Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Preconditioning for radial basis function partition of unity methods

Alfa Heryudono · Elisabeth Larsson · Alison Ramage · Lina von Sydow

Received: date / Accepted: date

Abstract Meshfree radial basis function (RBF) methods are of interest for solving partial differential equations due to attractive convergence properties, flexibility with respect to geometry, and ease of implementation. For global RBF methods, the computational cost grows rapidly with dimension and problem size, so localised approaches, such as partition of unity or stencil based RBF methods, are currently being developed. An RBF partition of unity method (RBF–PUM) approximates functions through a combination of local RBF approximations. The linear systems that arise are locally unstructured, but with a global structure due to the partitioning of the domain. Due to the sparsity of the matrices, for large scale problems, iterative solution methods are needed both for computational reasons and to reduce memory requirements. In this paper we implement and test different algebraic preconditioning strategies based on the structure of the matrix in combination with incomplete factorisations. We compare their performance for different orderings and problem settings and find that a no-fill incomplete factorisation of the central band of the original discretisation matrix provides a robust and efficient preconditioner.

Authors are listed in alphabetical order.
Heryudono was partially supported by AFOSR grant FA9550-09-1-0208 and by National Science Foundation grant DMS-1318427.

A. Heryudono
Department of Mathematics, University of Massachusetts, Dartmouth, Massachusetts, 02747, USA
Tel.: +1-508-999-8516
E-mail: aheryudono@umassd.edu

E. Larsson
Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden
Tel.: + 46-18-471 2768
E-mail: elisabeth.larsson@it.uu.se

A. Ramage
Department of Mathematics and Statistics, University of Strathclyde, Glasgow, G1 1XH, Scotland
Tel.: +44-141-548 3801
E-mail: a.ramage@strath.ac.uk

L. von Sydow
Department of Information Technology, Uppsala University, Box 337, 751 05 Uppsala, Sweden
Tel.: + 46-18-471 2785
E-mail: lina@it.uu.se

Keywords radial basis function · partition of unity · RBF–PUM · iterative method · preconditioning · algebraic preconditioner

Mathematics Subject Classification (2000) MSC 65F08 · MSC 65M70

1 Introduction

There is an increasing interest in using methods based on radial basis function (RBF) approximation [12] for the solution of partial differential equations (PDEs). The main advantages of these methods are that they are mesh free, which provides flexibility with respect to the geometry of the computational domain; they can be spectrally accurate for smooth solution functions [30, 31]; they are comparatively easy to apply to high-dimensional problems, which is vital for application areas such as finance, quantum dynamics, and systems biology.

The typical form of an RBF approximation $\hat{u}(\underline{x})$ to a solution function $u(\underline{x})$, where $\underline{x} = (x_1,\ldots,x_d) \in \mathbb{R}^d$, is

$$\hat{u}(\underline{x}) = \sum_{j=1}^{N} \lambda_j \phi_j(\underline{x}), \tag{1}$$

where λ_j are coefficients to be determined. Here $\phi(r)$ is a radial basis function, and $\phi_j(\underline{x}) = \phi(\varepsilon\|\underline{x}-\underline{x}_j\|)$, where $\underline{x}_j$, $j = 1,\ldots,N$ are the (scattered) node points at which the individual RBFs are centred. The parameter ε is called the shape parameter and controls the flatness of the RBFs. This shape parameter has a significant influence on the accuracy of the approximation, as well as on the conditioning of the resulting linear systems.

By requiring the RBF approximation to interpolate the solution at the node points, we arrive at a linear system

$$A\underline{\lambda} = \underline{u}, \tag{2}$$

where the $N \times N$ matrix A has entries $a_{ij} = \phi_j(\underline{x}_i)$, $i, j = 1,\ldots N$, $\underline{\lambda} = (\lambda_1,\ldots,\lambda_N)^T$, and $\underline{u} = (u(\underline{x}_1),\ldots,u(\underline{x}_N))^T$. If the basis function that is used has global support, the matrix A becomes dense, and the computational cost of solving the linear system (2), especially in higher dimensions, becomes prohibitive. Furthermore, as the shape parameter ε goes to zero, the RBFs become flat, and the linear system becomes severely ill-conditioned [18, 19]. As was noted in [8] for one-dimensional problems, and later in [23] for multivariate problems, this ill-conditioning is an artefact of the particular formulation of the problem, while the approximation result itself depends smoothly on the data. Several methods have been proposed to eliminate these conditioning problems. The Contour-Padé algorithm [17] came first, and was then followed by the RBF-QR method for the sphere [16] and for Cartesian space [14] and, more recently, the RBF-GA method [15]. All of these approaches compute the same end result as the ill-conditioned formulation, but through a stable reformulation. In this paper, we employ the RBF-QR method [14, 25] for constructing differentiation matrices.

In order to address the computational cost issues of the global RBF method, we need to introduce locality. An easy way to do that is to use compactly supported RBFs, such as the Wendland functions [41], but then the spectral convergence properties are lost. Here we take another approach, where the infinitely smooth RBFs are still used in the approximation but over local subregions of the computational domain. The possibility of using RBFs in a partition of unity scheme was mentioned in [2], further discussed in [12, 42], and implemented and analysed for elliptic PDEs in [24] (see also [34, 36], where an RBF partition of unity method (RBF–PUM) was applied to parabolic PDEs). In the recent benchmark paper [39], many different types of methods, including RBF methods, were evaluated for option pricing problems. The results demonstrated that out of the implemented RBF methods, the RBF–PUM approach was the most efficient computationally.

The RBF–PUM discretisation leads to sparse unstructured matrices. For larger problem sizes and in higher dimensions, it is therefore necessary in terms of computational efficiency to use an iterative solver for the linear systems that arise. Here we use the Krylov subspace method GMRES [33] in order to take advantage of its theoretical residual minimising property (see §4): other methods such as Bi-CGSTAB [40] or IDR [38] may be more suitable for practical implementation with large problems. The RBF-PUM matrices are non-symmetric and moderately ill-conditioned so iterative convergence is typically very slow. It is therefore important to have an effective preconditioner for these systems. In this paper, we design and evaluate the performance of algebraic preconditioners based on (incomplete) LU factorisation that take advantage of the underlying structure of the coefficient matrices. To the best of our knowledge, this is the first time that any preconditioner for this type of discretisation has been developed.

In the current literature, most papers on preconditioning for RBF interpolation or approximation consider global approximations like (1). In such circumstances, using preconditioners based on approximate cardinal basis functions computed on a reduced node set has been shown to be successful (see, e.g., [4,6,13,20,27]). In our case, we are using a local approximation, so the coefficient matrices are already sparse. Preconditioners utilising the Toeplitz structure of a discretisation with a logically Cartesian node layout are introduced in [3,7]. Although efficient, these may be hard to use for the unstructured sets of nodes which are useful for non-trivial geometries. In [11], algebraic preconditioners are constructed for compactly-supported RBFs, utilising the two-by-two block structure of the matrix arising from the separation of boundary and interior nodes, in combination with an additive Schwarz method. A similar type of algebraic preconditioner is investigated in [1], for a special case of complex matrices with symmetric positive definite real and imaginary parts. In the latter paper, sparsification is used for the RBF example, in the sense that small off-diagonal elements are removed, and their mass is added to the corresponding diagonal element.

The remainder of this paper is structured as follows. In §2, we present details of the RBF discretisation method. This is followed in §3 by a description of the set of Poisson test problems which we use throughout the paper, together with a discussion of some important issues concerning node numbering and matrix structure. In §4, the iterative method and new preconditioners are described, and in §5 we make some predictions concerning the asymptotic convergence rates of the resulting methods. Finally, in §6, we present the results of several numerical experiments and draw some conclusions in §7.

2 The Radial Basis Function Partition of Unity Method

Since in this paper we are mainly interested in the efficiency of different preconditioning approaches, we restrict our attention to a stationary linear PDE with Dirichlet boundary conditions. We note, however, that the techniques presented here can also be generalised to other problem settings, including time-dependent problems. We define our model PDE on a closed domain $\Omega \subset \mathbb{R}^d$, with boundary $\partial\Omega$ as follows:

$$\mathscr{L}u(\underline{x}) = f(\underline{x}), \quad \text{in } \Omega, \tag{3a}$$

$$u(\underline{x}) = g(\underline{x}), \quad \text{at } \partial\Omega, \tag{3b}$$

where $\underline{x} = (x_1, \ldots, x_d)$. In the numerical experiments presented later, we take $\mathscr{L} = -\Delta$ (the Laplace operator).

In a partition of unity method, the global approximation $\tilde{u}(\underline{x})$ to the solution $u(\underline{x})$ is constructed as a weighted sum of local solutions $\tilde{u}_j(\underline{x})$ on overlapping patches Ω_j, $j = 1, \ldots, P$. That is,

$$\tilde{u}(\underline{x}) = \sum_{j=1}^{P} w_j(\underline{x}) \tilde{u}_j(\underline{x}). \tag{4}$$

where w_j, $j = 1, \ldots, P$ are weight functions. The patches Ω_j need to form a cover of the domain in the sense that

$$\bigcup_{j=1}^{P} \Omega_j \supseteq \Omega.$$

There should also be an upper bound K for the number of patches that overlap at one given point $\underline{x} \in \Omega$.

An illustration of typical sets of circular patches used in this paper can be seen in Figure 1 in §3. We define the overlap γ relative to the minimal patch radius R_0 such that, for given patch centres, we fulfil the conditions required for a cover. That is, in subfigures 1(a) and 1(b), a patch radius of R_0 would correspond to patch boundaries just touching in the diagonal direction. With an overlap γ, we use a patch radius of $R = (1+\gamma)R_0$.

The partition of unity weight functions w_j are non-negative, compactly supported on Ω_j and satisfy

$$\sum_{j=1}^{P} w_j(\underline{x}) = 1, \quad \forall \underline{x} \in \Omega.$$

Furthermore, the weight functions need to be p times continuously differentiable, where p is the order of the PDE operator $\mathscr{L}$ (for $\mathscr{L} = -\Delta$, $p = 2$). We follow the approach in [24, 34, 36], and use Shepard's method [12, 37] applied to compactly supported C^2 Wendland functions [41] to construct the weight functions

$$w_j(\underline{x}) = \frac{\varphi_j(\underline{x})}{\sum_{j=1}^{P} \varphi_j(\underline{x})}, \qquad j = 1, \ldots, P,$$

where $\varphi_j(\underline{x})$ is the particular Wendland function supported on Ω_j.

The PDE (3) is discretised with a collocation method. We therefore define a global set of distinct nodes $X = \{\underline{x}_k\}_{k=1}^{N}$ in Ω, requiring the PDE (3a) to be satisfied at interior nodes, and the boundary condition (3b) to be satisfied at boundary nodes. That is, we require

$$\mathscr{L}\tilde{u}(\underline{x}_k) = \sum_{j=1}^{P} \mathscr{L}\left(w_j(\underline{x}_k)\tilde{u}_j(\underline{x}_k)\right) = f(\underline{x}_k), \qquad \underline{x}_k \in \Omega \setminus \partial\Omega, \tag{5a}$$

$$\tilde{u}(\underline{x}_k) = \sum_{j=1}^{P} w_j(\underline{x}_k)\tilde{u}_j(\underline{x}_k) = g(\underline{x}_k), \qquad \underline{x}_k \in \partial\Omega. \tag{5b}$$

For the particular case of the Poisson problem, where $\mathscr{L} = -\Delta$, the local operator can be expanded to give

$$\mathscr{L}\left(w_j(\underline{x}_k)\tilde{u}_j(\underline{x}_k)\right) = -\Delta w_j(\underline{x}_k)\tilde{u}_j(\underline{x}_k) - 2\nabla w_j(\underline{x}_k) \cdot \nabla\tilde{u}_j(\underline{x}_k) - w_j(\underline{x}_k)\Delta\tilde{u}_j(\underline{x}_k). \tag{6}$$

A general exposition for other linear operators can be found in [34].

When using partition of unity approximations such as (5), it is convenient to work at the patch level. We therefore now define local subsets of nodes as $X_j = \{\underline{x}_i^j\}_{i=1}^{n_j} = \{\underline{x}_k \in X \,|\, \underline{x}_k \in \Omega_j\}$, where n_j is the number of nodes that fall in Ω_j. In addition, we define the index mapping $k = \pi(i,j)$ that returns the global index k for a given local node $\underline{x}_i^j$. In the particular case of RBF-PUM, the local solutions $\tilde{u}_j(\underline{x})$ are RBF approximations

$$\tilde{u}_j(\underline{x}) = \sum_{i=1}^{n_j} \lambda_i^j \phi_i^j(\underline{x}), \tag{7}$$

where λ_i^j are coefficients to be determined and $\phi_i^j(\underline{x}) = \phi(\varepsilon\|\underline{x} - \underline{x}_i^j\|)$. However, in the partition of unity setting, it is inconvenient to use the coefficients λ_i^j as the degrees of freedom, as there is more than one coefficient per collocation node in the regions of overlap between patches. Instead, we solve for the nodal values $\tilde{u}_j(\underline{x}_i^j) \equiv \tilde{u}(\underline{x}_k)$, where $k = \pi(i,j)$. That is, we require the local solution from two adjacent patches to take on the same value at the node points in the overlap region. The same requirement expressed in terms of the coefficients would result in a non-local condition involving all coefficents in both patches.

We now define the vector of local nodal values $\underline{u}_j = (\tilde{u}_j(\underline{x}_1^j), \ldots, \tilde{u}_j(\underline{x}_{n_j}^j))^T$ and the local coefficient vector $\underline{\lambda}_j = (\lambda_1^j, \ldots, \lambda_{n_j}^j)^T$. From (7), we then have the relations

$$A_j \underline{\lambda}_j = \underline{u}_j \quad \Rightarrow \quad \underline{\lambda}_j = A_j^{-1} \underline{u}_j,$$

where $A_j = \{\phi_m^j(\underline{x}_i^j)\}_{i,m=1}^{n_j}$, and

$$\mathscr{L}\underline{u}_j = D_j^{\mathscr{L}} \underline{\lambda}_j = D_j^{\mathscr{L}} A_j^{-1} \underline{u}_j,$$

where $D_j^{\mathscr{L}} = \{\mathscr{L}\phi_m^j(\underline{x}_i^j)\}_{i,m=1}^{n_j}$. Note that, for distinct node points with positive definite RBFs such as the Gaussians used for the numerical experiments in this paper, the local RBF interpolation matrices A_j are guaranteed to be non-singular [35]. We also define a diagonal matrix

$$W_j^{\mathscr{L}} = \mathrm{diag}(\mathscr{L}w_j(\underline{x}_1^j), \ldots, \mathscr{L}w_j(\underline{x}_{n_j}^j))$$

associated with each patch. Now, using (6), we can express the discrete local Laplacian operators as

$$\tilde{L}_j = (W_j^{\Delta} A_j + 2W_j^{\nabla} \cdot D_j^{\nabla} + W_j D_j^{\Delta}) A_j^{-1},$$

where the gradient operators are vector valued, and the scalar product is applied in the appropriate way. To get the discrete local PDE operator, we also include the boundary conditions, which gives

$$L_j(i,m) = \begin{cases} \tilde{L}_j(i,m), & \underline{x}_i^j \in \Omega \setminus \partial\Omega, \\ \delta_{im}, & \underline{x}_i^j \in \partial\Omega, \end{cases}$$

where δ_{im} is the Kronecker delta. Finally, we obtain the global discrete operator by, as in a finite element method, assembling the local matrices L_j into the global matrix L such that

$$L_j(i,m) \xrightarrow{+} L(\pi(i,j), \pi(m,j)), \quad j = 1, \ldots, P, \quad i,m = 1, \ldots, n_j.$$

The global right hand side $\underline{f} = (f_1, \ldots, f_N)^T$ is defined through

$$f_k = \begin{cases} f(\underline{x}_k), & \underline{x}_k \in \Omega \setminus \partial\Omega, \\ g(\underline{x}_k), & \underline{x}_k \in \partial\Omega. \end{cases}$$

With the global vector of nodal values defined by $\underline{u} = (\underline{x}_1, \ldots, \underline{x}_N)^T$, the final (global) linear system to be solved is

$$L\underline{u} = \underline{f}. \tag{8}$$

For small values of the shape parameter ε, the matrices L_j, and consequently L, become highly ill-conditioned when computed as described above [18,19]. This is problematic because, for smooth solution functions, a small positive shape parameter value typically gives the best accuracy of the solution [17,22,23]. Furthermore, refining the patches in RBF-PUM for a fixed ε results in a decreasing 'effective' shape parameter value, that is, the shape parameter becomes smaller in relation to the patch size. However, the problem of ill-conditioning for small shape parameters can be avoided by employing stable evaluation methods such as the Contour-Padé approach [17], the RBF-QR method [14,16,25], or the RBF-GA method [15]. Here we employ the RBF-QR method which, simply put, corresponds to a change of basis from $\{\phi_m^j\}$ to $\{\psi_m^j\}$ in the local problems. This significantly reduces the condition number of A_j, and allows for stable evaluation of L_j for small shape parameter values.

3 Model problems and ordering issues

To fix ideas, we will focus for the remainder of the paper on two specific two-dimensional model problems. As stated above, we will solve the PDE (3) with $\mathscr{L} = -\Delta$ (the Laplace operator). For simplicity, we use a manufactured solution $u(\underline{x})$ from which we can compute the right-hand-side functions f and g, namely,

$$u(\underline{x}) = \sin(x_1^2 + 2x_2^2) - \sin(2x_1^2 + (x_2 - 0.5)^2). \tag{9}$$

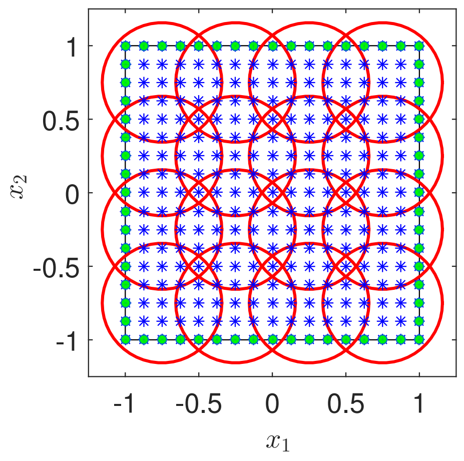
We solve this problem over two different two-dimensional physical domains Ω: for Model Problem I, the domain is the square $\Omega = [-1, 1]^2$, and for Model Problem II, the boundary of Ω is defined by

$$\partial\Omega = \{(r, \theta) | r(\theta) = 0.8 + 0.1(\sin(6\theta) + \sin(3\theta))\}. \tag{10}$$

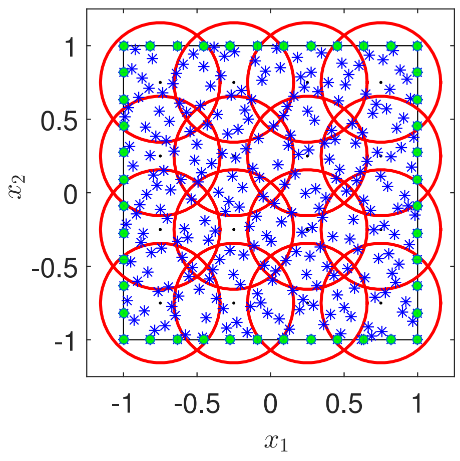
This region is illustrated in Figure 1(c).

In Figure 1 we show typical examples of patches and node distributions for Model Problem I (with 16 patches on the square domain) and Model Problem II (with 50 patches and domain boundary defined by (10)). In each case, the patch boundaries are shown in red, with patch centres marked as black dots. Points on the domain boundary (where the Dirichlet boundary conditions are applied) are represented by green circles. The amount of overlap between patches is $\gamma = 0.15$ for Model Problem I, and $\gamma = 0.3$ for Model Problem II. The square domain is shown with both Cartesian and Halton [21] nodes (shown as blue stars). The reason for choosing these two types of nodes is that they represent extremes in terms of node distributions: the Cartesian nodes are completely structured, while the Halton nodes are quasi random, and completely unstructured. For general geometries it is not possible to always have completely structured nodes. A typical scenario for a RBF-PUM discretisation would be to have unstructured nodes, but of a higher quality in terms of uniformity than Halton nodes. This is the case that is investigated for Model Problem II, see Figure 1(c).

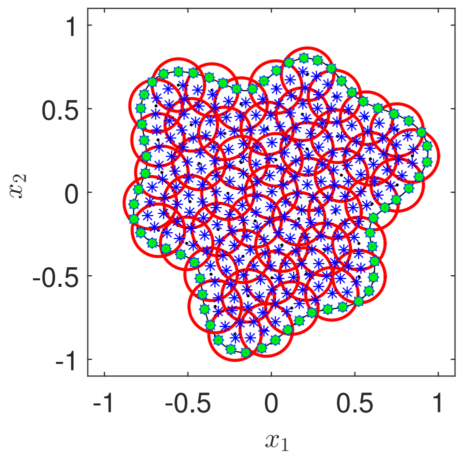
The patches similarly have a Cartesian layout for Model Problem I and an unstructured layout for Model Problem II. The number of patches is chosen such that the number of node points per patch is large enough ($\gtrsim 15$) to provide a reasonable local approximation, while still small enough ($\lesssim 100$) for the conditioning of the local problem to be manageable.



(a) Model Problem I, 289 Cartesian nodes, 16 patches.



(b) Model Problem I, 294 Halton nodes, 16 patches.

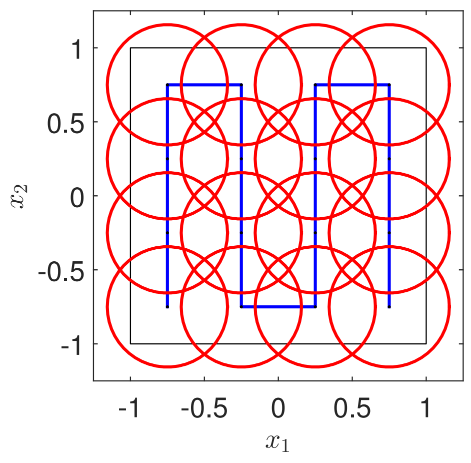


(c) Model Problem II, 298 unstructured nodes, 50 patches.

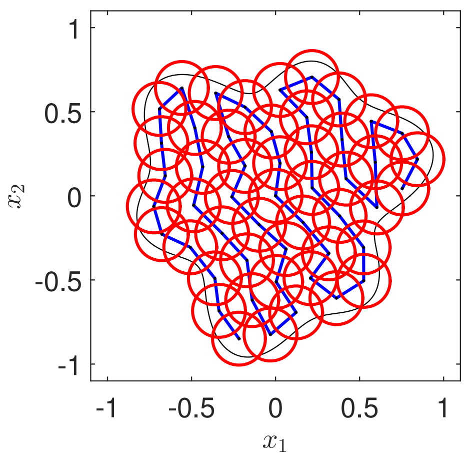
Fig. 1: Illustrations of typical patches and node distributions. Patch boundaries are marked with red circles, interior nodes with blue stars and boundary nodes with green circles.

One question which needs addressing is how the patches, and then the nodes within each patch, should be ordered. This is important as the sparsity pattern of L in (8) will have implications for the design of efficient fast solvers. In particular, as we will consider sparse factorisation techniques, we are interested in keeping the matrix entries as tightly banded as possible. To this end, we choose a *snake ordering* for the patches, where each patch (except the last) is followed by one of its neighbours. This ordering is illustrated for both model problems in Figure 2, where the patch ordering follows the blue line.

For Model Problem I, this is trivial to construct, beginning with a vertical ordering, and then alternating the direction in which the columns of patches are traversed (this could of course be done equivalently in a horizontal fashion). For Model Problem II, it is less ob-



(a) Patch order for Model Problem I.



(b) Patch order for Model Problem II.

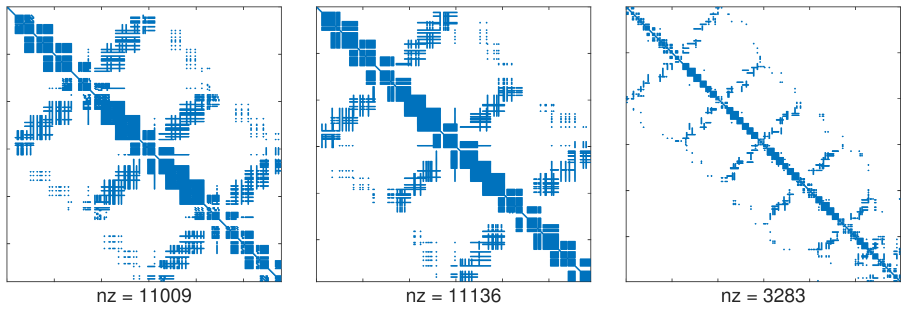
Fig. 2: Illustrations of the *snake* patch ordering strategy. The blue lines illustrate the order in which the patches are numbered.

vious how to proceed. We use the following simple heuristic approach. Starting with the patch whose centre has minimum y co-ordinate, we select a neighbour that is to the left or else above (in terms of the centre co-ordinates) for as long as possible. When this fails, we switch direction and look for a neighbour that is to the left or else below, continuing in this alternating way until all patches have been traversed. Although this approach may sometimes fail (for example, when the domain has thin sections with only one layer of patches such that changing direction is not possible), the general principle of ordering patches in terms of nearest neighbours in a linear-like way should still be followed where possible.

Having fixed an order for the patches, we now turn our attention to the ordering of nodes within each patch, with a similar aim of designing this to minimise the distance between neighbouring nodes. The strategy we use has two main components. First, each node $\underline{x}_k$ is allocated to a home patch, according to its largest weight. That is, it is associated with the patch Ω_j for which $w_j(\underline{x}_k) \geq w_i(\underline{x}_k)$, $i = 1, \ldots, P$, see (4). In the case of a tie, the first patch with this property is designated the home patch. Secondly, the nodes are then ordered within each patch as follows: first, nodes in the overlap of the current and preceding patch; then nodes only in the current patch; finally, nodes in the overlap of the current and the following patch. In this way, nodes that are located in the overlap regions between patches become close neighbours in the ordering, leading to a cleaner structure in the final global matrix. Examples of the sparsity of L resulting from this patch and node ordering are shown in Figure 3, where the three subfigures correspond to the three model problem configurations presented in Figure 1.

4 Iterative method and preconditioning

As the RBF-PUM coefficient matrix L is very sparse, solving system (8) with a direct method (based on the factorisation of L into easily invertible matrices) is not appropriate: the performance of direct methods scales poorly with problem size in terms of operation counts and



(a) Sparsity of L corresponding to patches and nodes in Figure 1(a). (b) Sparsity of L corresponding to patches and nodes in Figure 1(b). (c) Sparsity of L corresponding to patches and nodes in Figure 1(c).

Fig. 3: Illustrations of typical sparsity patterns of L corresponding to the sample patch and node combinations in Figure 1.

memory requirements, especially for high-dimensional PDE problems. Instead, we adopt an iterative approach to take full advantage of the sparsity induced by the local approximation. In this paper, we will focus on the Generalized Minimum Residual (GMRES) method [33]. As mentioned in the introduction, GMRES is usually not the most efficient method in practice, as it involves storing and re-orthogonalising against an increasing number of vectors at each iteration. For implementation purposes, the restarted version GMRES(m) should be used, or an alternative more cost effective Krylov method such as Bi-CGSTAB [40] or IDR [38]. However, we use GMRES here for its clear theoretical framework as outlined below.

It is well known that the convergence of GMRES (and other iterative methods) can be improved by introducing the concept of preconditioning. Theoretically, this is equivalent to replacing L by a preconditioned matrix whose eigenvalue spectrum facilitates faster iterative convergence (see below). Considerable research has been carried out in recent years to find inexpensive ways to generate suitable preconditioners for a wide variety of problems with different types of coefficient matrix (see, for example, [5] or any standard textbook on iterative methods). Here we will employ right preconditioning and solve linear systems equivalent to (8) of the form

$$LM^{-1}\underline{y} = \underline{f}, \qquad M\underline{u} = \underline{y}.$$

Note that in practice it is not necessary to form the preconditioned matrix LM^{-1} explicitly (which would again result in a loss of sparsity): we only need to solve 'inner' linear systems with M as coefficient matrix. The aim is therefore to find a preconditioner M such that LM^{-1} has an improved eigenvalue structure, while a system with coefficient matrix M is cheap to solve. This latter point is primarily what motivates the use of sparse factorisations as preconditioners.

The GMRES method has the attractive theoretical property of minimising the 2-norm of the residual at each iteration. That is, at iteration i, the residual vector $\underline{r}^i = \underline{f} - LM^{-1}\underline{y}^i$ satisfies

$$\|\underline{r}^i\|_2 = \min_{p_i \in \mathscr{P}_i, p_i(0)=1} \|p_i(LM^{-1})\underline{r}^0\|_2,$$

where $\mathscr{P}_i$ is the set of all polynomials of degree i. Furthermore, if the preconditioned coefficient matrix LM^{-1} is diagonalisable, it can be shown that

$$\frac{\|\underline{r}^i\|_2}{\|\underline{r}^0\|_2} \leq \text{cond}_2\left(W_{LM^{-1}}\right) \min_{p_i \in \mathscr{P}_i, p_i(0)=1} \max_{1 \leq \ell \leq N} |p_i(\alpha_\ell)|$$

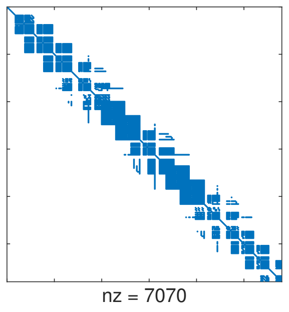
where α_ℓ, $\ell = 1, \ldots, N$ are the eigenvalues of LM^{-1} with corresponding eigenvector matrix $W_{LM^{-1}}$. If LM^{-1} is normal, then $\text{cond}_2\left(W_{LM^{-1}}\right) = 1$ and, in exact arithmetic, GMRES will converge in s iterations, where s is the number of distinct eigenvalues of LM^{-1}. In practice, although rounding error pollutes this theoretical result, the rate of convergence is still essentially bounded by the quantity

$$\rho_i = \min_{p_i \in \mathscr{P}_i, p_i(0)=1} \max_{1 \leq \ell \leq N} |p_i(\alpha_\ell)| \tag{11}$$

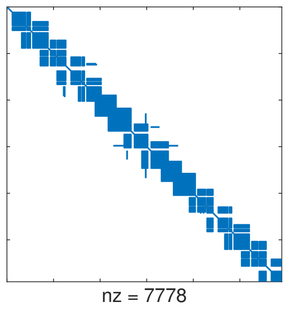
at each iteration, so fast convergence can be obtained if the eigenvalues of LM^{-1} are nicely clustered. Specifically, if the preconditioned eigenvalues lie in k dense clusters, we expect to obtain a good approximation to the solution vector in k GMRES iterations. If the preconditioned coefficient matrix is not normal (as is the case here), the factor $\text{cond}_2\left(W_{LM^{-1}}\right)$ reflects its degree of non-normality and convergence often exhibits an initial period of stagnation before bounds based on eigenvalues alone become descriptive [10]. This phenomenon can be observed in the convergence plots presented later (Figure7).

In the numerical experiments in §6, we will compare the performance of five different preconditioners with that of unpreconditioned GMRES. Two of these are based on a straightforward incomplete LU factorisation [29] of L. In the first (L-ILUn), no fill-in is allowed, that is, the sparsity pattern of the factors is fixed to the same as the sparsity pattern of the original matrix L (this method is often designated in the literature by ILU(0)). In the second variant (L-ILUd), a drop tolerance is specified (0.001 in our experiments), and any potential entries in the factors which are less than this value are ignored, again ensuring that the factors remain sparse.

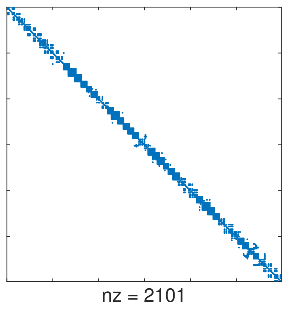
In addition to these two standard methods, we will also use three preconditioners based on factorisations of an alternative matrix, B, containing only the central band of L. Figure 4 shows the matrix B for the three configurations we have considered in Figures 1 and 3. In



(a) Sparsity of B corresponding to patches and nodes in Figure 1(a).



(b) Sparsity of B corresponding to patches and nodes in Figure 1(b).



(c) Sparsity of B corresponding to patches and nodes in Figure 1(c).

Fig. 4: Illustrations of typical sparsity patterns of B corresponding to the sample patch and node combinations in Figure 1.

each case, the bandwidth β of B (such that $b_{ij} = 0$ when $|i - j| > \beta$) has been set equal to $\max_j n_j - 1$ where n_j is the total number of nodes in patch j. This choice of bandwidth ensures that we retain information about the closest connections between nodes and patches, which is located in this central band thanks to the nearest-neighbour philosophy we have used in numbering of patches and nodes (see §3). In §6, we test three preconditioners based on B. The first of these is a full LU factorisation of B (B-LU). Although this will not be competitive in computational terms, the resulting iteration counts will give an indication of the amount of information lost by replacing the full coefficient matrix L with the banded approximation B. As more practical preconditioners, we also use the same two forms of ILU factorisation used for L, namely, with no fill-in (B-ILUn) and with a drop tolerance of 0.001 (B-ILUd). A summary of all five preconditioners implemented in §6, together with the acronyms used to refer to them in the following text, is given in Table 1.

Table 1: Summary of preconditioners implemented

B-LU	LU factorisation of B.
B-ILUn	Incomplete LU factorisation of B using no fill-in.
B-ILUd	Incomplete LU factorisation of B using drop tolerance 0.001.
L-ILUn	Incomplete LU factorisation of L using no fill-in.
L-ILUd	Incomplete LU factorisation of L using drop tolerance 0.001.

Note that, in terms of ILU fixed sparsity patterns, we have included here results only for the no-fill version (commonly called ILU(0)) and not the more general version, ILU(p) (see, for example, [32, §10.3.3]) which allows a higher level of fill-in. For the banded factorisation, we observed in our numerical experiments that most of the relevant information is already captured by B-ILUn, making versions with more fill-in essentially redundant. For the full factorisation of L, adding additional fill-in was more beneficial in terms of reducing iteration counts. However, the amount of extra storage required grew very quickly, making such methods unattractive when moving to high dimensional problems. We have therefore omitted results obtained using these methods from this paper.

5 Convergence estimates for GMRES

As described in §4, the asymptotic convergence phase of GMRES can be quantified by considering the factors ρ_i in (11) based on the eigenvalues α_ℓ of the coefficient matrix. In practice, however, the eigenvalues α_ℓ are not usually readily available, so it is common to use instead a related expression, based on a compact and continuous set S which contains the relevant eigenspectrum (but excludes the origin), of the form

$$\rho_i(S) = \min_{p_i \in \mathscr{P}_i, p_i(0)=1} \max_{\sigma \in S} |p_i(\sigma)|.$$

To remove the dependence on the iteration number i, it is often more convenient to consider the so-called *asymptotic convergence factor* of the set S (see e.g. [26, §5.7.6]) defined by

$$\rho(S) = \lim_{i \to \infty} (\rho_i(S))^{1/i}. \tag{12}$$

Although $\rho(S)$ can be difficult to quantify analytically, its value can be estimated using a computational technique based on conformal mappings. Specifically, if Φ is a conformal map from the exterior of S to the exterior of the unit disc that satisfies $\Phi(\infty) = \infty$, then $\rho(S)$ in (12) can be approximated by the value of $|\Phi(0)|^{-1}$ (see [9] for more details). In what follows, we apply this technique with S chosen to be the complex hull of the eigenvalue spectrum being studied.

We begin with Model Problem I with Cartesian grid points, but here using more points ($N = 1225$, with 64 (8×8) patches) than shown in Figure 1(a). Gaussian RBFs, $\phi(\varepsilon r) = \exp(-\varepsilon^2 r^2)$, with shape parameter $\varepsilon = 1.2$ are used. Here L (and its associated preconditioned versions) is positive definite, so the convex hull of the eigenvalues does not contain the origin, and the procedure for estimating the asymptotic convergence factor outlined above can be carried out in all cases. The pictures in Figure 5 show the eigenvalues (blue circles) of the coefficient matrix after the different preconditioners have been applied, with the convex hull outlined in red.

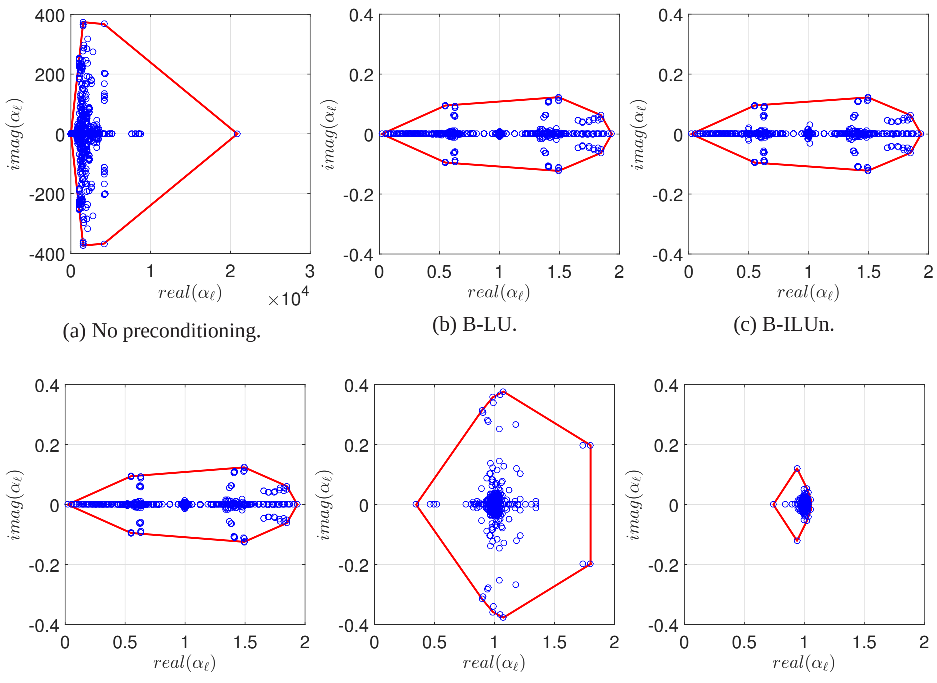


Fig. 5: Eigenvalues of the coefficient matrix with various preconditioners for Model Problem I with 1225 Cartesian points and 64 patches together with the associated convex hull used in the calculation of ρ in (12).

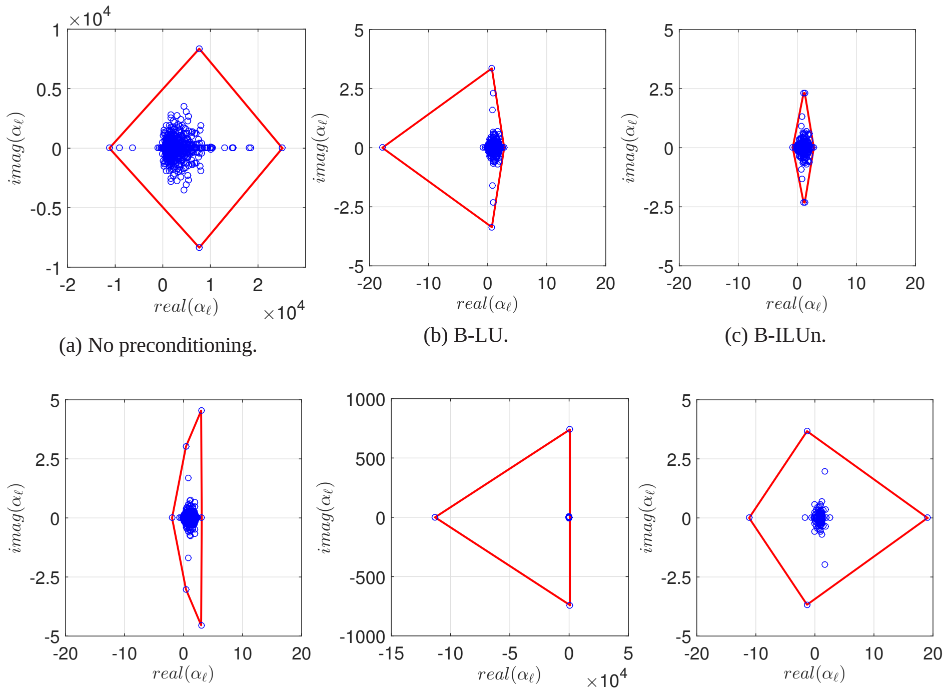
The values of ρ associated with the spectra represented in Figure 5 are listed in Table 2. When no preconditioning is applied, the value of ρ is close to one suggesting that the convergence of unpreconditioned GMRES will be slow: it can be seen that the eigenvalues of L itself (Figure 5(a)) are not well clustered. The three preconditioners based on B all lead to preconditioned spectra which look very similar (Figures 5(b), (c) and (d)), with essentially

Table 2: Approximate asymptotic convergence factor for Model Problem I with 1225 Cartesian points and 64 patches using different preconditioners.

None	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
0.990	0.839	0.839	0.840	0.506	0.132

the same convergence factor. An improvement in GMRES convergence rate is anticipated with all three. The fact that the two preconditioners based on L lead to very clustered eigenvalues (Figures 5(e) and (f)) is reflected in the much smaller values of ρ predicted for these methods, suggesting that they will require few iterations for convergence.

Analogous eigenvalue plots for Model Problem I with 1258 Halton points and 64 patches are shown in Figure 6. Here, the convex hull of the eigenvalues always contains the origin



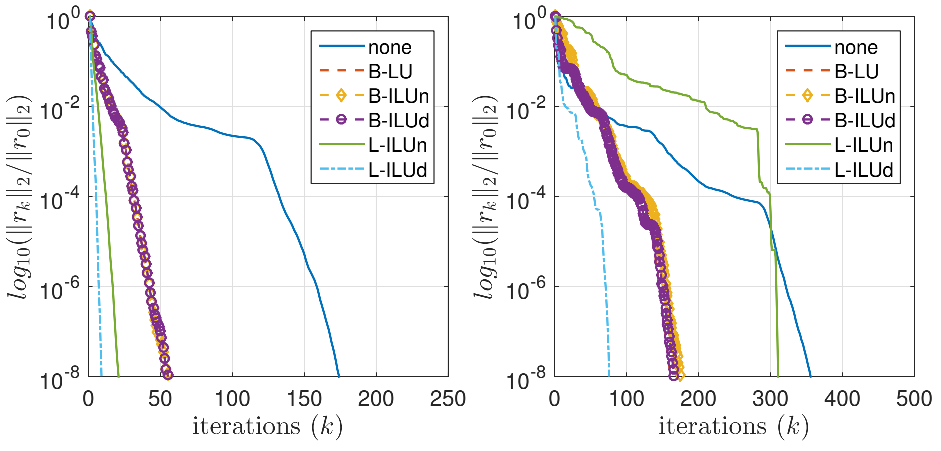
(a) No preconditioning. (b) B-LU. (c) B-ILUn.

(d) B-ILUd. (e) L-ILUn. (f) L-ILUd.

Fig. 6: Eigenvalues of the coefficient matrix with various preconditioners for Model Problem I with 1258 Halton points and 64 patches together with the associated convex hull. Note that the axis limits are the same for all preconditioned spectra except for the L-ILUn preconditioner.

so the above method for estimating the asymptotic convergence rate is not applicable.

Figure 7 shows the actual residual reduction for Model Problem I using 1225 Cartesian points (Figure 7(a)) and 1258 Halton points (Figure 7(b)). All calculations used a zero



(a) 1225 Cartesian points. (b) 1258 Halton points.

Fig. 7: Convergence for Model Problem I using 64 patches.

starting guess, and each GMRES iteration was terminated when

$$\|\underline{r}^i\|_2 \leq 10^{-8}\,\|\underline{r}^0\|_2. \tag{13}$$

These results can be used to compute the *residual reduction factor* $\tilde{\rho}$, defined by

$$\tilde{\rho} = \left(\frac{\|\underline{r}^i\|_2}{\|\underline{r}^0\|_2}\right)^{1/i},$$

where i is the number of iterations required for convergence. These values are shown in Table 3. The actual residual reduction factor for the Cartesian points is slightly better than expected from theory (Table 2) but exhibits the same relative behaviour over the range of preconditioners.

Table 3: Residual reduction for Model Problem I with 64 patches and 1225 Cartesian points and 1258 Halton points respectively, using different preconditioners.

	None	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
1225 Cartesian points	0.898	0.713	0.713	0.711	0.397	0.098
1258 Halton points	0.949	0.891	0.898	0.895	0.940	0.778

6 Numerical results

In this section we present the results of some numerical experiments which investigate the comparative performance of preconditioned GMRES for the preconditioners listed in Table 1. All calculations were carried out with MATLAB [28], and the timings were produced using `tic` and `toc`. The initial guess was zero, and all tests used the GMRES stopping criterion (13). The underlying PDE problems which we solve are Model Problems I and II as described in §3. However, as we are interested in studying the effect on solver performance of varying the number of patches, as well as the type and number of points used, we do not limit ourselves to the configurations shown in Figure 1. Instead we will introduce five sets of test problems which will help us to isolate these individual effects. In all experiments, Gaussian RBFs with shape parameter $\varepsilon = 1.2$ have been used. The RBF–QR method was employed in order to ensure stable evaluation of the matrices. It should however be noted that even if the RBF–QR method involves a change of basis, the resulting differentiation matrices are the same that would result from a direct use of the Gaussian basis if that was numerically feasible.

The first four of these test sets are for Model Problem I (on the square domain). For each test problem, the number of points (N), number of patches (P), average number of of points per patch (p), estimated condition number of L (κ, calculated using the MATLAB command `condest`), and the maximum error in the discrete solution are displayed in Table 4. Test sets

Table 4: Test sets 1–4 for Model Problem I

Cartesian nodes											
Test	N	P	p	κ	max error	Test	N	P	p	κ	max error
Test set 1						*Test set 2*					
1A	441	16	54.1	2.3e+4	2.4e−4	2A	400	25	35.6	4.8e+3	3.6e−3
1B	676	25	53.8	3.8e+4	3.7e−4	2B	576	25	46.2	1.9e+4	1.8e−4
1C	961	36	53.7	5.5e+4	2.9e−4	2C	676	25	53.8	3.8e+4	3.7e−4
1D	1024	49	42.7	2.6e+4	1.2e−4	2D	900	25	71.2	2.5e+5	4.7e−5
1E	1225	64	38.9	4.8e+4	2.3e−5	2E	1089	25	85.7	7.0e+5	1.3e−6
Halton nodes											
Test	N	P	p	κ	max error	Test	N	P	p	κ	max error
Test set 3						*Test set 4*					
3A	460	16	49.6	3.8e+6	1.2e−4	4A	436	25	30.8	3.9+e6	1.4e−2
3B	681	25	48.6	2.0e+7	1.2e−4	4B	583	25	41.2	1.0+e7	1.1e−3
3C	968	36	49.3	6.0e+7	1.7e−4	4C	681	25	48.6	2.0+e7	1.2e−4
3D	1056	49	40.1	6.1e+7	1.0e−4	4D	884	25	62.0	2.1+e8	1.3e−5
3E	1258	64	36.8	1.1e+8	2.0e−4	4E	1090	25	77.9	1.1+e9	1.6e−6

1 and 2 both use Cartesian points. In test set 1, N and P are both increased in such a way that the resulting error in the solution is kept at a similar level ($\simeq 1e-4$). This results in a set of coefficient matrices L which are similarly conditioned. Note that problem 1E corresponds to the configuration which leads to the eigenvalues in Figure 5. Test set 2 is designed to achieve different levels of accuracy by varying N while keeping P constant ($P = 25$). Here it is clear that there is a correlation between the accuracy of the discretisation and the condition of the resulting L. Test sets 3 and 4 are designed to be broadly similar, but use Halton points in place of a Cartesian lattice. The degradation of the condition number of L associated with the irregularly scattered points is apparent.

In Table 5 we display iteration counts and computational times for the different preconditioners defined in Table 1 employed on test sets 1 and 2. Let us first focus on iteration counts. As predicted by the asymptotic convergence rates in Table 2 (which were calculated

Table 5: Results for test sets 1 and 2 (Cartesian nodes, Model problem I). The lowest time for each problem is shown in bold.

Test set 1						
Test	none	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
Iteration counts						
1A	127	35	35	36	10	9
1B	165	43	43	44	12	11
1C	200	51	51	51	14	13
1D	170	52	52	52	25	8
1E	174	55	55	55	20	9
Computational time						
1A	1.8e−1	1.1e−1	3.1e−2	5.5e−2	**2.9e−2**	3.4e−2
1B	2.7e−1	9.5e−2	5.9e−2	7.8e−2	**4.1e−2**	6.6e−2
1C	4.3e−1	2.0e−1	8.4e−2	1.2e−1	**4.0e−2**	1.1e−1
1D	3.3e−1	1.3e−1	7.8e−2	1.2e−1	**3.9e−2**	9.5e−2
1E	4.4e−1	1.9e−1	1.0e−1	1.4e−1	**4.6e−2**	1.7e−1

Test set 2						
Test	none	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
Iteration counts						
2A	32	20	21	36	21	44
2B	127	38	38	38	12	7
2C	165	43	43	44	12	11
2D	170	48	49	49	21	11
2E	180	52	53	54	25	17
Computational time						
2A	6.4e−2	3.5e−2	3.2e−2	5.3e−2	**2.9e−2**	5.7e−2
2B	1.5e−1	6.1e−2	3.3e−2	7.1e−2	**1.5e−2**	3.9e−2
2C	3.5e−1	1.3e−1	6.0e−2	8.0e−2	**2.0e−2**	7.6e−2
2D	3.1e−1	1.7e−1	**1.1e−1**	2.5e−1	**1.1e−1**	1.7e−1
2E	4.1e−1	2.6e−1	9.6e−2	2.0e−1	**7.2e−2**	2.6e−1

using problem 1E), the method which converges in the smallest number of iterations is L-ILUd, followed by L-ILUn, while the three methods based on B have similar performance. As B-LU uses a full LU factorisation, iteration counts for this preconditioner give a lower bound on what can be expected from B-ILUn and B-ILUd. It can be seen that, as we have carefully numbered patches and points to ensure that the central band that is used to form B is relatively dense, very little is lost in replacing LU by an incomplete version. As well as considering iteration counts, however, it is of course important to consider the computational time taken by each method. Those are also listed in Table 5, and include the time taken for factorisation (where applicable) and the full GMRES solve. Using this measure, the methods of choice are the factorisations based on ILUn with, for these Cartesian examples, L-ILUn being slightly faster that the banded version, B-ILUn.

The equivalent results for test sets 3 and 4 (with Halton nodes) are shown in Table 6. As expected, the poorer conditioning of these matrices is reflected in increased iteration counts for all of the methods. In particular, the performance of L-ILUn degrades completely (cf. the eigenvalues for problem 3E shown in Figure 6(e)). This catastrophic behaviour appears to be restricted to the case with no fill-in only: allowing additional levels of fill-in brings the iteration counts more in line with the Cartesian results. However, as discussed above, this

Table 6: Results for test sets 3 and 4 (Halton nodes, Model Problem I). The lowest time for each problem is shown in bold.

Test set 3						
Test	none	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
Iteration counts						
3A	156	52	58	54	97	19
3B	231	100	112	106	167	49
3C	297	114	135	123	159	65
3D	310	108	118	115	202	60
3E	355	161	174	166	310	75
Computational time						
3A	2.5e−1	1.0e−1	**5.4e−2**	1.9e−1	2.2e−1	1.7e−1
3B	5.5e−1	3.0e−1	**1.6e−1**	2.6e−1	3.1e−1	4.3e−1
3C	8.6e−1	3.9e−1	**3.1e−1**	4.1e−1	3.4e−1	8.5e−1
3D	9.5e−1	3.4e−1	**2.1e−1**	3.5e−1	6.3e−1	7.9e−1
3E	1.4e−0	6.3e−1	**4.2e−1**	6.2e−1	1.1e−0	1.1e−0

Test set 4						
Test	none	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
Iteration counts						
4A	189	72	72	72	79	16
4B	209	80	91	84	147	34
4C	231	100	112	106	167	49
4D	262	107	125	124	176	66
4E	295	120	135	172	368	152
Computational time						
4A	3.1e−1	1.1e−1	**1.0e−1**	1.8e−1	1.6e−1	1.6e−1
4B	4.6e−1	2.7e−1	**1.9e−1**	2.5e−1	2.4e−1	2.3e−1
4C	4.9e−1	3.0e−1	**1.8e−1**	3.0e−1	4.2e−1	3.9e−1
4D	6.5e−1	5.2e−1	**2.8e−1**	4.2e−1	4.1e−1	8.8e−1
4E	9.0e−1	6.9e−1	**3.0e−1**	8.8e−1	1.6e−0	2.8e−0

comes at the price of significantly increasing the amount of memory needed. The comparative merits of other methods in Table 6 are essentially the same as for the Cartesian example, although the reduction in iteration counts (relative to the unpreconditioned case) is less in all cases here. This is not surprising as the Halton points do not have the rigid banded substructure of the Cartesian points, which is more amenable to efficient factorisation. In addition, the bandwidth measure β in B has been tailored to the Cartesian case (where it can be easily calculated): using the same measure in the Halton case does not necessarily guarantee inclusion of all closest connections due to the lack of structure. However, the factorisations based on B still work well as preconditioners and, in terms of computational times, B-ILUn is clearly the most efficient method overall.

To conclude this section, we introduce a final set of test problems, test set 5, for Model Problem II using unstructured nodes, and $P = 50$ patches. Details of the configurations used are given in Table 7. Note that the values of N have been chosen to roughly correspond to those used in test sets 1–4. The corresponding iteration counts and computational times are shown in Table 8. Despite the irregularity of the computational domain, the results are very similar to those obtained using Halton points on the square domain, with B-ILUn again being the method of choice.

Table 7: Test set 5 for Model Problem II

Test	N	P	p	κ	max error
5A	398	50	13.2	1.6e+6	8.8e−1
5B	695	50	23.4	9.7e+5	2.8e−4
5C	994	50	33.4	1.9e+7	2.2e−4
5D	1094	50	36.9	8.6e+6	8.4e−6
5E	1292	50	43.7	1.6e+7	1.7e−6

Table 8: Results for test set 5 (Model Problem II). The lowest time for each problem is shown in bold.

Test	none	B-LU	B-ILUn	B-ILUd	L-ILUn	L-ILUd
			Iteration counts			
5A	207	67	68	67	48	14
5B	235	76	78	76	85	13
5C	279	99	119	99	195	23
5D	304	102	120	102	255	26
5E	322	133	149	132	415	53
			Computational time			
5A	2.1e−1	7.3e−2	5.8e−2	7.3e−2	3.9e−2	**3.0e−2**
5B	4.9e−1	1.3e−1	8.8e−2	1.3e−1	9.9e−2	**8.4e−2**
5C	7.5e−1	2.6e−1	**2.1e−1**	2.6e−1	4.4e−1	2.8e−1
5D	9.1e−1	2.9e−1	**2.1e−1**	2.9e−1	7.4e−1	3.7e−1
5E	1.1e−0	5.6e−1	**3.3e−1**	5.3e−1	2.0e−0	7.9e−1

7 Conclusions

In this paper, we have introduced and evaluated algebraic preconditioners for RBF-PUM discretisations. These preconditioners are free of any assumptions on the node layout or geometry of the computational domain. The only property that is used is the knowledge that there is a patch structure and that nodes can be ordered accordingly. This is important so that the preconditioner is generally applicable.

The performance of the preconditioners, as well as the conditioning of the original matrix, is negatively affected by the use of the highly unstructured nodes. However, in our experiments we do not observe any adverse effect of changing the computational domain. The preconditioner that performed best overall, and that we recommend for use, is the no fill-in incomplete factorisation of the central band, denoted by B-ILUn.

The B-ILUn preconditioner is also the most sparse of the tested preconditioners. This property becomes increasingly important when moving to larger matrix sizes and/or higher dimensional problems, in which case memory requirements become a limiting factor. How the preconditioner performs in higher dimensions will be the subject of further studies.

References

1. Axelsson, O., Neytcheva, M., Ahmad, B.: A comparison of iterative methods to solve complex valued linear algebraic systems. Numer. Algorithms **66**(4), 811–841 (2014). DOI 10.1007/s11075-013-9764-1
2. Babuška, I., Melenk, J.M.: The partition of unity method. Internat. J. Numer. Methods Engrg. **40**(4), 727–758 (1997). DOI 10.1002/(SICI)1097-0207(19970228)40:4<727::AID-NME86>3.3.CO;2-E

3. Baxter, B.J.C.: Preconditioned conjugate gradients, radial basis functions, and Toeplitz matrices. Comput. Math. Appl. **43**(3-5), 305–318 (2002). DOI 10.1016/S0898-1221(01)00288-7
4. Beatson, R.K., Cherrie, J.B., Mouat, C.T.: Fast fitting of radial basis functions: methods based on preconditioned GMRES iteration. Adv. Comput. Math. **11**(2-3), 253–270 (1999). DOI 10.1023/A:1018932227617
5. Benzi, M.: Preconditioning techniques for large linear systems: A survey. J. Comp. Phys. **182**, 418–477 (2002). DOI 10.1006/jcph.2002.7176
6. Brown, D., Ling, L., Kansa, E., Levesley, J.: On approximate cardinal preconditioning methods for solving PDEs with radial basis functions. Eng. Anal. Bound. Elem. **29**(4), 343–353 (2005). DOI http://dx.doi.org/10.1016/j.enganabound.2004.05.006
7. Cavoretto, R., De Rossi, A., Donatelli, M., Serra-Capizzano, S.: Spectral analysis and preconditioning techniques for radial basis function collocation matrices. Numer. Linear Algebra Appl. **19**(1), 31–52 (2012). DOI 10.1002/nla.774
8. Driscoll, T.A., Fornberg, B.: Interpolation in the limit of increasingly flat radial basis functions. Comput. Math. Appl. **43**(3–5), 413–422 (2002). DOI 10.1016/S0898-1221(01)00295-4
9. Driscoll, T.A., Toh, K.C., Trefethen, L.N.: From potential theory to matrix iterations in six steps. SIAM Review **40**, 547–578 (1998). DOI 10.1137/S0036144596305582
10. Embree, M.: How descriptive are gmres convergence bounds? Tech. Rep. 99/08, Oxford University Computing Laboratory Numerical Analysis (1999)
11. Farrell, P., Pestana, J.: Block preconditioners for linear systems arising from multilevel RBF collocation. MIMS EPrint 2014.18, Manchester Institute for Mathematical Sciences, School of Mathematics, The University of Manchester (2014)
12. Fasshauer, G.E.: Meshfree approximation methods with MATLAB, *Interdisciplinary Mathematical Sciences*, vol. 6. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ (2007). DOI 10.1142/6437
13. Faul, A.C., Goodsell, G., Powell, M.J.D.: A Krylov subspace algorithm for multiquadric interpolation in many dimensions. IMA J. Numer. Anal. **25**(1), 1–24 (2005). DOI 10.1093/imanum/drh021
14. Fornberg, B., Larsson, E., Flyer, N.: Stable computations with Gaussian radial basis functions. SIAM J. Sci. Comput. **33**(2), 869–892 (2011). DOI 10.1137/09076756X
15. Fornberg, B., Lehto, E., Powell, C.: Stable calculation of Gaussian-based RBF-FD stencils. Comput. Math. Appl. **65**(4), 627–637 (2013). DOI 10.1016/j.camwa.2012.11.006
16. Fornberg, B., Piret, C.: A stable algorithm for flat radial basis functions on a sphere. SIAM J. Sci. Comput. **30**(1), 60–80 (2007). DOI 10.1137/060671991
17. Fornberg, B., Wright, G.: Stable computation of multiquadric interpolants for all values of the shape parameter. Comput. Math. Appl. **48**(5-6), 853–867 (2004). DOI 10.1016/j.camwa.2003.08.010
18. Fornberg, B., Wright, G., Larsson, E.: Some observations regarding interpolants in the limit of flat radial basis functions. Comput. Math. Appl. **47**(1), 37–55 (2004). DOI 10.1016/S0898-1221(04)90004-1
19. Fornberg, B., Zuev, J.: The Runge phenomenon and spatially variable shape parameters in RBF interpolation. Comput. Math. Appl. **54**(3), 379–398 (2007). DOI 10.1016/j.camwa.2007.01.028
20. Fuselier, E., Hangelbroek, T., Narcowich, F.J., Ward, J.D., Wright, G.B.: Localized bases for kernel spaces on the unit sphere. SIAM J. Numer. Anal. **51**(5), 2538–2562 (2013). DOI 10.1137/120876940
21. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer. Math. **2**, 84–90 (1960). DOI 10.1007/BF01386213
22. Larsson, E., Fornberg, B.: A numerical study of some radial basis function based solution methods for elliptic PDEs. Comput. Math. Appl. **46**(5–6), 891–902 (2003). DOI 10.1016/S0898-1221(03)90151-9
23. Larsson, E., Fornberg, B.: Theoretical and computational aspects of multivariate interpolation with increasingly flat radial basis functions. Comput. Math. Appl. **49**(1), 103–130 (2005). DOI 10.1016/j.camwa.2005.01.010
24. Larsson, E., Heryudono, A.: A partition of unity radial basis function collocation method for partial differential equations (2015). Manuscript in preparation
25. Larsson, E., Lehto, E., Heryudono, A., Fornberg, B.: Stable computation of differentiation matrices and scattered node stencils based on Gaussian radial basis functions. SIAM J. Sci. Comput. **35**(4), A2096–A2119 (2013). DOI 10.1137/120899108
26. Liesen, J., Strakoš, Z.: Krylov subspace methods: Principles and analysis, *Numerical Mathematics and Scientific Computation*, vol. 25. Oxford University press, Oxford, UK (2013)
27. Ling, L., Kansa, E.J.: A least-squares preconditioner for radial basis functions collocation methods. Adv. Comput. Math. **23**(1–2), 31–54 (2005). DOI 10.1007/s10444-004-1809-5
28. MATLAB: version 8.3.0.532 (R2014a). The MathWorks Inc., Natick, Massachusetts (2014)
29. Meijerink, J.A., van der Vorst, H.A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. **31**, 148–162 (1977). DOI 10.1090/S0025-5718-1977-0438681-4

30. Rieger, C., Zwicknagl, B.: Sampling inequalities for infinitely smooth functions, with applications to interpolation and machine learning. Adv. Comput. Math. **32**(1), 103–129 (2010). DOI 10.1007/s10444-008-9089-0
31. Rieger, C., Zwicknagl, B.: Improved exponential convergence rates by oversampling near the boundary. Constr. Approx. **39**(2), 323–341 (2014). DOI 10.1007/s00365-013-9211-5
32. Saad, Y.: Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics (2003). DOI 10.1137/1.9780898718003
33. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput. **7**, 856–869 (1986). DOI 10.1137/0907058
34. Safdari-Vaighani, A., Heryudono, A., Larsson, E.: A radial basis function partition of unity collocation method for convection–diffusion equations arising in financial applications. J. Sci. Comp. pp. 1–27 (2014). DOI 10.1007/s10915-014-9935-9
35. Schoenberg, I.J.: Metric spaces and completely monotone functions. Ann. of Math. (2) **39**(4), 811–841 (1938). DOI 10.2307/1968466
36. Shcherbakov, V., Larsson, E.: Radial basis function partition of unity methods for pricing vanilla basket options. Tech. Rep. 2015-001, Department of Information Technology, Uppsala University (2015)
37. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM national conference, ACM '68, pp. 517–524. ACM, New York, NY, USA (1968). DOI 10.1145/800186.810616
38. Sonneveld, P., van Gijzen, M.B.: IDR(s): A family of simple and fast algorithms for solving large non-symmetric linear systems. SIAM J. Sci. Comput. **31**, 1035–1062 (2008). DOI 10.1137/070685804
39. von Sydow, L., Höök, L.J., Larsson, E., Lindström, E., Milovanović, S., Persson, J., Shcherbakov, V., Shpolyanskiy, Y., Sirén, S., Toivanen, J., Waldén, J., Wiktorsson, M., Giles, M.B., Levesley, J., Li, J., Oosterlee, C.W., Ruijter, M.J., Toropov, A., Zhao, Y.: BENCHOP—The BENCHmarking project in Option Pricing (2015). Submitted
40. van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**, 631–644 (1992). DOI 10.1137/0913035
41. Wendland, H.: Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. Adv. Comput. Math. **4**(4), 389–396 (1995). DOI 10.1007/BF02123482
42. Wendland, H.: Fast evaluation of radial basis functions: methods based on partition of unity. In: Approximation theory, X (St. Louis, MO, 2001), Innov. Appl. Math., pp. 473–483. Vanderbilt Univ. Press, Nashville, TN (2002)