JOURNAL OF INTEGRAL EQUATIONS
AND APPLICATIONS
Volume 24, Number 4, Winter 2012

MULTILEVEL AUGMENTATION METHODS FOR NONLINEAR BOUNDARY INTEGRAL EQUATIONS II: ACCELERATED QUADRATURES AND NEWTON ITERATIONS

XIANGLING CHEN, ZHONGYING CHEN
BIN WU AND YUESHENG XU

Communicated by Kendall Atkinson

ABSTRACT. A fast multilevel augmentation method (MAM) was proposed recently by the same authors for solving a class of nonlinear boundary integral equations. In this paper, we develop accelerated quadrature formulas for computing the integrals involved in the MAM and approximate iteration for solving the resulting nonlinear system. Specifically, we employ a product integration scheme for computing the singular integrals which appear in the matrices involved in the MAM and introduce an approximation technique in the Newton iteration for solving the resulting nonlinear systems to avoid repeated computation in generating their Jacobian matrices. The use of these two techniques results in a modified MAM which speeds up its computation. We show that the modified MAM preserves the optimal convergence order of the original one while reducing computational costs. Numerical results are presented to demonstrate the approximation accuracy and computational efficiency of the proposed modified MAM, with a comparison to those of the original one and a known algorithm of Atkinson and Chandler.

1. Introduction.

Boundary value problems of the Laplace equation are commonly used mathematical models for many important applications, such as acoustics, elasticity, electromagnetics, fluid dynamics (see, for example, [**20, 23, 24, 27**] and the references cited therein).

2010 AMS *Mathematics subject classification.* Primary 65R20, 45E05.
Keywords and phrases. Multilevel augmentation method, nonlinear boundary integral equations.
This research was partially supported by Guangdong Provincial Government of China through the "Computational Science Innovative Research Team" program, by Guangdong Province Key Lab of Computational Science, by the Natural Science Foundation of China under grants 10771224, 10801138, 11071264, 11071286, by the Science and Technology Section of SINOPEC, by US Air Force Office of Scientific Research under grant FA9550-09-1-0511, by US National Science Foundation under grant 1115523, and by the Fundamental Research Funds for the Central Universities.
Received by the editors on March 24, 2011, and in revised form on August 15, 2011.

DOI:10.1216/JIE-2012-24-4-545

These boundary value problems may be transferred into integral equations defined on the boundary. When the boundary conditions are nonlinear (cf. [**7–9, 21**]), the resulting integral equations are also nonlinear.

The linearization approach (cf. [**1–3, 5, 6, 25, 26**]) is a popular numerical method for solving nonlinear boundary integral equations. In [**1**] several numerical methods for solving equations of this type are reviewed. The drawback of the linearization method is that it consumes high computational costs. A typical Newton iteration method requires generating the Jacobian matrix in the discretization subspace and updating it at each iteration step. This involves a large amount of integral evaluations.

Aimed at reducing computational costs of the linearization method, a fast multilevel augmentation method (MAM) was introduced in [**10**] for solving nonlinear boundary integral equations. The numerical solutions obtained from this method have an *almost optimal* convergence order and a *nearly linear* computational complexity order. The efficiency of the algorithm comes from two sources. One is that it inverts the nonlinear operator in a *much smaller* subspace rather than in the whole discretization subspace. Actually, outside the *fixed* subspace, we only need to solve linear systems. The other is that it truncates the matrices resulting from the discretization to sparse matrices. Since the entries of the matrices are defined by integrals, the truncation process avoids evaluating most of the integrals. Numerical experiments show that the algorithm runs much faster than the classical numerical schemes. Specifically, we compared in [**10**] the algorithm with a nice algorithm of Atkinson and Chandler [**4**] and observed obvious advantages in computational efficiency when a large discretization scale is used. For more information regarding the MAM for solving linear and nonlinear equations, readers are referred to [**12, 14, 16–18**].

Although in the MAM most of integral evaluations are avoided by using the truncation strategy, evaluating the remaining integrals still occupies a majority of the running time of the algorithm. We observe that most computational effort is spent in two key steps of the algorithm: generating the representation matrices of integral operators and updating the Jacobian matrices of the nonlinear equations. To overcome these computational challenges, we suggest in this paper two techniques to reduce the computational costs of the MAM. For computing the entries of the representation matrix of the weakly singular integral operator, we employ a product integration scheme. That is, we decompose the integral kernel into a singular term and a smooth term. The singular term is integrated exactly by explicit formulas, while the smooth term is integrated approximately by a quadrature of high accuracy. Integrating the weakly singular kernel in this way is remarkably efficient. In [**4**] the Nyström method was applied to solve the nonlinear boundary integral equation. A nice idea used in that paper is that, by choosing an appropriate quadrature rule to discretize the integral operators, the Jacobian matrix was expressed by linear operations of the representation matrices of the integral operators. In such a way, one can avoid numerical evaluation of integrals during updating the Jacobian matrix. Inspired by this idea, we propose in this paper to project the function which defines the nonlinear operator onto the approximation subspace so that the generation of the corresponding Jacobian matrix does not involve any integral evaluation. Numerical experiments show that it significantly reduces the running time for solving nonlinear equations. Employing the above two techniques, the run time of the MAM is speeded up remarkably.

The main purpose of this paper is to improve upon the computational performance of Algorithm 2 in [**10**] by using an efficient numerical integration method in numerical evaluation of the singular integrals and adopting an appropriate approximation technique in implementing the Newton iteration scheme for solving the nonlinear systems. The performance of Algorithm 2 in [**10**] was compared with that of the algorithm developed in [**4**], which will be called Algorithm AC throughout this paper. Since the computational complexity of Algorithm AC is of quadratic order while that of Algorithm 2 in [**10**] was of nearly linear order, for a large scale discretization of the nonlinear integral equation, Algorithm 2 in [**10**] was more efficient than Algorithm AC. However, we observed in our experiments that, when the nonlinear integral equation is solved numerically in a small scale, Algorithm AC is faster than Algorithm 2 in [**10**]. We find that two techniques used in Algorithm AC make it computationally efficient for a small scale discretization of the nonlinear integral equation. The first one is product integration in numerical evaluation of the singular integrals. The second one is the approximation method in the Newton iteration scheme for solving related nonlinear systems so as to update the Jacobian matrix very fast. The above comparisons and observations motivate us to modify Algorithm 2 in [**10**] in light of the ideas from Algorithm AC.

This paper is organized in five sections. In Section 2 we review the fast multilevel augmentation methods originally developed in [**10**], and in Section 3 we describe two techniques for the modified MAM and show that the modified MAM requires only a linear (up to a logarithmic factor) number of multiplications. We prove in Section 4 that the modified MAM preserves the approximation accuracy of the original MAM. In Section 5 we present numerical results to confirm the approximation accuracy and the computational efficiency of the proposed algorithms with a comparison to the original MAM and to the known method of Atkinson and Chandler.

2. Fast multilevel augmentation methods. We review in this section the multilevel augmentation method developed in [**10**] for solving the nonlinear boundary integral equation which is a reformulation of the nonlinear boundary value problem of the Laplace equation. We first describe the nonlinear boundary value problem of the Laplace equation. Let D be a simply connected bounded domain in $\mathbf{R}^2$ with a C^2 boundary Γ. We consider solving the following nonlinear boundary value problem

$$\begin{cases} \Delta u(x) = 0, & x \in D, \\ (\partial u/\partial \mathbf{n}_x)(x) = -g(x, u(x)) + g_0(x), & x \in \Gamma, \end{cases} \tag{2.1}$$

where $\mathbf{n}_x$ denotes the exterior unit normal vector to Γ at x.

We now reformulate the boundary value problem (2.1) as a nonlinear boundary integral equation. We assume that the boundary Γ has a parametrization $x = (\xi(t), \eta(t))$, $t \in [0,1]$, and let $\chi := \sqrt{(\xi')^2 + (\eta')^2}$. It is known (cf. [**4, 25**]) that equation (2.1) can be reformulated as a nonlinear integral equation

$$\begin{aligned} u(t) - \int_0^1 K(t,\tau)u(\tau)\,d\tau - \int_0^1 L(t,\tau)\chi(\tau)g(\tau, u(\tau))\,d\tau \\ = -\int_0^1 L(t,\tau)\chi(\tau)g_0(\tau)\,d\tau, \quad t \in [0,1], \end{aligned} \tag{2.2}$$

where, for $t, \tau \in [0,1]$,

$$K(t,\tau) := \begin{cases} (1/\pi)\frac{\eta'(\tau)(\xi(\tau)-\xi(t))-\xi'(\tau)(\eta(\tau)-\eta(t))}{(\xi(t)-\xi(\tau))^2+(\eta(t)-\eta(\tau))^2}, & t \neq \tau, \\ (1/\pi)\frac{\eta'(t)\xi''(t)-\xi'(t)\eta''(t)}{2[\xi'(t)^2+\eta'(t)^2]}, & t = \tau \end{cases}$$

and

$$L(t,\tau) := \frac{1}{\pi}\log\sqrt{(\xi(t)-\xi(\tau))^2+(\eta(t)-\eta(\tau))^2}, \quad t \neq \tau.$$

It is easily verified that, when Γ is of C^s with $s \geq 2$, the kernel K has continuous derivatives up to order $s-2$. Throughout this paper, we assume that s is sufficiently large so that K is sufficiently smooth. Specifically, there exists a positive constant Λ such that

$$(2.3) \qquad |D_t^\alpha D_\tau^\beta K(t,\tau)| \leq \Lambda, \quad t,\tau \in [0,1]$$

for positive integers α and β satisfying $\alpha+\beta \leq s-2$. We observe that kernel L is weakly singular and its singularity points are located at $\{(t,\tau) \in [0,1]\times[0,1] : t-\tau = 0,1,-1\}$ (cf. [**10**]). Accordingly, for each $t \in [0,1]$, $L(t,\cdot) \in C^\infty([0,1]\backslash\{t\})$, and there exist positive constants θ and $\sigma \in (0,1)$ such that

$$(2.4) \quad |D_t^\alpha D_\tau^\beta L(t,\tau)| \\ \leq \theta \cdot \max\{|t-\tau|^{-(\sigma+\alpha+\beta)}, |t-\tau+1|^{-(\sigma+\alpha+\beta)}, |t-\tau-1|^{-(\sigma+\alpha+\beta)}\}$$

for $\alpha+\beta \leq s$ and $t,\tau \in [0,1]$ with $t-\tau \neq -1,0,1$.

We next rewrite equation (2.2) in an operator form. To this end, we introduce two integral operators $\mathcal{K},\mathcal{L} : L^\infty(0,1) \to L^\infty(0,1)$ defined respectively by

$$(\mathcal{K}w)(t) := \int_0^1 K(t,\tau)w(\tau)\,d\tau,$$

$$(\mathcal{L}w)(t) := \int_0^1 L(t,\tau)w(\tau)\,d\tau,$$

$$t \in [0,1],$$

and the nonlinear operator

$$(\Psi u)(t) := g(t,u(t))\chi(t), \quad t \in [0,1].$$

In these notations, equation (2.2) is rewritten as

$$u - (\mathcal{K} + \mathcal{L}\Psi)u = f, \tag{2.5}$$

where the right hand side function is defined by

$$f := -\mathcal{L}(g_0\chi).$$

Due to the nonlinearity of the operator Ψ, equation (2.5) is a nonlinear integral equation.

We next recall the multiscale collocation method for solving the equation. For $n \in \mathbf{N}_0 := \{0, 1, \dots\}$, let π_n denote the mesh which divides the interval $[0,1]$ uniformly into μ^n pieces for a given integer $\mu > 1$, and let $\mathbf{X}_n$ be the piecewise polynomial space of order r with respect to the mesh π_n for a fixed positive integer r. Note that the sequence $\mathbf{X}_n, n \in \mathbf{N}_0$, is nested in the sense that

$$\mathbf{X}_n \subset \mathbf{X}_{n+1}, \quad \text{for all } n \in \mathbf{N}_0.$$

The collocation points are chosen from a subset G of $[0,1]$ which contains r distinct points. We require G to be *refinable* with respect to the family of contractive mappings $\Phi_\mu := \{\phi_e : e \in \mathbf{Z}_\mu\}$, where the index set $\mathbf{Z}_\mu := \{0, 1, \dots, \mu - 1\}$, and

$$\phi_e(x) := \frac{x+e}{\mu}, \quad x \in [0,1], \text{ for } e \in \mathbf{Z}_\mu.$$

The interested reader may refer to [**11**] for the concept of the *refinable set*. For $n \in \mathbf{N}_0$, let $\mathcal{P}_n$ denote the interpolatory projection from $C[0,1]$ onto $\mathbf{X}_n$ with the set of interpolation points:

$$G_n := \left\{ \frac{j+s}{\mu^n} : j \in \mathbf{Z}_{\mu^n}, s \in G \right\}.$$

Making use of the notations introduced above, we define the collocation scheme for solving (2.5) as finding $u_n \in \mathbf{X}_n$ to satisfy

$$u_n - \mathcal{P}_n(\mathcal{K} + \mathcal{L}\Psi)u_n = \mathcal{P}_n f. \tag{2.6}$$

According to Theorem 2.1 of [**10**], equation (2.6) is uniquely solvable for sufficiently large n, and there exist positive constants c_1 and c_2 such that

$$c_1\|u^* - \mathcal{P}_n u^*\|_\infty \le \|u^* - u_n\|_\infty \le c_2\|u^* - \mathcal{P}_n u^*\|_\infty,$$

where $u^* \in C[0,1]$ denotes an isolated solution of (2.5).

Solving equation (2.6) numerically encounters two difficulties: the density of the coefficient matrices and the nonlinearity of Ψ. The multilevel augmentation method developed in [**10**] successfully addresses these issues. By using multiscale basis functions and multiscale collocation functionals, we are able to compress the dense matrices to sparse matrices. This leads to a fast numerical algorithm. Moreover, we invert the nonlinear operator *only* in a fixed lower frequency scale without losing the approximation accuracy of the numerical solution. In this way, we solve equation (2.6) approximately with the optimal approximation order and *nearly linear* computational costs.

We now recall multiscale bases for $\mathbf{X}_n$ and the corresponding multiscale collocation functionals, which were originally introduced in [**13, 22**]. The nestedness of the sequence $\mathbf{X}_n$ allows us to decompose $\mathbf{X}_{n+1}$ as a sum of $\mathbf{X}_n$ and its orthogonal complement $\mathbf{W}_{n+1}$, that is,

$$\mathbf{X}_{n+1} = \mathbf{X}_n \oplus^{\perp} \mathbf{W}_{n+1}, \quad n \in \mathbf{N}_0.$$

Since $\mathbf{X}_n$ are spaces of piecewise polynomials of order r, the elements of $\mathbf{W}_n$ naturally have vanishing moments of order r. We next describe the construction of the corresponding collocation functionals. Generally speaking, for each $n \in \mathbf{N}_0$, $\mathbf{L}_n$ is the linear subspace of $(L^\infty(0,1))^*$, spanned by the point evaluation functionals at the points from G_n. Note that a point evaluation functional on $L^\infty(0,1)$ can be viewed as an extension of the point evaluation functional on $C[0,1]$. We assume that $\mathbf{V}_0$ consists of the point evaluation functionals corresponding to the points in set G_0 and $\mathbf{V}_1$ consists of linear combinations of point evaluation functionals with the points in G_1 having vanishing moments of order r. For $i > 1$, the elements of $\mathbf{V}_i$ are obtained from those of $\mathbf{V}_1$ through compositions of affine mappings from Φ_μ. See [**13**] for details of the construction of these functionals. It is not difficult to observe that

$$\dim(\mathbf{L}_n) = \dim(\mathbf{X}_n).$$

Let $w(0) := \dim(\mathbf{X}_0)$ and $w(i) := \dim(\mathbf{W}_i)$ for $i > 0$. We choose the bases for the subspaces such that

$$\mathbf{X}_0 = \operatorname{span}\{w_{0j} : j \in \mathbf{Z}_{w(0)}\}, \qquad \mathbf{L}_0 := \operatorname{span}\{\ell_{0j} : j \in \mathbf{Z}_{w(0)}\}$$

and

$$\mathbf{W}_i = \operatorname{span}\{w_{ij} : j \in \mathbf{Z}_{w(i)}\}, \qquad \mathbf{V}_i := \operatorname{span}\{\ell_{ij} : j \in \mathbf{Z}_{w(i)}\}, \quad i > 0.$$

By introducing the index set

$$J_n := \{(i,j) : i \in \mathbf{Z}_{n+1}, j \in \mathbf{Z}_{w(i)}\},$$

we have that

$$\mathbf{X}_n = \operatorname{span}\{w_{ij} : (i,j) \in J_n\}, \quad \mathbf{L}_n = \operatorname{span}\{\ell_{ij} : (i,j) \in J_n\},\ n \in \mathbf{N}_0.$$

With the bases and collocation functionals described above, we have the matrix representations for the operators $\mathcal{K}$ and $\mathcal{L}$. Specifically, for $n \in \mathbf{N}_0$, we define the matrix

$$\mathbf{K}_n := [K_{i'j',ij} : (i',j'),(i,j) \in J_n], \quad \text{with} \quad K_{i'j',ij} := \langle \ell_{i'j'}, \mathcal{K} w_{ij} \rangle$$

and

$$\mathbf{L}_n := [L_{i'j',ij} : (i',j'),(i,j) \in J_n], \quad \text{with} \quad L_{i'j',ij} := \langle \ell_{i'j'}, \mathcal{L} w_{ij} \rangle .$$

Matrices $\mathbf{K}_n$ and $\mathbf{L}_n$ will be compressed according to the regularity of the kernels K and L. Note that the kernel K is smooth and L is weakly singular, and their regularities are described in (2.3) and (2.4), respectively. Accordingly, as in [**10**], we adopt the following truncation strategies.

(T1) For each $n \in \mathbf{N}_0$, the matrix $\mathbf{K}_n$ is truncated to a sparse matrix

$$\widetilde{\mathbf{K}}_n := [\widetilde{K}_{i'j',ij} : (i',j'),(i,j) \in J_n],$$

where for $(i',j'),(i,j) \in J_n$,

$$\widetilde{K}_{i'j',ij} := \begin{cases} K_{i'j',ij}, & i'+i \le n, \\ 0, & \text{otherwise.} \end{cases}$$

(T2) We denote by $\operatorname{dist}(\cdot,\cdot)$ the distance of two point sets. For $(i,j) \in J_n$, we let $S_{ij} := \operatorname{supp}(w_{ij})$. For each $n \in \mathbf{N}_0$ and $(i',j'),(i,j) \in J_n$, we set

$$\widetilde{L}_{i'j',ij} := \begin{cases} L_{i'j',ij}, & \operatorname{dist}(S_{i'j'}, S_{ij}) \le \varepsilon^n_{i'i} \text{ or} \\ \quad \operatorname{dist}(S_{i'j'}, S_{ij}) \ge 1 - \varepsilon^n_{i'i}, & \\ 0, & \text{otherwise,} \end{cases}$$

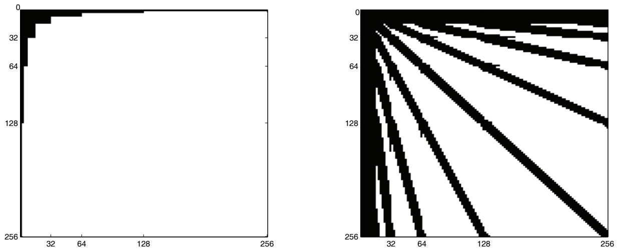

FIGURE 1. The distribution of nonzero entries of $\widetilde{\mathbf{K}}_7$ (left) and $\widetilde{\mathbf{L}}_7$ (right).

in which the *truncation parameters* $\varepsilon^n_{i'i}$ are chosen by

$$\varepsilon^n_{i'i} := \max\{a\mu^{-n+b(n-i)+b'(n-i')}, \rho(\mu^{-i} + \mu^{-i'})\} \tag{2.7}$$

for some constants b, b', $a > 0$ and $\rho > 1$. The truncated matrix of $\mathbf{L}_n$ is defined by

$$\widetilde{\mathbf{L}}_n := [\widetilde{L}_{i'j',ij} : (i', j'), (i, j) \in J_n].$$

When $n = 7$ and we use piecewise linear basis functions, we show in Figure 1 the matrices $\widetilde{\mathbf{K}}_7$ and $\widetilde{\mathbf{L}}_7$. It has been proved in [**10**, Lemma 3.1] that the number of nonzero elements of both $\widetilde{\mathbf{K}}_n$ and $\widetilde{\mathbf{L}}_n$ is of $\mathcal{O}(n\mu^n)$.

We now describe the multilevel augmentation method developed in [**10**]. For a fixed $k \in \mathbf{N}_0$ and any positive integer l, we define the index set $J_{kl} := J_{k+l} \backslash J_k$, that is, $J_{kl} = \{(i, j) : i \in \mathbf{Z}_{k+l+1} \backslash \mathbf{Z}_{k+1}, j \in \mathbf{Z}_{w(i)}\}$. For each $n \in \mathbf{N}_0$, we let

$$\mathbf{E}_n := [\langle \ell_{i'j'}, w_{ij} \rangle : (i', j'), (i, j) \in J_n].$$

For a fixed $k \in \mathbf{N}_0$ and an $l > 0$, we set

$$\mathbf{E}^H_{kl} := [\langle \ell_{i'j'}, w_{ij} \rangle : (i', j'), (i, j) \in J_{kl}],$$
$$\widetilde{\mathbf{K}}^H_{k,l-1} := [\widetilde{K}_{i'j',ij} : (i', j') \in J_{kl}, (i, j) \in J_{k+l-1}],$$

and

$$\widetilde{\mathbf{L}}^H_{kl} := [\widetilde{L}_{i'j',ij} : (i', j') \in J_{kl}, (i, j) \in J_{k+l}].$$

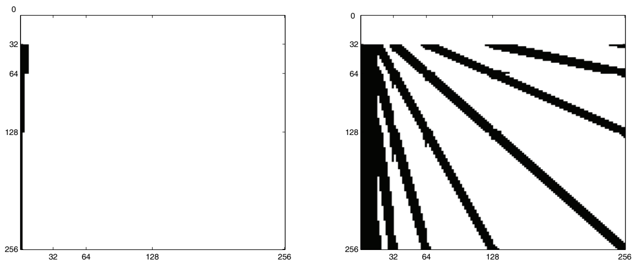

FIGURE 2. The distribution of nonzero entries of $\widetilde{\mathbf{K}}^H_{4,3}$ (left) and $\widetilde{\mathbf{L}}^H_{4,3}$ (right).

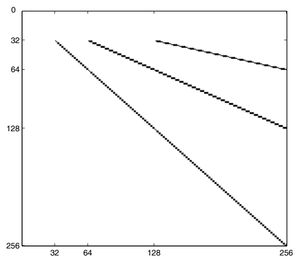

FIGURE 3. The distribution of nonzero entries of $\mathbf{E}^H_{4,3}$.

Let $k = 4, l = 3$; we show in Figures 2 and 3 the distribution of nonzero entries of $\widetilde{\mathbf{K}}^H_{4,3}, \widetilde{\mathbf{L}}^H_{4,3}$ and $\mathbf{E}^H_{4,3}$ in $J_7 \times J_7$ matrices in order to understand the MAM algorithm.

Each $v \in \mathbf{X}_{k+l}$ has the unique representation

$$v = \sum_{(i,j)\in J_{k+l}} (\mathbf{v})_{ij} w_{ij}.$$

We call $\mathbf{v} := [(\mathbf{v})_{ij} : (i,j) \in J_{k+l}]^T$ the *representation vector* of function v.

Algorithm 1: The fast multilevel augmentation method. Let k be a fixed positive integer. Given $m \in \mathbf{N}_0$, we carry out the following computing steps:

Step 1: Solve the nonlinear system

$$\left\langle \ell_{i'j'}, (\mathcal{I} - (\mathcal{K} + \mathcal{L}\Psi))\left(\sum_{(i,j)\in J_k} (\mathbf{u}_k)_{ij} w_{ij} \right) \right\rangle = \langle \ell_{i'j'}, f \rangle, \quad (i',j') \in J_k \tag{2.8}$$

for the solution $\mathbf{u}_k := [(\mathbf{u}_k)_{ij} : (i,j) \in J_k]^T$. Set $\mathbf{u}_{k,0} := \mathbf{u}_k$ and $l := 1$.

Step 2: Compute the representation vector $\widehat{\mathbf{u}}_{k+l}$ of $\widehat{u}_{k+l} := \mathcal{P}_{k+l}\Psi u_{k,l-1}$ and, generate the following vector::

$$\widehat{\mathbf{f}}_{kl} := [\langle \ell_{i'j'}, f \rangle : (i',j') \in J_{kl}]^T.$$

Compute

$$\widetilde{\mathbf{f}}_{kl} := \widehat{\mathbf{f}}_{kl} + \widetilde{\mathbf{K}}^H_{k,l-1}\mathbf{u}_{k,l-1} + \widetilde{\mathbf{L}}^H_{kl}\widehat{\mathbf{u}}_{k+l}. \tag{2.9}$$

Solve the linear system

$$\mathbf{E}^H_{kl}\mathbf{u}^H_{kl} = \widetilde{\mathbf{f}}_{kl} \tag{2.10}$$

for $\mathbf{u}^H_{kl} := [(\mathbf{u}_{kl})_{ij} : (i,j) \in J_{kl}]^T$, and define $u^H_{kl} := \sum_{(i,j)\in J_{kl}} (\mathbf{u}_{kl})_{ij} w_{ij}$.

Step 3: Solve the nonlinear system

$$\left\langle \ell_{i'j'}, (\mathcal{I} - (\mathcal{K} + \mathcal{L}\Psi))\left(\sum_{(i,j)\in J_k} (\mathbf{u}_{kl})_{ij} w_{ij} + u^H_{kl} \right) \right\rangle = \langle \ell_{i'j'}, f \rangle, \quad (i',j') \in J_k \tag{2.11}$$

for $\mathbf{u}^L_{kl} := [(\mathbf{u}_{kl})_{ij} : (i,j) \in J_k]^T$, and define $u^L_{kl} := \sum_{(i,j)\in J_k} (\mathbf{u}_{kl})_{ij} w_{ij}$ and $u_{kl} := u^L_{kl} + u^H_{kl}$.

Step 4: Set $l \leftarrow l + 1$ and go back to Step 2 until $l = m$.

Several remarks on the computational performance of Algorithm 1 are in order. Computational costs of Algorithm 1 can be divided into three parts. Part 1 is for generating the matrices $\widetilde{\mathbf{K}}_{k+l}$ and $\widetilde{\mathbf{L}}_{k+l}$, Part 2 is for computing (2.9) and solving (2.10) and Part 3 is for solving the resulting nonlinear systems, including (2.8) and (2.11). It takes much more time to generate matrix $\widetilde{\mathbf{L}}_{k+l}$ than to generate matrix $\widetilde{\mathbf{K}}_{k+l}$, since the kernel L is weakly singular. We observe from numerical experiments that, although Parts 1 and 3 both have high costs, when l is small Part 3 dominates the total computing time for implementing the algorithm, while when l increases Part 1 grows faster than Part 3.

3. The modified MAM algorithm.

In this section we address two computational issues of the MAM Algorithm. The use of accelerated quadratures and the Newton method reduces its running time significantly.

3.1. Product integration of singular integrals.

Two integration strategies were proposed in [**10**] for evaluating the nonzero entries of $\widetilde{\mathbf{K}}_n$ and $\widetilde{\mathbf{L}}_n$. Numerical experiments show that the generation of $\widetilde{\mathbf{L}}_n$ requires much more computing time than that of $\widetilde{\mathbf{K}}_n$ due to the singularity of kernel L. The quadrature rule proposed in [**10**] for computing the entries of $\widetilde{\mathbf{L}}_n$ uses graded meshes according to a general class of singular kernels, and it is suitable for the class of singular kernels. We observe that kernel L has a special structure which allows us to develop a more efficient quadrature method than the Gaussian quadrature method used in [**10**]. In this subsection, we shall develop a special product integration method for the specific kernel L so that the computing time for calculating the nonzero entries of the matrix $\widetilde{\mathbf{L}}_n$ is significantly reduced.

The product integration method has been widely used in the literature for computing singular integrals. For example, it was used in [**4, 19**] to discretize singular integral operators. Along this line, concrete formulas of product integration were given in [**3**] (see [**3**, pages 116–119]). These formulas were developed in the context of single scale approximation and were proved efficient for computation in that context. In the current multiscale approximation context, we shall establish product integration formulas suitable for the use of multiscale bases.

We now study the typical integral involved in the entries of matrix $\widetilde{\mathbf{L}}_n$. The nonzero entries of matrix $\widetilde{\mathbf{L}}_n$ involve integrals of the form

$$I_{ij}(s) := \int_{S_{ij}} L(s,t)w_{ij}(t)\,dt, \quad \text{for } (i,j) \in J_n,\ s \in [0,1],$$

where w_{ij} are multiscale piecewise polynomial basis functions. As suggested by [**3, 4, 19**], kernel L can be decomposed as

$$L(s,t) = \frac{1}{\pi}\left[B_0(s,t) + B_1(s,t)\right] \tag{3.1}$$

with

$$B_0(s,t) := \log\left|\frac{\sqrt{(\xi(s)-\xi(t))^2 + (\eta(s)-\eta(t))^2}}{(s-t)(s-t-1)(s-t+1)}\right|$$

and

$$B_1(s,t) := \log|s-t| + \log|s-t-1| + \log|s-t+1|.$$

The above decomposition in fact extracts the singularity of L. Specifically, B_1 possesses all the singularity features of L and B_0 is smooth. Kernel B_0 is easy to integrate numerically with a little computational cost, while the singularity of B_1 brings difficulty to its numerical integration. However, we observe that the expression of B_1 is very specific, which allows us to integrate it *exactly* with explicit formulas. The quantities I_{ij} can be written as the sum of two terms

$$I_{ij}^{\nu}(s) := \frac{1}{\pi}\int_{S_{ij}} B_\nu(s,t) w_{ij}(t)\,dt, \quad \nu = 0, 1. \tag{3.2}$$

We first compute the term I_{ij}^1. To this end, we re-express the multiscale basis functions w_{ij}. Note that, for $j \in \mathbf{Z}_{w(0)}$, w_{0j} is a polynomial of order r. Hence, it can be written as

$$w_{0j}(t) = \sum_{\gamma\in\mathbf{Z}_r} a_\gamma t^\gamma.$$

For $i > 0$ and $j \in \mathbf{Z}_{w(i)}$, w_{ij} is a piecewise polynomial of order r. According to the construction of the basis function w_{ij}, for all $(i,j) \in J_n$, the support S_{ij} can be divided into μ pieces $\Omega_\kappa = (a_\kappa, b_\kappa)$, $\kappa \in \mathbf{Z}_\mu$, on each of which w_{ij} is a polynomial of order r. Thus, we write w_{ij} as

$$w_{ij}(t) = \sum_{\gamma\in\mathbf{Z}_r} a_{\kappa,\gamma} t^\gamma, \quad t \in \Omega_\kappa,\ \kappa \in \mathbf{Z}_\mu.$$

The above discussion of the basis functions motivates us to define the following special integrals. For $\gamma \in \mathbf{Z}_r$, $\alpha \in \Sigma := \{0, 1, -1\}$ and for $a, b \in [0,1]$ with $a < b$, we set

$$I(a,b;\alpha,\gamma) := \int_a^b \log|s-t-\alpha| t^\gamma \, dt.$$

In this notation, we have that

$$\int_{S_{0j}} B_1(s,t) w_{0j}(t)\, dt = \sum_{\alpha\in\Sigma}\sum_{\gamma\in\mathbf{Z}_r} a_\gamma I(0,1;\alpha,\gamma) \tag{3.3}$$

and

$$\int_{S_{ij}} B_1(s,t) w_{ij}(t)\, dt = \sum_{\kappa\in\mathbf{Z}_\mu}\sum_{\alpha\in\Sigma}\sum_{\gamma\in\mathbf{Z}_r} a_{\kappa,\gamma} I(a_\kappa,b_\kappa;\alpha,\gamma). \tag{3.4}$$

The integral $I(a,b;\alpha,\gamma)$ can be computed exactly. We derive below the formula for the integral.

Lemma 3.1. *If $\gamma \in \mathbf{Z}_r$, $\alpha \in \Sigma$ and $a, b \in [0,1]$ with $a < b$, then*

$$I(a,b;\alpha,\gamma) = \frac{1}{\gamma+1}\left[(t^{\gamma+1} - (s-\alpha)^{\gamma+1})\log|s-t-\alpha| - \sum_{j=1}^{\gamma+1}(s-\alpha)^{\gamma-j+1}t^j\right]\Bigg|_a^b.$$

Proof. The formula in this lemma may be proved by integration by parts. □

Using the integration formula in Lemma 3.1, we are able to compute the integrals (3.3) and (3.4) exactly.

It remains to compute the term I_{ij}^0. For this purpose, we describe a Gaussian quadrature rule on interval $[a,b]$. For each positive integer j, we denote by g_j the Legendre polynomial of degree j, and by τ_ℓ^j, $\ell \in \mathbf{Z}_j$

the j zeros of g_j in the order $-1 < \tau_0^j < \cdots < \tau_{j-1}^j < 1$. We transfer these zeros to the interval $[a,b]$, by letting

$$\widehat{\tau}_\ell^j := \frac{a+b}{2} + \frac{b-a}{2}\tau_\ell^j, \quad \ell \in \mathbf{Z}_j.$$

The points $\widehat{\tau}_\ell^j$ are the j zeros of the Legendre polynomial of degree j on the interval $[a,b]$. Given a continuous function h defined on $[a,b]$, the j-point Gaussian quadrature rule is given by

$$G(h,[a,b],j) := \sum_{\ell \in \mathbf{Z}_j} \omega_\ell^j h(\widehat{\tau}_\ell^j),$$

where

$$\omega_\ell^j := \int_a^b \prod_{\substack{i \in \mathbf{Z}_j \\ i \neq \ell}} \frac{t - \widehat{\tau}_i^j}{\widehat{\tau}_\ell^j - \widehat{\tau}_i^j}\, dt.$$

This quadrature formula will be used to compute I_{ij}^0. We summarize below the integration strategy for computing the nonzero entries $\widetilde{L}_{i'j',ij}$ of $\widetilde{\mathbf{L}}_n$.

(**QL**) For a nonzero entry $\widetilde{L}_{i'j',ij}$ of $\widetilde{\mathbf{L}}_n$, we compute I_{ij}^0 and I_{ij}^1 separately. For computing I_{ij}^0 we divide the support S_{ij} of w_{ij} uniformly into N intervals I_ℓ, $\ell \in \mathbf{Z}_N$, where N is a positive integer such that the diameter of each of the intervals is less than or equal to $\mu^{-\kappa r}$ and w_{ij} is a polynomial on I_ℓ. The integral I_{ij}^0 is computed by the formula

$$\sum_{\ell \in \mathbf{Z}_N} G(B_0(s,\cdot)w_{ij}, I_\ell, \lceil (2\kappa)^{-1} n \rceil).$$

The integral I_{ij}^1 is expressed in terms of equations (3.3) and (3.4) and computed by using Lemma 3.1.

We now estimate the computational costs for generating the matrix $\widetilde{\mathbf{L}}_n$. For each pair of i and j, computing I_{ij}^1 requires only a fixed number of functional evaluations and multiplications. Moreover, computational costs for computing I_{ij}^0 may be estimated as in [**10**, Lemma 3.1]. We then have the following result.

Lemma 3.2. *For any $n \in \mathbf{N}_0$, the numbers of functional evaluations and multiplications needed for generating the matrix $\widetilde{\mathbf{L}}_n$ are both $\mathcal{O}(n\mu^n)$.*

3.2. Approximate iteration for solving nonlinear systems. Algorithm 1 requires solving two nonlinear systems (2.8) and (2.11). These equations are solved by using the Newton method. In each iteration step of the Newton method, we need to compute the entries of the Jacobian matrix. Specifically, for equation (2.8), the Newton iteration scheme has the following steps:

- Choose an initial guess, $\mathbf{u}_k^{(0)}$.
- For $m = 0, 1, \ldots$, compute

$$\mathbf{F}(\mathbf{u}_k^{(m)}) := (\mathbf{E}_k - \mathbf{K}_k)\mathbf{u}_k^{(m)} - \widehat{\mathbf{f}}_k^{(m)}$$

with

$$\widehat{\mathbf{f}}_k^{(m)} := \left[\left\langle \ell_{i'j'}, f + \mathcal{L}\Psi\left(u_k^{(m)}\right)\right\rangle : (i', j') \in J_k\right]^T,$$
$$u_k^{(m)} := \sum_{(i,j)\in J_k} (\mathbf{u}_k^{(m)})_{ij} w_{ij},$$

and compute the Jacobian matrix

$$J(\mathbf{u}_k^{(m)}) := [J_{i'j',ij} : (i', j'), (i, j) \in J_k]$$

with

$$J_{i'j',ij} := E_{i'j',ij} - K_{i'j',ij} - \left\langle \ell_{i'j'}, \mathcal{L}(w_{ij}\Psi'(u_k^{(m)}))\right\rangle.$$

- Solve $\Delta_k^{(m)}$ from the equation $J(\mathbf{u}_k^{(m)})\Delta_k^{(m)} = -\mathbf{F}(\mathbf{u}_k^{(m)})$,
- Compute $\mathbf{u}_k^{(m+1)} := \mathbf{u}_k^{(m)} + \Delta_k^{(m)}$.

It is easily seen that the evaluation of both $\mathbf{F}(\mathbf{u}_k^{(m)})$ and $J(\mathbf{u}_k^{(m)})$ involves computing integrals, which requires high computational costs. Solving equation (2.11) numerically also involves computing the integrals.

When evaluating $\mathbf{F}(\mathbf{u}_k^{(m)})$, we need to compute the integrals

$$\left\langle \ell_{i'j'}, \mathcal{L}\Psi\left(u_k^{(m)}\right)\right\rangle. \tag{3.5}$$

These integrals come from the integral operator $\mathcal{L}$. For different steps of the iteration, we are required to compute different integrals and, as a result, the computational cost is large. Note that for some $\widetilde{\Psi}(u_k^{(m)}) \in \mathbf{X}_k$, we can write it as

$$\widetilde{\Psi}\left(u_k^{(m)}\right) = \sum_{(i,j)\in J_k} c_{ij} w_{ij}$$

and

$$\left\langle \ell_{i'j'}, \mathcal{L}\widetilde{\Psi}\left(u_k^{(m)}\right)\right\rangle = \sum_{(i,j)\in J_k} c_{ij} L_{i'j',ij}. \tag{3.6}$$

Comparing (3.6) with (3.5), we observe that, although they both involve integral evaluations, (3.6) makes use of the values of the entries of the matrix $\mathbf{L}_n$, which have been previously obtained so that we do not have to recompute them. However, in general, $\Psi(u_k^{(m)}) \notin \mathbf{X}_k$. We cannot write $\Psi(u_k^{(m)})$ as a linear combination of the basis function w_{ij}. For this reason, we propose to project $\Psi(u_k^{(m)})$ into $\mathbf{X}_k$. Specifically, we do not solve (2.8) directly and, instead, we solve $\widetilde{\mathbf{u}}_k$ from the nonlinear system

$$\left\langle \ell_{i'j'}, (\mathcal{I} - (\mathcal{K} + \mathcal{L}\mathcal{P}_k\Psi))\Bigg(\sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_k)_{ij} w_{ij}\Bigg)\right\rangle = \langle \ell_{i'j'}, f\rangle, \ (i',j') \in J_k. \tag{3.7}$$

When we solve equation (3.7) by the Newton iteration method, we are required to compute the terms

$$\left\langle \ell_{i'j'}, \mathcal{L}\mathcal{P}_k\Psi\Bigg(\sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_k)_{ij} w_{ij}\Bigg)\right\rangle, \quad (i',j') \in J_k$$

and their partial derivatives with respect to the variables $(\widetilde{\mathbf{u}}_k)_{ij}, (i,j) \in J_k$. To this end, we suppose that

$$\mathcal{P}_k\Psi\Bigg(\sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_k)_{ij} w_{ij}\Bigg) = \sum_{(i,j)\in J_k} (\widehat{\mathbf{u}}_k)_{ij} w_{ij}.$$

Then each $(\widehat{\mathbf{u}}_k)_{ij}$ is a function with respect to the variables $(\widetilde{\mathbf{u}}_k)_{ij}, (i,j) \in J_k$. In fact, if we let

$$\mathbf{F} := \left[\left\langle \ell_{i'j'}, \Psi\Bigg(\sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_k)_{ij} w_{ij}\Bigg)\right\rangle, (i',j') \in J_k\right]^T,$$

then we have $\widehat{\mathbf{u}}_k = \mathbf{E}_k^{-1}\mathbf{F}$. Therefore, it follows that

$$\left\langle \ell_{i'j'}, \mathcal{L}\mathcal{P}_k\Psi\left(\sum_{(i,j)\in J_k}(\widetilde{\mathbf{u}}_k)_{ij}w_{ij}\right)\right\rangle = \sum_{(i,j)\in J_k} L_{i'j',ij}(\widehat{\mathbf{u}}_k)_{ij}.$$

In a similar manner, we compute the partial derivatives of the above quantities with respect to the variables $(\widetilde{\mathbf{u}}_k)_{ij}, (i,j) \in J_k$. Making use of the above observations, we describe the Newton iteration scheme for solving (3.7) as follows.

Algorithm 2: The Newton iteration method for solving (3.7). Set $m := 0$, $\mathbf{f}_k := [\langle \ell_{ij}, f\rangle : (i,j) \in J_k]^T$, and choose an initial guess $\widetilde{\mathbf{u}}_k^{(0)}$ and an iteration stopping threshold δ.

Step 1: Let $\widetilde{u}_k^{(m)} := \sum_{(i,j)\in J_k}(\widetilde{\mathbf{u}}_k^{(m)})_{ij}w_{ij}$, and set

$$\mathbf{G}(\widetilde{\mathbf{u}}_k^{(m)}) := \left[\left\langle \ell_{i'j'}, \Psi'(\widetilde{u}_k^{(m)})w_{ij}\right\rangle : (i',j'),(i,j) \in J_k\right].$$

Solve $\overline{\mathbf{F}}_k^{(m)}$ from $\mathbf{E}_k\overline{\mathbf{F}}_k^{(m)} = \mathbf{G}(\widetilde{\mathbf{u}}_k^{(m)})$, and compute the Jacobian matrix

$$\widetilde{\mathbf{J}}(\widetilde{\mathbf{u}}_k^{(m)}) := \mathbf{E}_k - \mathbf{K}_k - \mathbf{L}_k\overline{\mathbf{F}}_k^{(m)}.$$

Step 2: For $\mathbf{g}_k^{(m)} := \left[\left\langle \ell_{ij}, \Psi(\widetilde{u}_k^{(m)})\right\rangle : (i,j) \in J_k\right]^T$, solve $\widehat{\mathbf{u}}_k^{(m)}$ from $\mathbf{E}_k\widehat{\mathbf{u}}_k^{(m)} = \mathbf{g}_k^{(m)}$. Compute

$$\widetilde{\mathbf{F}}(\widetilde{\mathbf{u}}_k^{(m)}) := (\mathbf{E}_k - \mathbf{K}_k)\widetilde{\mathbf{u}}_k^{(m)} - \mathbf{L}_k\widehat{\mathbf{u}}_k^{(m)} - \mathbf{f}_k.$$

Step 3: Solve $\Delta_k^{(m)}$ from $\widetilde{\mathbf{J}}(\widetilde{\mathbf{u}}_k^{(m)})\Delta_k^{(m)} = -\widetilde{\mathbf{F}}(\widetilde{\mathbf{u}}_k^{(m)})$, and compute $\widetilde{\mathbf{u}}_k^{(m+1)} := \widetilde{\mathbf{u}}_k^{(m)} + \Delta_k^{(m)}$.

Step 4: Set $m \leftarrow m+1$ and go back to Step 1 until $\|\Delta_k^{(m)}\|_\infty < \delta$.

It is worth noticing that, in Algorithm 2, we need not evaluate any additional integrals but make use of the matrix $\mathbf{L}_k$. This saves tremendous computational effort and, thus, makes the algorithm very fast. We shall see from numerical examples in Section 5 that (3.7) is solved much faster than (2.8).

Equation (2.11) can be approximated in a similar manner. Specifically, in Step 3 of Algorithm 1, we replace (2.11) by

(3.8)
$$\left\langle \ell_{i'j'}, (\mathcal{I} - (\mathcal{K} + \mathcal{L}\mathcal{P}_{k+l}\Psi))\left(\sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_{kl})_{ij} w_{ij} + \widetilde{u}^H_{kl}\right)\right\rangle = \langle \ell_{i'j'}, f\rangle, \quad (i',j') \in J_k.$$

The Newton iteration scheme for solving (3.8) can be likewise developed and will be referred as Algorithm 2′.

We describe below the MAM algorithm with employing the above two techniques.

Algorithm 3: MAM with an accelerated quadrature and the Newton method. Let k be a fixed positive integer. Given $m \in \mathbf{N}_0$, we carry out the following computing steps:

Step 1: Use Algorithm 2 to solve the nonlinear system (3.7) and obtain the solution $\widetilde{\mathbf{u}}_k := [(\widetilde{\mathbf{u}}_k)_{ij} : (i,j) \in J_k]^T$. Let $\widetilde{\mathbf{u}}_{k,0} := \widetilde{\mathbf{u}}_k$ and $l := 1$.

Step 2: Follow Step 2 of Algorithm 1 to generate $\widetilde{\mathbf{f}}_{kl}$ and solve $\mathbf{E}^H_{kl}\widetilde{\mathbf{u}}^H_{kl} = \widetilde{\mathbf{f}}_{kl}$ to obtain $\widetilde{\mathbf{u}}^H_{kl} := [(\widetilde{\mathbf{u}}_{kl})_{ij} : (i,j) \in J_{kl}]^T$ and define $\widetilde{u}^H_{kl} := \sum_{(i,j)\in J_{kl}} (\widetilde{\mathbf{u}}_{kl})_{ij} w_{ij}$.

Step 3: Use Algorithm 2′ to solve the nonlinear system (3.8) and obtain the solution $\widetilde{\mathbf{u}}^L_{kl} := [(\widetilde{\mathbf{u}}_{kl})_{ij} : (i,j) \in J_k]^T$. Define $\widetilde{u}^L_{kl} := \sum_{(i,j)\in J_k} (\widetilde{\mathbf{u}}_{kl})_{ij} w_{ij}$ and $\widetilde{u}_{kl} := \widetilde{u}^L_{kl} + \widetilde{u}^H_{kl}$.

Step 4: Set $l \leftarrow l+1$ and go back to Step 2 until $l = m$.

When solving the nonlinear systems (3.7) and (3.8), we are required to generate matrices $\widetilde{\mathbf{K}}_n$ and $\widetilde{\mathbf{L}}_n$. The truncated matrix $\widetilde{\mathbf{K}}_n$ is evaluated by the same strategies as those with Algorithm 1, which was described in [**10**]. The truncated matrix $\widetilde{\mathbf{L}}_n$ is evaluated by using the quadrature method (QL). The next theorem gives an estimate of the computational cost required for Algorithm 3.

Theorem 3.3. *Let k be a fixed positive integer. For any $m \in \mathbf{N}_0$, the number of functional evaluations and multiplications used in Algorithm 3 is $\mathcal{O}((k+m)\mu^{k+m})$.*

Proof. The computational cost of Algorithm 3 is composed of two parts. One is the cost for generating the matrices $\widetilde{\mathbf{K}}_{k+m}$ and $\widetilde{\mathbf{L}}_{k+m}$. The other is that for carrying out the computing steps listed in the algorithm. Lemma 3.2 has shown that the computational cost of generating $\widetilde{\mathbf{L}}_{k+m}$ is $\mathcal{O}((k+m)\mu^{k+m})$. Moreover, we cite the result of Lemma 3.1 of [**10**] which ensures that the cost for generating matrix $\widetilde{\mathbf{K}}_{k+m}$ is also $\mathcal{O}((k+m)\mu^{k+m})$.

To estimate efforts of the computing steps in Algorithm 3, we only need to compare Algorithm 3 with Algorithm 1. We observe that Algorithm 3 replaces (2.8) and (2.11) by (3.7) and (3.8), respectively. We have shown that the modifications reduce computational cost. Therefore, the computational cost for carrying out the computing steps of Algorithm 3 is less than that of Algorithm 1. It is stated in [**10**] that the number of functional evaluations and multiplications in Algorithm 1 is $\mathcal{O}((k+m)\mu^{k+m})$. Thus, the theorem is proved. $\square$

We remark that, although Algorithm 3 has the same order of computational costs as Algorithm 1, the constant involved in the order of computational costs is improved.

4. Convergence analysis.

Since in Algorithm 3 we replace (2.8) and (2.11) by (3.7) and (3.8), respectively, it is necessary to analyze the effect of these modifications on the resulting numerical solutions. We estimate in this section the error of numerical solutions generated by Algorithm 3.

Let $\widetilde{\mathcal{K}}_n : \mathbf{X}_n \to \mathbf{X}_n$ be the operator corresponding to the truncated matrix $\widetilde{\mathbf{K}}_n$, that is,

$$\widetilde{K}_{i'j',ij} = \left\langle \ell_{i'j'}, \widetilde{\mathcal{K}}_n w_{ij} \right\rangle, \quad \text{for all } (i',j'),(i,j) \in J_n.$$

According to [**10**], the difference between operators $\mathcal{K}_n$ and $\widetilde{\mathcal{K}}_n$ is estimated in the following lemma.

Lemma 4.1. *If $u^* \in W^{r,\infty}[0,1]$, then there exists a positive constant c_1 such that, for all $n \in \mathbf{N}_0$ and all $v \in \mathbf{X}_n$,*

$$\|(\mathcal{K}_n - \widetilde{\mathcal{K}}_n)v\|_\infty \le c_1(n+1)^2\mu^{-rn}\|v\|_\infty. \tag{4.1}$$

Moreover, if

$$\|v - u^*\|_\infty \le \overline{c}\mu^{-r(n-1)}\|u^*\|_{r,\infty} \tag{4.2}$$

for some positive constant $\overline{c}$, then there exists a positive constant c_2 such that, for all $n \in \mathbf{N}_0$,

$$\|(\mathcal{K}_n - \widetilde{\mathcal{K}}_n)v\|_\infty \le c_2(n+1)\mu^{-rn}\|u^*\|_{r,\infty}. \tag{4.3}$$

We define $\widetilde{\mathcal{L}}_n : \mathbf{X}_n \to \mathbf{X}_n$ as the operator such that

$$\widetilde{L}_{i'j',ij} = \left\langle \ell_{i'j'}, \widetilde{\mathcal{L}}_n w_{ij} \right\rangle, \quad \text{for all } (i',j'),(i,j) \in J_n.$$

The strategy (T2) for truncating the matrix $\mathbf{L}_n$ is the same as that in [**10**], while the strategy (QL) for evaluating the nonzero elements of $\widetilde{\mathbf{L}}_n$ is different from that of [**10**]. We define the matrix blocks

$$\begin{aligned}\mathbf{L}_{i'i} &:= [L_{i'j',ij} : j' \in \mathbf{Z}_{w(i')}, j \in \mathbf{Z}_{w(i)}],\\ \widetilde{\mathbf{L}}_{i'i} &:= [\widetilde{L}_{i'j',ij} : j' \in \mathbf{Z}_{w(i')}, j \in \mathbf{Z}_{w(i)}], \quad i', i \in \mathbf{Z}_{n+1}.\end{aligned}$$

The difference between $\mathbf{L}_n$ and $\widetilde{\mathbf{L}}_n$ that results from both the truncation and quadrature errors has the bound

$$\|\mathbf{L}_{i'i} - \widetilde{\mathbf{L}}_{i'i}\|_\infty \le c\max\{\mu^{-rn}, (\varepsilon^n_{i'i})^{-(2r-\sigma')}\mu^{-r(i'+i)}\}.$$

By definition (2.7) of the truncation parameters $\varepsilon^n_{i'i}$, we conclude that

$$\|\mathbf{L}_{i'i} - \widetilde{\mathbf{L}}_{i'i}\|_\infty \le c(\varepsilon^n_{i'i})^{-(2r-\sigma')}\mu^{-r(i'+i)}.$$

This bound is the same as the one given in [**10**]. They can be used for the following estimation of the difference between $\mathcal{L}_n$ and $\widetilde{\mathcal{L}}_n$. This may be found in [**10**].

Lemma 4.2. *Suppose that $u^* \in W^{r,\infty}[0,1]$. Let $\sigma' \in (0,1)$, $\eta := 2r - \sigma'$, and set $b = 1$, $b' \in (r/\eta, 1)$ in (2.7). Then there exists a positive constant c_1 such that, for all $n \in \mathbf{N}_0$ and all $v \in \mathbf{X}_n$,*

$$\|(\mathcal{L}_n - \widetilde{\mathcal{L}}_n)v\|_\infty \le c_1(n+1)\mu^{-\sigma' n}\|v\|_\infty. \tag{4.4}$$

Moreover, if v satisfies (4.2) *for some positive constant $\overline{c}$, then there exists a positive constant c_2 such that*

$$\|(\mathcal{L}_n - \widetilde{\mathcal{L}}_n)\mathcal{P}_n \Psi v\|_\infty \le c_2 (n+1)^2 \mu^{-(r+\sigma')n} \|u^*\|_{r,\infty}. \tag{4.5}$$

In the next lemma, we estimate the error between the outputs of Algorithms 1 and 3.

Lemma 4.3. *If $u^* \in W^{r,\infty}[0,1]$, then there exists a positive constant c such that, for a sufficiently large integer k and for all $l \in \mathbf{Z}_{m+1}$,*

$$\|\widetilde{u}_{kl} - u_{kl}\|_\infty \le c(k+l+1)\mu^{-r(k+l)}\|u^*\|_{r,\infty}.$$

Proof. We prove this lemma by induction on l. For the case $l = 0$, we need to prove the solution $\widetilde{u}_k$ of

$$\widetilde{u}_k - \mathcal{P}_k(\mathcal{K} + \mathcal{L}\mathcal{P}_k\Psi)\widetilde{u}_k = \mathcal{P}_k f \tag{4.6}$$

satisfies

$$\|\widetilde{u}_k - u_k\|_\infty \le c(k+1)\mu^{-kr}\|u^*\|_{r,\infty}. \tag{4.7}$$

Following a standard argument, we may show that the solution of equation (4.6) has the bound

$$\|\widetilde{u}_k - u^*\|_\infty \le c\mu^{-kr}\|u^*\|_{r,\infty}.$$

Thus, by the triangle inequality we establish the estimate (4.7).

We now assume that the result of this lemma holds for $l-1$ and consider the case l. Note that Step 3 of Algorithm 3 is the same as Step 3 of Algorithm 1. According to Algorithms 1 and 3, we obtain that

$$u_{kl}^H - \widetilde{u}_{kl}^H = u_A + u_B, \tag{4.8}$$

where

$$u_A := (\mathcal{P}_{k+l} - \mathcal{P}_k)\widetilde{\mathcal{K}}_{k+l}(u_{k,l-1} - \widetilde{u}_{k,l-1})]$$

and

$$u_B := (\mathcal{P}_{k+l} - \mathcal{P}_k)\widetilde{\mathcal{L}}_{k+l}\mathcal{P}_{k+l}(\Psi u_{k,l-1} - \Psi\widetilde{u}_{k,l-1}).$$

Since the projections are uniformly bounded and it follows from (4.3) of Lemma 4.1, we have that

$$\|u_A\| \leq c(k+l+1)\mu^{-r(k+l)}\|u^*\|_{r,\infty} \tag{4.9}$$

By Lemma 4.2 and the induction hypothesis, this leads to

$$\|u_B\|_\infty \leq c(k+l+1)\mu^{-r(k+l)}\|u^*\|_{r,\infty}. \tag{4.10}$$

From equations (4.8)–(4.10), we conclude that

$$\|u_{kl}^H - \widetilde{u}_{kl}^H\|_\infty \leq c(k+l+1)\mu^{-r(k+l)}\|u^*\|_{r,\infty}. \tag{4.11}$$

It remains to estimate $\|\widetilde{u}_{kl}^L - u_{kl}^L\|$. Subtracting (2.11) from (3.8) yields the equation

$$\widetilde{u}_{kl}^L - u_{kl}^L = \mathcal{P}_k[(\mathcal{K} + \mathcal{L}\mathcal{P}_{k+l}\Psi)\widetilde{u}_{kl} - (\mathcal{K} + \mathcal{L}\Psi)u_{kl}].$$

Let $\mathcal{B} := (\mathcal{K} + \mathcal{L}\mathcal{P}_{k+l}\Psi)'$ and

$$\mathcal{R}(\widetilde{u}_{kl}, u_{kl}) := (\mathcal{K} + \mathcal{L}\mathcal{P}_{k+l}\Psi)\widetilde{u}_{kl} - (\mathcal{K} + \mathcal{L}\mathcal{P}_{k+l}\Psi)u_{kl} - \mathcal{B}(\widetilde{u}_{kl} - u_{kl}).$$

We then conclude that

$$\widetilde{u}_{kl}^L - u_{kl}^L = (\mathcal{I} - \mathcal{P}_k\mathcal{B})^{-1}\mathcal{P}_k[\mathcal{B}(\widetilde{u}_{kl}^H - u_{kl}^H) + \mathcal{R}(\widetilde{u}_{kl}, u_{kl}) + \mathcal{L}(\mathcal{P}_{k+l} - \mathcal{I})\Psi u_{kl}].$$

Following the analysis given in Lemma 4.4 of [**10**], we establish the estimate

$$\|\widetilde{u}_{kl}^L - u_{kl}^L\|_\infty \leq c'\|\widetilde{u}_{kl}^H - u_{kl}^H\|_\infty$$

for some positive constant c', which completes the proof for the case l. □

The above lemma leads to the following error estimate of the approximate solutions generated by Algorithm 3.

Theorem 4.4. *If $u^* \in W^{r,\infty}[0,1]$, then there exists a positive constant c such that, for a sufficiently large k and all $m \in \mathbf{N}_0$,*

$$\|u^* - \widetilde{u}_{km}\|_\infty \leq c(k+m+1)\mu^{-r(k+m)}\|u^*\|_{r,\infty}.$$

5. Numerical experiments. In this section, we present numerical results which compare approximation accuracy and computational efficiency of the proposed algorithm with those of the original MAM algorithm. All programs are run on a workstation with 3.38G CPU and 96G memory.

We consider boundary value problem (2.1) with $g(x, u(x)) := u(x) + \sin(u(x))$. Let D be the elliptical region $x_1^2 + (x_2/2)^2 < 1$. For comparison purposes, we choose the solution of (2.1) as $u_0(x) := \mathrm{e}^{x_1}\cos(x_2)$ with $x = (x_1, x_2)$. Correspondingly, we have that

$$g_0(x) := g(x, u_0) + \frac{\partial u_0(x)}{\partial \mathbf{n}_x}.$$

The corresponding solution of boundary integral equation (2.5) is given by

$$u^*(t) := \mathrm{e}^{\cos(2\pi t)}\cos(2\sin(2\pi t)).$$

In all experiments presented in this section, we choose $\mu = 2$ and $\mathbf{X}_n$ as the space of piecewise cubic polynomials with the knots at $j/2^n$, $j = 1, 2, \ldots, 2^n - 1$. It is easy to compute $\dim(\mathbf{X}_n) = 2^{n+2}$. The basis functions of $\mathbf{X}_0$ and $\mathbf{W}_1$ are given by

$$w_{00}(t) = -\frac{1}{6}(5t-2)(5t-3)(5t-4)$$

$$w_{01}(t) = \frac{1}{2}(5t-1)(5t-3)(5t-4),$$

$$w_{02}(t) = -\frac{1}{2}(5t-1)(5t-2)(5t-4),$$

$$w_{03}(t) = \frac{1}{6}(5t-1)(5t-2)(5t-3),$$

$$w_{10}(t) = \begin{cases} \frac{85}{32} - \frac{725}{12}t + \frac{575}{2}t^2 - \frac{1475}{4}t^3, & t \in [0, \frac{1}{2}], \\ -\frac{235}{32} + \frac{575}{12}t - \frac{175}{2}t^2 + \frac{575}{12}t^3, & t \in (\frac{1}{2}, 1], \end{cases}$$

$$w_{11}(t) = \begin{cases} \frac{1145}{288} - \frac{1775}{24}t + \frac{1675}{6}t^2 - \frac{4975}{18}t^3, & t \in [0, \frac{1}{2}], \\ -\frac{7495}{288} + \frac{3625}{24}t - \frac{525}{2}t^2 + \frac{2525}{18}t^3, & t \in (\frac{1}{2}, 1], \end{cases}$$

$$w_{12}(t) = \begin{cases} \frac{805}{288} - \frac{375}{8}t + \frac{475}{3}t^2 - \frac{2525}{18}t^3, & t \in [0, \frac{1}{2}], \\ -\frac{19355}{288} + \frac{8275}{24}t - 550t^2 + \frac{4975}{18}t^3, & t \in (\frac{1}{2}, 1], \end{cases}$$

$$w_{13}(t) = \begin{cases} \frac{95}{96} - \frac{50}{3}t + \frac{225}{4}t^2 - \frac{575}{12}t^3, & t \in [0, \frac{1}{2}], \\ -\frac{13345}{96} + \frac{1775}{3}t - \frac{3275}{4}t^2 + \frac{1475}{4}t^3, & t \in (\frac{1}{2}, 1]. \end{cases}$$

The corresponding collocation functionals are

$$\begin{aligned}
\ell_{00} &= \delta_{1/5}, \quad \ell_{01} = \delta_{2/5}, \quad \ell_{02} = \delta_{3/5}, \quad \ell_{03} = \delta_{4/5},\\
\ell_{10} &= \frac{2}{5}\delta_{1/10} - \frac{3}{2}\delta_{2/10} + 2\delta_{3/10} - \delta_{4/10} + \frac{1}{10}\delta_{6/10},\\
\ell_{11} &= \frac{3}{10}\delta_{2/10} - \delta_{3/10} + \delta_{4/10} - \frac{1}{2}\delta_{6/10} + \frac{1}{5}\delta_{7/10},\\
\ell_{12} &= \frac{1}{5}\delta_{3/10} - \frac{1}{2}\delta_{4/10} + \delta_{6/10} - \delta_{7/10} + \frac{3}{10}\delta_{8/10},\\
\ell_{13} &= \frac{1}{10}\delta_{4/10} - \delta_{6/10} + 2\delta_{7/10} - \frac{3}{2}\delta_{8/10} + \frac{2}{5}\delta_{9/10}.
\end{aligned}$$

The basis functions w_{ij} and collocation functionals ℓ_{ij} for $i > 1$ can be constructed recursively from w_{1j} and ℓ_{1j} (cf. [**13**]). For general information on basis functions and collocation functionals of this type, the reader is referred to [**11, 13, 22**].

Experiment 1. In this experiment we compare the computational efficiency of the proposed quadrature algorithm for generating the matrices $\widetilde{\mathbf{L}}_n$ with that of the algorithm presented in [**10**]. Note that we use the same truncation strategy in this paper and in [**10**] and different techniques for numerically evaluating the nonzero elements of $\widetilde{\mathbf{L}}_n$.

The numerical results are presented in Table 1. In columns 2, 3 and 4, we list, respectively, the dimension of $\widetilde{\mathbf{L}}_n$, the time ("T_n") for generating $\widetilde{\mathbf{L}}_n$ using the strategy in [**10**] and the time ("T'_n") using the algorithm proposed in this paper. The computing time is measured in seconds. We observe from the numerical results that the computing time for both strategies increases linearly with respect to the dimension of the matrices, but the proposed algorithm consumes much less than the one presented in [**10**]. Roughly speaking, the new method uses only 1/20 of the computing time of the old method.

Experiment 2. In this experiment, we use the multilevel augmentation methods with initial level $k = 4$ to solve the boundary integral equation (2.5). The purpose of this experiment is to test how much the proposed technique for solving the nonlinear systems speeds up the solution process. To this end, we run Algorithms 1 and 3 separately, and add up the time used in solving the nonlinear systems. Note that, in Algorithm 1, we need to solve (2.8) once and (2.11) m times, while,

TABLE 1. Computing time for generating $\widetilde{\mathbf{L}}_n$.

n	$\dim(\widetilde{\mathbf{L}}_n)$	T_n	T'_n
4	64	2.4	0.1
5	128	5.7	0.2
6	256	12.5	0.5
7	512	27.1	1.2
8	1024	56.2	2.7
9	2048	116.6	5.9
10	4096	241.0	12.8
11	8192	500.5	27.8
12	16384	1017.5	60.5

TABLE 2. Total time for solving the related nonlinear system

m	T_m	T'_m
0	24.9	0.016
1	37.0	0.046
2	44.7	0.078
3	53.7	0.093
4	63.8	0.140
5	74.7	0.203
6	85.2	0.382
7	99.1	0.757
8	113.2	1.592

in Algorithm 3, we need to solve (3.7) once and (3.8) m times. In Table 2, T_m denotes the total time spent in Algorithm 1 for solving the nonlinear systems, while T'_m denotes that in Algorithm 3. Obviously, we see that, in Table 2, T'_m is much less than T_m.

Experiment 3. We illustrate in this experiment the approximation accuracy and the total computational effects of Algorithm 3, with a comparison to those of Algorithm 1 and Algorithm AC (the algorithm of Atkinson and Chandler presented in [**4**]).

In Table 3, we report the numerical results of Algorithms 1 and 3. For any m, we denote by $u_{4,m}$ and $\widetilde{u}_{4,m}$, respectively, the numerical solutions resulting from Algorithms 1 and 3. Moreover, we let T_M and T'_M denote the total times for implementing Algorithms 1

TABLE 3. Comparison of the accuracy and running time of Algorithms 1 and 3.

m	d_{4+m}	$\|u^* - u_{4,m}\|_\infty$	$\|u^* - \widetilde{u}_{4,m}\|_\infty$	T_M	T'_M
0	64	4.107e-3	5.079e-3	85	0.2
1	128	3.079e-4	3.007e-4	89	0.4
2	256	2.152e-5	2.225e-5	112	0.8
3	512	1.491e-6	1.527e-6	146	1.7
4	1024	8.204e-8	8.624e-8	203	3.7
5	2048	5.041e-9	5.313e-9	309	7.9
6	4096	2.865e-10	2.970e-10	519	16.9
7	8192	1.795e-11	1.882e-11	880	36.1
8	16384	1.023e-12	1.101e-12	1961	76.8

TABLE 4. Numerical results of Algorithm AC.

N	$\|u^* - u_N\|_\infty$	T_N
128	1.780e-5	3
256	1.071e-6	11
512	6.559e-8	52
1024	4.076e-9	226
2048	2.695e-10	836
4096	9.870e-11	3877
8192	6.751e-11	16473

and 3, respectively. We observe in Table 3 that $u_{4,m}$ and $\widetilde{u}_{4,m}$ have nearly the same accuracy, while T'_M is significantly less than T_M. This indicates that, although Algorithms 1 and 3 have the same order of computational costs, the techniques proposed in this paper effectively reduce the absolute computational costs.

For comparison, we list in Table 4 the numerical results of Algorithm AC, which is a Nyström method. We let N be the number of unknowns, u_N the numerical solution, and T_N the running time of the program.

As is pointed out in [**10**], we cannot compare Algorithms 1 and AC in the same discretization scale since Algorithm 1 is a multiscale method and Algorithm AC is a single scale quadrature method. However, we may utilize a "numerical errors versus computing time" figure to compare these methods. We use the data in Tables 3 and 4 to generate

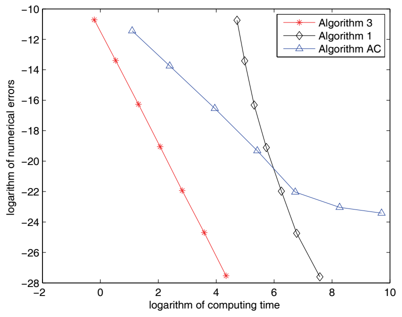

FIGURE 4. Comparison of Algorithms 1, 3 and Algorithm AC on numerical performance.

Figure 4. For convenience of display, we take logarithm on both numerical errors and computing times. It is seen that, for any accuracy level, Algorithm 3 uses the least time among the three algorithms.

REFERENCES

1. K.E. Atkinson, *A survey of numerical methods for solving nonlinear integral equations*, J. Integral Equat. Appl. **4** (1992), 15–46.

2. ———, *The numerical solution of a nonlinear boundary integral equation on smooth surfaces*, IMA J. Numer. Anal. **14** (1994), 461–483.

3. ———, *The numerical solution of integral equations of the second kind*, Cambridge University Press, Cambridge, 1997.

4. K.E. Atkinson and G. Chandler, *Boundary integral equation methods for solving Laplace's equation with nonlinear boundary conditions*: *The smooth boundary case*, Math. Comput. **55** (1990), 451–472.

5. K.E. Atkinson and J. Flores, *The discrete collocation method for nonlinear integral equations*, IMA J. Numer. Anal. **13** (1993), 195–213.

6. K.E. Atkinson and F. Potra, *Projection and iterated projection methods for nonlinear integral equations*, SIAM J. Numer. Anal. **24** (1987), 1352–1373.

7. R. Bialecki and A. Nowak, *Boundary value problems in heat conduction with nonlinear material and nonlinear boundary conditions*, Appl. Math. Model. **5** (1981), 417–421.

8. C. Brebbia, J. Telles and L. Wrobel, *Boundary element technique*, Springer, Berlin, 1984.

9. T. Carleman, *Über eine nichtlineare Randwertaufgabe bei der Gleichung* $\Delta u = 0$, Math. Z. **9** (1921), 35–43.

10. X. Chen, Z. Chen, B. Wu and Y. Xu, *Fast multilevel augmentation methods for a nonlinear boundary integral equation*, SIAM J. Numer. Anal. **49** (2011), 2231–2255.

11. Z. Chen, C.A. Micchelli and Y. Xu, *A construction of interpolating wavelets on invariant sets*, Math. Comput. **68** (1999), 1560–1587.

12. ———, *A multilevel method for solving operator equations*, J. Math. Anal. Appl. **262** (2001), 688–699.

13. ———, *Fast collocation methods for second kind integral equations*, SIAM J. Numer. Anal. **40** (2002), 344–375.

14. Z. Chen, B. Wu and Y. Xu, *Multilevel augmentation methods for solving operator equations*, Numer. Math. J. Chinese Univ. **14** (2005), 31–55.

15. ———, *Multilevel augmentation methods for differential equations*, Adv. Comput. Math. **24** (2006), 213–238.

16. ———, *Fast multilevel augmentation methods for solving Hammerstein equations*, SIAM. J. Numer. Anal. **47** (2009), 2321–2346.

17. Z. Chen, Y. Xu and H. Yang, *A multilevel augmentation method for solving ill-posed operator equations*, Inverse Problems **22** (2006), 155–174.

18. W. Fang, F. Ma and Y. Xu, *Multi-level iteration methods for solving integral equations of the second kind*, J. Integral Equations Appl. **14** (2002), 355–376.

19. H. Harbrecht and R. Schneider, *Wavelet Galerkin schemes for* 2D-BEM, in *Operator theory: Advances and applications* **121** (2001), Birkhäuser, Boston.

20. M. Jaswon and G. Symm, *Integral equation methods in potential theory and elastostatics*, Academic Press, London, 1977.

21. M. Kelmanson, *Solution of nonlinear elliptic equations with boundary singularities by an integral equation method*, J. Comput. Phys. **56** (1984), 244–283.

22. C.A. Micchelli and Y. Xu, *Using the matrix refinement equation for the construction of wavelets on invariant sets*, Appl. Comp. Harm. Anal. **1** (1994), 391–401.

23. J. Nedelec, *Approximation des Equations Integrales en Mecanique et en Physique*, Lecture Notes, Centre Math. Appl., Ecole Polytechnique, Palaiseau, France, 1977.

24. F. Rizzo, *An integral equation approach to boundary value problems of classical elastostatics*, Quart. Appl. Math. **25** (1967), 83–95.

25. K. Ruotsalainen and W. Wendland, *On the boundary element method for some nonlinear boundary value problems*, Numer. Math. **53** (1988), 299–314.

26. G.M. Vainikko, *Perturbed Galerkin method and general theory of approximate methods for nonlinear equations*, Zh. Vychisl. Mat. Fiz. **7** (1967), 723–751 (in Russian); USSR Comp. Math. Math. Phys. **7** (1967), 1–41 (in English).

27. W. Wendland, *On some mathematical aspects of boundary element methods for elliptic problems*, in *The mathematics of finite elements and applications* V, J. Whiteman, ed., Academic Press, London, 1985.

Guangdong Province Key Lab of Computational Science, Sun Yat-Sen University, Guangzhou 510275, P.R. China
Email address: chenxiangling828@163.com

Guangdong Province Key Lab of Computational Science, Sun Yat-Sen University, Guangzhou 510275, P.R. China
Email address: lnsczy@mail.sysu.edu.cn

Guangdong Province Key Lab of Computational Science, Sun Yat-Sen University, Guangzhou 510275, P.R. China
Email address: wubin@mail.sysu.edu.cn

Department of Mathematics, Syracuse University, Syracuse, NY 13244
Email address: yxu06@syr.edu