle.

5.3 Poisson's equation

In this section, we report the performance of our Newtonian potential evaluation algorithm in the context of solving Poisson's equation

$$\begin{aligned} \nabla^2 \varphi &= f \text{ in } \Omega, \\ \varphi &= g \text{ on } \partial\Omega, \end{aligned} \tag{58}$$

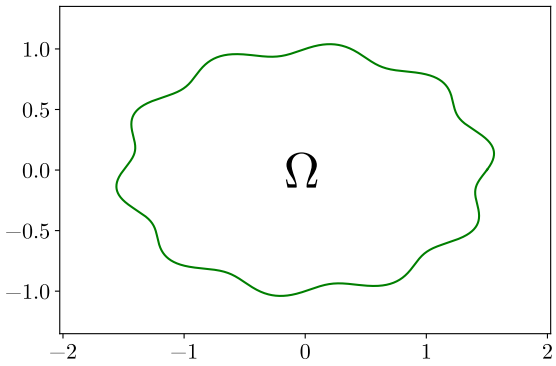
where the domain Ω is shown in Figure 6,

$$\begin{aligned} f(x,y) = &-\cos(50y) + (4x^2 + 4y^2 - 4)e^{-x^2-y^2} + \\ &\big((100 - 4y^2)\sin(10x - y^2) + 2\cos(10x - y^2)\big)/100, \end{aligned} \tag{59}$$

and

$$g(x,y) = \varphi(x,y)|_{\partial\Omega} = \big(\cos(50y)/2500 + e^{-x^2-y^2} + \sin(10x - y^2)\big)|_{\partial\Omega}. \tag{60}$$

We refer the readers to the numerical experiments in [26] for the procedure for solving Poisson's equation using the Newtonian potential.



(a)

Figure 6: **The domain** Ω.

We report the problem sizes in Tables 2, 3, and the performance of our algorithm in Table 4.

h_0	N_{tri}	$N_{\text{tot}}^{\text{tgt}}$
0.2	228	61668
0.1	1001	240231
0.05	4196	978276

Table 2: **The total number of mesh elements and targets for different mesh sizes** h_0. The domain is uniformly discretized into triangles. The targets denote the 20th order Vioreanu-Rokhlin nodes over all the mesh elements, plus 9000 uniformly sampled points along the boundary of the domain.

N_{ord}	h_0	$N_{\text{tot}}^{\text{src}}$
8	0.2	8208
	0.1	36030
	0.05	151056
14	0.2	12312
	0.1	54054
	0.05	226584
20	0.2	16416
	0.1	72072
	0.05	302112

Table 3: **The total number of sources for different orders of approximation** $N_{\mathbf{ord}}$ **and difference mesh sizes** h_0. The sources denote the Gauss-Legendre nodes of order $N_{\text{ord}} + 5$ along the boundaries of the mesh elements.

N_{ord}	h_0	T_{geom}	T_{init}	$T_{\mathcal{F}}$	$T_{\mathcal{N}}$	$T_{\mathcal{S}}$	T_{tot}	$\frac{\#\text{tgt}}{\text{sec}}$	$\widetilde{E}_{\text{abs}}$
8	0.2	0.03	0.01	0.17	0.22	0.08	0.51	1.22×10^{5}	1.81×10^{-5}
	0.1	0.12	0.03	0.72	0.81	0.30	1.98	1.21×10^{5}	1.43×10^{-8}
	0.05	0.56	0.15	3.32	3.39	1.33	8.75	1.12×10^{5}	2.26×10^{-11}
14	0.2	0.04	0.04	0.22	0.17	0.10	0.56	1.10×10^{5}	5.67×10^{-8}
	0.1	0.14	0.15	0.92	0.64	0.40	2.25	1.07×10^{5}	7.21×10^{-12}
	0.05	0.59	0.61	3.79	2.84	1.65	9.47	1.03×10^{5}	1.47×10^{-13}
20	0.2	0.04	0.12	0.25	0.14	0.13	0.68	9.04×10^{4}	3.80×10^{-9}
	0.1	0.14	0.54	1.05	0.63	0.49	2.86	8.39×10^{4}	1.30×10^{-13}
	0.05	0.59	2.30	4.27	2.46	1.99	11.6	8.42×10^{4}	4.46×10^{-13}

Table 4: **Performance of our algorithm for different orders of 2-D monomial approximation to the density function**. In these experiments, we set the order of the 1-D monomial expansion approximation to the trace of the anti-Laplacian to be $N_{\text{ord}} + 5$. We note that $\widetilde{E}_{\text{abs}}$ denotes the error of the approximation to the solution to Poisson's equation, as the analytical expression for the Newtonian potential is not available.

6 Conclusions and further directions

In this paper, we present a simple and efficient algorithm for the rapid evaluation of Newtonian potentials over a general planar domain. In this algorithm, we use Green's third identity to transform the Newtonian potential into a collection of boundary integrals that can be efficiently evaluated to high accuracy. As a result, after a fast precomputation, the speeds of close and self-evaluations for a single mesh element are almost the same as the speed of evaluating a layer potential over the element boundary by naive quadrature. Furthermore, the number of required quadrature nodes for computing the far field interactions is reduced significantly.

One of the key components of our algorithm is the high-order 2-D monomial expansion approximation of a function over a triangle, which is often regarded as an ill-conditioned problem. In this paper, we rigorously justify the feasibility of monomial approximation.

6.1 Quadrature error estimate for Newtonian potentials

The use of Green's third identity reduces the Newtonian potential evaluation to layer potential evaluations with single and double layer kernels. Since quadrature error estimates are available for the single and double layer potentials (see [19]), one could use them to estimate the quadrature error for Newtonian potentials.

6.2 Helmholtz and Yukawa volume potential

The Helmholtz volume potential and Yukawa volume potential also satisfy Green's third identity, and thus, one can transform the computation of these volume potentials into the computation of the corresponding layer potentials, with densities given by the traces of the anti-Helmholtzian (or anti-Yukawaian) of the volume density function. We note that the computation of anti-Helmholtzian (or anti-Yukawaian) is straightforward after approximating the volume density function by a monomial expansion (see Section 4.1). There also exist efficient algorithms for evaluating Helmholtz (or Yukawa) layer potentials (see, for example, [21, 23, 27]).

6.3 Newtonian potential in 3-D volumes

The generalization of our algorithm to the 3-D volume case is trivial. However, efficient algorithms for the evaluation of surface Newtonian potentials are an area of active research.

References

[1] T. G. Anderson, H. Zhu, and S. Veerapaneni, *A Fast, High-Order Scheme for Evaluating Volume Potentials on Complex 2D Geometries via Area-to-Line Integral Conversion and Domain Mappings*, arXiv:2203.05933 (2022).

[2] M. Anita, *The Rapid Evaluation of Volume Integrals of Potential Theory on General Regions*, J. Comput. Phys., 100.2 (1992), pp. 236–245.

[3] T. Askham, and A. J. Cerfon, *An Adaptive Fast Multipole Accelerated Poisson Solver for Complex Geometries*, J. Comput. Phys., 344 (2017), pp. 1–22.

[4] A. H. Barnett, *Evaluation of Layer Potentials Close to the Boundary for Laplace and Helmholtz Problems on Analytic Planar Domains*, SIAM J. Sci. Comput., 36.2 (2014), pp. A427–451.

[5] Å. Björck, and V. Pereyra, *Solution of Vandermonde Systems of Equations*, Math. Comput. 24.112 (1970), pp. 893–903.

[6] J. Bremer, and Z. Gimbutas, *A Nyström Method for Weakly Singular Integral Operators on Surfaces*, J. Comput. Phys., 231.14 (2012), pp. 4885–4903.

[7] J. Bremer, Z. Gimbutas, and V. Rokhlin, *A Nonlinear Optimization Procedure for Generalized Gaussian Quadratures*, SIAM J. Sci. Comput., 32.4 (2010), pp. 1761–1788.

[8] O. P. Bruno, and J. Paul, *Two-Dimensional Fourier Continuation and Applications*, SIAM J. Sci. Comput. 44.2 (2022), pp. A964–992.

[9] F. Ethridge, *Fast Algorithms for Volume Integrals in Potential Theory*, Ph.D. dissertation, New York University.

[10] F. Ethridge, and L. Greengard, *A New Fast-Multipole Accelerated Poisson Solver in Two Dimensions*, SIAM J. Sci. Comput., 23.3 (2001), pp. 741–760.

[11] C. Epstein, and S. Jiang. *A Stable, Efficient Scheme for C^n Function Extensions on Smooth Domains in $\mathbb{R}^d$*, arXiv:2206.11318, 2022.

[12] F. Fryklund, E. Lehto, and A. K. Tornberg, *Partition of Unity Extension of Functions on Complex Domains*, J. Comput. Phys. 375 (2018), pp. 57–79.

[13] W. J. Gordon, and C. A. Hall, *Construction of Curvilinear Co-Ordinate Systems and Applications to Mesh Generation*, Int. J. Numer. Methods Eng., 7.4 (1973), pp. 461–477.

[14] Z. Gimbutas, and L. Greengard, *Computational Software: Simple FMM Libraries for Electrostatics, Slow Viscous Flow, and Frequency-Domain Wave Propagation*, Commun. Comput. Phys., 18.2 (2015), pp. 516–528.

[15] Z. Gimbutas, and H. Xiao, *Quadratures for triangles, squares, cubes and tetrahedra*, `https://github.com/zgimbutas/triasymq` (2020).

[16] L. Greengard, M. ONeil, M. Rachh, and F. Vico, *Fast Multipole Methods for the Evaluation of Layer Potentials with Locally-Corrected Quadratures*, J. Comput. Phys.: X, 10 (2021), pp. 100092.

[17] L. Greengard, and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comput. Phys., 135.2 (1997), pp. 280–292.

[18] L. af Klinteberg, and A. H. Barnett. *Accurate Quadrature of Nearly Singular Line Integrals in Two and Three Dimensions by Singularity Swapping*, BIT Numerical Mathematics 61.1 (2021): 83–118.

[19] L. af Klinteberg, and A. K. Tornberg, *Error Estimation for Quadrature by Expansion in Layer Potential Evaluation*, Adv. Comput. Math., 43.1 (2017), pp. 195–234.

[20] J. Helsing, and R. Ojala. *On the Evaluation of Layer Potentials Close to Their Sources*, J. Comput. Phys. 227.5 (2008): 2899–2921.

[21] J. Helsing, and A. Holst. *Variants of an Explicit Kernel-Split Panel-Based Nystrm Discretization Scheme for Helmholtz Boundary Value Problems*, Adv. Comput. Mathe. 41.3 (2015), pp. 691–708.

[22] N. J. Higham, *Error analysis of the Bjrck-Pereyra algorithms for solving Vandermonde systems*, Numer. Math. 50.5 (1987), pp. 613–632.

[23] A. Klöckner, A. Barnett, L. Greengard, and M. ONeil. *Quadrature by Expansion: A New Method for the Evaluation of Layer Potentials*, J. Comput. Phys. 252 (2013), pp. 332–349.

[24] S. Nintcheu Fata, *Treatment of Domain Integrals in Boundary Element Methods*, Appl. Numer. Math., 62.6 (2012), pp. 720–735.

[25] P. W. Patridge, C. A. Brebbia, L. W. Wrobel, *The Dual Reciprocity Boundary Element Method*, Computational Mechanics Publication, Southampton, 1992.

[26] Z. Shen, and K. Serkh. *Accelerating Potential Evaluation over Unstructured Meshes in Two Dimensions*, arXiv:2205.04401 (2022).

[27] S. Pålsson, and A. K. Tornberg, *Spectrally Accurate Ewald Summation for the Yukawa Potential in Two Dimensions*, arXiv:1911.04875 (2019).

[28] J. D. Towers, *A Source Term Method for Poisson Problems on Irregular Domains*, J. Comput. Phys. 361 (2018), pp. 424–441.

[29] B. Wu, H. Zhu, A. Barnett, and S. Veerapaneni. *Solution of Stokes Flow in Complex Nonsmooth 2D Geometries via a Linear-Scaling High-Order Adaptive Integral Equation Scheme*, J. Comput. Phys. 410 (2020): 109361.

[30] P. G. Martinsson, *Fast Direct Solvers for Elliptic PDEs*, CBMS-NSF Conference Series, vol. CB96. SIAM, Philadelphia, 2019.

[31] B. Vioreanu, and V. Rokhlin, *Spectra of Multiplication Operators as a Numerical Tool*, SIAM J. Sci. Comput., 36.1 (2014), pp. A267–288.

[32] H. Zhu, *Fast, High-Order Accurate Integral Equation Methods and Application to PDE-Constrained Optimization*, PhD thesis, University of Michigan (2021).