A FAST BLOCK-GREEDY ALGORITHM FOR QUASI-OPTIMAL MESHLESS TRIAL SUBSPACE SELECTION

LEEVAN LING*

Abstract. Meshless collocation methods are often seen as a flexible alternative to overcome difficulties that may occur with other methods. As various meshless collocation methods gain popularity, finding appropriate settings becomes an important open question. Previously, we proposed a series of sequential-greedy algorithms for selecting quasi-optimal meshless trial subspaces that guarantee stable solutions from meshless methods, all of which were designed to solve a more general problem: "Let A be an $M \times N$ matrix with full rank M; choose a large $M \times K$ submatrix formed by $K \leq M$ columns of A such that it is numerically of full rank." In this paper, we propose a block-greedy algorithm based on a primal/dual residual criterion. Similar to all algorithms in the series, the block-greedy algorithm can be implemented in a matrix-free fashion to reduce the storage requirement. Most significantly, the proposed algorithm reduces the computational cost from the previous $\mathcal{O}(K^4 + NK^2)$ to at most $\mathcal{O}(NK^2)$. Numerical examples are given to demonstrate how this efficient and ready-to-use approach can benefit the stability and applicability of meshless collocation methods.

Key words. Kansa method, kernel collocation, radial basis function, adaptive greedy algorithm, basis selection.

AMS subject classifications. 65D15, 65N35, 41A63.

1. Introduction. Since they were first proposed, meshless collocation methods, a.k.a. the Kansa method [19], have been used to solve many problems in science and engineering. Consider a time-independent linear partial differential equation,

$$\mathcal{L}u = f \text{ in } \Omega \subset \mathbb{R}^d \quad \text{and} \quad \mathcal{B}u = g \text{ on } \partial\Omega,$$

which can be formulated as a *generalized interpolation* problem by writing the equations as uncountably many simultaneous scalar equations $\lambda[u] = f_\lambda := \lambda[u^*]$ for all $\lambda \in \Lambda$. Here, u^* denotes the exact solution and the set of test functionals Λ is a collection of both the differential $\mathcal{L}$ and boundary $\mathcal{B}$ operators. A functional $\lambda_i \in \Lambda$ corresponding to a point $x_i \in \overline{\Omega}$ is defined by the differential operator and function evaluation, namely, $\lambda_i = \delta_{x_i}\mathcal{L}$; for $x_i \in \partial\Omega$, we have $\lambda_i = \delta_{x_i}\mathcal{B}$.

The unsymmetric collocation approach can then be specified by a radial basis kernel Φ along with $(X_M, \Xi_N, \mathbf{c}_N)$. The set $X_M \subset \overline{\Omega}$ specifies the M collocation locations. The set of trial centers $\Xi_N \subset \mathbb{R}^d$ and a (variable [7, 8, 20]) shape parameter vector $\mathbf{c}_N \in \mathbb{R}_+^N$ determine the kernels' positions and shapes to yield the set of *trial functions*

$$\{\Phi_j = \Phi(\|\cdot - \xi_j\|/c_j) \,:\, \xi_j \in \Xi_N,\, c_j = [\mathbf{c}_N]_j,\, 1 \leq j \leq N\}. \tag{1.1}$$

Selecting a set of collocation points X_M is equivalent to discretizing Λ by a subset of M functionals $\Lambda_M \subset \Lambda$. Once the kernel Φ is fixed, the numerical expansion is then

$$u^*(x) \approx \sum_{j=1}^{N} \eta_j\, \Phi_j(x) = \sum_{j=1}^{N} \eta_j\, \Phi(\|\cdot - \xi_j\|/c_j),$$

*Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. (lling@hkbu.edu.hk).

in which the unknown coefficients $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_N)^T$ can be determined by collocating the PDE at X_M and "solving" the resultant $M \times N$ system:

$$A(X_M, \Xi_N)\, \boldsymbol{\eta} = \mathbf{b}(X_M), \tag{1.2}$$

with entries

$$\big[\, A(X_M, \Xi_N) \,\big]_{i,j} = \lambda_i[\Phi_j], \quad \big[\, \mathbf{b}(X_M) \,\big]_i = \lambda_i[u^*], \tag{1.3}$$

where $x_i \in X_M$ for $i = 1, \ldots, M$ and $\xi_j \in \Xi_N$ for $j = 1, \ldots, N$. The original Kansa formulation [19] can now be seen as a special case under our generalized interpolation settings: $X_M = \Xi_N$ with $M = N$ and $\mathbf{c}_N = c \cdot \mathbb{1}$ for some constant $c > 0$. If meshless collocation methods are set up with some care, the resulting accuracy and efficiency are usually satisfactory and promising. Moreover, researchers observed that tweaking the Kansa method can improve accuracy and stability, for example, by collocating $\mathcal{L}$ and putting more trial functions on boundary [24, 31], using variable shape parameters [9, 20, 38], etc. Although the solvability of Kansa's formulation was questioned by Hon and Schaback [14], we showed that it is possible to find a proper trial space such that the matrix in (1.2) has rank M for any linearly independent Λ_M [27, 28]. Generally speaking, the search for quasi-optimal meshless trial spaces can be categorized into two types: the first involves determining the optimal shape parameter c by considering Ξ_N fixed [4, 18, 32, 35], and the second aims to seek a suitable problem-oriented trial subspace of a large trial space [15, 33] for the sake of stable computations. This paper focuses on the latter.

Previously proposed trial subspace selection algorithms, a.k.a. adaptive greedy algorithms, were iterative methods that built up non-singular subsystems of (1.2). In each iteration, the subsystem is expanded by adding a pair comprised of a collocation point and a trial center. First, a new collocation point associated with the largest local residual is added to the *selected set*. Then, a new trial center is selected by some other strategy, for example, by the determinant of the expanded matrix [27] or by a primal/dual residual criterion [29]. This *sequential* expansion process continues until the problem of ill-conditioning is detected. After K iterations, the algorithm has a subsystem of (1.2) specified by collocation points $X_K \subseteq X_M$ and trial centers $\Xi_K \subseteq \Xi_N$. Thus, the algorithm found a K-dimensional trial subspace for K linearly independent functionals. Using the submatrix specified by Ξ_K, an approximation to the unknown solution $\boldsymbol{\eta}$ of (1.2) can be obtained by solving a well-conditioned $M \times K$ linear system $A(X_M, \Xi_K)\boldsymbol{\eta} = \mathbf{b}$ in the least-squares sense [23] for computational efficiency or the L^∞ sense minimization [28] for convergence theory. The applications of these algorithms include both direct problems [1] and inverse problems [39]; see [36, 37] for problems with moving domains.

Example 1.1. This is a schematic example showing the basic idea of a sequential-greedy algorithm. Consider the following matrix system:

$$A\boldsymbol{\eta} = \begin{bmatrix} 1 & 4 & 3 \\ 2 & 5 & 2 \\ 3 & 6 & 1 \end{bmatrix} \boldsymbol{\eta} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} = \mathbf{b}.$$

Suppose a 1×1 subsystem with row-3 and column-2 has been selected, i.e., $X_1 = \{3\}$ and $\Xi_1 = \{2\}$. To expand the subsystem to 2×2, we first solve the subsystem $A(X_1, \Xi_1)\boldsymbol{\eta}^{(1)} = \mathbf{b}(X_1)$. In this simple demonstration, we have $[6]\boldsymbol{\eta}^{(1)} = [6]$ with solution $\boldsymbol{\eta}^{(1)} = 1$. The local residual can be evaluated by $\mu^{(2)} := A\widehat{\boldsymbol{\eta}}^{(1)} - \mathbf{b} = [2,\, 1,\, 0]^T$,

where $\widehat{\boldsymbol{\eta}}^{(1)} = [0,\, \boldsymbol{\eta}^{(1)},\, 0]$ is the upsample vector from the current subspace to the whole space such that $\widehat{\boldsymbol{\eta}}^{(1)}(\Xi_1) = \boldsymbol{\eta}_1$. By a *greedy* criterion, we select row-1 and get a bigger selected set $X_2 = \{3,\, 1\}$. To complete the subsystem expansion, we have to select a column. Let us use the intuitive determinant criterion [27] to complete the column selection. The to-be-determined 2×2 matrix can be seen as a function of column index $A_{2\times 2}(j) = A(X_2, \Xi_1 \cup \{x_j\})$ for $j = 1, 2, 3$. Obviously, $A_{2\times 2}(2)$ is singular, as column-2 has already been selected. Out of the two remaining candidates, e.g.,

$$\underbrace{\begin{bmatrix} 1 & 4 \\ 3 & 6 \end{bmatrix}}_{=A_{2\times 2}(1)} \quad \text{or} \quad \underbrace{\begin{bmatrix} 4 & 3 \\ 6 & 1 \end{bmatrix}}_{=A_{2\times 2}(3)},$$

we select the one with unsigned determinant closest to 1. In the actual implementation, one can use Cramer's rule to compute $A_{2\times 2}(j)$ for all j. As $|\det(A_{2\times 2}(1))| = 6$ and $|\det(A_{2\times 2}(3))| = 14$, column-1 will be selected. The expanded 2×2 subsystem now has rows $X_2 = \{3, 1\}$ and columns $\Xi_2 = \{2, 1\}$. This completes one iteration of a greedy algorithm. Since $A_{2\times 2}$ is well-conditioned, the next iteration can be started. □

The main drawback of the sequential approach is its high computational cost for large K. Take an $N \times N$ system as an example; if a sequential-greedy algorithm terminates in the full N steps, its complexity is $\mathcal{O}(N^4)$. This happens if the original linear system is well-conditioned, which is not straightforward to tell *a priori* without computing all entries in (1.3). After checking the whole trial space, the greedy algorithm returns the original system and all extra work spent on adaptivity is in vain. To improve efficiency, we propose running greedy algorithms in block. In each iteration, the algorithm attempts to double the dimension of the trial subspace. In Section 2, we thoroughly describe the procedures in one iteration of the block-greedy algorithm by the following four basic steps:

Step-1: Compute local residuals.
Step-2: Select rows to be added.
Step-3: Select columns to be added.
Step-4: Check condition number.

We also discribe motivations and reasons. As one can already surmise from Example 1.1, the main challenge is to properly select multiple rows and columns in each iteration. In Section 3, we use a pseudocode to summarize the proposed algorithm and study its complexity and storage requirement. To better help readers utilize the proposed algorithm, we demonstrate two novel applications after verifying the efficiency and effectiveness of the block-greedy algorithm in Section 4.

2. Updates in block by primal/dual residual criterion. The basic principles of the proposed block-greedy algorithm and the sequential ones are analogous. In this section, we will describe how to complete one iteration of a greedy algorithm in block, which can be interpreted as a combination of Householder reflections (when we call the built-in QR function) and the Gram–Schmidt process (when we use projection matrices). Some theorems that direct the actual implementation are reviewed or proven.

Assume there are two (potentially very large) sets X_M of collocation points and Ξ_N of trial centers. Let $A = A(X_M, \Xi_N) \in \mathbb{R}^{M\times N}$ and $\mathbf{b} = \mathbf{b}(X_M) \in \mathbb{R}^M$ with entries defined as in (1.3) denote the full matrix and right-hand side vector of the

original system (1.2). We will soon see that only some entries of A and $\mathbf{b}$ are needed for the iteration, and they can be computed on-the-fly. After completing the $(\ell-1)$st iteration, the algorithm has built up a determined or overdetermined $m(\ell-1)\times n(\ell-1)$ subsystem and a reduced QR factorization for $A(X_{m(\ell-1)}, \Xi_{n(\ell-1)})$. In the ℓth iteration, we aim to expand the subsystem from

$$A(X_{m(\ell-1)}, \Xi_{n(\ell-1)})\, \boldsymbol{\eta}^{(\ell-1)} = \mathbf{b}(X_{m(\ell-1)}) \quad \text{for} \quad \boldsymbol{\eta}^{(\ell-1)} \in \mathbb{R}^{n(\ell-1)} \tag{2.1}$$

to

$$A(X_{m(\ell)}, \Xi_{n(\ell)})\, \boldsymbol{\eta}^{(\ell)} = \mathbf{b}(X_{m(\ell)}) \quad \text{for} \quad \boldsymbol{\eta}^{(\ell)} \in \mathbb{R}^{n(\ell)} \tag{2.2}$$

such that the expanded system has twice as many rows and columns, i.e., $2m(\ell-1) \approx m(\ell) \geq n(\ell) \approx 2n(\ell-1)$.

For ease of readability, let us give an overview of the notation to come. We simply write $m = m(\ell-1)$ and $n = n(\ell-1)$ to denote the sizes of the pre-expanded subsystem. We use the prime notation to indicate that the update process of the corresponding item is completed. For simplicity, we also assume that there are enough unselected rows and columns to be chosen from. To expand the preselected sets of $m \leq M$ collocation points and $n \leq N$ trial centers, denoted by $X_m = X_{m(\ell-1)}$ and $\Xi_n = \Xi_{n(\ell-1)}$ respectively, we first add $p \approx m$ rows and then $q \lesssim p$ columns to expand the subsystem by selecting some appropriate subsets $X_p \subset X_M \backslash X_m$ and $\Xi_q \subset \Xi_N \backslash \Xi_n$ from the unselected points. The subscripts we use to expand the $(\ell-1)$st selected sets to the ℓth are

$$(m,n) \to \underbrace{(m+p,n)}_{\text{Add rows}} \to \underbrace{(m+p,n+q)}_{\text{Add columns}} =: (m',n').$$

The expanded ℓ-th subsystem then uses $m' = m(\ell)$ collocation points in $X' := X_{m'} = X_m \cup X_p$ and $n' = n(\ell)$ trial centers in $\Xi' := \Xi_{n'} = \Xi_n \cup \Xi_q$. For the corresponding expansion of the submatrix, we use the following notation:

$$A_1 := A(X_m, \Xi_n) \to \underbrace{\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = A_3}_{\text{Add rows}} \to \underbrace{[A_3\ A_4] = A'}_{\text{Add columns}},$$

where $A_2 = A(X_p, \Xi_n)$, $A_3 = A(X', \Xi_n)$, $A_4 = A(X', \Xi_q)$, and $A' = A(X', \Xi')$. Throughout the section, ideas are presented as points selections so that we have a geometric point of view. If one is only interested in the selection of a good set of columns to span the column space of matrix $A \in \mathbb{R}^{M\times N}$, the right-hand side vector $\mathbf{b}$ can be chosen arbitrarily, say as the one vector $\mathbb{1} \in \mathbb{R}^M$. Readers can interpret X_m as the row index and Ξ_n as the column index respectively; see Example 4.1.

2.1. (Step-1) Compute local residuals. *To solve a general interpolation subproblem and its dual problem in order to evaluate the primal and dual residuals at the unselected collocation points and trial centers respectively.*

The primal and dual residuals for a linear system with a rank M matrix $A \in \mathbb{R}^{M\times N}$ and an arbitrary vector $\mathbf{b} \in \mathbb{R}^M$ are built upon the following constrained minimization problem,

$$\begin{array}{rl} \text{minimize} & \frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{\eta} \\ \text{subject to} & A\boldsymbol{\eta} = \mathbf{b}, \end{array}$$

and the method of Lagrange multipliers, which is given in block form as

$$\begin{bmatrix} I_{N\times N} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\zeta} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}. \tag{2.3}$$

In our context, $A = A(X_M, \Xi_N)$ and $\mathbf{b} = \mathbf{b}(X_M)$ as in (1.2).

With X_m and Ξ_n preselected from the $(\ell-1)$st iteration, the primal and dual subsystems to (2.1) associated with these selected points are given by

$$\begin{cases} A(X_m, \Xi_n)\, \boldsymbol{\eta}^{(\ell-1)} &= \mathbf{b}(X_m), \\ A(X_m, \Xi_n)^T \boldsymbol{\zeta}^{(\ell-1)} &= -\boldsymbol{\eta}^{(\ell-1)}. \end{cases} \tag{2.4}$$

Since we assume $m \geq n$, solving (2.4) (in the least-squares sense if $m > n$) yields two vectors $\boldsymbol{\eta}^{(\ell-1)} \in \mathbb{R}^n$ and $\boldsymbol{\zeta}^{(\ell-1)} \in \mathbb{R}^m$. To get a local approximation to (2.3), define an upsampled vector $\widehat{\boldsymbol{\eta}}^{(\ell-1)} \in \mathbb{R}^N$ to be the extension of $\boldsymbol{\eta}^{(\ell-1)}$ with zeros patched into entries associated with unselected trial centers in $\Xi_N \setminus \Xi_n$. The upsampled $\widehat{\boldsymbol{\zeta}}^{(\ell-1)} \in \mathbb{R}^M$ is defined similarly using the set X_m. The primal residual for (2.4) is defined as

$$\boldsymbol{\mu}^{(\ell)} = A\widehat{\boldsymbol{\eta}}^{(\ell-1)} - \mathbf{b} = A(X_M, \Xi_n)\, \boldsymbol{\eta}^{(\ell-1)} - \mathbf{b}(X_M) \in \mathbb{R}^M. \tag{2.5}$$

In the case of $m = n$, all the entries in the primal residual corresponding to the selected collocation points X_m are zeros. This property no longer holds when $m > n$. The dual residual is defined similarly as

$$\boldsymbol{\nu}^{(\ell)} = \widehat{\boldsymbol{\eta}}^{(\ell-1)} + A^T \widehat{\boldsymbol{\zeta}}^{(\ell-1)} = \widehat{\boldsymbol{\eta}}^{(\ell-1)} + A(X_m, \Xi_N)^T \boldsymbol{\zeta}^{(\ell-1)} \in \mathbb{R}^N. \tag{2.6}$$

We omit the less relevant details and refer readers to the original article [29]. The following theorem is required for justifying our point selection strategies later.

THEOREM 2.1 ([29]). *The ith entry of the primal residual $\boldsymbol{\mu}^{(\ell)} \in \mathbb{R}^M$ (and the dual residual $\boldsymbol{\nu}^{(\ell)} \in \mathbb{R}^N$) is a scaled distance between the approximate solution and the ith hyperplane of the affine space containing the exact solution (and the Lagrange multiplier).*

In order to implement Step-1, we use the precomputed reduced QR factorization of $A(X_m, \Xi_n)$, which is available from Step-3 of the previous iteration, to solve (2.4) for $\boldsymbol{\eta}^{(\ell-1)}$ and $\boldsymbol{\zeta}^{(\ell-1)}$. As we shall only select points from the unselected points, we only need to compute the primal and dual residuals at the unselected points. We compute the primal residuals at the unselected collocation points $X_M \setminus X_m$ by

$$\boldsymbol{\mu}^{(\ell)}_{|_{X_M \setminus X_m}} = A(X_M \setminus X_m, \Xi_n)\, \boldsymbol{\eta}_n^{(\ell-1)} - \mathbf{b}(X_M \setminus X_m) \in \mathbb{R}^{M-m}. \tag{2.7}$$

Similarly, instead of (2.6), the dual residuals at the unselected trial centers $\Xi_N \setminus \Xi_n$ are given by

$$\boldsymbol{\nu}^{(\ell)}_{|_{\Xi_N \setminus \Xi_n}} = A(X_m, \Xi_N \setminus \Xi_n)^T \boldsymbol{\zeta}_m^{(\ell-1)} \in \mathbb{R}^{N-n}, \tag{2.8}$$

since $\widehat{\boldsymbol{\eta}}_n^{(\ell-1)}$ has zero entries at $\Xi_N \setminus \Xi_n$ by construction.

2.2. (Step-2) Select rows to be added. *To select a set $X_p \subset X_M \setminus X_m$ with $p \approx m$ collocation points based on the primal residual in* (2.7).

The following theorem shows that adding more collocation points to expand the $m \times n$ subsystem has minimal effect on the growth of the condition number.

THEOREM 2.2. *Let $A \in \mathbb{R}^{m_1 \times n}$ and $E \in \mathbb{R}^{m_2 \times n}$ with $m_1 \geq n$ be two arbitrary matrices. The condition number of the $\mathbb{R}^{(m_1+m_2)\times n}$ row-augmented matrix satisfies the following bound*

$$\kappa\left(\begin{bmatrix} A \\ E \end{bmatrix}\right) \leq \left(\frac{\sigma_1^2(E)+\sigma_1^2(A)}{\sigma_n^2(E)+\sigma_n^2(A)}\right)^{\frac{1}{2}} = \left(\frac{1+\sigma_1^2(E)/\sigma_1^2(A)}{1+\sigma_n^2(E)/\sigma_n^2(A)}\right)^{\frac{1}{2}} \kappa(A).$$

PROOF. Denote the row-augmented matrix by B. When the eigenvalues of $B^TB = A^TA+E^TE \in \mathbb{R}^{n\times n}$ are ordered in a nonincreasing order, they satisfy [10, Thm 8.1.5]

$$\lambda_j(A^TA) + \lambda_n(E^TE) \leq \lambda_j(B^TB) \leq \lambda_j(A^TA) + \lambda_1(E^TE)$$

for all $j = 1, \ldots, n$, or equivalently

$$\sigma_j^2(A) + \sigma_n^2(E) \leq \sigma_j^2(B) \leq \sigma_j^2(A) + \sigma_1^2(E),$$

the proposed upper bound can be derived from the inequalities associated with $j = 1$ and $j = n$. □

Theorem 2.2 can be applied to the two collocation matrices $A_1 := A(X_m; \Xi_n)$ and $A_2 := A(X_p\,, \Xi_n)$, which can be interpreted as two independent collocation processes at X_m and X_p using the same set of trial functions centered at Ξ_n. We could be dealing with collocation settings that are already ill-conditioned. As all the problem parameters (such as the shape parameter, trial centers, etc) are identical, it can be expected that the smallest and largest singular values of A_1 and A_2 are similar in magnitude. Theorem 2.2 then suggests that the row-augmented matrix will still have a condition number in the order of $\kappa(A_1)$.

Hence, Theorem 2.2 suggests that any approach for picking new collocation points X_p *at this stage* is rather safe with respect to the condition number. Any bad selection will only give us problems in later stages. Hence, we shall not work too much to select a set of "perfect X_p" and will simply pick $p \approx m$ new collocation points such that their primal residuals are well-separated.

STRATEGY 2.2.1. *Select $X_p \in X_M \setminus X_m$ such that the absolute primal residuals corresponding to X_p are approximately* $\text{floor}\big((M-m)/p\big)$ *positions apart in the sorted list of $|\boldsymbol{\mu}^{(\ell)}|$, defined as in* (2.7).

Theorem 2.1 suggests that the new collocation points in X_p will not be closely clustered. The whole set X_p of new collocation points will be added without further checking and we have $X' := X_m \cup X_p$ of size $m' := |X'| = m+p$. By working with an overdetermined subsystem, the extra rows can buffer any bad row selections. This is a new low-cost feature that the previous sequential-greedy algorithms did not offer. Computing a reduced QR factorization of $A_3 := A(X', \Xi_n) = Q_3R_3$ completes Step-2.

2.2.1. QR update. It is not within the scope of this paper to explore QR update algorithms, which were already studied, implemented, and tested (see, e.g., [11]). Nonetheless, it is possible to update the precomputed QR factorization of A_1 to include the added collocation points and obtain $A_3 = Q_3R_3$.

Using the precomputed reduced QR factorization $A_1 = Q_1 R_1$ with $Q_1 \in \mathbb{R}^{m\times n}$ and $R_1 \in \mathbb{R}^{n\times n}$, the row-augmented matrix can be written as

$$A_3 := A(X', \Xi_n) = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 & \\ & I_{p\times p} \end{bmatrix} \begin{bmatrix} R_1 \\ A_2 \end{bmatrix}.$$

By row pivoting A_2 to the top, the zeros in R_1 can speed up the householder QR algorithm, i.e., due to the fact that the number of nonzero entries is constant in all householder vectors instead of increasing in general cases. Performing another reduced QR factorization without pivoting to the $m' \times n$ matrix gives

$$\begin{bmatrix} A_2 \\ R_1 \end{bmatrix} = Q_2 R_2 = \begin{bmatrix} Q_2^u \\ Q_2^l \end{bmatrix} R_2,$$

with the lower $n \times n$ part of Q_2 denoted by Q_2^l. Using a customized code to take advantage of the sparsity can speed up this QR factorization. Combining with the matrix decomposition above yields the desired reduced QR factorization for the row-augmented matrix

$$A_3 := \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 & \\ & I_{p\times p} \end{bmatrix} \begin{bmatrix} Q_2^l \\ Q_2^u \end{bmatrix} R_2 = \begin{bmatrix} Q_1 Q_2^l \\ Q_2^u \end{bmatrix} R_2 =: Q_3 R_3. \tag{2.9}$$

The orthogonal matrix Q_3 should be constructed alongside R_3 in the customized code; otherwise, some extra work is needed to compute the matrix product $Q_1 Q_2^l$. When this update procedure is implemented in MATLAB exactly as described without a customized code, the runtime is almost identical to the time required to factor the row-augmented matrix A_3 from scratch. For this reason, we *will not* use QR update in the proposed algorithm.

2.3. (Step-3) Select columns to be added. *To select a set $\Xi_c \subset \Xi_N \setminus \Xi_n$ of $q_c > p$ candidates based on the dual residual in* (2.8) *and get a set $\Xi_q \subset \Xi_c$ of $q \lesssim p$ trial centers by QR.*

After Step-2, the expanded collocation center X' is kept fixed throughout the rest of the iteration. In the pursuit more trial centers, we begin by adding an arbitrary set Ξ_q to the subsystem in order to study its effect on the condition number. Let $P := Q_3 Q_3^T \in \mathbb{R}^{m'\times m'}$ denote the orthogonal projection matrix that projects onto the column space of A_3. Let $Q_4 \in \mathbb{R}^{m'\times q}$ and $R_4 \in \mathbb{R}^{q\times q}$ be a reduced QR factorization of $(I_{m'\times m'} - P)A_4$, where $A_4 := A(X', \Xi_q) \in \mathbb{R}^{m'\times q}$, i.e.,

$$(I_{m'\times m'} - P)A_4 = (I_{m'\times m'} - Q_3 Q_3^T)A_4 = Q_4 R_4. \tag{2.10}$$

With $\Xi' := \Xi_n \cup \Xi_q$ and $n' := n + q$, the column-augmented matrix can now be factorized as

$$A' := A(X', \Xi') = [A_3\ A_4] = [Q_3\ Q_4] \begin{bmatrix} R_3 & Q_3^T A_4 \\ & R_4 \end{bmatrix} =: Q' R'. \tag{2.11}$$

These steps are nothing but the standard (unmodified) Gram–Schmidt process in block. The updated matrix $Q' = [Q_3\ Q_4] \in \mathbb{R}^{m'\times n'}$ has orthonormal columns by construction. Thus, we obtain a QR decomposition for the new column-augmented matrix A'.

2.3.1. Growth of condition number. Finding upper bounds for condition numbers has been an important topic for years, and there are two main types of theorems based on either the Gersgorin disk [16] or arithmetic-geometric mean inequalities [40]. Since the submatrix A_3 gives us extra information about A', we can prove tighter bounds for column-augmented matrices within our context.

THEOREM 2.3. *Let $A_3 \in \mathbb{R}^{m\times n_3}$ and $A_4 \in \mathbb{R}^{m\times n_4}$ with $n_3 + n_4 \le m$ be two arbitrary matrices. Moreover, let $Q_3R_3 = A_3$ and $Q_4R_4 = (I - Q_3Q_3^T)A_4$ be a reduced QR factorization. The condition number of the column-augmented matrix $[A_3\ A_4] \in \mathbb{R}^{m\times(n_3+n_4)}$ satisfies the bounds (multiplicative-form)*

$$\kappa([A_3\ A_4]) \le \max\left\{1,\ \frac{\|R_4\|}{\sigma_1(A_3)},\ \sigma_n(A_3)\|R_4^{-1}\|\right\}\cdot\left(\|R_3^{-1}Q_3^TA_4\|+1\right)^2\kappa(A_3),$$

and (additive-form):

$$\kappa([A_3\ A_4]) \le \left(\max\left\{1,\ \frac{\|R_4\|}{\sigma_1(A_3)}\right\}+\frac{\|Q_3^TA_4\|}{\sigma_1(A_3)}\right)\cdot\left(\max\left\{1,\ \sigma_n(A_3)\|R_4^{-1}\|\right\}+\sigma_n(A_3)\|R_3^{-1}Q_3^TA_4R_4^{-1}\|\right)\kappa(A_3).$$

LEMMA 2.4. *Let B and E be $n\times n$ matrices. The set of singular values of B is $\{\sigma(B)\}$ and similarly for E and BE. Then, for $j = 1,\dots,n$, we have*

$$\max\{\sigma_n(B)\sigma_j(E),\sigma_j(B)\sigma_n(E)\} \le \sigma_j(BE) \le \min\{\sigma_1(B)\sigma_j(E),\sigma_j(B)\sigma_1(E)\}.$$

PROOF. This lemma follows from the multiplicative property of the spectral norm [17, Sec. 7.3, p.18]:

> "If $B, C^T \in \mathbb{R}^{m\times n}$ and $k = \min\{m,n\}$, then $\sigma_{i+j-1}(BC) \le \sigma_i(B)\sigma_j(C)$ for $i,j = 1,\dots,k$ and $i+j \le k+1$."

By setting $k = m = n$ and $i = 1$, we have

$$\sigma_j(BC) \le \sigma_1(B)\sigma_j(C) \quad \text{for } j = 1,\dots,n. \tag{2.12}$$

The upper bound arises by combining with the inequality with $i = 1,\dots,n$ and $j = 1$.

To obtain the lower bound, we can assume both $\sigma_n(B)$ and $\sigma_n(C)$ are strictly positive; otherwise, the corresponding lower bound is just the trivial one. Suppose $\sigma_n(B) > 0$; replacing B by B^{-1} and C by BC in (2.12) yields one of the lower bounds: $\sigma_j(B^{-1}BC) \le \sigma_1(B^{-1})\sigma_j(BC)$. The other part of the lower bound follows by arguments similar to C^{-1}. □

PROOF [Theorem 2.3]. Denote $R_{34} = Q_3^TA_4$; then the column-augmented matrix $[A_3\ A_4]$ has a reduced QR factorization:

$$[A_3\ A_4] = [Q_3\ Q_4]\begin{bmatrix} R_3 & R_{34} \\ & R_4 \end{bmatrix}.$$

Decompose the upper triangular matrix by the following products:

$$R := \begin{bmatrix} R_3 & R_{34} \\ & R_4 \end{bmatrix} = \underbrace{\begin{bmatrix} R_3 & \\ & R_4 \end{bmatrix}}_{=B}\underbrace{\begin{bmatrix} I_{n_3\times n_3} & R_3^{-1}R_{34} \\ & I_{n_4\times n_4} \end{bmatrix}}_{=C}. \tag{2.13}$$

Applying Lemma 2.4 (with $j = 1$), we have $\sigma_1(R) \le \sigma_1(B)\sigma_1(C)$. Since B is block diagonal, we have $\sigma_1(B) = \max\{\sigma_1(R_3), \sigma_1(R_4)\}$. To complete proving a bound for $\sigma_1(R)$, it remains to estimate the norm of

$$C := \begin{bmatrix} I_{n_3\times n_3} & R_3^{-1}R_{34} \\ & I_{n_4\times n_4} \end{bmatrix} = \underbrace{\begin{bmatrix} I_{n_3\times n_3} & \\ & I_{n_4\times n_4} \end{bmatrix}}_{=\tilde{B}} + \underbrace{\begin{bmatrix} 0 & R_3^{-1}R_{34} \\ & 0 \end{bmatrix}}_{=\tilde{C}}.$$

We use an SVD perturbation theory [10, Cor. 8.6.2]:

"If $\tilde{B}$ and $\tilde{B}+\tilde{C}$ are in $\mathbb{R}^{m\times n}$ with $m \ge n$, then for $j = 1, \ldots, n$

$$|\sigma_j(\tilde{B}+\tilde{C}) - \sigma_j(\tilde{B})| \le \sigma_1(\tilde{C}) = \|\tilde{C}\|_2."$$

By doing so, we know that $|\sigma_1(C) - 1| \le \|R_3^{-1}R_{34}\|$. Together with the fact that $\|C\| \ge 1$, we have a multiplicative upper bound for $\sigma_1(R)$,

$$\sigma_1(R) \le \max\{\|R_3\|, \|R_4\|\} \cdot \big(\|R_3^{-1}R_{34}\| + 1\big). \tag{2.14}$$

By applying similar techniques [17, Sec. 7.3, p.16] on decomposition

$$R = \begin{bmatrix} R_3 & R_{34} \\ & R_4 \end{bmatrix} = \begin{bmatrix} R_3 & \\ & R_4 \end{bmatrix} + \begin{bmatrix} 0 & R_{34} \\ & 0 \end{bmatrix},$$

an additive upper bound can be found,

$$\sigma_1(R) \le \max\{\|R_3\|, \|R_4\|\} + \|R_{34}\|. \tag{2.15}$$

Since both (2.14) and (2.15) hold in the 2-norm, these upper bounds for $\sigma_1(R)$ remain valid with the weaker Frobenius norm.

By inverting (2.13) as

$$R^{-1} = \begin{bmatrix} R_3^{-1} & -R_3^{-1}R_{34}R_4^{-1} \\ & R_4^{-1} \end{bmatrix} \tag{2.16}$$

the following lower bounds for the minimum singular value of R hold in both the 2- and the Frobenius norms:

$$\sigma_n(R) \ge \Big(\max\{\|R_3^{-1}\|, \|R_4^{-1}\|\} \cdot \big(\|R_3^{-1}R_{34}\| + 1\big)\Big)^{-1},$$
$$\sigma_n(R) \ge \Big(\max\{\|R_3^{-1}\|, \|R_4^{-1}\|\} + \|R_3^{-1}R_{34}R_4^{-1}\|\Big)^{-1}.$$

The bounds for condition numbers in the theorem result directly from these bounds on singular values. □

Example 2.1. Figure 2.1 gives two demonstrations for the bounds in Theorem 2.3. The matrices used in (a) and (b) result from a Kansa discretization of, respectively, a Poisson problem (by the multiquadric kernel) and an interpolation problem (by the Gaussian kernel) in the unit square. Using a set of 10×10 regular data points, we denote both $\mathbb{R}^{100\times 100}$ matrices as A'. The employed constant shape parameters are fine-tuned so that the condition numbers of A' fall to around $1.00(+16) := 1.00 \times 10^{16}$. The matrices $A_3 \in \mathbb{R}^{100\times 50}$ are formed by 50 randomly

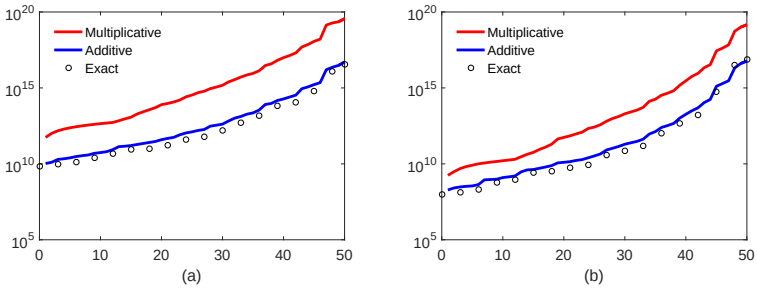

FIG. 2.1. *Comparisons of the bounds for condition numbers in Theorem 2.3 when column-augmenting two matrices of sizes* 100×50 *and* $100 \times q$ *arise from the meshless discretization of (a) a Poisson problem and (b) an interpolation problem*

chosen columns of A'. The matrices $A_4 \in \mathbb{R}^{100\times q}$ for $q = 1, \ldots, 50$ are formed by randomly chosen q columns out of the remaining 50 in A'. Figure 2.1 shows the condition numbers of $[A_3\ A_4] \in \mathbb{R}^{100\times(50+q)}$ for various q. In both cases, the additive-form gives a tighter estimate. □

The close agreement between the additive bound and the exact condition number is promising, but it also reveals the potential risk when we attempt to select trial centers in block. The bounds in Theorem 2.3 suggest that the growth of condition numbers is proportional to a factor involving $\|R_4^{-1}\|$. Furthermore, evaluating these bounds requires repeated computations of matrix-matrix products and norms, which are too computationally intensive. We need to select new trial centers by some other means.

2.3.2. Partition by dual residual. One of the benefits of using the dual residual is that it provides a cheap way to shortlist some potential candidates for new trial centers. Furthermore, we can avoid using the geometry of Ξ_N and allow the proposed algorithm to work on linear systems resulting from other utilizations. By Theorem 2.1, spreading the candidates with respect to their corresponding dual residuals ensures certain separating distances in between. Recall that p collocation points were added in Step-2. By shortlisting a (modest sized) set of $q_c > p$ trial candidates, we can employ more expensive algorithms to select the $q \lesssim p$ finalists without hurting the efficiency. We adopt the following strategy to select new trial centers.

STRATEGY 2.3.1. *Select* $\Xi_c \subset \Xi_N \setminus \Xi_n$ *such that the absolute dual residuals corresponding to* Ξ_c *are approximately* $\mathrm{round}\big((N-n)/q_c\big)$ *positions apart in the sorted list of* $|\boldsymbol{\nu}^{(\ell)}|$*, defined as in* (2.8)*. Then, select a set* $\Xi_q \subset \Xi_c$ *of* $q \lesssim p$ *new trial centers by a permuted QR factorization.*

EXAMPLE 2.2. We run a greedy algorithm on a linear system resulting from applying the Kansa method with multiquadric kernels to solve a Poisson problem. After solving the $n = 1024$ subproblems, the full set Ξ_N is partitioned according to the dual residuals. The resulting partitions are shown in Figure 2.2. In any sequential-greedy algorithm, a trial center associated with the largest dual residual in absolute value will be selected and the algorithm can move on to the next iteration. In block form, we attempt to select 1024 centers for the next iteration. It is obvious that choosing those only from the high end will make the selection clustered in a small region. □

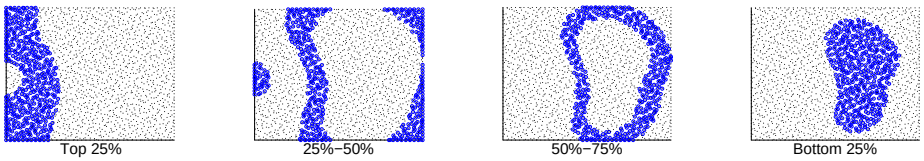

FIG. 2.2. *Schematic demonstration of partitioning trial centers by dual residuals.*

Selecting $\Xi_q \subset \Xi_c$ is now equivalent to selecting q columns out of $A_c := A(X', \Xi_c) \in \mathbb{R}^{m' \times q_c}$. To do so, we first compute a permuted QR factorization:

$$(I_{m' \times m'} - P)A_c = Q_c R_c E_c^T, \tag{2.17}$$

where $P := Q_3 Q_3^T$ is the same orthogonal projection used in (2.10). Now, pick the set $\Xi_q \subset \Xi_c$ to be the first q candidates according to the ordering in E_c. In matrix notation, A_4 consists of the first q columns of $A_c E_c$, e.g., $A_4 = A_c E_c \, [I_{q \times q} \; 0]^T$, and we can modify (2.17) to obtain a reduced QR factorization for

$$\begin{aligned}(I_{m' \times m'} - P)A_4 &= (I_{m' \times m'} - P)A_c E_c \, [I_{q \times q} \; 0]^T \\ &= (Q_c R_c E_c^T) E_c \, [I_{q \times q} \; 0]^T \\ &= (Q_c [I_{q \times q} \; 0]^T)([I_{q \times q} \; 0] R_c \, [I_{q \times q} \; 0]^T).\end{aligned}$$

Hence, Q_4 and R_4 in (2.10) consist of the first q columns of Q_c and the upper-left $q \times q$ submatrix of R_c respectively. This brings us back on track, and we can get $A' = Q'R'$ by (2.11) to complete Step-3.

2.4. (Step-4) Check condition number. *To check whether the extended submatrix A' is well-conditioned to allow further expansion.*

After expanding the subsystem in both rows and columns, we need to check the condition number of $A' = A(X', \Xi')$. If A' is ill-conditioned, we simply stop and return Ξ' in a sequential-greedy algorithm. The situation is more complicated in block form, especially when q is large. Returning the whole selected set $\Xi' = \Xi_n \cup \Xi_q$ could be $q - 1$ too many trial centers in the worst case.

Suppose we aim to have $\kappa(A') \leq 1/\varepsilon$ for some user-defined parameter ε, which is usually set to the machine epsilon $\varepsilon_{\text{mach}}$. In this final step, we have to achieve two objectives:

Objective 1: To estimate the condition number $\kappa(A')$ efficiently.

Objective 2: To select a large set $\Xi_K \subset \Xi'$ such that $\kappa(A(X', \Xi_K)) \leq 1/\varepsilon$ if $\kappa(A') > 1/\varepsilon$.

The block-greedy algorithm terminates based on the value of $\kappa(A')$, which should be estimated somehow. A practical estimator does not always have to be more accurate than the others. When the subsystem is well-conditioned, the algorithm will still go on running even with an inaccurate estimate; say, a lousy estimate to $\kappa(A') = 100$ with an error to a factor of thousands is still far below the tolerance. Accuracy is required only when we need to terminate the algorithm. The upper bounds in Theorem 2.3 are good estimators. However, they cost $\mathcal{O}(q^3)$ to compute. To make the proposed algorithm more efficient than its sequential predecessors, we want to keep the monitoring cost down to $\mathcal{O}(q^2)$ at most whenever we add q trial centers.

Remark: Step-3 is a numerically unstable Gram–Schmidt process, and if we use QR updates in Step-2, it will gradually make $\kappa(R') < \kappa(A')$. Recomputing QR

from scratch in Step-2 can reintroduce orthogonality into the factorization and keep $\kappa(R') \approx \kappa(A')$. Having said that, QR updates need not to be avoided completely. If one tries to choose a basis out of a well-conditioned underdetermined matrix, then the inaccurate R' results from QR updates will work as well. If the matrix is ill-conditioned, our numerical evidence suggests that the two condition numbers will be off by about three orders of magnitude. For example, if one wants to stop the algorithm when the condition number of A' is around $1.00(+15)$, the tolerance for $\kappa(R')$ should be about three orders of magnitude smaller.

2.4.1. Condition number estimators. The condition number of A' in (2.11) can be monitored by two norm estimators. To estimate $\kappa(A')$, we use a forward algorithm to compute $\|R'\| = \|A'\|$ and a backward algorithm for $\|R'^{-1}\| = \|A'^{-1}\|$.

First, we modify the forward algorithm found in [13] to continuously record the 2-norms of all the principal minors of the triangular matrix R'. The algorithm runs SVDs incrementally among all upper-left submatrices of R' to approximate the norms of all principal minors and the corresponding singular vector of R'. Despite their high complexity, all SVDs are applied to small matrices with two columns only. For $R' \in \mathbb{R}^{n' \times n'}$, we need $n' - 1$ SVDs of $2 \times j$ matrices, $j = 2, \ldots, n'$. The forward algorithm can also estimate $\|R'^{-1}\|$ if R'^{-1} is computed. Even with the inverse update, e.g., (2.16), using R'^{-1} in the algorithm will lead to an extra $\mathcal{O}(n^3+q^3)$ overhead. We will then end up with the same cost as using the upper bounds in Theorem 2.3. The solution is to give up having the norms of all principal minors of R'^{-1}. A backward algorithm in [30], without modification, allows us to approximate $\|R'^{-1}\|$ using entries of R' with a cost of $\mathcal{O}(n'^2)$. The idea of this algorithm is similar to the forward one but was built upon backward substitutions. For this reason, we cannot collect any information on the principal minors of R^{-1}.

Combining these norm estimators, we can estimate $\kappa(A') = \kappa(R') = \|R'\| \, \|R'^{-1}\|$ after each subsystem expansion. This addresses the first objective. These norm estimators can also help achieve the second objective. We know that $\kappa(R_3) < 1/\varepsilon$, and, therefore, the algorithm continued to the current iteration. If $\kappa(R') > 1/\varepsilon$, we are already in the right setting to start a root finding search:

- Let T_k be the first $k \times k$ block of R'. Define $F(k) := \|T_k\|\|T_k^{-1}\| - 1/\varepsilon$.
- Find $K \in (n, n')$ such that $F(K) < 0$ and $F(K+1) > 0$, provided that $F(n) < 0$ and $F(n') > 0$.

We already have the values of all $\|T_k\|$ in the memory. Evaluating $F(k)$ for any $k \in (n, n')$ is only subject to the estimation of $\|T_k^{-1}\|$. Since the bisection search must terminate after $\log_2(n' - n) = \log_2 q$ steps, the cost for finding K is at most $\mathcal{O}(q^2 \log_2 q)$. Estimating the condition number has been a topic of interest for a long time; see [12] for a survey. New algorithms are still appearing in the literature; see [34] for an example. Readers can employ any new estimator into the block-greedy algorithm to further improve its efficiency.

3. Implementation, complexity, and storage. We summarize a matrix-free block-greedy algorithm with the pseudocode in Algorithm 1, in which only the necessary entries in the matrix A are computed and stored. The stopping criteria include

- the largest absolute primal residual of the $m' \times n'$ subsystem (stop if $< \tau$),
- the size of subsystem (stop if $m' = M$ or $n' = N$), or
- the condition number of A' (stop if $\kappa(R') > 1/\varepsilon$ and return the first K indices of Ξ' such that $\kappa(A(X', \Xi_K)) \leq 1/\varepsilon$)

for some user-defined parameters $\tau, \varepsilon \ll 1$. MATLAB syntax is used for submatrix operations in the pseudocode. Some extra conditions (in lines–14, 18, 21) were in-

Decomposition/Operation		Operation counts	
Matrix-matrix product	AB	$2mnp$	
	AT	mn^2	
Inverse of triangular matrix	T^{-1}	$n^3/3$	
Full QR with pivoting	$QR = AE$	$3m^2n + m^3/3$	$(m \le n)$
		$4m^2n - mn^2 + n^3/3$	$(m \ge n)$
Reduced QR factorization	$QR = A$	$4mn^2 - 4n^3/3$	$(m \ge n)$
	$QR = AE$	$5mn^2 - 5n^3/3$	$(m \ge n)$
Reduced SVD factorization	$U\Sigma V^T = A$	$14mn^2 + 8n^3$	$(m \ge n)$
Add rows to $A = QR$	$\tilde{Q}\tilde{R} = [A^T\ B^T]^T$	$4n^2p + mn^2$	

TABLE 3.1

Operation counts for some matrix decompositions and operations. $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$ are arbitrary, and $T \in \mathbb{R}^{n\times n}$ is triangular.

cluded to keep track of the number of added rows, p, and columns, q, in order to keep the subsystems' dimensions within the size range of the input matrix A. The implementation for estimating $\kappa(R)$ (lines–25, 26) and the search for K (line–28) were implemented as in Section 2.4.1.

Assume m, n, p, q are of the same order of magnitude, which we denote by k. The ideas in Section 2 cover most of the lines inside the while-loop in Algorithm 1:

Step-1: Lines–11 to 12 work on A_1 of size $\sim k \times k$.
Step-2: Lines–14 to 17 work on A_3 of size $\sim 2k \times k$.
Step-3: Lines–18 to 23 work on A_c of size $\sim 2k \times q_c$.
Step-4: Lines–25 to 31 work on A' of size $\sim 2k \times 2k$.

Complexity counts in each line are performed carefully (with leading constant) according to those listed in Table 3.1 and listed on the right-hand side of Algorithm 1. In line–20, we assume that A_c is underdetermined to allow shortlisting of all unselected trial centers. Altogether, the cost per iteration is

$$\sim 20k^2q_c + 28k^3/3 + \mathcal{O}(k^2 + Mk + Nk),$$

in which M and N only appear in the lower-order terms.

Suppose that the block-greedy algorithm ceases to expand after selecting n' columns and n' is some power of 2. Once we choose a way to determine the number of shortlisted columns q_c, the total cost for the block-greedy algorithm to build up subsystems of size $k \times k$ for $k = 1, \ldots, 2^j, \ldots, n'/2$ can be obtained easily by the following equalities:

$$\sum_{j=1}^{\log_2(n'/2)} 2^j = n' - 2, \qquad \sum_{j=1}^{\log_2(n'/2)} \left(2^j\right)^2 = \frac{n'^2}{3} - \frac{4}{3}, \qquad \sum_{j=1}^{\log_2(n'/2)} \left(2^j\right)^3 = \frac{n'^3}{7} - \frac{8}{7}.$$

First, if all unselected columns are included in the shortlist of each iteration, then the total cost of running the block-greedy algorithm to select the final set $\Xi_{n'}$ is

$$\sim \frac{20}{3}Nn'^2 - \frac{32}{21}n'^3 + \mathcal{O}(Mn' + Nn') \quad \text{for } q_c = N - k. \tag{3.1}$$

Algorithm 1 Block-greedy algorithm

function BLOCKGREEDY(A, b, τ, ε)
  Inputs: $A \in \mathbb{R}^{M\times N}$ or entry-generator $(x_{(i)}, \xi_{(j)}) \to [A]_{i,j}$, $b \in \mathbb{R}^M$, $\tau, \varepsilon > 0$
  Initialize: $k = 1$
  $\mu = b$, find an index $x_{(1)}$ by the largest magnitude
  Compute and store all entries of the row $A(x_{(1)}, :)$
  $\nu = A(x_{(1)}, :)$, find an index $\xi_{(1)}$ by the largest magnitude
  Compute and store all entries of the column $A(:, \xi_{(1)})$
  $k = 1$, $X_{sel} = \{x_{(1)}\}$, $\Xi_{sel} = \{\xi_{(1)}\}$ ▷ User can select starting X_{sel}, Ξ_{sel}
  [Q,R] = qr($A(X_{sel}, \Xi_{sel})$, 0)
  while $k < \min(M, N)$ **do**
    Solve dual linear system (2.4) by Q, R in memory ▷ $\mathcal{O}(k^2)$
    Compute μ and ν by (2.7)–(2.8) ▷ $\mathcal{O}(Mk + Nk)$
    if $\max(|\mu|) < \tau$ **then** Break
    Select X_p by Strategy 2.2.1
    Compute and store all new entries of the rows $A(X_p, :)$ ▷ $\mathcal{O}(Nk)$
    $X_{sel} \leftarrow X_{sel} \cup X_p$
    $[Q, R] = \mathrm{qr}(\ A(X_{sel}, \Xi_{sel}),\ 0\ )$ ▷ $\sim 20k^3/3$
    Select Ξ_c by Strategies 2.3.1 and 3.0.1
    $R_{12} = Q^T * A(X_{sel}, \Xi_c)$ and $B = A(X_{sel}, \Xi_c) - Q * R_{12}$ ▷ $\sim 2 \times 4k^2 q_c$
    $[Q_c, R_c, E] = \mathrm{qr}(\ B,\ 0\ )$ ▷ $\sim 12k^2 q_c + 8k^3/3$
    $q = \min(\min(M, N) - k, k)$
    $Q \leftarrow [Q,\ Q_c(:, 1:q)]$ and $R \leftarrow [R,\ R_{12}(:, 1:q)\ ;\ \mathbf{0},\ R_c(1:q, 1:q)]$
    $\Xi_{sel} \leftarrow \Xi_{sel} \cup \Xi_q$ according to the permutation E
    Compute and store all new entries of the columns $A(:, \Xi_q)$ ▷ $\mathcal{O}(Mk)$
    Estimate and store $\|R(1:\ell, 1:\ell)\|$, $1 < \ell \le q$ ▷ $\mathcal{O}(k^2)$
    Estimate $\|R^{-1}\|$ and $\kappa(R) \approx \|R(1:q, 1:q)\|\|R^{-1}\|$ ▷ $\mathcal{O}(k^2)$
    if $\kappa(R) > 1/\varepsilon$ **then**
      Bisection to find K by computing $\|R^{-1}(1:\ell, 1:\ell)\|$, $k < \ell < q$
      $\Xi_{sel} \leftarrow \Xi_{sel}(1:K)$
      Break
    $k \leftarrow 2k$
  return Ξ_{sel}

If we pick $q_c = \rho k$ for some constant $\rho > 1$, the overall cost must not exceed (3.1). For small n', the cost of the block-greedy algorithm is $\sim (4/3 + 20\rho/7)n'^3 + \mathcal{O}(Mn' + Nn')$, which is an overestimate for large n', as the number of unselected columns will eventually be smaller than q_c. Therefore, we can conclude that the overall complexity of selecting $\Xi_{n'}$ is

$$\sim \min\left\{\frac{20}{3}N - \frac{32}{21}n',\ (\frac{4}{3} + \frac{20\rho}{7})n'\right\} n'^2 + \mathcal{O}(Mn' + Nn') \quad \text{for } q_c = \rho k. \tag{3.2}$$

Last but not least, the bisection search for determining $K \le n'$ could cost up to $\mathcal{O}(n'^2 \log n')$ in the most unfortunate situation. The search for K will be slow when A' is ill-conditioned ($K < M$) and K is large ($K \approx M$). In such a scenario, the block-greedy algorithm has an $\mathcal{O}(M^3)$ complexity and the search for K costs at most $\mathcal{O}(M^2 \log_2 M)$ operations, which does not affect the leading cost of the algorithm. Therefore, by $n' < 2K$, we can conclude that the cost of running Algorithm 1 is *at*

most

$$\sim \min\left\{\frac{20}{3}N - \frac{64}{21}K,\ (\frac{8}{3} + \frac{40\rho}{7})K\right\} 4K^2 + \mathcal{O}(MK + NK) \quad \text{for } q_c = \rho k, \qquad (3.3)$$

or, by $K \leq M \leq N$, simply $\mathcal{O}(NK^2)$ as in the abstract.

EXAMPLE 3.1. Let us analyze the worst case scenario that motivates the development of the block-greedy algorithm. Suppose one begins with a well-conditioned square $N \times N$ system $A\boldsymbol{\eta} = \mathbf{b}$. Without any *a priori* knowledge of the conditioning of the linear system, solving it by the regularized-QR approach costs $\sim 10N^3/3$ for a $QR = AE$ factorization. In contrast to the sequential approach, which takes a very large $\mathcal{O}(N^4)$ work to select all N trial centers, the block-greedy algorithm takes $\sim (20/3 - 32/21)N^3 \approx 5N^3$ works to complete the same task. Here, we use the complexity in (3.2) with $n' = N$ because there is no need to search for K. In other words, the cost of adaptivity is reduced to just a fraction of the cost of finding a QR factorization. □

Empirical evidence suggests that $\rho \geq 2$ is sufficient to ensure the robustness of Algorithm 1. By balancing the $\mathcal{O}(NK^2)$ and $\mathcal{O}(K^3)$ terms in (3.2) along with the $\mathcal{O}(M^2 \log_2 M)$ cost in the bisection search, we adopt the following strategy for selecting the search size q_c in each iteration.

STRATEGY 3.0.1. *Select $q_c = N - k$ to search all unselected columns if $M < 7N/9$, or otherwise $q_c = \max(2, \log_{10} M)\, k$.*

The total storage requirement of the block-greedy algorithm is slightly higher than the sequential version in [29], which requires $\sim K(M + N)$ for storing the necessary matrix entries (in lines–15 and 24) and an approximated matrix inverse, i.e., A'^{-1} in our notation. Algorithm 1 needs the same storage for the entries of A and Q'. On top of that, it needs extra storage for the $K \times K$ upper triangular matrix R'. The memory needed to run the block-greedy algorithm is therefore

$$\sim K(M + N) + \frac{1}{2}K^2 \quad \text{(storage)}.$$

A MATLAB script for the block-greedy algorithm is available for download.[1] To make use of the matrix-free feature, readers can modify the script according to the comments within to generate matrix entries on-the-fly.

4. Numerical demonstrations. In this section, we will illustrate the efficiency, quasi optimality, accuracy, and possible applications of the proposed block-greedy algorithm. We present five examples, in all of which the block-greedy algorithm is run with stopping tolerances $\tau = \varepsilon = \varepsilon_{\text{mach}}$. The first example aims to compare the performance and efficiency of the proposed algorithm and a permuted QR algorithm in the problem basis selection for linear dependency. The other four examples focus on the applications of the block-greedy algorithm to the original Kansa method. In these examples, we do not apply any scaling to the collocation conditions for accuracy improvement [26] and solve linear systems as in (1.2). The trial functions in (1.1) are generated either by the multiquadric (MQ) or the Gaussian (GA) kernels. To avoid confusion by using either one of the common RBF notations,

$$\Phi_j = \Phi(\|\cdot - \xi_j\|/c) = \Phi(\epsilon\|\cdot - \xi_j\|),$$

[1] See `http://www.mathworks.com/matlabcentral/fileexchange/55342` or `www.math.hkbu.edu.hk/~lling/blockgreedy.m`

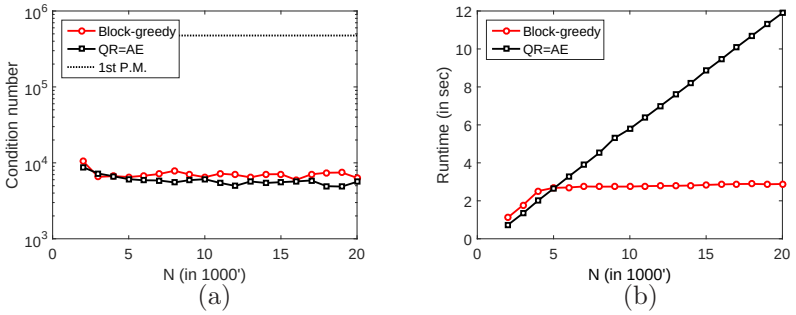

FIG. 4.1. *Example 4.1: (a) Condition numbers of various $M \times M$ matrices B formed by the selected M columns out of $A \in \mathbb{R}^{M\times N}$ with $M = 1500$ and $2000 \le N \le 20000$ using the block-greedy algorithm and a permuted QR factorization and (b) the respective computational times.*

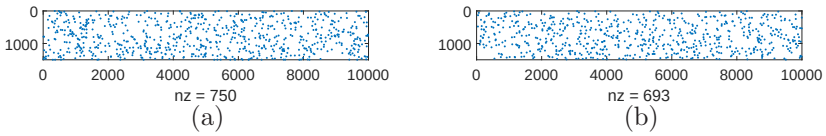

FIG. 4.2. *Example 4.1: Entries of $B^{-1}A$ with absolute values greater than 1, where B is the $M \times M$ submatrix formed by the selected M columns out of $A \in \mathbb{R}^{M\times N}$ with $M = 1500$ and $N = 10000$, using (a) the block-greedy algorithm and (b) a permuted QR factorization.*

we use a constant shape parameter $c := 1/\epsilon = 1$, unless otherwise specified. Note that these unscaled kernels are not optimized for any of our test problems. It is consistently observed in both two and three dimensions that more *unscaled* MQ than GA trial functions are needed in the presented examples; however, this is not numerical evidence for comparing the performance of the MQ and GA kernels.

For all PDEs in two dimensions, the corresponding right-hand side functions and boundary data are generated by either of the following exact solutions:

- *Full peaks*: `peaks(3x,3y)` from MATLAB.
- *Zoom-in peaks*: `peaks(x,y)` from MATLAB.

All scattered data are generated by the Halton sequences; see Figure 2.2. In all of the examples, we report numerical errors in the L^∞ norm, which are estimated by sufficiently dense sets of evaluation points.

EXAMPLE 4.1. To give readers a glimpse of the efficiency, we first consider the standard problem of basis selection for linear dependency. This test is all about the column selection strategies, and the norm estimators in Section 2.4.1 have no role to play. Reported runtime in MATLAB is obtained from an Intel-i7 desktop computer.

The experiment is set up as follows: a sequence of matrices $A \in \mathbb{R}^{M\times N}$ with a fixed number of rows $M = 1500$ and various number of columns $2000 \le N \le 20000$ is generated by the MATLABfunction `rand` with the random number generator reset, i.e., `rng(0)` to be specific. The goal is to select M columns out of A and we denote the selected submatrix by $B \in \mathbb{R}^{M\times M}$. Without any selection, if we take the first $M \times M$ block of A as B, which is fixed by construction, the condition number of B is 4.7513(+5). For any good selection, the condition number of B is expected to be small. Figure 4.1(a) compares condition numbers of the selections, $\kappa(B)$, for two different matrices B selected by the proposed block-greedy algorithm and by the MATLABcommand `qr` command respectively. As seen, the selection qualities of the

block-greedy algorithm are highly competitive with the results from QR. With M being fixed, the block-greedy algorithm requires a constant $\mathcal{O}(M^3)$ time, whereas the QR factorization of an underdetermined matrix requires an $\mathcal{O}(M^2N)$ linear time in N; see the actual run times in Figure 4.1(b).

In [22], it is shown that, for any $\theta > 1$, it is possible to select M columns of A to form a submatrix B, such that the absolute values of all entries of $B^{-1}A$ are less than or equal to θ, in polynomial time in M and N. Any entry in $B^{-1}A$ that is greater than 1 in absolute value indicates the corresponding column is lying outside the parallelepiped formed by the columns of B. What we show in Figure 4.2 is the sparsity pattern of such entries with the submatrix B selected by the block-greedy algorithm and the MATLABcommand `qr`. The maximum magnitudes of these entries are 1.266 and 1.148 in the block-greedy and QR algorithms respectively. For comparison, the maximum is 67.31 if we simply pick B by the first $M \times M$ block of A. For these measures, the QR algorithm yields a slightly better column selection (i.e., a bigger parallelepiped) but bears a higher computational cost. If we allow the block-greedy algorithm to search over all unselected columns to find new columns, the resulting maximum entry in $B^{-1}A$ will drop to 1.059 in absolute value with a trade-off of higher computational cost. □

Example 4.2. This experiment aims to demonstrate that the greedy algorithms with primal and dual residuals (2.5)–(2.6) are more robust when run in block as proposed than sequentially as in [29]. The improvement in robustness is due to the error buffer for bad collocation point selections; i.e., the block-greedy algorithm uses more rows than columns in its construction of subsystems. To study the difference in convergence and accuracy, we solve the Poisson equation with Dirichlet boundary conditions in $\Omega = [-1, 1]^2$ by the original Kansa method with the MQ kernel. To keep our numerical results easily reproducible, the tests are run with $M = N = 6^2, 11^2, \ldots, 101^2$ regular grids.

Figure 4.3 shows the observed results for two different exact solutions: the zoom-in peaks (top) and the full peaks (bottom). In Figures 4.3(a)–(b), we show the numerical errors against increasing $\sqrt{N}$. For each N, the matrix in the collocation system (1.2) are identical for both exact solutions. Yet, the observed convergence rates are rather different. Figures 4.3(c)–(d) show the numbers of trial centers, K, being selected, and Figures 4.3(e)–(f) show the two numerical solutions for $N = 31^2$ and the selected trial centers by the block-greedy algorithm.

As both the block- and the sequential-greedy algorithms are built upon the same greedy criterion, their performances are identical for small N when the linear systems are still well-conditioned. As N increases, with a fixed shape parameter $c = 1$, the linear systems soon suffer from the problem of ill-conditioning. Greedy algorithms are designed to deal with such situations by selecting a subspace from a given trial space to yield a well-conditioned subsystem. In this sense, the block-greedy algorithm is doing a better job by selecting more trial centers and, hence, a larger trial subspace.

Also note that the number of selected columns K is due to the properties of the given trial spaces. The fact that the growth of K stagnates for large N in Figures 4.3(c)–(d) can be interpreted as follows: *the maximum dimension of a "well-conditioned" trial space is around $K = 880$ for the unscaled MQ kernel centered on a set of quasi-uniform trial centers in the unit square domain* (repeating the same test with the GA kernel, we will see that the numbers of selected GA trial functions stagnate at around $K = 160$). The convergence profiles with respect to $\sqrt{N}$ in Fig-

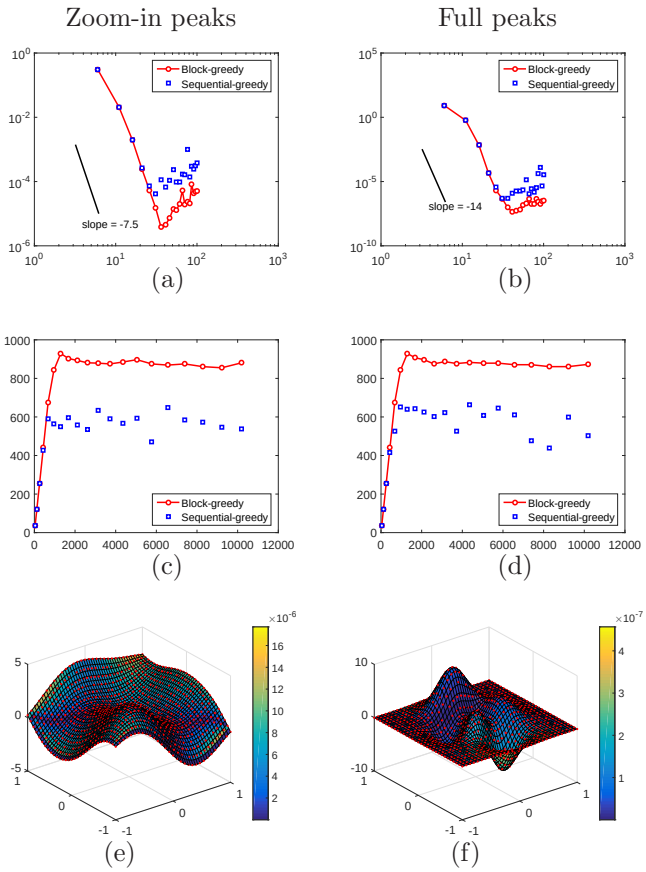

FIG. 4.3. *Example 4.2: (a)–(b) Convergence profiles, (c)–(d) the numbers of selected trial centers of the block- and sequential-greedy algorithms, and (e)–(f) selected trial centers by the block-greedy algorithm on numerical solutions.*

ures 4.3(a)–(b) are no longer purposeful after the fill distances of the selected trial centers Ξ_K stop reducing with increasing N. More importantly, by using the block-greedy algorithm, the numerical approximations can be obtained in $\mathcal{O}(NK^2)$ works instead of $\mathcal{O}(N^3)$. Computational cost only increases in linear time with respect to the original problem size. □

EXAMPLE 4.3. It has long been known that the performance of meshless methods, particularly those using global kernels, depends heavily on the choice of the shape parameter. For quite some time, researchers have been trying to overcome this problem by finding either the optimal shape parameter or the data/geometry-driven variable shape parameter. Even though our approaches are very different in nature, our block-greedy algorithm is similar to various stable evaluations of GA kernel systems [2, 3, 5, 6] in the sense that all methods try to safeguard the solution of severely ill-conditioned linear systems from being numerical garbage if one picks a "bad" shape parameter.

Kernel	c	K, out of 961	L^∞ error
MQ	2	251	8.6928(-5)
	1	813	5.4408(-5)
	0.5	961	0.0039
GA	2	77	0.2747
	1	163	1.6022(-8)
	0.5	413	1.7666(-5)

TABLE 4.1

Example 4.3: Accuracy of adaptive meshless collocation methods with constant shape parameters.

Kernel	c	K, out of 961	L^∞ error
MQ	$(1/2, 2)$	859	7.3448(-7)
	$(1/3, 3)$	894	5.6504(-7)
	$(1/4, 4)$	920	1.4412(-6)
GA	$(1/2, 2)$	385	2.0364(-11)
	$(1/3, 3)$	574	2.5763(-10)
	$(1/4, 4)$	673	7.6422(-10)

TABLE 4.2

Example 4.3: Accuracy of adaptive meshless collocation methods with uniformly random shape parameters.

Let us see what happens if we employ the Kansa method to the following problem:

$$\begin{aligned} \triangle u + [1,\ 1]^T \cdot \bigtriangledown u + 5u = f \quad & \text{in } \Omega = [-1,1]^2, \\ \partial_n u = g_1 \quad & \text{on } \Gamma = \{(x,y) :\ x \in [-1,1],\ y = 1\}\ , \\ u = g_0 \quad & \text{on } \partial\Omega \setminus \Gamma\ . \end{aligned} \tag{4.1}$$

Let the zoom-in peaks be the exact solution to (4.1). Using a set of $M = N = 961$ scattered data and different shape parameters $c = 0.5, 1, 2$, the resulting errors are shown in Table 4.1. Both kernels have the potential to yield highly accurate results whenever they are fed with some appropriate shape parameters. More than that, a just-right parameter can also achieve high accuracy with a small number of trial functions. However, a bad one (e.g. the peaky MQ kernel with $c = 0.5$) uses up all provided trial centers and yields a less accurate numerical approximation.

One may try to pick a good shape parameter based on the errors in Table 4.1; say, the optimal c are around 1.5 and 0.8 for the MQ and GA kernels respectively. Just how hard is it to find the optimal shape parameter, which literally depends on everything in the problem? Even with all of the work from collecting the data, we say it is impossible. Using elementary statistical arguments, if we can use an interval estimate instead of a point estimate, then we can start talking about confidence interval and so on. Besides empirical evidence and luck, we now have a reliable adaptive block-greedy algorithm. Suppose we make an empirical guess that both optimal parameters are inside an interval $(1/c, c)$. Without using the geometry of the scattered data, we take an even more controversial approach and assign shape parameters randomly. This is more or less like a variable shape parameter approach, in which different trial centers

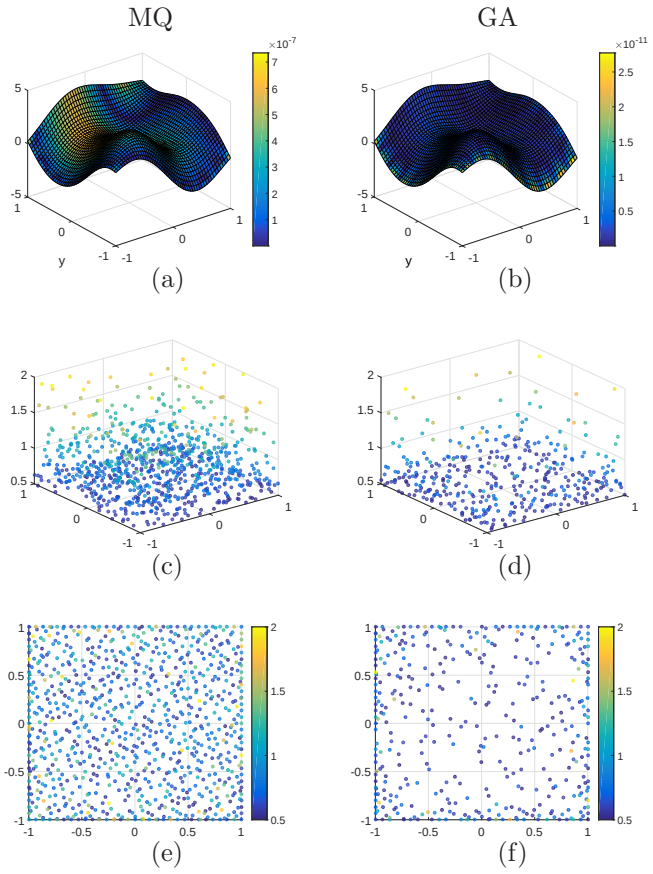

FIG. 4.4. *Example 4.3: Selected trial centers and shape parameters in meshless collocations with random shape parameters.*

use kernels with different scales. Such settings immediately destroy the reproducing kernel Hilbert space of the kernel and send us away from all theoretical support. Hopefully, the block-greedy algorithm can deal with this.

Table 4.2 shows the results of using the MQ and GA kernels with *uniformly random* shape parameters. The random shape parameter approach results in good approximations that are even more accurate than those obtained by any constant shape parameter in the range. Figures 4.4(a)–(b) show the numerical solutions obtained by the MQ and GA kernels with random shape parameters in $[0.5, 2]$. The selected MQ and GA trial centers and the corresponding shape parameters (in the z-axis) are shown in Figures 4.4(c)–(d). Apparently, the block-greedy algorithm prefers peaky trial functions (small c) in general. From Figures 4.4(e)–(f), which are the projections of Figures 4.4(c)–(d) onto the xy-plane, we can also see that flatter trial functions (large c) are more favorable near the boundary.

To ensure that this is not a coincidence, Figure 4.5 demonstrates the accuracy of the same test on increasing numbers of scattered data. For each set of scattered data, we run the test for 10 sets of random shape parameters generated by different seeds, i.e. `rng(SD)` with $\texttt{SD} = 0, \ldots, 9$. The convergence behavior and the trend in numbers of selected trial centers are indeed very similar to the constant shape parameter case

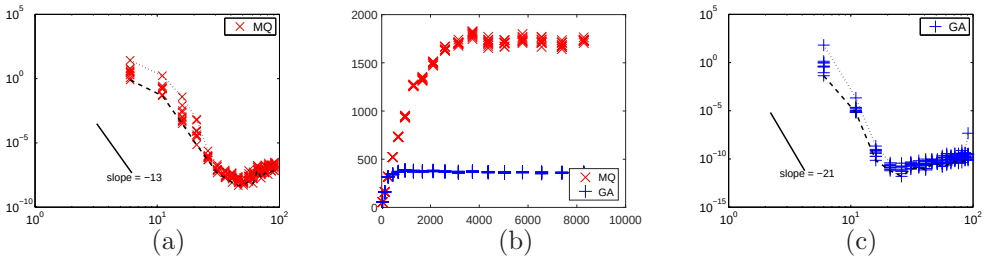

FIG. 4.5. *Example 4.3: Convergence profiles and the numbers of selected trial centers for meshless collocations using multiquadric (MQ) and Gaussian (GA) kernels with random shape parameters in* [0.5, 2].

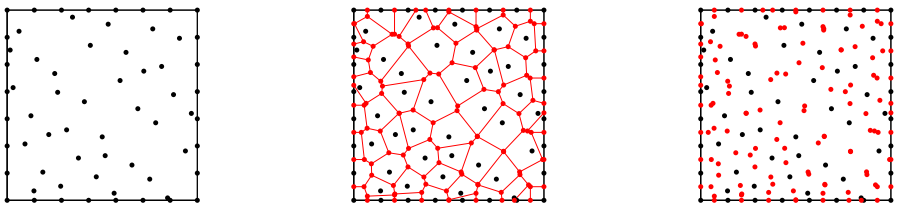

FIG. 4.6. *Example 4.4: Adaptively generated data points within the block-greedy algorithm by Voronoi diagram.*

when compared with Figure 4.3(a)–(b). Again, we can see an improvement in accuracy by a few orders of magnitude. We hope that these observations will open up a new research direction[2] but we shall not proceed further in this work. □

EXAMPLE 4.4. In this example, we want to demonstrate some possibilities for how one can tweak the block-greedy algorithm to yield a *system-free* algorithm; i.e. we run the block-greedy algorithm without prefixing the linear system (1.2). Instead of starting the block-greedy algorithm with fixed sets of collocation points and trial centers, we start Algorithm 1 with small sized X_M and Ξ_N (i.e., a small original linear system) but generate more points whenever we need more unselected points.

To expand X_M and Ξ_N on-the-fly, we need to generate new distinct points that are not too close to the existing point sets. Voronoi diagrams are designed to have this desired property. We can consider the points in Figure 4.6(a) as either set of the preselected points and then compute the Voronoi diagram as in Figure 4.6(b). Finally, we add in all the vertices of the diagram to form a new data set. As seen in Figure 4.6(c), some new points are clustered and we will let the block-greedy algorithm deal with the problem.

As a demonstration, we employ the GA kernel to solve the PDE in (4.1) with Dirichlet boundary conditions in an amoeba shape domain whose boundary is given in polar form by

$$r(\theta) = \exp(\sin\theta)\sin^2 2\theta + \exp(\cos\theta)\cos^2 2\theta.$$

We take the zoom-in peaks to generate the boundary data. Figure 4.7 shows the

[2]For readers' information, the MQ kernel with random shape parameter can achieve high accuracy with many fewer trial functions if the parameter range is in the form of (ϵ, c) with $1 \approx \epsilon \ll c$.

expansion process of the sets of all collocation points X_M (left) and trial centers Ξ_N (right). In the first row, we show the identical set of 53 points that is used to start the block-greedy algorithm. At first, the algorithm runs exactly as described in Algorithm 1. After a few iterations, when the block-greedy algorithm attempts to add in a majority of the unselected points, two Voronoi diagrams are drawn based on the selected collocation points and trial centers respectively. By adding in the vertices of these diagrams, both sets of unselected points are expanded; see the second row in Figure 4.7 for the results of one such expansion, in which all the selected points are labelled black. Since the block-greedy algorithm builds up overdetermined subsystems, we see more points in X_M than in Ξ_N after expansion. The process goes on until ill-conditioning is detected as in Algorithm 1. The resulting numerical solution uses 203 selected GA trial functions and has an error of 1.1733(-7). Figure 4.8(a) shows the resulting error profile. For comparison, running the system-free block-greedy algorithm with the MQ kernel in the same setting generates a final set of 1122 trial centers Ξ_N. Figure 4.8(b) shows the error profile of the resulting numerical solution using 495 selected MQ trial functions.

Although the system-free approach is very preliminary, this extra level of adaptivity will be beneficial to problems in complicated domains and higher dimensions [21, 25] by generating points wherever necessary and by further reducing the computational cost. □

Example 4.5. Our last example will illustrate how meshless collocation methods can benefit from the block-greedy algorithm. We solve the three-dimensional modified Helmholtz problem with unit wave number subject to Neumann boundary conditions in the Dupin cyclide defined by the implicit representation of

$$(x^2 + y^2 + z^2 + b^2 - d^2)^2 - 4(ax - cd)^2 - 4b^2y^2 = 0,$$

with $a = 2$, $b = 1.9$, $c = \sqrt{a^2 - b^2}$ and $d = 1$. We use `z*peaks(x,y)` as the exact solution to generate the Neumann data. Data points on the surface were simply generated by some parametric formulas $[x, y, z](\theta, \phi)$ with regular data in the (θ, ϕ)-space; that is, $[x, y, z]$ is non-uniform on the surface. Using the GA kernel and the standard setup procedures for the Kansa method, we get linear systems as in (1.2). The unknown coefficient $\boldsymbol{\eta}$ is solved by `mldivide` of MATLAB(without adaptivity) and by the block-greedy algorithm. In Figure 4.9, we show a sequence of error profiles with a selected number of trial functions, N. In each row of Figure 4.9, the colors of both surfaces are scaled to range from 0 to the maximum error of the non-adaptive numerical solution. The maximum error of the numerical solution obtained from a smaller trial subspace selected by the block-greedy algorithm is labelled in the color bar. Generally speaking, numerical approximations are more accurate with the block-greedy algorithm in place.

The convergence rate of the non-adaptive solutions is monotonous. Ignoring the non-uniformity of data locations, the errors reduce at the rate of $N^{-3.5}$. In contrast, the adaptive solutions converge in a staircase pattern. As N increases from 1318 to 4237, the errors reduce at the rate of $N^{-4.5}$. But the numbers of selected trial functions do not always go up with N. For $N = 6156$, the block-greedy algorithm selects fewer trial functions, but the resulting accuracy remains similar. Nonetheless, the convergence pattern resumes as the trial space enlarges; see $N = 9365$. What happens within this convergence test is not trivial. We believe that the non-uniform data distribution has some influence on the approximation powers of the trial spaces.

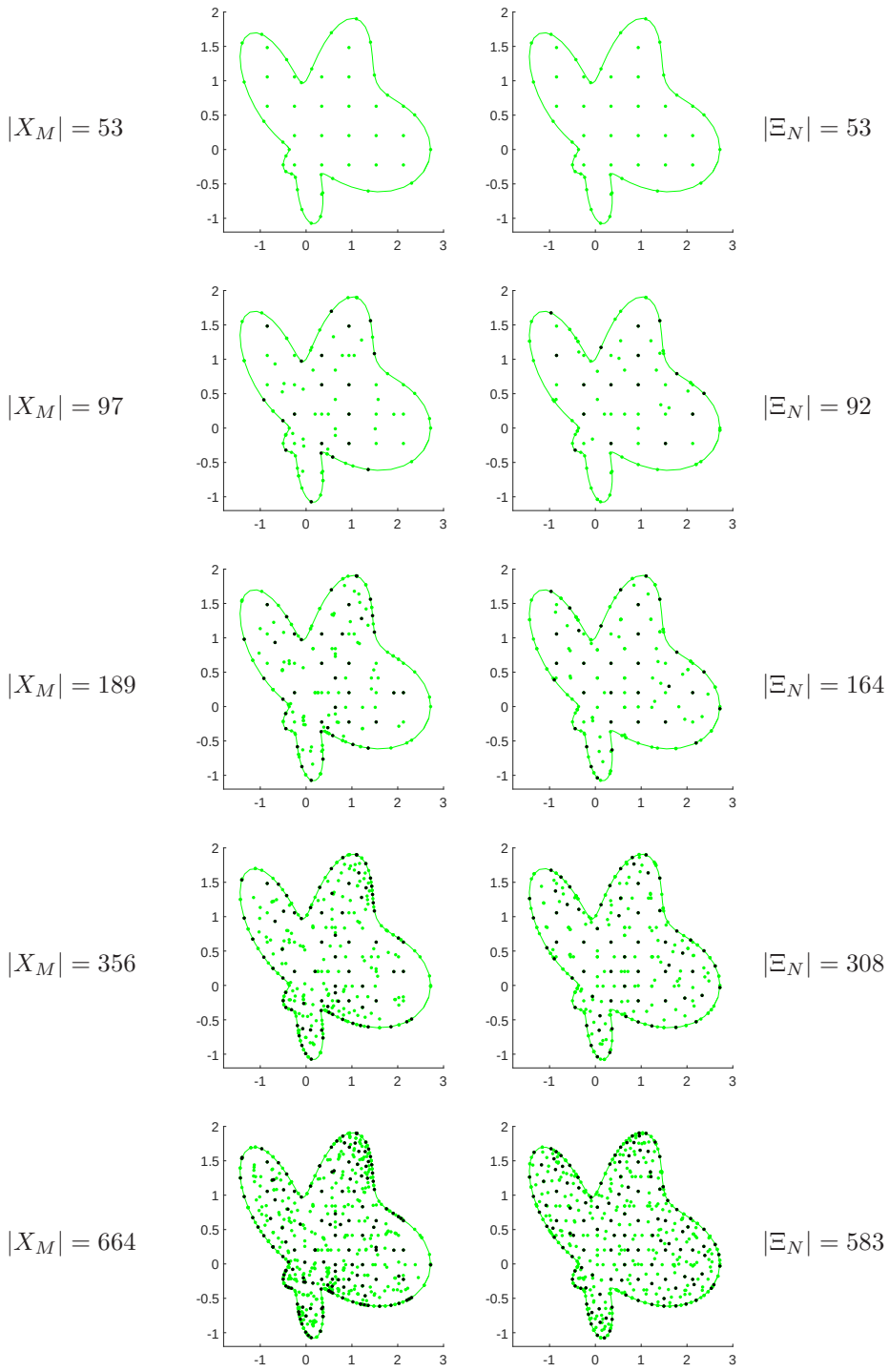

FIG. 4.7. *Example 4.4: Schematic demonstration of generating the sets of (left) collocation points and (right) trial centers on-the-fly.*

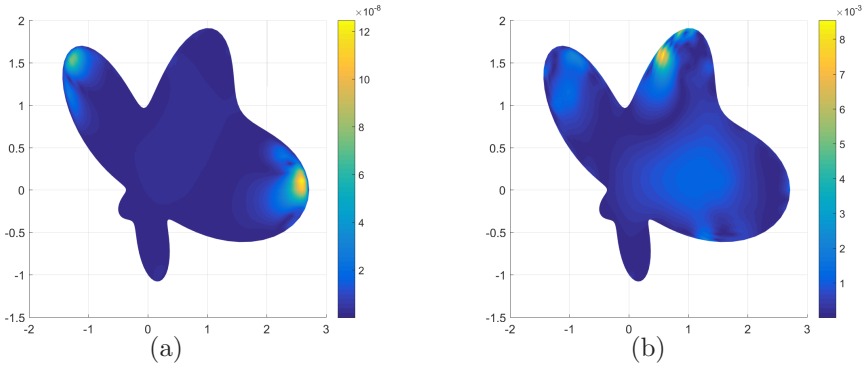

FIG. 4.8. *Example 4.4: Error profiles of numerical approximations obtained by a system-free Kansa method with (a) the unscaled GA kernels and (b) the unscaled MQ kernels.*

Note that it is more appropriate to measure the convergence of the adaptive solutions with K instead of N. Using the complexity count in (3.3), it costs $\mathcal{O}(NK^2)$ to select K out of N trial functions. So, the cost of selecting "1947 out of 6156" is indeed slightly lower than "3359 out of 4237." In terms of the complexity, the convergence of this example of an *adaptive*-Kansa method is monotonous. □

5. Conclusion. We proposed a block-greedy algorithm to select a large set of K columns out of any given $M \times N$ matrix, which is of full rank, but potentially ill-conditioned, so that the resulting $M \times K$ submatrix formed by the selected columns is well-conditioned. The main improvements over the previously proposed algorithms are the great reduction in computational cost and the consistent selection power. The complexity of the block-greedy algorithm is at most $\mathcal{O}(NK^2)$; see (3.2) and (3.3) for detailed estimates. The cost of adaptivity is now comparable to the costs of direct methods. When $N \gg M$, the cost of the proposed algorithm becomes $\mathcal{O}(M^3)$, making it an attractive alternative to the problem of basis selection out of a huge spanning set. The most expensive subroutines in the block-greedy algorithm are QR factorizations and matrix-matrix multiplications, both of which can be carried out in parallel. Thus, the block-greedy algorithm is already parallelizable to some extent, but an adequate version for large scale applications remains unexplored.

We have five numerical examples in this paper. Besides verifying the robustness and numerical performance of the block-greedy algorithm, a three-dimensional demonstration is included to show the benefit of employing the proposed algorithm with meshless collocation methods. It also ensures readers that the block-greedy algorithm works in regular domains as well as complicated ones in three dimensions. We hope that our numerical demonstrations can open new research discussions and possible applications. In particular, we show how the block-greedy algorithm can successfully deal with meshless collocation systems with random shape parameters and yield great improvements in accuracy. In addition, we demonstrate that another level of adaptivity can be added to create a system-free algorithm, in which linear systems are generated on-the-fly. Despite being presented separately, it is straightforward to use random shape parameters in a system-free algorithm. However, such an ad hoc combination does not yield further improvement in accuracy and, hence, is omitted from this paper. It is our future research topic to properly incorporate these concepts.

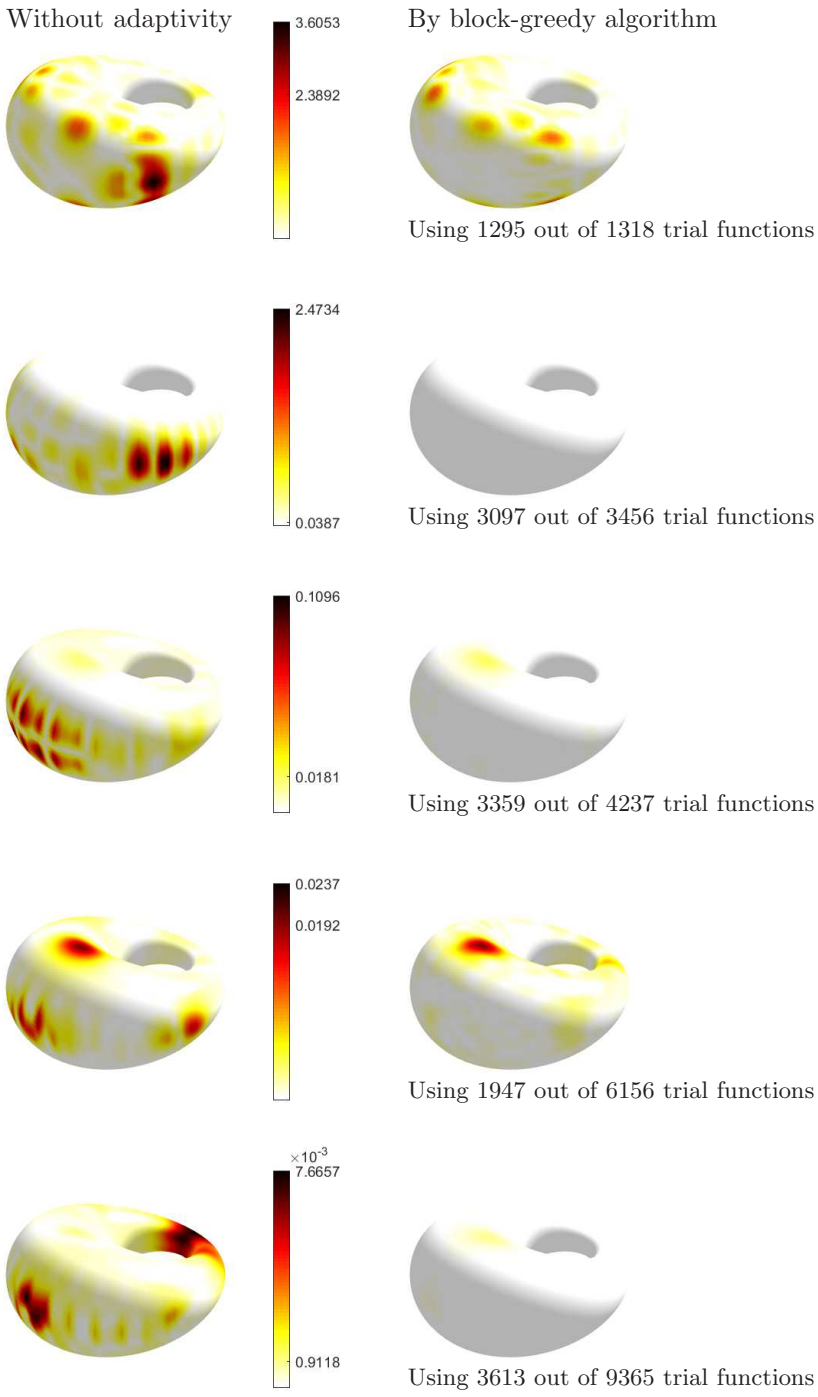

FIG. 4.9. *Example 4.5: Numerical errors in solving a modified Helmholtz problem with Neumann boundary with various numbers of GA trial function: (Left) Using all the trial functions and (Right) Using a subset selected by the block-greedy algorithm.*

Acknowledgements. This project was supported by a GRF Grant of the Hong Kong Research Grant Council and an FRG Grant of the Hong Kong Baptist University.

REFERENCES

[1] H. Brunner, L. Ling, and M. Yamamoto, *Numerical simulations of 2D fractional subdiffusion problems*, J. Comput. Phys. **229** (2010), no. 18, 6613–6622.
[2] G. E. Fasshauer and M. J. McCourt, *Stable evaluation of Gaussian radial basis function interpolants*, SIAM J. Sci. Comput. **34** (2012), no. 2, A737–A762.
[3] ———, *Kernel-based approximation methods using MATLAB*, Interdiscip. Math. Sci. 19. Hackensack, NJ: World Scientific., 2015.
[4] G. E. Fasshauer and J. G. Zhang, *On choosing "optimal" shape parameters for RBF approximation*, Numer. Algorithms **45** (2007), no. 1-4, 345–368.
[5] B. Fornberg, E. Larsson, and N. Flyer, *Stable computations with Gaussian radial basis functions in 2-D*, Tech. Report 2009-020, Department of Information Technology, Uppsala University, August 2009.
[6] ———, *Stable computations with Gaussian radial basis functions*, SIAM J. Sci. Comput. **33** (2011), no. 2, 869–892.
[7] B. Fornberg and C. Piret, *On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere.*, J. Comput. Phys. **227** (2008), no. 5, 2758–2780 (English).
[8] B. Fornberg and J. Zuev, *The Runge phenomenon and spatially variable shape parameters in RBF interpolation*, Comput. Math. Appl. **54** (2007), no. 3, 379–398.
[9] A. Golbabai, E. Mohebianfar, and H. Rabiei, *On the new variable shape parameter strategies for radial basis functions*, Compt. Appl. Math. **34** (2015), no. 2, 691–704.
[10] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[11] S. Hammarling and C. Lucas, *Updating the QR factorization and the least squares problem*, MIMS EPrint 2008.111, Manchester Institute for Mathematical Sciences, School of Mathematics, University of Manchester, Manchester, UK, 2008.
[12] N. J. Higham, *A survey of condition number estimation for triangular matrices*, SIAM Review **29** (1987), 575–596.
[13] ———, *Estimating the matrix p-norm*, Numer. Math **62** (1992), 511–538.
[14] Y. C. Hon and R. Schaback, *On unsymmetric collocation by radial basis functions*, Appl. Math. Comput. **119** (2001), no. 2-3, 177–186.
[15] Y. C. Hon, R. Schaback, and X. Zhou, *An adaptive greedy algorithm for solving large RBF collocation problems*, Numer. Algorithms **32** (2003), no. 1, 13–25.
[16] Y. P. Hong and C.-T. Pan, *A lower bound for the smallest singular value*, Linear Algebra Appl. **172** (1992), 27–32.
[17] R. A. Horn and C. R. Johnson (eds.), *Matrix analysis*, Cambridge University Press, New York, NY, USA, 1986.
[18] C.-S. Huang, H.-D. Yen, and A. H.-D. Cheng, *On the increasingly flat radial basis function and optimal shape parameter for the solution of elliptic PDEs*, Eng. Anal. Bound. Elem. **34** (2010), no. 9, 802–809.
[19] E. J. Kansa, *Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates*, Comput. Math. Appl. **19** (1990), no. 8-9, 127–145.
[20] E. J. Kansa and R. E. Carlson, *Improved accuracy of multiquadric interpolation using variable shape parameters*, Compt. Math. Appl. **24** (1992), no. 12, 99–120.
[21] E. J. Kansa and J. Geiser, *Numerical solution to time-dependent 4D inviscid Burgers' equations*, Eng. Anal. Bound. Elem. **37** (2013), no. 3, 637–645.
[22] D. E. Knuth, *Semi-optimal bases for linear dependencies*, Linear Multilinear Algebra **17** (1985), no. 1, 1–4.
[23] C. F. Lee, L. Ling, and R. Schaback, *On convergent numerical algorithms for unsymmetric collocation*, Adv. Comput. Math **30** (2009), no. 4, 339–354.
[24] J. Li, A. H.-D. Cheng, and C. S. Chen, *A comparison of efficiency and error convergence of multiquadric collocation method and finite element method*, Eng. Anal. Bound. Elem. **27** (2003), no. 3, 251–257.
[25] M. Li, W. Chen, and C. S. Chen, *The localized RBFs collocation methods for solving high dimensional PDEs*, Eng. Anal. Bound. Elem. **37** (2013), no. 10, 1300–1304.

[26] L. Ling, *An adaptive–hybrid meshfree approximation method*, Int. J. Numer. Methods Eng. **89** (2011), no. 5, 637–657.
[27] L. Ling, R. Opfer, and R. Schaback, *Results on meshless collocation techniques*, Eng. Anal. Bound. Elem. **30** (2006), no. 4, 247–253.
[28] L. Ling and R. Schaback, *Stable and convergent unsymmetric meshless collocation methods*, SIAM J. Numer. Anal. **46** (2008), no. 3, 1097–1115.
[29] ———, *An improved subspace selection algorithm for meshless collocation methods*, Int. J. Numer. Methods Eng. **80** (2009), no. 13, 1623–1639.
[30] C. Van Loan, *On estimating the condition of eigenvalues and eigenvectors*, Linear Algebra Appl. **88-89** (1987), no. 0, 715–732.
[31] R. B. Platte and T. A. Driscoll, *Computing eigenmodes of elliptic operators using radial basis functions*, Comput. Math. Appl. **48** (2004), no. 3–4, 561–576.
[32] C. M. C. Roque and A. J. M. Ferreira, *Numerical experiments on optimal shape parameters for radial basis functions*, Numer. Methods for PDEs **26** (2010), no. 3, 675–689.
[33] R. Schaback, *Adaptive numerical solution of MFS systems*, The method of fundamental solutions – A meshless method (C. S. Chen, A. Karageorghis, and Y. S. Smyrlis, eds.), Dynamic Publishers, 2008, pp. 1–27.
[34] J. D. Tebbens and M. Tuma, *On incremental condition estimators in the 2-norm*, SIAM Matrix Anal. Appl. **35** (2014), no. 1, 174–197.
[35] C. H. Tsai, J. Kolibal, and M. Li, *The golden section search algorithm for finding a good shape parameter for meshless collocation methods*, Eng. Anal. Bound. Elem. **34** (2010), no. 8, 738–746.
[36] L. Vrankar, E. J. Kansa, L. Ling, F. Runovc, and G. Turk, *Moving-boundary problems solved by adaptive radial basis functions.*, Comput. Fluids **39** (2010), no. 9, 1480–1490.
[37] L. Vrankar, N. A. Libre, L. Ling, G. Turk, and F. Runovc, *Solving moving-boundary problems with the wavelet adaptive radial basis functions method*, Comput. Fluids **86** (2013), no. 5, 37–44.
[38] J. Wertz, E. J. Kansa, and L. Ling, *The role of the multiquadric shape parameters in solving elliptic partial differential equations*, Comput. Math. Appl. **51** (2006), no. 8, 1335–1348.
[39] F. L. Yang, L. Ling, and T. Wei, *An adaptive greedy technique for inverse boundary determination problem*, J. Comput. Phys. **229** (2010), no. 22, 8484–8496.
[40] Y.-S. Yu and D.-H. Gu, *A note on a lower bound for the smallest singular value*, Linear Algebra Appl. **253** (1997), no. 1–3, 25–38.