Greedy Trial Subspace Selection in Meshfree Time-Stepping Scheme with Applications in Coupled Bulk-Surface Pattern Formations

Yichen Su[a,*], Leevan Ling[a]

[a] *Department of Mathematics, Hong Kong Baptist University, Hong Kong*

Abstract

Combining kernel-based collocation methods with time-stepping methods to solve parabolic partial differential equations can potentially introduce challenges in balancing temporal and spatial discretization errors. Typically, using kernels with high orders of smoothness on some sufficiently dense set of trial centers provides high spatial approximation accuracy that can exceed the accuracy of finite difference methods in time. The paper proposes a greedy approach for selecting trial subspaces in the kernel-based collocation method applied to time-stepping to balance errors in both well-conditioned and ill-conditioned scenarios. The approach involves selecting trial centers using a fast block-greedy algorithm with new stopping criteria that aim to balance temporal and spatial errors. Numerical simulations of coupled bulk-surface pattern formations, a system involving two functions in the domain and two on the boundary, illustrate the effectiveness of the proposed method in reducing trial space dimensions while maintaining accuracy.

Keywords: Parabolic PDEs, Block-greedy algorithm, Stopping criteria, Couple bulk-surface reaction-diffusion equations, Kernel-based collocation method, Radial basis functions
2000 MSC: 00A20, 00B10

1. Introduction

As kernel-based methods gain popularity, finding appropriate settings remains an open problem: how to stably solve the severely ill-conditioned linear systems that arise. Various strategies have been employed to address instability issues in computations. Regularization, as demonstrated in [1, 2], is effective in mitigating ill-conditioning problems. In this paper, we consider selecting a trial subspace to achieve more stable solution approximations. The Greedy algorithm was initially explored for symmetric kernel-based interpolation matrix systems by Schaback et al [3] and remain an active topic. Traditional methods for adaptive trial subspace selection include the P-greedy algorithm, which applies to symmetric PDEs. For the latest developments on power function-based P-greedy, see [4]. Additionally, [5] proposes the residual-based f-greedy algorithm and combinations of the P- and f-greedy algorithms to achieve higher convergence rates. The greedy algorithm has advantages for solving both time-dependent and time-independent problems. In the context of time-independent problems, it is directly employed to

*Corresponding author
Email addresses: 21481210@life.hkbu.edu.hk (Yichen Su), lling@hkbu.edu.hk (Leevan Ling)

choose trial subspaces for approximating solutions. Regarding time-dependent problems, we first apply time-stepping methods and subsequently apply the selected trial subspaces for solution approximation at each time step. Following the pioneering work on greedy methods [6] for solving time-independent partial differential equations (PDEs) using collocation methods, a.k.a. Kansa methods, we proposed various sequential-greedy algorithms [7, 8] to select quasi-optimal sets of trial subspaces that guarantee stable solutions. They built up nonsingular subsystems iteratively by selecting a row and then a column in a greedy fashion. Although each version brings improved accuracy, they suffer from high complexity. For instance, in a well-conditioned system, a greedy algorithm might select subsystems that yield unexpectedly high accuracy. For an $n \times n$ asymmetric collocation system, selecting k columns by any of these algorithms costs $O(k^4 + nk^2)$. The selection of multiple rows and columns results in large computations, clearly reflecting the trade-off between achieving higher accuracy and managing algorithm complexity. The block-greedy algorithm addresses the constraints of sequential algorithms [9] and can be implemented in a matrix-free manner, offering a reduced complexity of $O(nk^2)$. It can also be extended to a fully adaptive algorithm for boundary value problems [10] of elliptic type. It allows new data points to be added gradually, dynamically refining the trial subspace without relying on fixed data distribution and enhances the efficiency by solving least-squares problems.

A strategy ensures that fill distances of boundary/domain points are similar. Block-greedy to address ill-conditioning from small minimum separating distances. Specifically, during the subspace selection iterations of the block greedy algorithm, each iteration adds a set of new data points according to the primal/dual residual criterion outlined in [8]. This criterion helps avoid point clustering and maintains uniform fill distances among boundary and domain points, as detailed in [9] Theorem 2.1. As the selection proceeds, the algorithm monitors the condition number of the selected subsystem, and the greedy algorithm also checks whether the expanded submatrix is well-conditioned to allow further expansion. The stability of these kernel-based methods also heavily depends on the kernel's shape parameter. In this paper, a data-driven shape parameter generator fine-tunes this parameter through extensive testing to optimize system stability and numerical accuracy, enhancing the overall performance and reliability.

Sobolev kernels work well numerically, producing reasonable approximations with refinement for increased accuracy. This fully automatic meshfree algorithm benefits Kansa method users. While effective for elliptic PDEs, experimental evidence shows that using the greedy algorithm with time-stepping methods for solving parabolic PDEs is not as numerically stable as desired. We provide evidence with unstable numerical experiments in Example 3.1. One significant reason for the instability observed here is the mismatch in the accuracy of spatial and temporal discretizations of PDEs. To address this, we modified the original greedy algorithm by introducing different tolerance values for residuals and condition numbers. This approach efficiently handles issues arising from the accuracy mismatch problem. In Section 2, we overview the block-greedy algorithm. In Section 3, we explore how to properly fine-tune the algorithm's stopping criteria to fix the problem. In this paper, we first discretize PDEs in time and then in space, as a fixed time step is essential to ensure balanced convergence between time and spatial dimensions. We thoroughly studied the least-squares, kernel-based collocation method of lines in [11, 12]. These approaches have both theoretically and numerically demonstrated the importance of oversampling, which can also be achieved by using the block-greedy algorithm. But this will not be considered in this paper. Numerical experiments verify the effectiveness of the proposed algorithm, followed by conclusions.

2. Block-greedy algorithm for column subspace selections

From here on, *greedy algorithm* refers to the block-version in [9]. Although designed for kernel-based collocation methods, it can be viewed purely in terms of linear algebra. The following overview is not meant to be exhaustive but rather provides sufficient details for this paper. Consider the linear system $A\boldsymbol{\lambda} = \mathbf{b}$, where A is an $m \times n$ matrix with full rank m. The greedy algorithm is applied to the linear system and selects a column space of A that allows accurate and stable computations, ensuring that provides an effective approximation of the vector $\mathbf{b}$.

Suppose at a non-initial step, rows indexed by $\boldsymbol{m} \subset \mathbb{N}_m := \{1, 2, \ldots, m\}$ and columns indexed by $\boldsymbol{n} \subset \mathbb{N}_n$ have been selected. Let $A(\boldsymbol{m}, \boldsymbol{n})$ denote the corresponding submatrix. The goal is to expand the selections to $\boldsymbol{m}'$ and $\boldsymbol{n}'$ in batches rather than one by one, such that:

- $|\boldsymbol{m}'| \gtrsim 2|\boldsymbol{m}|$ while $|\boldsymbol{m}'| \leq m$, and
- $|\boldsymbol{n}'| = 2|\boldsymbol{n}|$ while $|\boldsymbol{n}'| < n$,

to build an overdetermined subsystem as it iterates. Here $|\cdot|$ represents the size of the row/ column set, and $|\boldsymbol{m}'| \gtrsim 2|\boldsymbol{m}|$ indicates that the number of rows in $\boldsymbol{m}'$ is greater than or approximately equal to two times of the number of rows in $\boldsymbol{m}$. When running out of unselected rows or columns, the greedy algorithm will take all m rows or n columns.

Instead of searching all unselected rows in $\mathbb{N}_m \setminus \boldsymbol{m}$ and columns in $\mathbb{N}_n \setminus \boldsymbol{n}$, the greedy algorithm shortlists candidates by sorting the residual magnitudes. To obtain the primal and dual residuals, we formulate a constrained minimization problem. The objective function aims to minimize the Euclidean norm of the solution vector $\boldsymbol{\eta}$, subject to the linear constraint $A\boldsymbol{\eta} = \mathbf{b}$. To solve this constrained optimization problem, we employ the method of Lagrange multipliers. The optimisation problem is transferred in the form of

$$\mathscr{L}(\boldsymbol{\eta}, \boldsymbol{\zeta}) = \frac{1}{2}\boldsymbol{\eta}^T\boldsymbol{\eta} + \boldsymbol{\zeta}(A\boldsymbol{\eta} - \mathbf{b}), \tag{1}$$

where $\boldsymbol{\zeta}$ denotes the Lagrange multiplier. We take the partial derivatives of the Lagrangian function $\mathscr{L}(\boldsymbol{\eta}, \boldsymbol{\zeta})$ with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\zeta}$ to derive the primal-dual subsystem. The primal-dual subsystem is defined in the form of

$$\begin{aligned} A(\boldsymbol{m}, \boldsymbol{n})\boldsymbol{\eta} &= \mathbf{b}(\boldsymbol{m}), & (2)\\ A(\boldsymbol{m}, \boldsymbol{n})^T\boldsymbol{\zeta} &= -\boldsymbol{\eta}. & (3) \end{aligned}$$

Then the primal and dual residuals are defined by

$$\mathbf{r}_{\text{primal}} := A(\mathbb{N}_m, \boldsymbol{n})\boldsymbol{\eta} - \mathbf{b} \in \mathbb{R}^m, \quad \text{and} \quad \mathbf{r}_{\text{dual}} := \mathscr{E}_{\mathbb{R}^{|\boldsymbol{n}|}\to\mathbb{R}^n}\boldsymbol{\eta} + A(\boldsymbol{m}, \mathbb{N}_n)^T\boldsymbol{\zeta} \in \mathbb{R}^n, \tag{4}$$

where $\mathscr{E}_{\mathbb{R}^{|\boldsymbol{n}|}\to\mathbb{R}^n}\boldsymbol{\eta} \in \mathbb{R}^n$ is the extension of $\boldsymbol{\eta} \in \mathbb{R}^{|\boldsymbol{n}|}$ with zeros patched into entries not in $\boldsymbol{n}$. Then, the greedy algorithm identifies and selects candidate rows and columns from those that remain unselected, choosing those associated with the largest magnitudes of $\mathbf{r}_{\text{primal}}$ and $\mathbf{r}_{\text{dual}}$ in the primal and dual systems, respectively. The final selection is made from shortlisted candidates based on these smaller submatrices.

The greedy algorithm is matrix-free because, as seen in (4), it only needs to store entries of selected rows and columns, i.e., $A(\boldsymbol{m}, \mathbb{N}_n)$ and $A(\mathbb{N}_m, \boldsymbol{n})$. The computational cost of the greedy algorithm is determined by the size of the shortlisted candidate sets in each iteration. We refer

readers to [9] for details on the complexity analysis. As mentioned in the introduction, the greedy algorithm achieves $O(n^2\mathscr{k})$ complexity, where $\mathscr{k}$ denotes the number of total selected columns.

The greedy algorithm can be initialized with pre-selected rows/columns, i.e., input some $\boldsymbol{m}$ and $\boldsymbol{n}$. If none is specified, it begins with rows/columns associated with the largest primal/dual residuals.

After each expansion that (roughly) doubles the number of selected rows/columns, the stopping criteria of the greedy algorithm use two tolerance values, τ_r and τ_κ. Besides stopping by a small enough residual, the greedy algorithm will check the condition number of $A(\boldsymbol{m}', \boldsymbol{n}')$ to determine whether it is well-conditioned for further expansion; otherwise, the expansion terminates because of the ill-conditioning. In particular, the greedy algorithm stops when

- **[SC-**1**]** the condition number of the expanded system $\kappa(A(\boldsymbol{m}', \boldsymbol{n}')) > 1/\tau_\kappa$ exceeds the tolerance, or
- **[SC-**2**]** the norm of the primal residual $\|\mathbf{r}_{\text{primal}}\|_\infty < \tau_r$ is below the tolerance.

It is cheap to compute the residual and check the condition number since the algorithm updates [13] QR factorizations of $A(\boldsymbol{m}, \boldsymbol{n})$ to $A(\boldsymbol{m}', \boldsymbol{n}')$ in each iteration. Recall that in the block-style greedy algorithm, the number of selected columns doubles at each iteration. When we stop due to SC-1, a backtracking process is employed to prevent including too many columns, which would lead to a condition number much larger than what the user tolerance allows. This backtracking process uses the bisection method to select the first $\mathscr{k}$ indices in $\boldsymbol{n}'$ to generate the largest subset $\boldsymbol{n}'' \subset \boldsymbol{n}'$ so that the final column subspace selection yields a condition number just below tolerance, i.e., $\kappa(A(\boldsymbol{m}', \boldsymbol{n}'')) \leqslant 1/\tau_\kappa$. In the case of SC-2, the greedy algorithm returns $\boldsymbol{n}'$ as its final selection. In the original article, a single tolerance $\tau_r = \tau_\kappa = \varepsilon_{mech}$, the machine epsilon, was used as the default value, which works well in various time-independent elliptic PDEs. To clearly demonstrate the column subspace selection process in the greedy algorithm, the brief framework is outlined in Algorithm 1.

Algorithm 1 Column subspace selection algorithm

Inputs: $A(\mathbb{N}_m, \mathbb{N}_n) \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ τ_κ, τ_r
Initialize:
$\boldsymbol{m} \leftarrow$ row index with the maximum primal residual
$\boldsymbol{n} \leftarrow$ column index with the maximum dual residual
while $|\boldsymbol{n}| < n$ **do**
    Solve primal-dual subsystems (2) and (3) by QR factorization of $A(\boldsymbol{m}, \boldsymbol{n})$
    Calculate $\mathbf{r}_{\text{primal}}$ and $\mathbf{r}_{\text{dual}}$
    Break if $\mathbf{r}_{\text{primal}} < \tau_r$ ▷ **[SC-2]**
    Select rows $\boldsymbol{m}'$ and columns $\boldsymbol{n}'$ in batches, $\boldsymbol{m} \leftarrow \boldsymbol{m}'$, $\boldsymbol{n} \leftarrow \boldsymbol{n}'$
    Estimate the condition number $\kappa(A(\boldsymbol{m}', \boldsymbol{n}'))$
    if $\kappa(A(\boldsymbol{m}, \boldsymbol{n})) > \tau_\kappa$ **then** ▷ **[SC-1]**
        Bisection method to select the first $\mathscr{k}$ indices in $\boldsymbol{n}'$ to generate $\boldsymbol{n}''$
        $\boldsymbol{n} \leftarrow \boldsymbol{n}''$
        Break
    end if
end while

3. Controlling spatial accuracy in time-stepping methods

Before applying the block-greedy algorithm to solve the resultant matrix systems arising from kernel-based collocation and time-stepping methods, we first walk through the main steps of this discretization procedure. Consider second-order parabolic partial differential equations defined in a bounded domain $\Omega \subset \mathbb{R}^d$ with smooth boundary $\partial\Omega$. The equations are complemented by initial and boundary conditions that we assume are compatible.

As a demonstrative example, the heat equation with Dirichlet boundary conditions is defined in the form:

$$\begin{array}{rlcl} \dfrac{\partial u}{\partial t} = D\nabla^2 u + f & \text{in} & \Omega & \times \ [0,T] \\ u = g & \text{on} & \partial\Omega & \times \ [0,T] \\ u = u_0 & \text{on} & \Omega & \times \ \{0\} \end{array} \tag{5}$$

with constant diffusion $D > 0$ and source function f. While this simple example illustrates the approach, the proposed method applies more generally to quasi-linear parabolic PDEs of the above form with variable coefficients.

We define a partition $\{t_k\}_{k=0}^K$ of [0,T] with time steps $\triangle t_k = t_k - t_{k-1}$. We denote the approximate solution at time t_k by $U^k \approx u(\cdot, t_k)$ for $k = 0, 1, \ldots, K$. The commonly used Crank-Nicolson scheme semi-discretizes the PDE (5) in time and results in a sequence of elliptic PDEs of the form

$$\frac{U^k - U^{k-1}}{\Delta t_k} = \frac{1}{2}(D\nabla^2 U^k + f^k + D\nabla^2 U^{k-1} + f^{k-1}) \quad \text{for } k = 1, \ldots, K, \tag{6}$$

where $f^k := f(\cdot, t_k)$. This paper also employs semi-implicit backward differentiation formulas (SBDF) in [14], which use explicit schemes for reaction terms and implicit schemes for diffusion terms. The system generated by the first-order SBDF (SBDF1) is

$$\frac{U^k - U^{k-1}}{\triangle t_k} = f^{k-1} + D\nabla^2 U^k, \quad \text{for } k = 1, \ldots, K, \tag{7}$$

and that generated by the second-order SBDF (SBDF2) is

$$\frac{1}{2\triangle t_k}\Big(3U^k - 4U^{k-1} + U^{k-2}\Big) = 2f^{k-1} - f^{k-2} + D\nabla^2 U^k, \quad \text{for } k = 2, \ldots, K, \tag{8}$$

with U^1 obtained by (7).

We solve the semi-discretized systems (6), (7) or (8) using a kernel-based collocation method. We choose a radius basis kernel Φ on $\mathbb{R}^d$ giving rise to a positive definite kernel with a smoothness order greater than 2. Common examples include the Gaussian and multiquadrics kernels. Reproducing kernels of Sobolev spaces, like the Whittle-Matern-Sobolev (MS) kernel and compactly supported Wendland functions, are also suitable choices that come with convergence theory for elliptic PDEs [15, 16, 17]. The theoretical approximation powers of employing these kernels on greedy points were addressed in [18].

Given a set $\Xi := \{\boldsymbol{\xi}^j\}_{j=1}^n \subset \Omega$, we define the finite-dimensional trial space

$$\mathcal{U} := \text{span}\Big\{\Phi(\cdot - \boldsymbol{\xi}) \,\Big|\, \boldsymbol{\xi} \in \Xi\Big\}. \tag{9}$$

We seek approximation to the solution U^k within $\mathcal{U}$ in the form of

$$\sum_{j=1}^{n} \lambda_j^k \Phi(\cdot - \boldsymbol{\xi}^j), \qquad \text{for } k = 1, \ldots, K, \tag{10}$$

where $\boldsymbol{\lambda}^k = \{\lambda_j^k\}_{j=1}^n$ are the unknown coefficients. We determine $\boldsymbol{\lambda}^k$ by imposing collocation at points $Z = Z_\Omega \cup Z_{\partial\Omega} = \{\mathbf{z}^i\}_{i=1}^m \subset \Omega \cup \partial\Omega$ for the governing equation and boundary conditions respectively. Taking the CN scheme (6), we obtain an $m \times n$ matrix system:

$$\begin{pmatrix} (2\Phi - \triangle t_k[\nabla^2\Phi])(Z \cap \Omega, \Xi) \\ \Phi(Z \cap \partial\Omega, \Xi) \end{pmatrix} = \begin{pmatrix} (2U^{k-1} + \triangle t(f^k + D\nabla^2 U^{k-1} + f^{k-1}))(Z \cap \Omega) \\ g^k(Z \cap \partial\Omega) \end{pmatrix}, \tag{11}$$

where $g^k := g(\cdot, t_k)$ and the data dependent matrix with entries

$$[\Phi(Z_\Omega, \Xi)]_{i,j} = \Phi(\boldsymbol{z}_i - \boldsymbol{\xi}_j) \qquad \text{for } \boldsymbol{z}_i \in Z_\Omega \text{ and } \boldsymbol{\xi}_i \in \Xi.$$

The other two matrices $\Phi(Z_{\partial\Omega}, \Xi)$ and $[\nabla^2\Phi](Z_\Omega, \Xi)$ are defined similarly. The right-hand side vector is known; given $\boldsymbol{\lambda}^{k-1}$, we can evaluate the approximate solution U^{k-1} and its Laplacian $\nabla^2 U^{k-1}$ using (10). In linear algebra notation, we can express the fully-discretized systems (11) as

$$A\boldsymbol{\lambda}^k = \mathbf{b}(\boldsymbol{\lambda}^{k-1}), \quad \text{for } k = 1, \ldots, K, \tag{12}$$

with $\boldsymbol{\lambda}^0$ or $\mathbf{b}(\boldsymbol{\lambda}^0)$ determined from the initial condition. Using a greedy algorithm to select a column subspace of A is equivalent to selecting a subset of trial centers from Ξ. In the following numerical experiments, we implement the greedy algorithm based on the linear system only at the initial time step, which makes sense for parabolic problems with slowly varying solutions. Running the greedy algorithm at a later time is certainly a possibility for solutions that exhibit more variation over time. However, doing so efficiently requires a reliable error indicator, which is beyond the scope of this work.

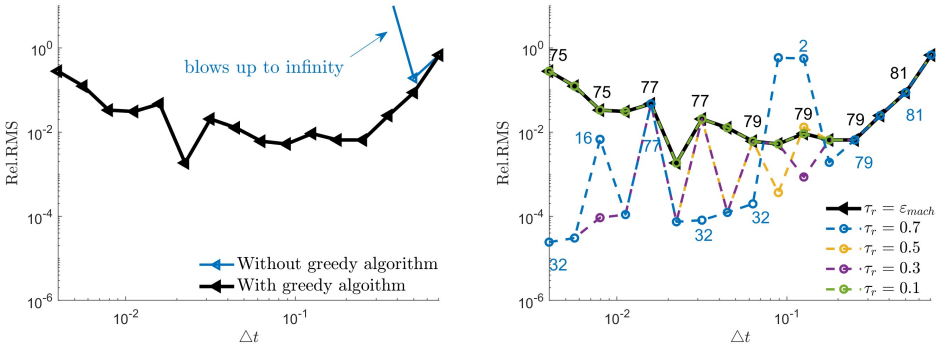
Example 3.1 (Greedy Algorithm in action)

We compare the performance of the meshfree time-stepping method with and without the greedy algorithm applied. Our comparison is based on a linear heat equation (5) with $D = 1$ and $\Omega = [0, 1]^2$. The right-hand functions f and g were computed from the exact solution expressed as:

$$u^*([x_1, x_2]^T, t) = \sin(2\pi x_1)\sin(\pi x_2)\exp(-2\pi^2 t) + (1 - \exp(-2\pi^2 t))/2,$$

which exhibits the standard decay property of heat equations. The Crank-Nicolson scheme was used for temporal discretization, and a fixed time step of $\triangle t$ was used for all k. For spatial discretization, we employed the Halton sequence to generate a set of $n = 300$ data points, which were used as both the trial centers and collocation points. This exactly determined setup will become an overdetermined system if the greedy algorithm selects a proper subset of columns. The kernel used in this example is the (unscaled) Gaussian kernel, which results in ill-conditioned matrices in the fully discretised system. The relative root mean square error of solutions obtained by the meshfree time-stepping method with various $\triangle t$ is shown in Figure 1.

The power of the greedy method is illustrated in Figure 1a. The solution obtained without the greedy algorithm is unstable and diverges for most tested cases. By applying the greedy algorithm (with default parameter values) to the matrix system $A\boldsymbol{\lambda}^1 = \mathbf{b}(\boldsymbol{\lambda}^0)$ for the first time step, we obtained a subset of columns of A indexed by $\boldsymbol{n}'$. For all runs, SC-1 stopped the algorithm with large condition numbers around $1/\varepsilon_{mech}$. The resulting submatrix $A(\mathbb{N}_m, \boldsymbol{n}')$ was



(a) PDE error with/without the greedy algorithm

(b) Using different tolerances in SC-2

Figure 1: For Example 3.1, the relative root mean error profiles of solving a heat equation by meshfree time-stepping method with various $\triangle t$ (1a) without and with the greedy algorithm in default settings; (1b) with greedy algorithm using different tolerance in stopping criteria. Colored numbers in (1b) are the number of selected columns in A by the greedy algorithm.

then used in the sequence of the fully-discretized system to update solutions in time. This process regularizes the solution but does not fully prevent divergence, and it fails to achieve second-order convergence. We labeled the number of selected columns by the greedy algorithm in Figure 1b with black numbers and noticed that they do not show significant differences. Regarding the CPU time for the greedy algorithm under different time step settings, note that all tested scenarios concluded after a small number of iterations. Consequently, the CPU times were relatively low[1] and provide limited value for comparative analysis. Reported runtime in MATLAB is obtained from an Intel-i7 processor.

While in the greedy algorithm, we observe that SC-2 uses a default tolerance $\tau_r = \varepsilon_{mech}$ for the residual, which is too small to effectively stop the greedy iteration. This forces the selection of extra basis functions, inducing unnecessary ill-conditioning and instability. Therefore, choosing appropriate stopping criteria for the greedy algorithm is important to prevent these issues. As a test, we ran the greedy algorithm with various tolerance values $0.1 \leq \tau_r \leq 0.7$ to produce the error profiles in Figure 1b. Compared to the default (black solid line), some runs with large τ_r terminate earlier (as expected) by selecting fewer columns (numbers in the figure). Using fewer columns resulted in 2-3 orders of magnitude accuracy improvement. However, we noted that these settings' errors lack stability; some runs with only 2 columns are inaccurate. If $\tau_r = 0.1$ is too small to stop the iteration with SC-2, then SC-1 will stop the algorithm, yielding identical errors as the default.

These observations indicate the greedy algorithm can stabilize solutions compared to methods without it. However, choosing appropriate stopping criteria is critical to achieving the desired accuracy. ■

The kernel-based collocation method using greedy algorithms could efficiently solve semi-discretized PDEs. However, when spatial and temporal discretization accuracies are mismatched, the resulting fully discrete systems may become ill-conditioned, impacting solution accuracy.

[1]The CPU time of the greedy algorithm used in each point-selecting procedure is approximately 0.20 seconds.

As demonstrated in Example 3.1, using a smaller τ_r results in more selected columns to achieve higher spatial accuracy. When the order of spatial discretization significantly surpasses that of temporal discretization, a substantial mismatch in the scales of numerical representation between the two arises. This mismatch can lead to numerical instability.

In the rest of this section, we develop two stopping strategies for the greedy algorithm to improve condition number and accuracy when solving fully-discrete systems. We propose to:

- **[SC-1′]** run a residual search within the newly added columns and output the trial sub-subspace that minimizes the least-squares residual, when the greedy algorithm stops due to a large condition number $\kappa(A(\boldsymbol{m}', \boldsymbol{n}')) > 1/\tau_\kappa$ with τ_κ, and
- **[SC-2′]** apply a backtracking process to reduce selected column set accuracy down to a target residual τ_r' when the greedy algorithm stops due to a small residual $\|\mathbf{r}_{\text{primal}}\|_\infty < \tau_r$.

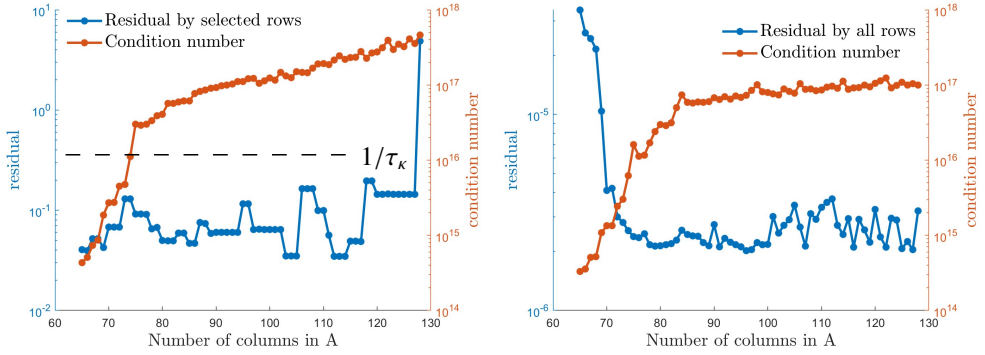
These techniques enable the greedy algorithm to balance accuracy, stability, and efficiency based on the conditioning properties of the resulting discrete systems. We implement new stopping criteria by modifying lines 8 to 16 in algorithm 1:

Algorithm 2 Column subspace selection algorithm with new stopping criteria

1: **Inputs:** $A(\mathbb{N}_m, \mathbb{N}_n) \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ τ_κ, τ_r
2: **Initialize:**
3: $\boldsymbol{m} \leftarrow$ row index with the maximum primal residual
4: $\boldsymbol{n} \leftarrow$ column index with the maximum dual residual
5: **while** $|\boldsymbol{n}| < n$ **do**
6: Solve primal-dual subsystems (2) and (3) by QR factorization of $A(\boldsymbol{m}, \boldsymbol{n})$
7: Calculate $\mathbf{r}_{\text{primal}}$ and $\mathbf{r}_{\text{dual}}$
8: **if** $\mathbf{r}_{\text{primal}} < \tau_r$ **then** ▷ **[SC-2′]**
9: Run backtracking process, $\boldsymbol{n}' \leftarrow$ columns set that makes the residual closest to
10: the target residual τ_r', break
11: **end if**
12: Select rows $\boldsymbol{m}'$ and columns $\boldsymbol{n}'$ in batches, $\boldsymbol{m} \leftarrow \boldsymbol{m}'$, $\boldsymbol{n} \leftarrow \boldsymbol{n}'$
13: Estimate the condition number $\kappa(A(\boldsymbol{m}', \boldsymbol{n}'))$
14: **if** $\kappa(A(\boldsymbol{m}, \boldsymbol{n})) > \tau_\kappa$ **then** ▷ **[SC-1′]**
15: Run residual search, $\boldsymbol{n}'' \leftarrow$ columns set corresponding to the residual minimum
16: value
17: $\boldsymbol{n} \leftarrow \boldsymbol{n}'$
18: Break
19: **end if**
20: **end while**

3.1. [SC-1′] Stop by large condition number

Figure 2a schematically illustrates the behavior of the residuals and the condition numbers of the subsystem as selected columns are incrementally added. This progression continues until the subsystem reaches ill-conditioning, at which point the original greedy method is stopped by SC-1. The red curve shows the condition numbers of submatrices of A with rows $\boldsymbol{m}'$ against expanding columns from the index set $\boldsymbol{n}$ in the previous iteration to that of the current iteration



(a) Using selected rows in computation

(b) Using all rows in computation

Figure 2: Greedy algorithm stops by large condition number, 128 columns are selected in the column space – A schematic demonstration of residual value and condition numbers of submatrices of A with expending columns $\boldsymbol{n} \to \boldsymbol{n}'$ and (a) selected rows $\boldsymbol{m}$ and (b) all rows $\mathbb{N}_m$ in the column space.

$\boldsymbol{n}'$, denoted by $\boldsymbol{n} \to \boldsymbol{n}'$. Its left endpoint satisfies $\kappa(A(\boldsymbol{m}, \boldsymbol{n})) < 1/\tau_\kappa$ and so the iteration continues; whereas its right endpoint satisfies $\kappa(A(\boldsymbol{m}', \boldsymbol{n}')) > 1/\tau_\kappa$ and so the greedy algorithm stops. In the original SC-1, a backtracking process traces back along the same curve. It seeks the point with $\kappa(A(\boldsymbol{m}, \boldsymbol{n})) < 1/\tau_\kappa$ to identify the final number of column selection $\hat{k}$, as discussed in Section 2. The blue curve shows the ℓ^∞-norm residual vectors of all rows for least-squares subproblems similar to (2) with the selected rows in $\boldsymbol{m}$ and expanding $\boldsymbol{n} \to \boldsymbol{n}'$ columns in A.

We remark that the observed large residual is not the true approximation of the trial subspace. Large error is expected at the remaining rows in $\mathbb{N}_m \setminus \boldsymbol{m}'$ whose information was left out from the least-squares subproblems. The greedy method is used for columns (or trial functions) selection and PDEs were solved with all rows (or collocation points).

For the new stopping criterion SC-1′, we stop the greedy iteration using the same mechanism:

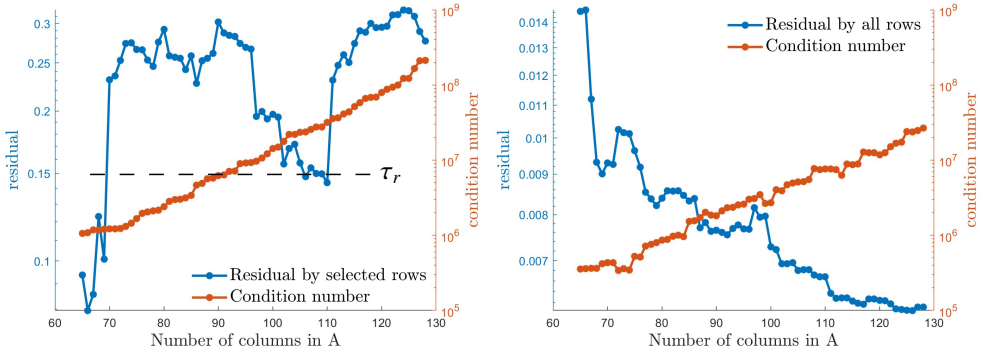
$$\kappa(A(\boldsymbol{m}, \boldsymbol{n})) < 1/\tau_\kappa \quad \textit{and} \quad \kappa(A(\boldsymbol{m}', \boldsymbol{n}')) > 1/\tau_\kappa.$$

For small τ_κ, we are dealing with ill-conditioned subsystems and should take a stability-first approach. The goal of the greedy algorithm is then to select column subspaces that are computationally stable.

Figure 2b corresponds to Figure 2a but uses all rows in the submatrices. Note the different y-axis ranges in these figures. First, using more rows has a minimal effect on the condition numbers and the red curves are similar in both figures. The blue curve shows the ℓ^∞-norm error of using all rows and expanding the $\boldsymbol{n}$ to $\boldsymbol{n}'$ columns in A to least-squares approximate $\mathbf{b}$, i.e.,

$$A(\mathbb{N}_m, \boldsymbol{n} \to \boldsymbol{n}')\boldsymbol{\lambda} = \mathbf{b}.$$

This residual curve shows the true approximation power of trial subspaces. We also observe that the trial subspace that yields the smallest approximation error in approximating $\mathbf{b}$ is not easy to detect from the curve of the condition number. We propose searching this residual curve instead and having the greedy algorithm return $\hat{k}$ columns at which the residual is minimum and the corresponding column index set $\boldsymbol{n}''$.



(a) Using selected rows in computation

(b) Using all rows in computation

Figure 3: Greedy algorithm stops by small residual, 128 columns are selected in the column space –A schematic demonstration of residual value and condition numbers of submatrices of A with expending columns $\boldsymbol{n} \to \boldsymbol{n}'$ and (a) selected rows $\boldsymbol{m}$, and (b) all rows $\mathbb{N}_m$ in the column space.

The computational overhead is the difference in cost of QR factoring the greedy selected $|\boldsymbol{m}'| \times |\boldsymbol{n}''|$ submatrix and the $|\boldsymbol{m}'| \times |\boldsymbol{n}'|$ that allows us to compute the whole residual curve in Figure 2b. The estimate is $O(|\boldsymbol{m}'| \times (|\boldsymbol{n}'|^2 - |\boldsymbol{n}''|^2))$.

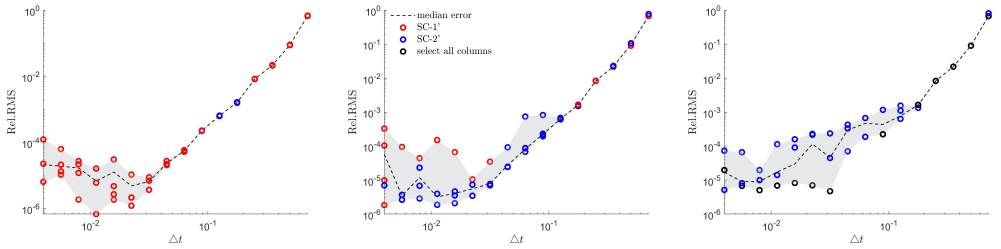
3.2. [SC-2′] Stop by small residual

This is an aspect the original greedy algorithm largely ignores. The greedy algorithm only runs on residual values using the selected rows and columns, see equations (2)–(3). Setting $\tau_r = \varepsilon_{mech}$ means the greedy algorithm almost never stops because of this small residual criteria. Even in the case when the discretized matrix system is well-conditioned, the greedy algorithm in the default setting will select the whole column space (but without a huge overhead in the block version) to yield the best possible accuracy. Figure 3 were the counterpart of Figure 2 but the greedy algorithm was stopped by a small residual for some given (large) tolerance τ_r. In Figure 3, the condition numbers are all well below the dangerous zone $1/\varepsilon_{mech}$. In Figure 3a, we once again observed that the residual values obtained using the selected rows are large and do not reflect the true approximation power of the trial subspace. The residual using all rows in Figure 3b is not a quantity that exists within the greedy algorithm. Yet it could be way smaller than the tolerance value τ_r.

Suppose the greedy algorithm selected column index set $\boldsymbol{n}'$ after stopping by small residual (based on selected rows indexed by $\boldsymbol{m}'$), $\|\mathbf{r}_{\text{primal},\boldsymbol{m}'}\|_\infty < \tau_r$. The original greedy algorithm does not have any postprocessing. In the new SC-2′, we propose to run the stopping criteria using residual from all rows when

$$\|\mathbf{r}_{\text{primal},\mathbb{N}_m}\|_\infty < \tau_r.$$

That is, we use the blue curve in Figure 3b instead of that in Figure 3a to stop the greedy iteration. Then we apply a backtracking process, similar to the one presented in Section 2, to select the first k indices in $\boldsymbol{n}'$ to generate the largest subset $\boldsymbol{n}'' \subset \boldsymbol{n}'$ so that the final column subspace selection yields a residual for the $A(\mathbb{N}_m, \boldsymbol{n}'')$ subsystem is just below another tolerance τ_r'. If the residual using all rows and $\boldsymbol{n}'$ is above τ_r', we simply set $\boldsymbol{n}'' = \boldsymbol{n}'$ and return all selected columns. Like in SC-1′, the QR factorization of $A(\mathbb{N}_m, \boldsymbol{n}')$ allows us to evaluate any data point



(a) $\epsilon = 1$ (b) $\epsilon = 3$ (c) $\epsilon = 6$

Figure 4: Example 3.2, error profiles for solving a 2D heat equation using a meshfree time-stepping method with greedy algorithm and the MS kernel with $\mu = 6$. Different values of ε and $\triangle t$ were used, and the shaded areas in each plot show the error range for $n \in [500, 1000]$ data points, with the median as a dashed line. The stopping criteria that terminate the greedy algorithm are shown as black circles for all columns being selected, red circles for SC-1′, and blue circles for SC-2′.

on the residual curve. Since we double the number of selected columns in each (block-version) greedy iteration, the overhead of SC-2′ is the extra QR factorization to intermediate submatrix $A(\mathbb{N}_m, 1 \to \boldsymbol{n})$ in the previous iteration that costs $O(m\boldsymbol{k}^2)$.

In the context of meshfree time-stepping schemes, a highly accurate spatial scheme is wasteful in terms of computational cost when the overall error is dominated by temporal discretization. It makes sense to greedily select just enough spatial approximation power to match the temporal error in order to reduce spatial accuracy and compensate for computational efficiency. The tolerance τ_r' should be based on the finite difference scheme and time step $\triangle t$ used in temporal discretization, which yields the semi-discrete systems. For the second-order scheme in this paper, we propose:

$$\tau_\kappa = O(\varepsilon_{mech}/\triangle t), \quad \tau_r = O(\triangle t), \quad \text{and} \quad \tau_r' = O(\triangle t^2). \tag{13}$$

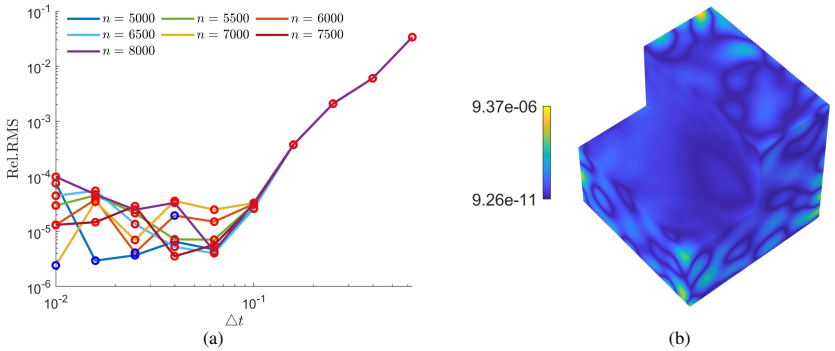
In the rest of this paper, we use 1 as the Big-Oh constant, yielding satisfactory results in 2D and 3D test problems.

Example 3.2 (New stopping criteria in action) We utilize SC-1′ and SC-2′ into the greedy algorithm and solve the heat equation in Example 3.1 again. For temporal discretization, we still use the CN scheme. For spatial discretization, we use the MS kernel in dimension $d = 2$ with smoothness order $\mu = 6$

$$\Phi_\mu(x - y) := \|x - y\|_2^{\mu - d/2} \mathcal{K}_{\mu - d/2}(\|x - y\|_2) \qquad \text{for } x, y \in \mathbb{R}^2 \tag{14}$$

and scale input argument of this translation-invariant kernel by $\Phi(r) \leftarrow \Phi(\epsilon r)$ with shape parameters $\epsilon = 1$, 3, and 6. We take $n = 500, 550, \ldots, 1000$ Halton points to form the sets of trial centers and collocation points.

Instead of one curve per tested n, we show the range of relative root mean errors in Figure (4) along with the median of all tested n against different $\triangle t$. Note that the $\triangle t$ value determines all greedy tolerance by (13). The red circle for SC-1′ and blue circle for SC-2′ in the figures indicate the active stopping criteria that terminate the greedy algorithm in that particular run. If the greedy algorithm selects all columns without activating any stopping criteria (usually when n is small and the system matrix A is well-conditioned), it is labelled by black circles in the figure.



(a) (b)

Figure 5: Example 3.2, (a) error profiles for the solution of a 3D heat equation using a meshfree time-stepping method with the new greedy stopping criteria for different numbers of data points n. (b) error function for $n = 7000$, $\triangle t = 0.01$.

Our experimental results show that the new stopping criteria generate more stable solutions with a wide range of meshfree setups, as evidenced by the fluctuation range of the error in Figure 4. In the case of $\epsilon = 1$, the MS kernel quickly leads to an ill-conditioned matrix in the fully-discretized system as n increases. We observe in Figure 4a that most runs were stopped SC-1′ by large conditioned numbers. With everything else fixed, increasing ϵ yields a system matrix with a smaller condition number. We choose to show the results for $\epsilon = 3$ since it clearly demonstrates errors of similar magnitude when greedy is terminated by either stopping criteria. The errors using all columns are generally smaller since SC-2′ is designed for the sake of computational cost. Selecting all columns yields the highest possible approximation power in the full discretization and the lowest error, though at a higher computational cost.

We set up a similar test problem in 3D with the following heat solution:

$$u^*([x_1, x_2, x_3]^T, t) = \sin(2\pi x_1)\sin(\pi x_2)\sin(\pi x_3)\exp(-2\pi^2 t) + (1 - \exp(-2\pi^2 t))/2.$$

Using the MS kernel (14) with $d = 3$ and $\epsilon = 3$ centered at $n = 5000$ to 8000 Halton points in $[0, 1]^3$ as trial centers and collocation points, we solve the heat equation (as in the 2D case) with various time step $\triangle t$ (hence, various tolerance for the greedy algorithm). The resulting relative root mean squared errors are shown in Figure 5a. Similar to Figure 4b, we see that the blue circle for SC-2′ is more likely to yield smaller errors for any $\triangle t$. Figure 5b shows the error function for $n = 7000$ and $\triangle t = 0.01$ with the greedy algorithm.

■ The previous examples have provided motivation and insights into the effectiveness of the new stopping criteria across different setups. However, a more thorough investigation through extensive numerical experiments is needed to verify the robustness of our proposed stopping criteria and tolerance selection strategies.

4. Coupled bulk-surface pattern formations

Semi-linear coupled bulk-surface reaction-diffusion equations are crucial in modeling phenomena where processes at the surface significantly influence or are influenced by the dynamics occurring in the bulk. Such models find applications in various fields including biology, chemistry, materials science, and environmental engineering. These equations consist of reaction-diffusion terms in the bulk and on the surface, with coupling typically occurring through bound-

ary conditions linking the fluxes or concentrations between the bulk and the surface. For example, in developmental biology, it is crucial to understand how polarized states arise and are maintained, characterized by uneven distributions of chemical substances such as proteins and lipids [19, 20]. Another application of the bulk-surface reaction-diffusion equation is in developing solid-state hydrogen storage materials using metal hydrides, where slow kinetics of hydrogen uptake and release involve adsorption and dissociation of hydrogen, its diffusion through the solid, and a phase transition in the metal-hydride from lower to higher hydrogen content [21]. This section explores the application of these equations in developmental biology, specifically in the area of pattern formation.

Let $\Omega \subset \mathbb{R}^d$ be some bounded (bulk) domain and $\mathcal{S} = \partial\Omega$ be its smooth boundary with well-defined normal vector field $\boldsymbol{n} : \mathcal{S} \to \mathbb{R}^d$. We verify the new greedy stopping criteria with the following semi-linear coupled bulk-surface reaction-diffusion equations in both 2D and 3D for pattern formation:

$$\begin{cases} \dfrac{\partial u}{\partial t} = D_u \nabla^2 u + f_1(u, v), \\ \dfrac{\partial v}{\partial t} = D_v \nabla^2 v + f_2(u, v), \end{cases} \quad \text{in } \Omega \times [0, T]$$
$$\begin{cases} \dfrac{\partial w}{\partial t} = D_w \nabla_{\mathcal{S}}^2 w + f_1(w, s) - h_1(u, w), \\ \dfrac{\partial s}{\partial t} = D_s \nabla_{\mathcal{S}}^2 s + f_2(w, s) - h_2(v, s), \end{cases} \quad \text{on } \mathcal{S} \times [0, T] \tag{15}$$

with coupling boundary conditions

$$\begin{cases} D_u \nabla u \cdot \boldsymbol{n} = h_1(u, w), \\ D_v \nabla v \cdot \boldsymbol{n} = h_2(v, w). \end{cases} \quad \text{on } \mathcal{S} \times [0, T]. \tag{16}$$

The surface equations in (15) contain differential operators on surfaces, that can be defined via a projection to the tangent space of $\mathcal{S}$

$$P := I_d - \boldsymbol{n}\boldsymbol{n}^T,$$

where I_d denotes the $d \times d$ identity matrix. The surface gradient and the Laplace-Beltrami operator, a.k.a. surface Laplacian, are defined respectively by

$$\nabla_{\mathcal{S}} = P\nabla \quad \text{and} \quad \nabla_{\mathcal{S}}^2 = \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}} = (P\nabla) \cdot (P\nabla),$$

using the standard gradient operator ∇ for functions defined in $\mathbb{R}^d$.

The source functions in (15)–(16) are given by the nonlinear reaction kinetics and the coupling of the internal bulk dynamics in Ω to the surface dynamics on $\mathcal{S}$. The interaction between unknown functions $u, v : \Omega \to \mathbb{R}$ and $w, s : \mathcal{S} \to \mathbb{R}$ yields Turing patterns in the bulk and on the surface. The nonlinear reaction kinetics are:

$$f_1(u, v) = \gamma(a - u + u^2 v), \qquad f_2(u, v) = \gamma(b - u^2 v).$$

The coupling of the internal dynamics to the surface dynamics is:

$$h_1(u, w) = \alpha_1 w - \beta_1 u, \qquad h_2(v, s) = \alpha_2 s - \beta_2 v,$$

Table 1: Parameters for Turing's pattern formation used in coupled bulk-surface reaction-diffusion equations (15)–(16).

	a	b	α_1	α_2	β_1	β_2	D_v	D_s	q	γ
Fig. 6–9	⋮	⋮	⋮	⋮	⋮	⋮	2	2	1/12	30
Fig. 9							1	1	1/12	30
Fig. 10	1/10	9/10	5/12	5	5/12	5	5	5	1/10	500
Fig. 11–12							3	3	1/12	40
Fig. 13	⋮	⋮	⋮	⋮	⋮	⋮	6	6	1/12	30
Fig. 14							3	3	1/12	30

with parameter values listed in Table 1 with $D_u = qD_v$, $D_w = qD_s$. As shown in [22], the coupled bulk-surface PDEs (15) has a unique homogeneous equilibrium state when

$$f_1(w, s) - h_1(u, w) = 0 \quad \text{and} \quad f_2(w, s) - h_2(v, s) = 0. \tag{17}$$

For any γ, solving (17) yields corresponding (constant) function values at equilibrium as

$$\Big(a + b,\ \frac{b}{(a+b)^2},\ a + b,\ \frac{b}{(a+b)^2}\Big) = (u_0,\ v_0,\ w_0,\ s_0). \tag{18}$$

We set the initial conditions $(u_0,\ v_0,\ w_0,\ s_0)$ for the coupled bulk surface PDEs (15) to be the homogeneous equilibrium state with a, b as shown in Table 1. Small approximation errors in these initial conditions then serve as sources of perturbations. According to Turing instability theory [23], in the absence of diffusion a uniform steady state remains stable. However, it becomes unstable to small spatial perturbations. In our meshfree context, minor disturbances from numerical approximation error can cause an otherwise stable homogeneous equilibrium state to become unstable, resulting in the formation of patterns.

4.1. Discretizing PDEs for the greedy algorithm

We use the SBDF2 scheme in equation (8) for temporal discretization, where all nonlinear reaction terms in equation (15) were evaluated at the previous time step. Let U^k, V^k, W^k, S^k represent the semi-discretized solutions as in Section 3. We also use an explicit scheme for bulk-surface dynamics. The semi-discretized equation for the bulk function u is given by:

$$\frac{1}{2\triangle t_k}\Big(3U^k - 4U^{k-1} + U^{k-2}\Big) = 2f_1(U^{k-1}, V^{k-1}) - f_1(U^{k-2}, V^{k-2}) + D_u\nabla^2 U^k \quad \text{in } \Omega, \tag{19}$$

subject to the boundary condition

$$D_u\nabla U^k \cdot \boldsymbol{n} = h_1(U^{k-1}, W^{k-1}) \quad \text{on } \mathcal{S}, \tag{20}$$

which is decoupled from the other solutions. The semi-discretized equation for the surface function w is

$$\begin{aligned}\frac{1}{2\triangle t_k}\Big(3W^k - 4W^{k-1} + W^{k-2}\Big) = 2f_1(W^{k-1}, S^{k-1}) - 2h_1(W^{k-1}, S^{k-1})\\ -f_1(W^{k-2}, V^{k-2}) + h_1(W^{k-2}, V^{k-2}) + D_w\nabla_{\mathcal{S}}^2 W^k \quad \text{on } \mathcal{S}.\end{aligned} \tag{21}$$

We obtain the semi-discretized systems for V^k and S^k similarly to the equation for U^k and W^k. In the end, this yields four sequences of linear elliptic PDEs that incorporate solution history only through the right-hand source functions.

For spatial discretization, we construct two trial spaces to approximate the solutions in the bulk and on the surface, respectively, using the MS kernel in (14) that reproduces the Sobolev space $H^\mu(\mathbb{R}^d)$. Let $\mu_\Omega \geq \max(2, d/2)$ be the smoothness order used in the bulk kernel Φ_{μ_Ω}. When we restrict the kernel $\Phi_{\mu_S+1/2}$ on the surface $\mathcal{S}$ for any $\mu_\mathcal{S} \geq \max(2, (d-1)/2)$, the restricted kernel $\Psi_{\mu_S} := \Phi_{\mu_S+1/2}\big|\mathcal{S} : \mathcal{S} \times \mathcal{S}$ is a Sobolev space $H^{\mu_S-1/2}(\mathcal{S})$ reproducing kernel with smoothness $\mu_\mathcal{S}$, as shown in [24]. Based on convergence estimates for meshfree least-squares collocation for elliptic PDEs in the bulk [17] and on the surface [15], it is suggested [25] to choose the smoothness orders of the kernels such that

$$\begin{cases} \text{[SO-1]} & \mu_\mathcal{S} \geqslant \mu_\Omega + d/2 - 2, \quad \mu_\Omega \geq (9+d)/2, \\ \text{[SO-2]} & \mu_\Omega \geqslant \mu_\mathcal{S} - d/2, \quad \mu_\mathcal{S} \geq 3 + d, \end{cases} \qquad \text{or} \tag{22}$$

in order to balance error estimates in the bulk and on the surface. Condition SO-1 arises from the error estimate for meshfree least-square collocation methods solving elliptic PDEs in the bulk and the error of surface functions treated as perturbations. A second condition results from reversing the roles of the bulk and surface functions within the same framework.

To simplify notations, we use two sets of data points $\Xi_\Omega \subset \overline{\Omega}$ and $\Xi_\mathcal{S} \subset \mathcal{S}$ for the bulk and surface functions, respectively, i.e.,

$$U^k, V^k \in \mathcal{U}_{\Xi_\Omega,\Omega} := \operatorname{span}\{\Phi_{\mu_\Omega}(\cdot, \boldsymbol{\xi}) \,|\, \boldsymbol{\xi} \in \Xi_\Omega\},$$

and

$$W^k, S^k \in \mathcal{U}_{\Xi_\mathcal{S},\mathcal{S}} := \operatorname{span}\{\Psi_{\mu_\mathcal{S}}(\cdot, \boldsymbol{\xi}) \,|\, \boldsymbol{\xi} \in \Xi_\mathcal{S}\} = \operatorname{span}\{\Phi_{\mu_\mathcal{S}+1/2}(\cdot, \boldsymbol{\xi}) \,|\, \boldsymbol{\xi} \in \Xi_\mathcal{S}\}.$$

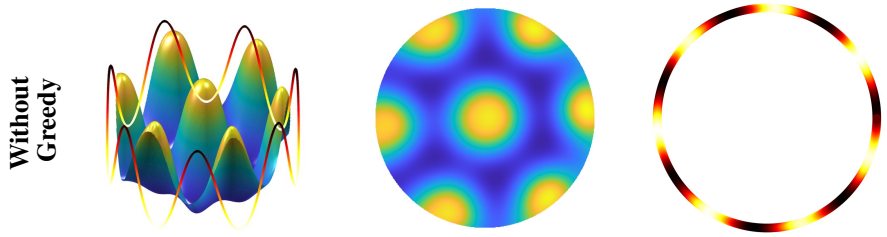
For the upcoming 2D and 3D simulations, we simply use $\mu_\Omega = 6$ and $\mu_\mathcal{S} = 5.5$ which satisfy the first condition SO-1 in (22). Consequently, we will use the MS kernel Φ_6 in constructing both the bulk and surface collocation systems.

We use sets of collocation points $Z_\Omega \subset \Omega \cup \mathcal{S}$ and $Z_\mathcal{S} \subset \mathcal{S}$ for bulk and surface functions respectively to set up the meshfree collocation matrix system, as discussed in Section 3. Specifically, for bulk functions U^k, V^k, we can obtain matrix equations similar to (11) except with different constant factors and a new boundary part in the matrix, namely $\Phi(Z_\Omega \cap \mathcal{S}, \Xi_\Omega) \leftarrow [\nabla\Phi_{\mu_\Omega} \cdot \boldsymbol{n}](Z_\Omega \cap \mathcal{S}, \Xi_\Omega)$.

We assume prior analytical knowledge of the normal vector field $\boldsymbol{n}$ and use the convergent Kansa-type analytical projection method in [15] to approximate the surface Laplacian operator. This extrinsic approach allows collocations directly on surfaces and can fully discretize the surface equations. The elliptic PDE in (21) for semi-discretized surface function W^k leads to the following meshfree collocation matrix when fully-discretized:

$$A_w(\triangle t_k) = \Big[(3\Phi_{\mu_\mathcal{S}+1/2} - 2\triangle t_k [D_w \nabla^2_\mathcal{S} \Phi_{\mu_\mathcal{S}+1/2}])(Z_\mathcal{S}, \Xi_\mathcal{S})\Big],$$

depending on the step size. The meshfree collocation matrix for V^k looks almost identical but with a different diffusion coefficient D_v. If we use an equal time step $\triangle t$, then all four meshfree collocation matrices (denoted by A_U, A_V, A_W, and A_S) remain fixed for all time, which is a cost-effective approach that we will take in the following calculations. Now we have a sequence of matrix equations in the form of (12) for each unknown functions sequence U^k, V^k, W^k and S^k for $k = 1, \ldots, K$.



(a) $n = 717 + 100$, $\triangle t = 0.005$ (b) Bulk solution u (c) Surface solution w

Figure 6: Example 4.1: Spot pattern formation by solving a 2D coupled bulk-surface reaction-diffusion equation, using the meshfree time stepping method, but without using the greedy algorithm. The meshfree method uses $n = n_\Omega + n_\mathcal{S} = 717 + 100$ data points and a time step of $\triangle t = 0.005$. (a) Bulk and surface solutions together are shown in 3D, (b) pattern formed in the bulk solution u, and (c) pattern formed in the surface solution w.

Because our initial conditions (18) correspond to the homogeneous equilibrium state, analytically applying SBDF1 at the first time step yields constant functions U^1, V^1, W^1, S^1. This implies that, in the second time step, the right-hand side vector of the SBDF2 matrix system is a constant vector. In this case, we can run the greedy algorithm to select columns for each of the four unknown functions without needing to spend time computing the actual right-hand side vector. It is worth noting that the original greedy algorithm also runs with an all-one right-hand vector as part of the patch test procedure in the absence of a specified input vector. Next, we obtain solutions in the second time step by solving the SBDF2 matrix system with all rows but only the greedily selected columns. This is where the perturbation comes into play. Solving the reduced system (or the full system in cases without the greedy algorithm) introduces small errors that serve as perturbations, allowing patterns to form according to the Turing instability mechanism described earlier.

In summary, here is the main numerical setup described above: We use the MS Φ_6 kernel in (14) with smoothness order $\mu_\Omega = 6$ and $\mu_\mathcal{S} = 5.5$. The tolerances of the greedy algorithm are set according to (13) with a Big-Oh constant of 1. All examples use an exactly determined setup with identical sets of trial centers $\Xi_\cdot$ and collocation points $Z_\cdot = \Xi_\cdot$ in bulk domain and surface, respectively. The number of data points $n = |\Xi|$ varies for different examples due to different domain volumes and will be reported individually.

Example 4.1 (Robustness verifications in 2D) Let Ω be the 2D unit ball. We solve the coupled bulk-surface pattern formation partial differential equations (15) in bulk Ω and surface $\mathcal{S} = \partial\Omega$ subject to the boundary condition (16) and initial condition (18). Under this simple setting, even in the absence of an analytic solution, we can use symmetry to gain insight into the interfacial patterns and identify pattern defects in the numerical solutions. Such interfacial patterns would not form in isolated bulk or surface systems; patterns in the bulk and on the surface are expected to be highly correlated.

Using Turing's pattern formation parameters in [26] for spots pattern formation, as shown in Table 1, we solved the 2D coupled bulk-surface reaction-diffusion equations to study pattern formation, using a meshfree time stepping method. Firstly, without the greedy algorithm, Figure 6 shows the meshfree solutions u in the bulk and w on the surface obtained by employing $n = n_\Omega + n_\mathcal{S} = 717 + 100$ data points and a time step of $\triangle t = 0.005$. Here n_Ω and $n_\mathcal{S}$ denote the number of basis functions in Ω and on $\mathcal{S}$. To ensure that the solutions converge to steady states, we solve the reaction-diffusion until a sufficiently long time, say, $T = 1000$. Figure 6a shows the bulk and surface solutions together in 3D, revealing their complex spatial interactions.

Table 2: The number of selected basis functions and the stopping criteria corresponding to the solutions corresponding greedy cases in Figure 7.

	$n = 717 + 100$					$n = 1031 + 120$				
	$n'_\Omega(717)$	$n'_\mathcal{S}(100)$	SC(Ω)	SC($\mathcal{S}$)	t_{CPU}	$n'_\Omega(1031)$	$n'_\mathcal{S}(120)$	SC(Ω)	SC($\mathcal{S}$)	t_{CPU}
$\triangle t = 0.01$	341	23	SC-2′	SC-2′	7.27	186	39	SC-2′	SC-2′	5.74
$\triangle t = 0.005$	451	23	SC-2′	SC-2′	5.90	465	24	SC-2′	SC-2′	5.58

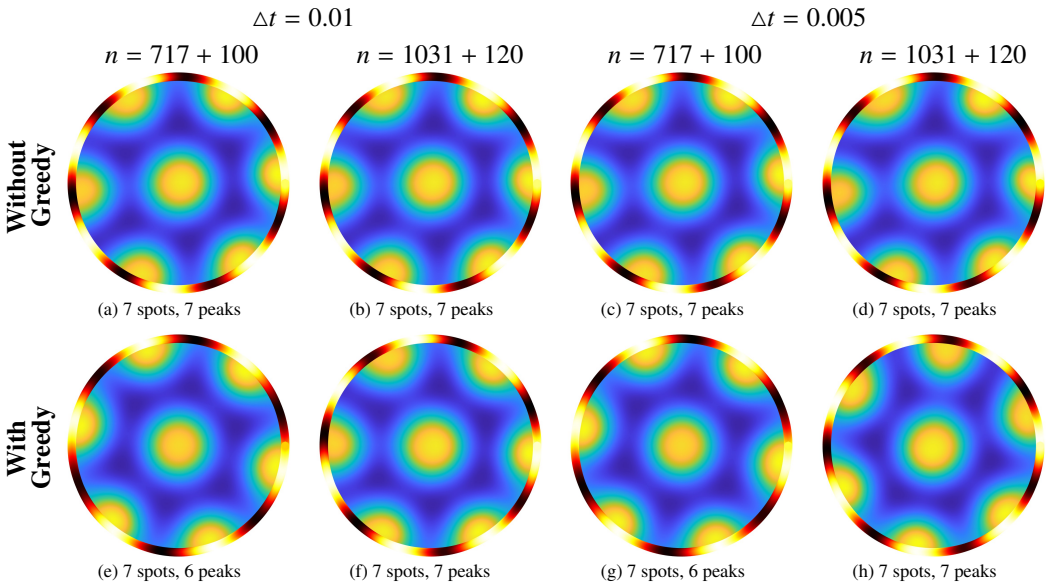
Figures 6b and 6c show bird's eye views of the bulk solution u and surface solution w, respectively. The bulk solution exhibits a spotted pattern while the surface solution develops a wavelike pattern around the circular domain.

We implement the greedy algorithm to determine if the patterns can be generated with fewer basis functions. To ensure numerical convergence, we vary the time steps and the number of data points. Figure 7 displays overlaid bulk and surface solutions of $\triangle t = 0.01$, $\triangle t = 0.005$, and $n = 717 + 100$, $n = 1031 + 120$, with and without the greedy algorithm. Upon initial inspection, all four meshfree parameter sets, irrespective of whether the greedy algorithm was used, generate similar bulk patterns, characterized by 7 spots in the bulk and either 6 or 7 peaks on the surface. We also examine the interfacial patterns. In Figure 8, we plot all eight (non-unique) surface patterns w, shifted so that the first peak coincides. The numerical solutions for the surface exhibit less consistency than those in the bulk. Notably, the greedy surface solutions with small n, say $n = 717+100$ exhibit a distinct characteristic from the other solutions, displaying only a six-peak wave-like pattern. We plot these special solutions in red, while all other solutions are plotted in blue. Figure 7e and 7g correspond to these solutions showing that the bulk and surface solutions are coupled together. We recorded the CPU times used to select the trial space for all variables using the greedy algorithm in Table 2. To reduce random variability, we counted the average CPU time of executing the greedy algorithm ten times and represented it by t_{CPU} (seconds).

In Table 2, we present the stopping criteria and the number of basis functions selected for the greedy solutions corresponding to Figure 7 and Figure 8, with varying values of n_Ω and $n_\mathcal{S}$. For both functions u and w, fewer basis functions are selected to produce consistent patterns. Using smaller $\triangle t$ results in more basis functions being selected. We remark that all kernels use $\epsilon = 6$ and run to $T = 1000$. Figures 6–7 is kept at the same range for the sake of fair comparison of spot size.

We present additional experiments that demonstrate how the greedy algorithm can rectify unstable numerical results that arise due to poor numerical settings. Figure 9 shows the solutions obtained with the greedy algorithm under different parameter settings. Specifically, building upon the scenarios depicted in Figure 7, we take more data points wherein n was set to $1612 + 150$ with $\triangle t$ being 0.01 and 0.005 in Figure 9a-9b. Additionally, in Figure 9c-9d, we varied the diffusion coefficients from $D_v = D_s = 2$ to $D_v = D_s = 1$ with $\triangle t = 0.005$, $n = 717 + 100$, and $n = 1031 + 120$. While there remains some uncertainty regarding whether the greedy algorithm accurately captures the distribution of the patterns, particularly in cases where $D_v = D_s = 1$, it is worth noting that the appearance of the patterns is indicative of the success of the greedy algorithm in comparison to the blow-up solution without the greedy algorithm under these settings.

Before tackling 3D problems, we also attempt to solve the numerically more challenging stripe pattern formation problem. We do this by simply changing the parameters for Turing's pattern formation, as shown in Table 1. Since generating stripe patterns typically requires a higher level of spatial and temporal resolution than spot patterns due to their more complex and dynamic



(a) 7 spots, 7 peaks (b) 7 spots, 7 peaks (c) 7 spots, 7 peaks (d) 7 spots, 7 peaks

(e) 7 spots, 6 peaks (f) 7 spots, 7 peaks (g) 7 spots, 6 peaks (h) 7 spots, 7 peaks

Figure 7: Example 4.1: Bulk and surface solutions obtained by solving the 2D coupled bulk-surface reaction-diffusion equation with and without the greedy algorithm. The meshfree method uses $n = 717 + 100$, $n = 1031 + 120$ data points and $\triangle t = 0.005$, $\triangle t = 0.01$ time steps. The number of bases used and stopping criteria in the greedy cases are in Table 2.

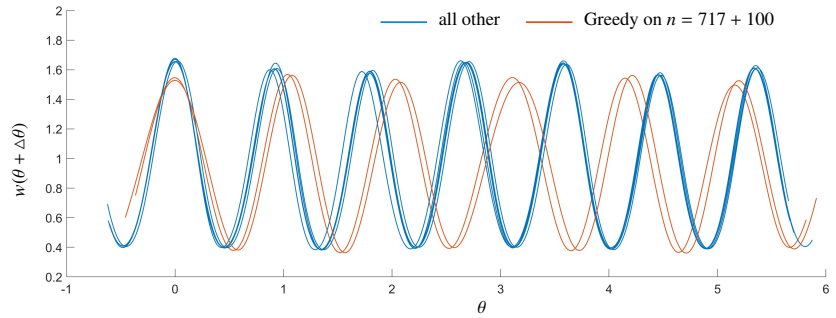
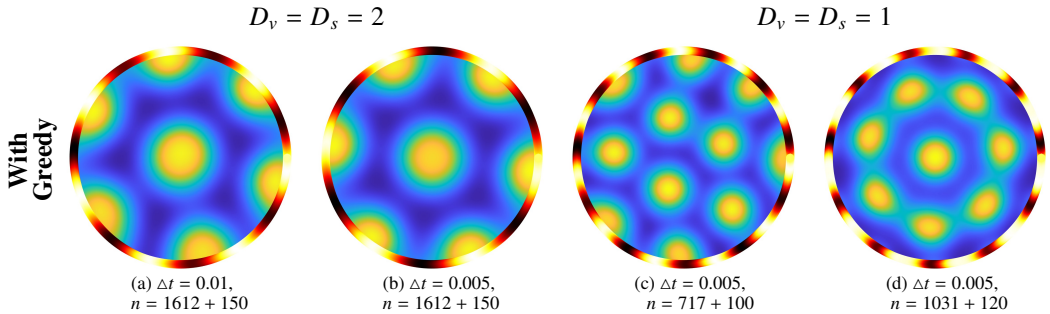


Figure 8: Example 4.1: The surface solutions from Figure 7 are individually shifted (with different $\triangle t$) to align the location of the first peak and plotted against θ.



(a) $\triangle t = 0.01$, $n = 1612 + 150$ (b) $\triangle t = 0.005$, $n = 1612 + 150$ (c) $\triangle t = 0.005$, $n = 717 + 100$ (d) $\triangle t = 0.005$, $n = 1031 + 120$

Figure 9: Example 4.1: Corresponding to the greedy cases in Figure 7, bulk and surface solutions by solving 2D coupled bulk-surface reaction-diffusion equation using $D_v = D_s = 2$ and $D_v = D_s = 1$.

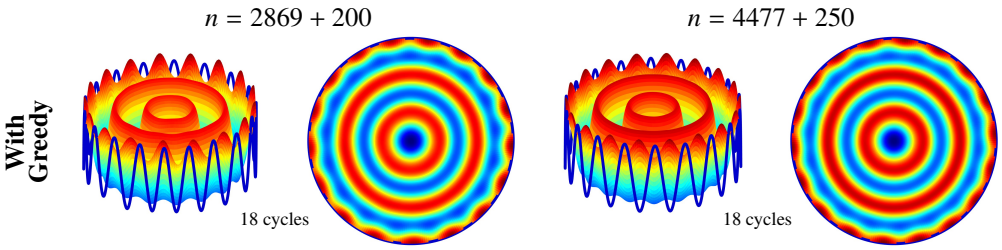


Figure 10: Example 4.1: Stripe pattern formation by solving a 2D coupled bulk-surface reaction-diffusion equation, using the meshfree time stepping method and the greedy algorithm. The meshfree method uses $n = 2869 + 200$ and $4477 + 250$ data points and a time step of $\triangle t = 0.001$. For each test case, the bulk and surface patterns formed are shown together in 3D view and bird's eye view.

nature, we take a smaller time step as $\triangle t = 0.001$ and more data points as $n = 2869 + 200$ and $n = 4477 + 250$ in this case. The run time here is still taken as $T = 1000$. Initially, we attempted to solve for stripe patterns under these settings without utilizing the greedy algorithm. However, as we had anticipated, the solution experienced a blow-up in the first few iterations. Then we employ the greedy algorithm, which selects fewer bases to obtain reasonable stripe patterns. Figure 10 showcases both solutions in the bulk and on the surface, which successfully capture the formation of stripe patterns in the bulk domain and exhibit very high-frequency wavelike surface solutions. The amplitudes of surface solutions appear uniform and consistent. In both patterns, the bulk solution at the boundary and the surface solution demonstrate high-frequency oscillations and are perfectly synchronized[2]

Example 4.2 (Coupled Bulk-Surface Pattern formations in 3D)

Moving forward with our simulation efforts, we now explore 3D coupled bulk-surface pattern formation. Our aim is to test whether the greedy algorithm can produce patterns with relatively fewer basis functions in the cases when the solution is less oscillatory and smoother, i.e., spot formation. Compared to the 2D domain, generating patterns in 3D requires more accurate discretization due to the higher dimensionality. To simplify our approach, we focus on larger spot patterns in 3D domains instead of small patterns or slim stripes as seen in 2D cases.

We implement the greedy algorithm approach as before without modification. Our initial experiments involve a torus with a simple geometrical structure, and we provide sufficient numerical solutions to verify the robustness of our approach. Subsequently, we will extend our experimentation to other 3D shapes, namely Dupin's cyclide and ellipsoid, to further validate our findings.

Observation 1: The greedy algorithm recovers from unsuccessful runs

As in the 2D demonstration, we obtain bulk and surface solutions by solving the three-dimensional coupled bulk-surface reaction-diffusion equation was achieved by applying the meshfree time stepping method both with and without the greedy algorithm in Figure 11. We still use the MS Φ_6 kernel in 3D experiments. For the meshfree method, the value of n was set to $2644 + 1430$, ϵ was fixed at 1 and the time step $\triangle t$ was allocated as 0.005 until the final time $T = 1000$. We observe that in the absence of the greedy algorithm, faint and irregular pat-

[2]Similar to the peak-to-trough matching seen in cases of 7 spots in the bulk in Figure 7, there is a similar matching in these stripe patterns.

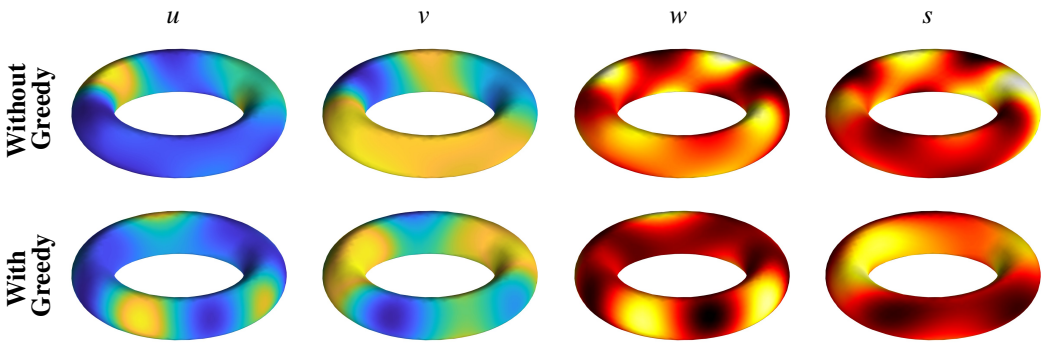


Figure 11: Example 4.2: Bulk and surface solutions by solving the 3D coupled bulk-surface reaction-diffusion equation, using the meshfree time stepping method with and without the greedy algorithm in/on a torus. The meshfree method uses $n = 2644 + 1430$, $\epsilon = 1$ and time step $\triangle t = 0.005$. Without the greedy algorithm, faint and irregular patterns emerge, while using the algorithm produces clearer, symmetric and well-separated spot patterns.

Table 3: The number of selected basis functions and the stopping criteria corresponding to the solutions corresponding greedy cases in Figure 12.

	$\epsilon = 4$					$\epsilon = 5$				
	$n'_\Omega(2644)$	$n'_S(1430)$	SC(Ω)	SC($\mathcal{S}$)	t_{CPU}	$n'_\Omega(2644)$	$n'_S(1430)$	SC(Ω)	SC($\mathcal{S}$)	t_{CPU}
$\triangle t = 0.01$	753	116	SC-2′	SC-2′	130.56	771	222	SC-2′	SC-2′	61.86
$\triangle t = 0.005$	776	125	SC-2′	SC-2′	115.96	2039	208	SC-2′	SC-2′	79.79

terns materialized, whereas the employment of the algorithm produced clearer, symmetric and well-separated spot configurations.

Observation 2: The greedy algorithm improves efficiency in successful runs

Next, we implement the experiments on the torus with some larger values of the shape parameter, say, $\epsilon = 4$ and $\epsilon = 5$. We fix the number of basis functions as $n = 2644 + 1430$ and use time steps as $\triangle t = 0.01$ and $\triangle t = 0.005$. As seen in Figure 11, the patterns of u and v, as well as w and s, are reversely correlated. Accordingly, we focus our discussion on the solutions of u and w for both the bulk and surface functions.

Figure 12 displays the bulk and surface function solutions in/on torus under different RBF shape parameters and time step settings. We observe that the patterns obtained with and without the greedy algorithm are nearly identical. Across all solutions, the spots tend to converge to the same pattern, with 4 spots in the bulk and 8 spots on the surface. This trend suggests that even with different numbers of basis functions, the greedy algorithm can facilitate pattern formation.

Additionally, we list the stopping criteria and the number of selected basis functions out of $n = 2644 + 1430$ in Table 3 provided by the greedy algorithm. Except for the situation when taking $\epsilon = 5$ on the surface, taking $\triangle t = 0.005$ tends to select more basis than taking $\triangle t = 0.01$. Since using a different number of basis functions produces nearly identical patterns, it nearly has no effect on pattern formation compared to a solution without a greedy algorithm. Therefore, the larger time step $\triangle t = 0.01$ with the greedy algorithm appears to be the most effective setting for generating patterns in this experiment.

Observation 3: The greedy algorithm performs robustly in various domains

In our continued exploration of the effectiveness of our approach, we further examine the solutions of (15) on additional 3D domains. Figures 13 and 14 present solutions of u and w in a

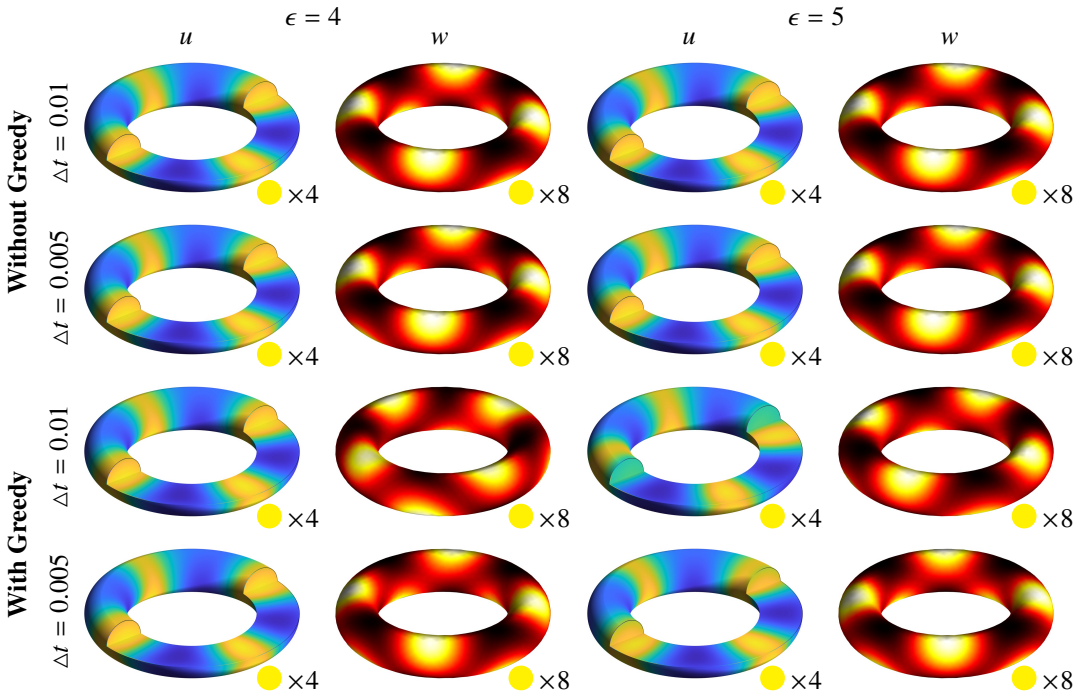


Figure 12: Example 4.2: Bulk and surface solutions by solving the 3D coupled bulk-surface reaction-diffusion equation corresponding with Figure 11 with time steps $\triangle t = 0.01$ and $\triangle t = 0.005$ and larger RBF shape parameters $\epsilon = 4$ and $\epsilon = 5$. The number of bases used and stopping criteria in the greedy cases are in Table 3.

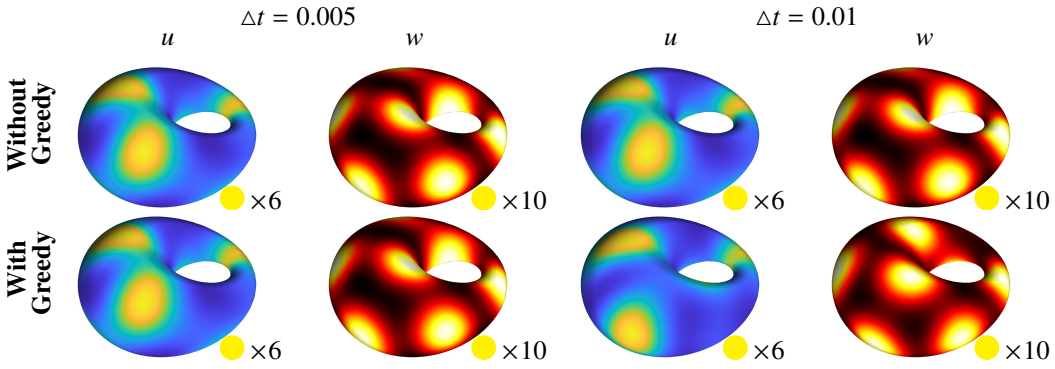


Figure 13: Example 4.2: Solutions by solving the 3D coupled bulk-surface reaction-diffusion equation with and without the greedy algorithm in/on a Dupin's cyclide. The meshfree method uses $n = 6760 + 2956$ and $\epsilon = 4$. The time steps are taken as $\triangle t = 0.005$ and $\triangle t = 0.01$.

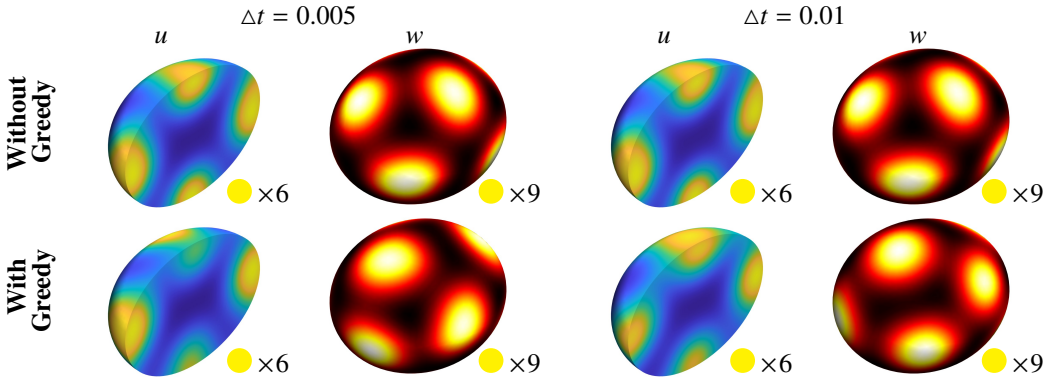


Figure 14: Example 4.2: Solutions by solving the 3D coupled bulk-surface reaction-diffusion equation with and without the greedy algorithm in/on an ellipsoid. The meshfree method uses $n = 3395 + 1164$ and $\epsilon = 6$. The time steps are taken as $\triangle t = 0.005$ and $\triangle t = 0.01$.

Dupin's cyclide and an ellipsoid. For the cyclide, we use $\epsilon = 4$, and the time stepping method with $\triangle t = 0.01$ and $\triangle t = 0.005$. Due to the complex geometry, the spots formed in/on the cyclide are not as uniformly distributed as in the torus. Nonetheless, we observe that the greedy algorithm produces the same number of clear and separated spots as the non-greedy solutions, both in the bulk and on the surface. Specifically, there are consistently 6 spots in the bulk and 10 on the surface. However, under different settings, the distributions of spots may not always be identical. For example, when applying the greedy algorithm with $\triangle t = 0.01$, the solution in the cyclide domain appears different from other solutions. This difference is due to the fact that the positions of the spots are influenced by the number and position of the selected basis functions.

Likewise, the solutions on the ellipsoid with the greedy algorithm also exhibit slight differences when compared to those without. Specifically, when we apply the greedy algorithm with $\triangle t = 0.005$, the spots on the top of the ellipsoid appear different from the patterns in other solutions. Nonetheless, despite using or not using the greedy algorithm, the number of spots in the bulk and on the surface remains almost the same.

5. Conclusion

We present two stopping criteria for the block greedy algorithm to improve stability and accuracy when solving fully-discrete matrix systems of a meshfree time-stepping method for solving parabolic PDEs. When stopped by a large condition number, we propose a minimum residual search within newly added columns to maximize the approximation power. When stopped by a small residual, a backtracking process is used to reduce selected columns while keeping the residual below a threshold. Together with suggested time step-dependent tolerance selection strategies, these stopping criteria enable the greedy algorithm to balance accuracy and stability. This is demonstrated in solving 2D and 3D heat equations, where the solutions' accuracy is somewhat independent of which stopping criteria is used. We applied the greedy algorithm to coupled bulk-surface pattern formation as a practical first step. We use symmetry arguments on interfacial patterns to identify numerical defects.

The numerical examples presented in both 2D and 3D demonstrate that, under appropriate temporal and spatial conditions, utilizing the greedy algorithm along with varying numbers of basis functions can lead to the emergence of reasonable patterns. We have made improvements to the greedy algorithm to enhance its stability when solving PDEs using meshfree and time-stepping methods. The patterns efficiently simulated using the greedy algorithm successfully maintain the general characteristics of the original configurations. In our future work, we will explore the possibility of combining our approach with variably scaled RBF [27, 28] to enhance the stability of the system even further. We may also consider combining our approach with [29, 30] in solving the coupled bulk-surface reaction-diffusion equations on domains with moving surfaces.

Acknowledgements

This work was supported by the General Research Fund (GRF No. 12303818, 12301419, 12301520) of Hong Kong Research Grant Council.

[1] M. Amirfakhrian, M. Arghand, E. Kansa, A new approximate method for an inverse time-dependent heat source problem using fundamental solutions and rbfs, Eng. Anal. Bound. Elem. 64 (2016) 278–289.

[2] E. de Vito, L. Rosasco, A. Caponnetto, M. Piana, A. Verri, Some properties of regularized kernel methods, J. Mach. Learn. Res. 5 (2004) 1363–1390.

[3] R. Schaback, H. Wendland, Adaptive greedy techniques for approximate solution of large RBF systems, Numer. Algorithms 24 (3) (2000) 239–254.

[4] R. Campagna, S. De Marchi, E. Perracchione, G. Santin, Stable interpolation with exponential-polynomial splines and node selection via greedy algorithms, Adv. Comput. Math. 48 (6) (2022).

[5] T. Wenzel, G. Santin, B. Haasdonk, Analysis of target data-dependent greedy kernel algorithms: Convergence rates for f-, $f \cdot p$- and f/p-greedy, Constr. Approx. 57 (1) (2023) 45–74.

[6] Y. C. Hon, R. Schaback, X. Zhou, An adaptive greedy algorithm for solving large RBF collocation problems, Numer. Algorithms 32 (1) (2003) 13–25.

[7] L. Ling, R. Opfer, R. Schaback, Results on meshless collocation techniques, Eng Anal Bound Elem 30 (4) (2006) 247–253.

[8] L. Ling, R. Schaback, An improved subspace selection algorithm for meshless collocation methods, Int J Numer Methods Eng 80 (13) (2009) 1623–1639.

[9] L. Ling, A fast block-greedy algorithm for quasi-optimal meshless trial subspace selection, SIAM J. Sci. Comput. 38 (2) (2016) A1224–A1250.

[10] L. Ling, S. N. Chui, Fully adaptive kernel-based methods, Int J Numer Methods Eng 114 (4) (2018) 454–467.

[11] M. Chen, L. Ling, Exploring oversampling in rbf least-squares collocation method of lines for surface diffusion (2023). arXiv:2203.08579.

[12] M. Chen, K. C. Cheung, L. Ling, A kernel-based least-squares collocation method for surface diffusion, SIAM J. Numer. Anal. 61 (3) (2023) 1386–1404.

[13] S. Hammarling, 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).
[14] S. J. Ruuth, Implicit-explicit methods for reaction-diffusion problems in pattern formation, J. Math. Biol. 34 (2) (1995) 148–176.
[15] M. Chen, L. Ling, Extrinsic meshless collocation methods for PDEs on manifolds, SIAM J. Numer. Anal. 58 (2) (2020) 988–1007.
[16] K. C. Cheung, L. Ling, A kernel-based embedding method and convergence analysis for surfaces PDEs, SIAM J. Sci. Comput. 40 (1) (2018) A266–A287.
[17] K. C. Cheung, L. Ling, R. Schaback, H^2–convergence of least-squares kernel collocation methods, SIAM J. Numer. Anal. 56 (1) (2018) 614–633.
[18] G. Santin, T. Karvonen, B. Haasdonk, Sampling based approximation of linear functionals in reproducing kernel Hilbert spaces, BIT Numer. Math. 62 (1) (2021) 279–310.
[19] A. Rätz, M. Röger, Symmetry breaking in a bulk-surface reaction-diffusion model for signaling networks, Nonlinearity 27 (2014).
[20] A. Rätz, M. Röger, Turing instabilities in a mathematical model for signaling networks, J. Math. Biol. 65 (2011) 1215 – 1244.
[21] Q. Li, X. Lin, Q. Luo, Y. Chen, J. Wang, B. Jiang, F. Pan, Kinetics of the hydrogen absorption and desorption processes of hydrogen storage alloys: A review, Int. J. Miner. Metall. Mater. 29 (1) (2022) 32–48.
[22] A. Madzvamuse, H. Chung, C. Venkataraman, Stability analysis and simulations of coupled bulk-surface reaction-diffusion systems, Proc R Soc (London) A. 471 (2015).
[23] J. Murray, Mathematical Biology II: Spatial Models and Biomedical Applications, Springer, 1997.
[24] E. J. Fuselier, G. B. Wright, Scattered data interpolation on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates, SIAM J. Numer. Anal. 50 (3) (2012) 1753–1776.
[25] M. Chen, L. Ling, Kernel-based meshless collocation methods for solving coupled bulk–surface partial differential equations, J. Sci. Comput. 81 (2019) 375–391.
[26] C. B. Macdonald, B. Merriman, S. J. Ruuth, Simple computation of reaction–diffusion processes on point clouds, Proceedings of the National Academy of Sciences 110 (2013) 9209 – 9214.
[27] M. Chen, L. Ling, Y. Su, Solving interpolation problems on surfaces stochastically and greedily, Dolomites Res. Notes Approx. 15 (3) (2022) 26–36.
[28] F. Marchetti, L. Ling, A stochastic extended Rippa's algorithm for LpOCV, Appl. Math. Lett. 129 (2022).
[29] A. Petras, L. Ling, C. Piret, S. Ruuth, A least-squares implicit RBF-FD closest point method and applications to PDEs on moving surfaces, J. Comput. Phys. 381 (2019) 146–161.
[30] A. Petras, S. Ruuth, PDEs on moving surfaces via the closest point method and a modified grid based particle method, J. Comput. Phys. 312 (2016) 139–156.