An adaptive meshfree diffusion wavelet method for partial differential equations on the sphere

Kavita Goyal, Mani Mehra *
Indian Institute of Technology Delhi, India

ARTICLE INFO
Article history:
Received 18 March 2013
Received in revised form 11 January 2014
Accepted 27 April 2014
Available online 6 May 2014

Keywords:
Multiresolution analysis (MRA)
Diffusion wavelet
Adaptive node arrangement
Numerical method

ABSTRACT
An adaptive meshfree diffusion wavelet method for solving partial differential equations (PDEs) on the sphere is developed. Approximation formulae for Laplacian–Beltrami ($\nabla^2$) and gradient ($\vec{\nabla}$) operators are derived using radial basis functions (RBFs), and the convergence of these approximations to $\nabla^2$ and $\vec{\nabla}$ is verified for two test functions. The matrix approximating the Laplace–Beltrami operator is used for the construction of the diffusion wavelet. The diffusion wavelet is used for the adaptation of node arrangement as well as for the fast computation of dyadic powers of the matrices involved in the numerical solution of the PDE. The efficiency of the method is that the same operator is used for the construction of the diffusion wavelet and for the approximation of the differential operators. As a part of the wavelet method the behaviour of the compression error with respect to different parameters involved in the construction of the diffusion wavelet is tested. The CPU time taken by the proposed method is compared with the CPU time taken by the RBF based collocation method and it is observed that the proposed method performs better. The method is tested on three test problems namely spherical diffusion equation (linear), problem of computing a moving steep front (nonlinear) and problem of Turing patterns (system of nonlinear reaction–diffusion equations).

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

Partial differential equations (PDEs) are widely used for realistic representation of real word problems such as fluctuations in stock markets, epidemiological models, climate modelling etc. In some situations the domain of these PDEs is a general manifold [2]. Many attractive mathematical properties of wavelets (namely efficient multiscale decompositions, compact support, vanishing moments, and the existence of fast wavelet transform etc.) in conjunction with the techniques for preconditioning and compression of operators and matrices, have motivated their use for numerical solutions of PDEs. Wavelet methods have been developed for most of the linear PDEs such as Laplace/Poisson equations [8] and advection diffusion problems [29]. Also for nonlinear PDEs, there exists a large spectrum of wavelet methods, which have been applied to Burger’s equation [27,41], reaction–diffusion equations [18] and Stokes equation [11].

While solving a PDE numerically one can either choose to work on a static node arrangement [46] constructed at the beginning of the computation or can opt for an adaptive node arrangement which will keep on modifying itself according to the numerical solution of PDE at different times [32,3,23,1]. With a static node arrangement we need a large set of node points to discover all the features of the solution but this will increase the computational as well as storage cost.

* Corresponding author.
E-mail addresses: kavita@maths.iitd.ac.in (K. Goyal), mmehra@maths.iitd.ac.in (M. Mehra).

http://dx.doi.org/10.1016/j.jcp.2014.04.044
0021-9991/© 2014 Elsevier Inc. All rights reserved.
In some cases the set required for a static node arrangement to capture all the features of the solution may exceed the practical limitations. To deal with these problems we work on an adaptive node arrangement. In case of an adaptive node arrangement, instead of taking a larger set of node points, more node points are added only in the areas where the solution of PDE possesses sharp variations. Computational and storage costs will be saved by using an adaptive node arrangement.

In recent years many wavelet based adaptive numerical methods for solving PDEs were developed [27,22,6,33] on the manifolds with zero curvature. The critical issue is to solve PDEs on general manifolds. Many techniques have been developed to construct wavelets on general manifolds. For example:

1. In [5,9], wavelet bases are constructed on a special type of manifolds which can be represented as a disjoint union of smooth parametric images of a standard cube. The construction is based solely on smooth parametrization of the unit cube, which has several disadvantages from a practical point of view.
2. The above problem is resolved in [10], where wavelet basis functions are constructed from an initial finite element discretization of the domain.
3. In [17], spherical Shannon wavelets are constructed that form an orthogonal multiresolution analysis on the sphere.
4. The second generation wavelet is developed by Sweldens and co-workers in [37] and the main advantage of this wavelet is that it can be custom designed for general manifolds.

Despite of vast literature of wavelets on general manifolds, the wavelet theory for numerical solutions of PDEs on general manifolds is still in its nascent stage. The second generation wavelet led to wavelet based solutions of PDEs initially in [41], where one-dimensional Burger’s equation, modified Burger’s equation and laminar diffusion frame problems are solved. This paper was a major turning point for the people using wavelets for numerical solutions of PDEs because second generation wavelet has many practical advantages over the existing wavelets. This wavelet was later used to solve PDEs on the sphere in [28], where a dynamic adaptive numerical method for solving PDEs on the sphere is developed using second generation spherical wavelet. The main difficulty with second generation wavelet is that we require an initial mesh structure to approximate the manifold (e.g. sphere can be approximated using an icosahedron mesh). However, generating an initial mesh for an arbitrary manifold is difficult.

This difficulty can be handled with meshfree methods. Meshfree methods are formulated based on a set of scattered nodes and mesh-related difficulties are avoided as no mesh is used. Hitherto, the meshfree methods based on wavelets are very less developed and to best of our knowledge, the developed wavelet-meshfree methods are limited to flat geometry [21,45,43]. In this paper, we develop an adaptive meshfree diffusion wavelet method on the sphere (AMDWMS).

The diffusion wavelet was introduced by Coifman and his co-workers in 2006 [7]. The important features of this wavelet are that it can be constructed on general manifolds and its construction does not require an initial mesh discretizing the manifold. These features make diffusion wavelet a suitable choice for meshfree methods on general manifolds.

The AMDWMS developed in this paper can solve PDEs on an adaptive node arrangement on the sphere and can be generalized to general manifolds. AMDWMS will use RBFs for interpolation of functions and for approximation of differential operators. The theory of interpolation of continuous functions by RBFs is well understood [24,30,4,34]. In 1990, E.J. Kansa introduced the concept of solving PDEs using RBFs [25,26]. Since then RBFs are continuously used for solving PDEs on manifolds of zero curvature [16,15,12,38], but RBFs are not widely used to solve PDEs on general manifolds. To best of our knowledge the only work done is by Q.T.L. Gia [20,19], where Wendland’s RBFs are used in approximation of elliptic and parabolic PDEs on the sphere. In their work, the initial conditions for the PDEs are chosen in such a way that the Laplace–Beltrami and gradient operators are reduced to a very simple form. Therefore approximation formulae for these operators are not derived in their work. In the present paper we have obtained the approximation formulae for Laplace–Beltrami and gradient operator on the unit sphere and unit cube (in most generalised form) using Wendland’s RBFs.

The beauty of the proposed method is that the approximation of Laplace–Beltrami operator obtained using RBFs is used both for the construction of the diffusion wavelet as well as for the approximation of the differential operator involved in PDEs. The diffusion wavelet is used for adapting node arrangement and for the fast computation of the dyadic powers of the matrices involved in the numerical solutions of the PDEs.

The technique in this paper somehow resembles with the technique used in [14], where PDEs on an interval with periodic boundary conditions are considered. In the review paper [35] the wavelet methods for solving PDEs are classified into following categories

1. Pure wavelet methods.
3. Lagrangian wavelet methods.
4. Space–time wavelet methods.
5. Wavelet optimized adaptive methods.

The proposed method will obviously fit in the second category. Moreover, it will also fit in the last category as the method uses nontensorial wavelet viz. diffusion wavelet to solve the PDEs on a general manifold.
The paper is organised as follows: Section 2 gives a brief description of the diffusion wavelet. In Section 3, different tools required in AMDWMS for numerical solutions of PDEs are developed. Section 4 contains the numerical results of the AMDWMS. In this section three test problems namely spherical diffusion equation, problem of computing a moving steep front and famous problem of Turing patterns are solved. Section 5 concludes the paper and gives a brief idea of the future work.

2. A brief description of diffusion wavelet

Diffusion wavelet [7] is constructed on any general manifold \( X \). Multiresolution analysis (MRA) is built using a diffusion operator \( T \) on \( L_2(X) \) which satisfies following conditions:

- \( T \) is local, i.e. \( T(\delta_k) \), where \( \delta_k \) is a mollification of the Dirac \( \delta \)-function at \( k \in X \) \( (X = \{x_1, x_2, \ldots, x_N\} \) is a discretization of \( X \) using \( N \) points), has small support.
- High powers of \( T \) have low numerical rank \( \text{Ran}_T \), where \( \text{Ran}_T \) is defined as follows:
  - Let \( H \) be a Hilbert space and \( V \subseteq H \), then a subset \( \{\xi_i\}_{i \in K} \) of \( V \) \( \tau \)-spans \( V \) if for each \( v \in V \); \( \|P_{\{\xi_i\}_{i \in K}} v - v\|_H \leq \tau \). The \( \tau \) dimension of \( V \) and \( \text{Ran}_T(T) \) are defined as
    \[
    \dim_{\tau}(V) = \inf\{\dim(V) : V \text{ is an } \tau \text{-span of } V\}, \quad \text{Ran}_T(T) = \dim_{\tau}(\text{range}(T)).
    \]
- \( T \) is a self-adjoint operator.

For example \( I - T \) with \( I \) as an identity operator on \( L_2(X) \) could be the Laplace–Beltrami operator.

2.1. Multiresolution analysis

MRA is a natural technique for the construction of a wavelet. The goal of MRA is to express an arbitrary function \( f \in L_2(X) \) at various levels of detail [31]. For a classical wavelet, MRA of \( L_2(X) \) is a sequence \( \{V^j\}_{j \in \mathbb{Z}} \) satisfying following axioms:

1. \( \emptyset \subseteq \cdots \subseteq V^{-1} \subseteq V^0 \subseteq V^1 \subseteq \cdots \subseteq L_2(X) \).
2. \( \bigcup_{j \in \mathbb{Z}} V^j = L_2(X) \).
3. Invariance to dilations, i.e. \( f \in V^j \) iff \( f(2^j \cdot) \in V^{j+1} \).
4. Invariance to translations, i.e. \( \{\phi^j_k \text{ (scaling function)} = \phi(x - k) \mid k \in \mathbb{Z}\} \) is an orthonormal basis for \( V^0 \).

2.1.1. Construction of MRA for diffusion wavelet

A precision \( \tau > 0 \) is fixed. Following notations are used:

- \( [T]_{B_1}^{B_2} \): Matrix representing the operator \( T \) with respect to the basis \( B_1 \) in the domain and basis \( B_2 \) in the range.
- \( [B_1]_{B_2} \): Matrix of transition from basis \( B_2 \) to \( B_1 \).

- **Construction of the space \( V^j \)**: Let \( \phi^j = \{\delta_k \}_{k \in X^j} \) (here \( \delta_k \) is an \( N \times 1 \) vector having 1 at \( k \)th place and 0 otherwise), then space \( V^j \) is defined as:
  \[
  V^j = \text{span}\{\phi^j\}.
  \]

- **Construction of the space \( V^{j-1} \)**: Columns of \( [T]\phi^j \) are set of functions \( \phi^{j-1} = \{T\delta_k \}_{k \in X^j} \). A local multiscale orthogonalization procedure [7] is used to orthonormalize these columns to get a basis \( \phi^{j-1} = \{\phi^{j-1}_k \}_{k \in X^{j-1}} \) written with respect to the basis \( \phi^j \), of the range of \( T \) up to a precision of \( \tau \).
  \[
  [T]\phi^j \xrightarrow{\text{Orthogonalization}} [\phi^{j-1}]_{\phi^j}.
  \]

From the basis \( \phi^{j-1} \) we get the space \( V^{j-1} = \text{span}(\phi^{j-1}) = \text{range}_\tau(T) \), where \( \text{range}_\tau(T) \) is a space which is \( \tau \) close to the range of the operator \( T \). It is clear that \( |X^{j-1}| \leq |X^j| \).

- **Construction of the space \( V^{j-2} \)**: Using the matrices \( [T]\phi^j \) and \( [\phi^{j-1}]_{\phi^j} \), the matrix \( [T]\phi^{j-1} \) is obtained as follows:
  \[
  [T]\phi^{j-1} = [T]\phi^j ([\phi^{j-1}]_{\phi^j}).
  \]

Having computed \( [T]\phi^{j-1} \), \( [T^2]\phi^{j-1} \) is computed as follows
\[
[T^2]\phi^{j-1} = [\phi^{j-1}]_{\phi^j} (T^2)[\phi^j ([\phi^{j-1}]_{\phi^j})) = [T]\phi^{j-1} ((T)\phi^{j-1})^*.
\]
The last equality holds because the operator \( T \) is self-adjoint. Columns of the matrix \([T^2]\Phi_{j-1}\) are set of the functions \( \Phi_{j-1}^T = [T^2]^{\phi_{j-1}^T}_{k}\). Again the local multiscale orthogonalization procedure is used to orthonormalize these columns to get a basis \( \Phi_{j-2} = [\phi_{j-2}^T]_{k\in X} \) and \( \Phi_{j-1}\), of the range of \( T^{2+1} \) up to a precision of \( \varepsilon \). From the basis \( \Phi_{j-2} \) we get the space \( \mathcal{V}_{j-2} = \text{span}[\Phi_{j-2}^T] = \text{range}_{\varepsilon}(T^{2+1}) \).

- **Construction of the space \( \mathcal{V}_{j-1} \):** After \( j-1 \) steps in above fashion, we will have a representation of \( T^{2j-1} \) with respect to the basis \( \Phi_{j-1} = [\phi_{j-1}^T]_{k\in X} \), encoded in the matrix \([T^{2j-1}]\Phi_{j-1}^T\). The columns of \([T^{2j-1}]\phi_{j-1}^T\) are the set of the functions \( \Phi_{j-1}^T = (T^{2j-1})^{\phi_{j-1}^T}_{k}\). The local multiscale orthogonalization procedure on this set will yield an orthonormal basis \( \Phi_{j-1}^T = [\phi_{j-1}^T]_{k\in X} \) for \( \text{range}_{\varepsilon}(T^{2j-1}) \) and hence we get the space \( \mathcal{V}_{j-1} = \text{span}[\Phi_{j-1}^T] = \text{range}_{\varepsilon}(T^{2j-1}) \).

- For the operator \( T \) it is clear that
  \[
  \cdots \subseteq \text{range}_{\varepsilon}(T^{1+2+\cdots+2^j-1}) \subseteq \cdots \subseteq \text{range}_{\varepsilon}(T) \subseteq \text{span}[\Phi] \subseteq \cdots \subseteq \mathcal{L}_2(X),
  \]
  or
  \[
  \cdots \subseteq \mathcal{V}_{j-1} \subseteq \cdots \subseteq \mathcal{V}_{j-1} \subseteq \cdots \subseteq \mathcal{L}_2(X),
  \]
  which is analogous to the axiom 1 of MRA.

- Clearly \( \bigcup_{j\in\mathbb{Z}} \mathcal{V}_j = \mathcal{L}_2(X) \), which is analogous to axiom 2 of MRA.

- Axiom 3 of MRA means that the functions in the space \( \mathcal{V}_j \) are dilations of the functions in the space \( \mathcal{V}_{j+1} \). In the construction of diffusion wavelet the operator \( T \) being the diffusion operator dilates the functions on which it is operated. The functions in \( \mathcal{V}_j \) are obtained by applying the operator \( T \) on the functions of the space \( \mathcal{V}_{j+1} \), hence the functions of the space \( \mathcal{V}_j \) are dilations of the functions in the space \( \mathcal{V}_{j+1} \).

- Also \( \Phi_j \) \( \tau \)-spans the space \( \mathcal{V}_j \), which is analogous to axiom 4 of MRA.

In this way an MRA is constructed for the space \( \mathcal{L}_2(X) \), i.e., we have constructed the approximation spaces \( \mathcal{V}_j \). The detail spaces \( \mathcal{W}_j \) are constructed in such a way that \( \mathcal{V}_j = \mathcal{V}_{j+1} \oplus \mathcal{W}_{j-1} \). The sets \( \mathcal{W}_j \) which \( \tau \)-span the spaces \( \mathcal{V}_j \) are constructed simply by continuing the orthogonalization procedure till the domain (instead of the range) of \([T^{2j}]\phi_j\) is exhausted. In practice, to preserve numerical accuracy, this is achieved by starting with the columns of \( \text{I}_j \Phi_{j-1} - \Phi_j \).

Note that we can either construct the MRA till the range of \([T^{2j}]\phi_j\) is exhausted or we can always fix a coarsest level say \( J_0 \) (in that case \( j = J_0 \) is called the depth). If the dim \( \mathcal{V}_j \) is small and does not change in moving from level \( j = j_1 \) to level \( j = j_1 - 1 \), then it is recommended that \( j_1 \) should be fixed as coarsest level.

### 2.2. Diffusion scaling function and inverse diffusion scaling function transform

For any \( f \in \mathcal{L}_2(X) \) we have \( P_{\mathcal{V}_j} f(x) = \sum_{k \in X} c_k^j \phi_k^j(x) \). The coefficients \( c_k^j \) are called the scaling function coefficients of the function \( f(x) \) at the level \( j \). Because of the orthonormality of \( \{\phi_k^j\}_{k \in X} \) we have \( c_k^j = \langle f, \phi_k^j \rangle \). Let \( c^j = (c_k^j)_{k \in X} \) so that \( c^j = (f, \Phi^j) \). The set \( (f, \Phi^j) \) is nothing but \( (\Phi^j)^T f \), where \( f = (f(x_1), f(x_2), \ldots, f(x_N))^T \), i.e.,

\[
  c^j = (\Phi^j)^T f.
\]

Eq. (1) is called diffusion scaling function transform (DST). From (1)

\[
  f = [(\Phi^j)^T]^{-1} (\Phi^j)_\phi c^j.
\]

Eq. (2) is called the inverse diffusion scaling function transform (IDST).

### 2.3. Diffusion wavelet and inverse diffusion wavelet transform

For any function \( f \in \mathcal{L}_2(X) \), \( P_{\mathcal{V}_j} f = P_{\mathcal{V}_j} f + P_{\mathcal{W}_j} f \). So we can write

\[
  (P_{\mathcal{V}_j} f)(x) = \sum_{k \in X} c_k^{j-1} \phi_k^{j-1}(x) \sum_{k \in Y^{j-1}} d_k^{j-1} \psi_k^{j-1}(x),
\]

where \( Y^{j-1} \) is the index set. Given the set \( c^j \) we want to obtain the sets \( c^{j-1} = (c_k^{j-1})_{k \in X} \) and \( d^{j-1} = (d_k^{j-1})_{k \in Y} \).

The set \( c^{j-1} \) can be obtained as follows:

\[
  c^{j-1} = (f, \Phi^{j-1}) = (\Phi^{j-1})^T f = (\Phi^{j-1})^T (\Phi^j)^T f = (\Phi^j)^T (\Phi^{j-1})^T f.
\]
The set $d^{j-1} = (f, \psi^{j-1})$ can be obtained as follows:

$$d^{j-1} = (f, \psi^{j-1}) = [\psi^{j-1}]_{\phi_j} f = [\psi^{j-1}]_{\phi_j} [\Phi^{j-1}]_{\phi_j} f = [\psi^{j-1}]_{\phi_j} c^j.$$ 

Hence we have the relations

$$c^{j-1} = [\Phi^{j-1}]_{\phi_j} c^j, \quad d^{j-1} = [\psi^{j-1}]_{\phi_j} c^j.$$ 

The above relations are termed as partial diffusion wavelet transform (PDWT). Now for the coarsest level $J_0$ and the finest level $J$, we can decompose the space $V^j$ as $V^j = V^{j_0} \bigoplus V^{j_1}$. Therefore

$$(P_{V^j} f)(x) = \sum_{k \in V^{j_0}} c_k^{j_0} \Phi_k(x) + \sum_{j=j_0}^{j_1-1} \sum_{k \in V^j} d_k^j \psi_k(x).$$  

PDWT can be applied on $c^j$ for $j = J, J - 1, \ldots, J_0 + 1$ to obtain the full diffusion wavelet transform (FDWT) which will give all the coefficients in (3).

Next we want to construct the set $c^j$ from the sets $c^{j-1}$ and $d^{j-1}$. Since $\Phi^{j-1}$ is the basis of $V^{j-1}$ and $\psi^{j-1}$ is the basis of $V^j$, $\Phi^{j-1} \cup \psi^{j-1}$ (call it $\Phi^j$) is the basis of $V^j$. The matrix $[\Phi^j]_{\phi_j}$ is given as

$$[\Phi^j]_{\phi_j} = \begin{pmatrix} [\Phi^{j-1}]_{\phi_j} & 0 \\ 0 & [\psi^{j-1}]_{\phi_j} \end{pmatrix}.$$ 

Therefore

$$c^j = \begin{pmatrix} [\Phi^{j-1}]_{\phi_j} & 0 \\ 0 & [\psi^{j-1}]_{\phi_j} \end{pmatrix}^{-1} \begin{pmatrix} c^{j-1} \\ d^{j-1} \end{pmatrix} \quad \Rightarrow \quad c^j = [\Phi^{j-1}]_{\phi_j} c^{j-1} + [\psi^{j-1}]_{\phi_j} d^{j-1}. \tag{4}$$

Eq. (4) is called inverse partial diffusion wavelet transform (IPDWT). Inverse full diffusion wavelet transform (IFDWT) is obtained by applying IPDWT recursively.

3. Adaptive meshfree diffusion wavelet method on the sphere

In this section we will explain AMDWMS to solve PDEs on the sphere. Wendland’s RBFs are used for approximating the Laplace–Beltrami and gradient operators on the sphere. The approximated Laplace–Beltrami operator will serve the purpose of the matrix $[T]_{\phi_j}$ mentioned in Section 2.

3.1. Wendland’s RBFs

Let $X$ be the unit sphere. The RBFs are derived from bizonal kernels $\Phi : X \times X \to \mathbb{R}$ of the form $\Phi(x, y) = \varphi(x \cdot y)$, $x, y \in X$. Here $\varphi$ is a univariate continuous function defined on the closed interval $[-1, 1]$ and $x \cdot y$ is the Euclidean dot product of the position vectors of the points $x, y \in X$.

For a fixed value of $x$, the value of $\Phi(x, y)$ depends only on the geodesic distance from $x$ to $y$, hence the function $\Phi(x, \cdot)$ is radially symmetric function with respect to the point $x$ and is therefore called RBF [20]. For every point $x_j \in X^j$, an RBF is defined as

$$\Phi_j(x) = \Phi(x, x_j) = \varphi(x \cdot x_j) = \varphi(|x - x_j|). \tag{5}$$

One special type of the bizonal functions is

$$\Phi(x, y) = \varphi(x \cdot y) = \sum_{l=0}^{\infty} a_l P_l(n + 1; x \cdot y), \quad a_l > 0, \quad \sum_{l=0}^{\infty} a_l < \infty,$$ 

where $\{P_l(n + 1; t)\}_{l=0}^{\infty}$ is a sequence of $(n + 1)$-dimensional Legendre polynomials. Such a $\Phi$ is positive definite on $X$ [36], i.e., the matrix $A = [\Phi_j(x_i)]_{i,j=1}^N$ is positive semi-definite for every $X^j$ and for every positive integer $N$. When the coefficients $a_l$ in (6) are positive for every $l$, $\Phi$ is said to be strictly positive definite [44]. In this case the matrix $A$ becomes positive definite and hence invertible for every $X^j$ and for every positive integer $N$. These bizonal functions will be used to derive Wendland’s RBFs.

Given a continuous function $f$ on $X$, there uniquely exists a sequence of numbers $\tilde{f}^{j_1}$ such that the function

$$I_{X^j} f(x) = \sum_{j=1}^{N} \tilde{f}^{j_1} \Phi_j(x). \tag{7}$$
satisfies the interpolating condition \( I_X f(x_k) = f(x_k), \ 1 \leq k \leq N \). Put \( x = x_1, x_2, \cdots, x_N \) in (7)

\[
\begin{bmatrix}
  f(x_1) \\
  f(x_2) \\
  \vdots \\
  f(x_N)
\end{bmatrix}
= \begin{bmatrix}
  \Phi_1(x_1) & \Phi_2(x_1) & \cdots & \Phi_N(x_1) \\
  \Phi_1(x_2) & \Phi_2(x_2) & \cdots & \Phi_N(x_2) \\
  \vdots & \vdots & \ddots & \vdots \\
  \Phi_1(x_N) & \Phi_2(x_N) & \cdots & \Phi_N(x_N)
\end{bmatrix} \begin{bmatrix}
  \hat{f}_1 \\
  \hat{f}_2 \\
  \vdots \\
  \hat{f}_N
\end{bmatrix},
\]

which can be written in matrix form as \( f = A\hat{f} \). Since the matrix \( A \) is invertible \( \hat{f} = A^{-1}f \), where \( f = [f(x_1), f(x_2), \cdots, f(x_N)]^T \) etc. Hence

\[
f(x) \approx I_X f(x) = \sum_{j=1}^{N} \tilde{f}_j \phi_j(x).
\]

There are plenty of choices for the function \( \Psi \) of (5), we will choose it as Wendland’s compactly supported RBF [42].

The function \( \Psi \) in case of Wendland’s RBF depends on two parameters \( d \) and \( k \), where \( d \) is the dimension of the space and \( 2k \) is the order of the smoothness of the function \( \Psi \). Since \( \Psi \) depends on \( d \) and \( k \), we will use the notation \( \Psi_{d,k} \) to denote it. It is defined as follows

\[
\Psi_{d,k}(r) = \begin{cases}
  p_{d,k}(r) & 0 \leq r \leq 1, \\
  0 & \text{if } r > 1,
\end{cases}
\]

where

\[
p_{d,k}(r) = \sum_{j=0}^{l+2k} d^{(j)}_{j,k} r^j,
\]

with \( l = \left[ \frac{d}{2} \right] + k + 1 \). The coefficients \( d^{(j)}_{j,k} \) can be computed recursively for \( 0 \leq s \leq k - 1 \) as follows

\[
d^{(j)}_{j,0} = (-1)^j j! \binom{l}{j}, \quad 0 \leq j \leq l,
\]

\[
d^{(j)}_{0,s+1} = \sum_{j=0}^{l+2s} d^{(j)}_{j,s} j^2, \quad s \geq 0,
\]

\[
d^{(j)}_{1,s+1} = 0, \quad s \geq 0,
\]

\[
d^{(j)}_{j,s+1} = -\frac{d^{(j-2s)}_{j-2}}{j}, \quad s \geq 0, \ 2 \leq j \leq l + 2s + 2.
\]

From (9), we obtain RBFs for the space \( \mathbb{R}^3 (d = 3) \) as

\[
\begin{align*}
\Psi_{3,0}(r) &= (1 - r)^2, \\
\Psi_{3,1}(r) &= (1 - r)^4 (4r + 1), \\
\Psi_{3,2}(r) &= (1 - r)^6 (35r^2 + 18r + 3), \\
\Psi_{3,3}(r) &= (1 - r)^8 (32r^3 + 25r^2 + 8r + 1),
\end{align*}
\]

upto a constant. Note that \( x_+^d = \max\{|x|^d, 0\} \) denotes power truncated function. Since \( \Psi_{3,0}(r) \in C^0 \) and we have to compute the Laplace–Beltrami and gradient operators, we will not use this for further calculations. Throughout this paper we use \( \Psi_{3,1}, \Psi_{3,2} \) and \( \Psi_{3,3} \). Further, observe from (5) that the argument \( r \) for the function \( \Psi \) is \( |x - x_j| \), if \( x \) and \( x_j \) are the points on the unit sphere \( X \), then \( |x - x_j| = \sqrt{2 - 2x \cdot x_j} \), therefore \( r = \sqrt{2 - 2x \cdot x_j} \).

3.1.1. Error estimates

There exist convergence estimates for interpolation error for Wendland’s functions. We will give a brief description of the results here, and for details one can see [42]. We would like to see the behaviour of the interpolation error \( f - I_X f \) as a function of the data density. Therefore the space of the functions \( f \) to be approximated or interpolated has to be specified and a measure for the data density has to be introduced. The space will depend on the basis function \( \Psi \) and will be denoted by \( \mathcal{R}_\Psi \). For a set of points \( X^l = \{x_1, x_2, \cdots, x_n\} \subset X \subset \mathbb{R}^d \), a measure called density measure is defined as

\[
h = \sup_{x \in X} \min_{1 \leq i \leq N} ||x - x_i||_2.
\]
We have the following results:

1. For Wendland’s function $\Psi_{d,k}$ (with $k \geq 1$ for $d = 1, 2$), the space $\mathcal{R}_{d,k}$ coincides with Sobolev space $H^s(\mathbb{R}^d)$ with $s = \frac{d}{2} + k + \frac{1}{2}$ and the space norm is equivalent to the Sobolev norm.

2. Let $s = \frac{d}{2} + k + \frac{1}{2}$ and $k \geq 1$ for $d = 1, 2$. For every $f \in H^s(\mathbb{R}^d)$ and every compact $X \subset \mathbb{R}^d$, the interpolant $I_X f$ satisfies the estimate

$$\|f - I_X f\|_\infty \leq C \|f\|_{H^s(\mathbb{R}^d)} h^{k + \frac{1}{2}},$$  \hspace{1cm} (11)

with $h$ sufficiently small.

3. Since we are working on the unit sphere, so we have $d = 3 \implies s = 2 + k$. Therefore, we have the following error estimates

$$\|f - I_X f\|_\infty \leq C \|f\|_{H^s(\mathbb{R}^d)} h^{\frac{3}{2}},$$  \hspace{1cm} (12)

$$\|f - I_X f\|_\infty \leq C \|f\|_{H^s(\mathbb{R}^d)} h^{\frac{5}{2}},$$  \hspace{1cm} (13)

and

$$\|f - I_X f\|_\infty \leq C \|f\|_{H^s(\mathbb{R}^d)} h^{\frac{7}{2}},$$  \hspace{1cm} (14)

for $k = 1, 2$ and 3 respectively.

If the interpolation error is defined as $E^{(k)}(f) = \|f - I_X f\|_\infty$, then we expect that $E^{(k)}(f) = O(N^{-(k + \frac{1}{2})})$ (from (11)). We considered $f(x) = c$ (with $x = (a, b, c)$) shown in Fig. 1(a) as our first test function. Fig. 1(b) shows the graph of the interpolation error ($E^{(k)}(f)$) versus $N$ for different values of $k$ for this function. It can be observed from the graph that we obtained the expected order of convergence. As a second test function we considered a localized function $f(x) = \frac{4}{5^2} e^{-\frac{(x_1^2 + x_2^2 + x_3^2)}{2}}$ shown in Fig. 1(c). Fig. 1(d) shows the variation of $E^{(k)}(f)$ with $N$ for different values of $k$ for this test function.
3.2. Approximation of the Laplace–Beltrami operator on the sphere using RBFs

Applying the Laplace–Beltrami operator on both sides of (8)

\[ \nabla^2 f(x) \approx \sum_{j=1}^{N} \tilde{f}^j \nabla^2 \Phi_j(x) = B \mathbf{f} = B \mathbf{A}^{-1} \mathbf{f}, \quad B = [\nabla^2 \Phi_j(x_i)]_{i,j=1}^N. \]

In this way the matrix \( B \mathbf{A}^{-1} \) is an approximation of the Laplace–Beltrami operator \( \nabla^2 \) on the sphere.

3.2.1. Computation of the entries of the matrix \( B \)

The \( ij \)th entry of the matrix \( B \) is

\[ \nabla^2 \Phi_j(x_i) = \nabla^2 \Phi(x_i, x_j) = \nabla^2 \varphi(x_i \cdot x_j) = \nabla^2 \psi(\sqrt{x_i \cdot x_j}) = \nabla^2 \psi(\sqrt{2 - 2x_i \cdot x_j}). \]

For a fixed \( x_j = (a, b, c) \) with \( a^2 + b^2 + c^2 = 1 \), we calculate \( \nabla^2 \psi(\sqrt{2 - 2x_i \cdot x_j}) \). If \( x = (y_1, y_2, y_3) \) then

\[ \nabla^2 \psi(\sqrt{2 - 2x_i \cdot x_j}) = \nabla^2 \psi(\sqrt{2 - 2(ay_1 + by_2 + cy_3)}). \]

In spherical polar coordinates with radial distance \( r \), polar angle \( \theta_1 \) and azimuthal angle \( \theta_2 \), we have

\[ y_1 = R \sin \theta_1 \cos \theta_2, \]
\[ y_2 = R \sin \theta_1 \sin \theta_2, \]
\[ y_3 = R \cos \theta_1. \]

On the unit sphere \( R = 1 \), hence \( y_1 = \sin \theta_1 \cos \theta_2, \ y_2 = \sin \theta_1 \sin \theta_2, \ y_3 = \cos \theta_1 \). Therefore, we need to compute \( \nabla^2 \psi(\sqrt{2 - 2(\sin \theta_1 \cos \theta_2 + b \sin \theta_1 \sin \theta_2 + c \cos \theta_1)}) \). The Laplace–Beltrami operator \( \nabla^2 \psi \) in spherical polar coordinates on the unit sphere is given by

\[ \nabla^2 \psi = \frac{\partial^2 \psi}{\partial \theta_1^2} + \frac{\cos \theta_1}{\sin \theta_1} \frac{\partial \psi}{\partial \theta_1} + \frac{1}{\sin^2 \theta_1} \frac{\partial^2 \psi}{\partial \theta_2^2}. \]

Let us start with \( \psi = \psi_{3,1} \) and with this choice

\[ \psi(\theta_1, \theta_2) = (1 - r)^4 (4r + 1), \]
\[ r = \sqrt{2 - 2t}, \]
\[ t = a \sin \theta_1 \cos \theta_2 + b \sin \theta_1 \sin \theta_2 + c \cos \theta_1. \]

Hence

\[ \frac{\partial \psi}{\partial r} = \begin{cases} -20r(1 - r)^3 & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]
\[ \frac{\partial \psi}{\partial t} = \frac{-2}{2\sqrt{2 - 2t}} = \frac{-1}{r}, \]
\[ \frac{\partial \psi}{\partial \theta_1} = \frac{\partial \psi}{\partial r} \frac{\partial r}{\partial \theta_1} + \frac{\partial \psi}{\partial t} \frac{\partial t}{\partial \theta_1} = \begin{cases} 20(1 - r)^3 \frac{\partial t}{\partial \theta_1} & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]
\[ \frac{\partial^2 \psi}{\partial \theta_1^2} = \begin{cases} 60(1 - r)^2 \left( \frac{\partial t}{\partial \theta_1} \right)^2 + 20(1 - r)^3 \frac{\partial^2 t}{\partial \theta_1^2} & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]
\[ \frac{\partial \psi}{\partial \theta_2} = \frac{\partial \psi}{\partial r} \frac{\partial r}{\partial \theta_2} + \frac{\partial \psi}{\partial t} \frac{\partial t}{\partial \theta_2} = \begin{cases} 20(1 - r)^3 \frac{\partial t}{\partial \theta_2} & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]
\[ \frac{\partial^2 \psi}{\partial \theta_2^2} = \begin{cases} 60(1 - r)^2 \left( \frac{\partial t}{\partial \theta_2} \right)^2 + 20(1 - r)^3 \frac{\partial^2 t}{\partial \theta_2^2} & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases}. \]
Insert the values from (16), (17) and (19) into (15) to obtain

\[ \nabla^2 \psi = \begin{cases} 20(1 - r)^3 (\frac{\partial^2}{\partial r^2} + \frac{\cos \theta_1}{\sin \theta_1} \frac{\partial}{\partial \theta_1} + \frac{1}{\sin \theta_1} \frac{\partial^2}{\partial \theta_2^2}) + 60 \frac{(1 - r)^2}{r^2} ((\frac{\partial}{\partial r})^2 + \frac{1}{\sin \theta_1} (\frac{\partial}{\partial \theta_2})^2) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]  

(20)

Next

\[ \frac{\partial t}{\partial \theta_1} = a \cos \theta_1 \cos \theta_2 + b \cos \theta_1 \sin \theta_2 - c \sin \theta_1, \]  

(21)

\[ \frac{\partial^2 t}{\partial \theta_2^2} = -a \sin \theta_1 \cos \theta_2 - b \sin \theta_1 \sin \theta_2 - c \cos \theta_1 = -t, \]  

(22)

\[ \frac{\partial t}{\partial \theta_2} = -a \sin \theta_1 \sin \theta_2 + b \sin \theta_1 \cos \theta_2. \]  

(23)

\[ \frac{\partial^2 t}{\partial \theta_2^2} = -a \sin \theta_1 \cos \theta_2 - b \sin \theta_1 \sin \theta_2. \]  

(24)

Insert (23) and (24) into (20) to obtain, for \( r < 1 \),

\[ \nabla^2 \psi = 20(1 - r)^3 (r^2 - 2) + 60 \frac{(1 - r)^2 r^2 (4 - r^2)}{4} \]

\[ = 20(1 - r)^3 (r^2 - 2) + 15r(1 - r)^2 (4 - r^2) \]

\[ = 5(1 - r)^2 (-7r^3 + 4r^2 + 20r - 8), \]

and 0 otherwise. Therefore we have

\[ \nabla^2 \psi_{2,1} = \begin{cases} 5(1 - r)^2 (-7r^3 + 4r^2 + 20r - 8) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]  

(25)

Moving on similar lines, one can obtain

\[ \nabla^2 \psi_{3,2} = \begin{cases} 28(1 - r)^4 (-25r^4 + 8r^3 + 82r^2 - 16r - 4) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]  

(26)

and

\[ \nabla^2 \psi_{3,3} = \begin{cases} 22(1 - r)^6 (-52r^5 + 3r^4 + 182r^3 + 7r^2 - 12r - 2) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]  

(27)

If the error is defined as \( E^{(k)}(\nabla^2 f) = \frac{\| \nabla^2 f - D^{(2)} f \|_\infty}{\| \nabla^2 f \|_\infty} \), then we expect that \( E^{(k)}(\nabla^2 f) = O(N^{-k \frac{1}{2}}) \) (from (11)). Figs. 2(a) and 2(b) show the convergence of \( D^{(2)} \) to the Laplace–Beltrami operator \( \nabla^2 \) for the two test functions. We obtained the desired order of convergence.
3.3. Approximation of the gradient operator on the sphere using RBFs

Applying gradient operator on both sides of (8)

\[ \tilde{\nabla} f(x) \approx \sum_{j=1}^{N} \tilde{f} \tilde{\nabla} \Phi_j(x) = C \mathbf{A}^{-1} \mathbf{f}, \quad C = [\tilde{\nabla} \Phi_j(x_j)]_{i,j=1}^{N}. \]

It is clear that the matrix \( \mathbf{D}^{(1)} = \mathbf{C \mathbf{A}^{-1}} \) approximates the gradient operator \( \tilde{\nabla} \) on the sphere.

3.3.1. Computation of the entries of the matrix \( C \)

The \( ij \)th entry of the matrix \( C \) is

\[ C_{ij} = \tilde{\nabla} \Phi_j(x_i) = \tilde{\nabla} \Psi(|x_i - x_j|) = \tilde{\nabla} \Psi(\sqrt{2 - 2x_i \cdot x_j}). \]

For a fixed \( x_j = (a, b, c) \) with \( a^2 + b^2 + c^2 = 1 \), we calculate \( \tilde{\nabla} \Psi(\sqrt{2 - 2x_i \cdot x_j}) = \tilde{\nabla} \Psi(\sqrt{2 - 2(ay_1 + by_2 + cy_3)}) = \tilde{\nabla} \Psi(\sqrt{2 - 2(a \sin \theta_1 \cos \theta_2 + b \sin \theta_1 \sin \theta_2 + c \cos \theta_1)}). \)

The gradient of a function \( \Psi \) in spherical polar coordinates on the unit sphere is given by

\[ \tilde{\nabla} \Psi = \frac{\partial \Psi}{\partial \theta_1} \hat{\theta}_1 + \frac{1}{\sin \theta_1} \frac{\partial \Psi}{\partial \theta_2} \hat{\theta}_2, \]

where \( \hat{\theta}_1 \) and \( \hat{\theta}_2 \) are the unit vectors in \( \theta_1 \) and \( \theta_2 \) directions respectively. We will start with \( \Psi = \psi_{3,1} \). Using (16) and (18), we get

\[ \tilde{\nabla} \Psi = \begin{cases} 20(1 - r)^3 \left( \frac{\partial}{\partial \theta_1} \hat{\theta}_1 + \frac{1}{\sin \theta_1} \frac{\partial}{\partial \theta_2} \hat{\theta}_2 \right) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]

Using (21) and (22) in (29), we get

\[ \tilde{\nabla} \Psi = \begin{cases} 20(1 - r)^3 \left( a \cos \theta_1 \cos \theta_2 \hat{\theta}_1 - \sin \theta_2 \hat{\theta}_2 \right) + b \cos \theta_1 \sin \theta_2 \hat{\theta}_1 + \cos \theta_2 \hat{\theta}_2 - c \sin \theta_1 \hat{\theta}_1) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \]

The relations between the unit vectors in Cartesian coordinates, i.e., \( \hat{i}, \hat{j}, \hat{k} \) and the unit vectors in spherical polar coordinates, i.e., \( \hat{\theta}_1, \hat{\theta}_2 \) are

\[ \hat{i} = \sin \theta_1 \cos \theta_2 \hat{R} + \cos \theta_1 \cos \theta_2 \hat{\theta}_1 - \sin \theta_1 \hat{\theta}_2, \]
\[ \hat{j} = \sin \theta_1 \sin \theta_2 \hat{R} + \cos \theta_1 \sin \theta_2 \hat{\theta}_1 + \cos \theta_1 \hat{\theta}_2, \]
\[ \hat{k} = \cos \theta_1 \hat{R} - \sin \theta_1 \hat{\theta}_1, \]
\[ \hat{R} = \sin \theta_1 \cos \theta_2 \hat{i} + \sin \theta_1 \sin \theta_2 \hat{j} + \cos \theta_1 \hat{k}, \]
\[ \hat{\theta}_1 = \cos \theta_1 \cos \theta_2 \hat{i} + \cos \theta_1 \sin \theta_2 \hat{j} - \sin \theta_1 \hat{k}, \]
\[ \hat{\theta}_2 = -\sin \theta_2 \hat{i} + \cos \theta_2 \hat{j}. \]
Fig. 3. Error while computing gradient operator for $f(x) = c$.

Fig. 4. Error while computing gradient operator for $f(x) = \frac{4}{5} e^{-\frac{r^2}{2}}$.

Using these relations we obtain, for $r < 1$

\[
\tilde{\nabla} \psi(\sqrt{2 - 2x \cdot x_j}) = 20(1 - r)^3 (a \hat{i} + b \hat{j} + c \hat{k} - t \hat{R}) \\
= 20(1 - r)^3 (x_j - t \sin \theta_1 \cos \theta_2 \hat{i} + \sin \theta_1 \sin \theta_2 \hat{j} + \cos \theta_1 \hat{k}) \\
= 20(1 - r)^3 (x_j - t \tilde{x}).
\]

where $x_j = a \hat{i} + b \hat{j} + c \hat{k}$ and $\tilde{x} = y_1 \hat{i} + y_2 \hat{j} + y_3 \hat{k}$ and 0 otherwise. Therefore, we have

\[
\tilde{\nabla} \psi_{3,1} = \begin{cases} 
20(1 - r)^3 (x_j - t \tilde{x}) & \text{if } r < 1 \\
0 & \text{otherwise} 
\end{cases}.
\]

In a similar way one can obtain

\[
\tilde{\nabla} \psi_{3,2} = \begin{cases} 
56(1 - r)^5 (5r + 1)(x_j - t \tilde{x}) & \text{if } r < 1 \\
0 & \text{otherwise} 
\end{cases},
\]

and

\[
\tilde{\nabla} \psi_{3,3} = \begin{cases} 
22(1 - r)^7 (16r^2 + 7r + 1)(x_j - t \tilde{x}) & \text{if } r < 1 \\
0 & \text{otherwise} 
\end{cases}.
\]

If the error is defined as $E^{(k)}((\tilde{\nabla} f)_a) = \frac{1}{\|\tilde{\nabla} f_a\|_2} \|\tilde{\nabla} f_a - (\nabla \psi)_a\|_2$, then we expect that $E^{(k)}((\tilde{\nabla} f)_a) = O(N^{-(k-\frac{1}{2})})$ (from (11)). Note that here $((\tilde{\nabla} f)_a$ stands for component of the vector $\tilde{\nabla} f$ in the direction of the unit vector $\hat{i}$ etc. Fig. 3 shows the convergence of $\nabla^{(1)}$ to the gradient operator $\nabla$ for $f(x) = c$. Fig. 4 shows the convergence for $f(x) = \frac{4}{5} e^{-\frac{r^2}{2}}$. The expected order of convergence is achieved for both the functions.

3.4. Approximation of the Laplace–Beltrami and gradient operator on the unit cube using RBFs

To show that the proposed method is meshfree and allows arbitrary manifold to be used, we demonstrate that the order of accuracy is independent of the choice of the manifold. In that direction, we considered the unit cube. We computed $\nabla^2 \psi$ and $\nabla \psi$ which are required for computation of the entries of the matrices $A$ and $B$. 
3.5. Reconstruction and compression error

For any $f(x) \in L_2(X)$ and a given threshold $\epsilon$, (3) can be written as $P_{\triangledown j} f(x) = f_{\geq \epsilon}(x) + f_{< \epsilon}(x)$, where

$$f_{\geq \epsilon}(x) = \sum_{k \in X^j} c_k^{j_0} d_k^{j_0}(x) + \sum_{j=j_0}^{j-1} \sum_{|d_k^j| \geq \epsilon} d_k^j \psi_k^j(x) \quad \text{and} \quad f_{< \epsilon}(x) = \sum_{j=j_0}^{j-1} \sum_{|d_k^j| < \epsilon} d_k^j \psi_k^j(x).$$

The number of significant coefficients $N(\epsilon)$ is defined as $N(\epsilon) = \# X^{j_0} + \sum_{j=j_0}^{j-1} \# \{|k| \in \mathcal{Y}^j \text{ and } |d_k^j| \geq \epsilon\}$. $\|f - f_{\geq \epsilon}\|_p$ is called compression error and for $\epsilon = 0$ there is no compression and hence it is reconstruction error. Donoho in [13] proved...
that for sufficiently smooth $f$

$$\|f - f_{\geq \epsilon}\|_{\infty} < C\epsilon,$$

where $C$ is a constant. Note that wavelet coefficients $d^k_l$ can be arranged in decreasing order and a fraction of coefficients from the top can be used to reconstruct the function. If $d$ is the fraction of the top wavelet coefficients used for reconstruction then it means that absolute value of $(d + 1)$th wavelet coefficient is used as threshold $\epsilon$.

- **Test function 1** $f(x) = c$. For $N = 2^{11} = 2048$ the finest level is $J = 11$. Following results are obtained:
  - Reconstruction error is of the order $10^{-14}$.
  - Fig. 9(a) shows the relation between the number of significant coefficients $N(\epsilon)$ and compression error $\|f - f_{\geq \epsilon}\|_p$. A good compression can be observed from the graph. Fig. 9(b) shows the graph between the compression error $\|f - f_{\geq \epsilon}\|_p$ and the threshold $\epsilon$. It can be observed that the estimate (31) is verified.
  - Fig. 9(c) shows the relation between the compression error $\|f - f_{\geq \epsilon}\|_p$ and the precision $\tau$ used in the construction of the wavelet (75% wavelet coefficients are used for reconstruction).
It is to be noted that for large value of $\tau$ the decay in the dimension of $V_j$ will be fast. The fast decay in dimension of $V_j$ means the computational cost will be less. It can be seen from Fig. 9(c) that the compression error does not change much with the change in the value of $\tau$, hence one can choose high value of $\tau$.

- **Test function 2** $f(x) = \frac{4}{5} e^{-\left(a-1^2+b^2+c^2\right)/5^2}, v = \frac{1}{2\pi^2}$. The following results are obtained:
  - Reconstruction error is of the order $10^{-14}$.
  - Fig. 10(a) shows the compression error as a function of number of significant coefficients. Fig. 10(b) shows the graph between the compression error and the threshold $\epsilon$.

### 3.6. Efficient computation of $[T^2^m, m > 0]$ 

Suppose we are given a function $f \in L_2(X)$ and we want to compute $T^2^m f$. Note that $T^2^m f \approx [T^2^m]_{\phi^j}^l f$. We have the result

$$[T^2^m]_{\phi^j}^l f = [T^2^m]_{\phi^j}^{l-m} [T^2^m]_{\phi^j}^{l-(m-1)} [T^2^m]_{\phi^j}^{l-(m-2)} \cdots [T^2^m]_{\phi^j}^{l-2} [T^2^m]_{\phi^j}^{l-1} [T^2^m]_{\phi^j}^l f.$$

Next we know that $[T^2^m]_{\phi^j}^{l-m} f$ will give us the coordinates of $[T^2^m]_{\phi^j}^l f$ in the basis $\phi^{j-m}$ of the space $V^{j-m}$, and these coordinates are nothing but scaling function coefficients of $[T^2^m]_{\phi^j}^l f$ in the space $V^{j-m}$, i.e., $\epsilon^{l-m}$. From $\epsilon^{l-m}$ we can compute $\epsilon^l$ using IDST. $\epsilon^l$ is nothing but vector of coefficients of $[T^2^m]_{\phi^j}^l f$ in the basis $\Phi^l$ which is $[T^2^m]_{\phi^j}^l f$ itself. Algorithm to compute $[T^2^m]_{\phi^j}^l f$ is

$\begin{align*}
T^2^m f & \approx \epsilon^l = \text{ALGORITHM}(T, f, m) \\
& \quad \text{where } g = [T]_{\phi^j}^l f. \\
& \quad \text{For } k = 0, 1, \ldots, m - 1, \\
& \quad \text{if } [T^2^m]_{\phi^j}^{l-k} g \neq \epsilon^{l-k} \text{ then } \epsilon^{l-k} \\
& \quad \text{end} \\
& \quad \text{This gives us } \epsilon^{l-m}.
\end{align*}$
Numerical solution of a PDE is computed by approximating the solution at a discrete set of node points. To discover all the features of the solution we need a large set of node points but this will increase the computational as well as storage cost. In some cases the set required to capture all the features of the solution may exceed the practical limitations. Moreover, if a function has a discontinuity in one of its derivatives then the wavelet coefficients will decrease slowly only near the point of discontinuity and maintain fast decay where the function is smooth. This property of wavelet makes it suitable to detect where in the numerical solution of a PDE the shocks are located and hence an adaptive node arrangement can be generated.

Now we demonstrate the adaptation of node arrangements. Suppose $X^c$ is the current coarse node arrangement and $\{f(x_j)\}_{j \in X^c}$ is known. Using RBF interpolation we compute expanded $f(x)$ on $X^I$ as follows:

- Apply IDST on $c^{I-m}$ to get $c^I$.
- $c^I$ so obtained is nothing but coefficients of $[T^{2^m}]_{\phi^I}^f$ in the basis $\Phi^I$ of the space $V^I$ which further is nothing but $[T^{2^m}]_{\phi^I}^f$ itself.

We took $f(x, y, z) = z$ and computed $[T^{2^{12}}]_{\phi^I}^f$ ($T = I - D^{(2)}$ is the diffusion operator used for the construction of the diffusion wavelet) both analytically (note that $[T^{2^{12}}]_{\phi^I}^f = ([T]\phi^I)^{2^{12}}$ and analytical computation means the matrix $[T]\phi^I$ is multiplied 12 times and finally with $f$ and numerically with the help of the algorithm explained above. Fig. 11 shows the point wise error between analytic and numerical value of $[T^{2^{12}}]_{\phi^I}^f$. The errors in $l_2$ and $l_\infty$ norm are $1.145 \times 10^{-10}$ and $3 \times 10^{-11}$ respectively.

In this case if we compute $[T^{2^{12}}]_{\phi^I}^f$ analytically (i.e. without exploiting the fact that higher powers of $T$ have low numerical rank), then we have to multiply a $256 \times 256$ matrix 12 times and finally the multiplication of a $256 \times 256$ matrix with a $256 \times 1$ matrix. While computing $[T^{2^{12}}]_{\phi^I}^f$ using diffusion wavelet, we are exploiting that higher powers of $T$ have low numerical rank. From Table 1 it can be observed that we need to multiply twelve matrices to get $[T^{2^{12}}]_{\phi^I}^f$ and ten of these are of the size less than 20% of the size of $T$. The CPU time taken for the numerical computation of $[T^{2^{12}}]_{\phi^I}^f$ is 12% of the CPU time taken for its analytic computation.

### Table 1

<table>
<thead>
<tr>
<th>$k$</th>
<th>Size of $[T^{2^k}]_{\phi^{I-k+1}}^{f^I}$ for $\tau = 10^{-9}$</th>
<th>Size of $[T^{2^k}]_{\phi^{I-k+1}}^{f^I}$ for $\tau = 10^{-2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$256 \times 256$</td>
<td>$256 \times 256$</td>
</tr>
<tr>
<td>1</td>
<td>$221 \times 256$</td>
<td>$241 \times 256$</td>
</tr>
<tr>
<td>2</td>
<td>$157 \times 221$</td>
<td>$173 \times 212$</td>
</tr>
<tr>
<td>3</td>
<td>$86 \times 157$</td>
<td>$52 \times 117$</td>
</tr>
<tr>
<td>4</td>
<td>$43 \times 86$</td>
<td>$27 \times 52$</td>
</tr>
<tr>
<td>5</td>
<td>$24 \times 43$</td>
<td>$12 \times 27$</td>
</tr>
<tr>
<td>6</td>
<td>$6 \times 24$</td>
<td>$3 \times 12$</td>
</tr>
<tr>
<td>7</td>
<td>$1 \times 6$</td>
<td>$1 \times 3$</td>
</tr>
</tbody>
</table>

3.7. Adaptive node arrangement
the geodesic distance between the point \( x_m \) using RBFs for space and forward scheme in time to get means less computational cost. But as clear from (31) that the compression error is linearly proportional to \( \epsilon \). Therefore, there is a trade off between the computational cost and the compression error. We need to choose the value of \( \epsilon \) optimally.

Fig. 12 explains the construction of the coarse node arrangement \( \tilde{X} \) on the sphere. Fig. 13 shows the variation of \( N(\epsilon) \) with \( \epsilon \) for \( f(x) = \frac{4}{3\pi} e^{-\frac{(x-1)^2+2x^2+4x^2}{10}} \). It can be observed from the graph that larger is the value \( \epsilon \), smaller is the value of \( N(\epsilon) \) which means less computational cost. But as clear from (31) that the compression error is linearly proportional to \( \epsilon \). Therefore, there is a trade off between the computational cost and the compression error. We need to choose the value of \( \epsilon \) optimally.

3.8. AMDWMS for partial differential equations on the sphere

Consider the differential equation on the unit sphere \( X \) given by

\[
\begin{align*}
\frac{\partial u}{\partial t} - \nabla^2 u &= f(x, t), \\
u(x, 0) &= u_0(x), & u_0(x) \in \mathcal{L}_2(X).
\end{align*}
\]

Eq. (32) describes the heat diffusion process on the surface of the sphere with external heat source \( f(x, t) \). Discretize (32) using RBFs for space and forward scheme in time to get

\[
\mathbf{u}^n = A\mathbf{u}^{n-1} + \Delta tf, \quad \mathbf{u}^0 = \mathbf{u}_0,
\]
Fig. 13. $N(\epsilon)$ versus $\epsilon$ for $f(x) = \frac{A}{\pi} e^{-\frac{|a_1^2 + a_2^2|}{2}}$, $\nu = \frac{1}{2\pi^2}$.

where $u^n$ is the vector of the unknowns $u^n(x_j), j = 1, 2, \ldots, N$ at time $t = n\Delta t$ and $A = I + \Delta t D^{(2)}$. Eq. (33) can be written as

$$u^n = \frac{\|A\|}{\|A\|} A u^{n-1} + \Delta t f = \|A\| B u^{n-1} + \Delta t f.$$  \hspace{1cm} (34)

where $B = \frac{A}{\|A\|}$ (note that $\|A\|$ stands for Frobenius norm given by $\left(\sum_{i} \sum_{j} |a_{ij}|^2 \right)^{\frac{1}{2}}$). Using (34) recursively we obtain

$$u^n = \|A\|^n B^n u^0 + \sum_{k=0}^{n-1} \|A\|^k B^k \Delta t f.$$  \hspace{1cm} (35)

Now let $n = 2^m$ for some $m$, then we have

$$\sum_{k=0}^{n-1} \|A\|^k B^k \Delta t f = \sum_{k=0}^{2^m-1} \|A\|^k B^k \Delta t f = (I + \|A\|B + \|A\|^2 B^2 + \|A\|^3 B^3 + \cdots + \|A\|^{2^m-1} B^{2^m-1}) \Delta t f$$

$$= \prod_{k=0}^{m-1} (I + \|A\|^{2k} B^{2k}) \Delta t f.$$  \hspace{1cm} (36)

Hence (35) can be written as

$$u^{2^m} = \|A\|^{2^m} B^{2^m} u^0 + \prod_{k=0}^{m-1} (I + \|A\|^{2k} B^{2k}) \Delta t f.$$ \hspace{1cm} (36)

The matrix $B$ is suitable for the construction of diffusion wavelet. We construct diffusion wavelet using the operator $B$ and then the dyadic powers of $B$ required in (36) can be computed efficiently as explained in Section 3.6. The same diffusion wavelet is used for the adaptation of node arrangement. Hence in AMDWMS we use diffusion wavelet for two purposes, one for the computation of dyadic powers of $B$ and second for the adaptation of node arrangement. The algorithm for AMDWMS is as follows:

- Construct the new coarse node arrangement using $u^{2^{m-1}}$.
- Compute the matrix $B$ and the vector $f$ for the new coarse node arrangement.
- Compute dyadic powers of the matrix $B$.
- Compute $u^{2^m}$ using (36) on the new coarse node arrangement.
- Perform several time steps on the new coarse node arrangement (Note that how many steps to take between node arrangement evaluations depends on the problem at hand. If the problem is such that its solution is rapidly changing then the node arrangement is needed to be evaluated frequently and if the solution is slowly changing then the same node arrangement can be used for several time steps).

It should be noted that the order of accuracy of the proposed method before introducing adaptivity is $O(N^{-3/2} + \Delta t)$ (for $k = 3$, in numerical experiments we will use $k = 3$). As clear from (31), for a threshold $\epsilon$, the error introduced during adaptivity is $O(\epsilon)$. Therefore, order of accuracy of the final scheme is $O((N(\epsilon))^{-3/2} + \Delta t + \epsilon)$.
with the CPU time taken by simple collocation method using RBFs (we call it CPU($X_{5670}$ @ 2.93 GHz and 16 GB RAM (provided by IIT Delhi private cloud).

We consider the spherical diffusion equation

\[
\frac{\partial u(x, t)}{\partial t} = \nu \nabla^2 u(x, t) + f, \quad x \in X,
\]

where $f$ is a localized source chosen in such a way that the solution of the diffusion equation is given by

\[
u(x, t) = 2\epsilon^{-\frac{1}{2}} e^{-\frac{1}{4}\epsilon^{-\frac{1}{2}} \left(\sin(\theta_1) \sin(\theta_2) \cos(\phi) - \sin(\theta_0) \sin(\phi)\right)^2}.
\]

where $(1, \theta_1, \theta_2)$ is the spherical polar representation of the point $x \in X$. The initial condition is obtained from the analytical solution, and the problem is solved for the parameters $\nu = 1/(4\pi^2)$, $\theta_0^2 = 0$, $\phi_0^2 = 0$. Table 2 shows the node arrangement modifications at different times while computing the solution for different values of $\epsilon$. It is clear from Table 2 that for large value of $\epsilon$, the value of $N(\epsilon)$ is small and hence the computational cost will be small. But for large value of $\epsilon$ the error will be large. Hence we need to choose the value of $\epsilon$ optimally which makes a balance between the error and the computational cost. Fig. 14 shows the solution of (37) and the corresponding adaptive node arrangement at different times. Figs. 15 and 16 verifies the convergence of the method with respect to $N(\epsilon)$ and the threshold $\epsilon$ respectively (note that $u_{\text{num}}$ stands for the numerical solution on the adaptive node arrangement computed using AMDWMS and $u_{\text{ana}}$ stands for the analytic solution of the problem). It can be observed from these figures that the expected order of accuracy is achieved.

In order to verify the efficiency of AMDWMS, we compare the CPU time taken by the AMDWMS (we call it CPU($\epsilon$)) with the CPU time taken by simple collocation method using RBFs (we call it CPU$(\epsilon = 0)$, it should be noted that simple collocation method does not use adaptivity and the fast computation of the powers of the matrices). For its measurement we define time compression coefficient as $\Theta = \frac{\text{CPU}(\epsilon = 0)}{\text{CPU}(\epsilon = \epsilon)}$. Larger is the value of the time compression coefficient, more efficient is the adaptive algorithm. Table 3 gives variation of CPU($\epsilon$) with $\epsilon$ for CPU$(\epsilon = 0) = 11.4$. It can be observed that for $\epsilon = 0.5$, $\Theta = 21.11$ and hence AMDWMS is performing quite good for this value of $\epsilon$ but again we have to choose a value of $\epsilon$ which makes a balance between the efficiency of the method and the error introduced.

### 4. Numerical results

Here we will discuss the numerical results obtained using AMDWMS. Most of the experiments are performed on Matlab (7.6.0 (R2008a)) software, on a 32 bit machine with Intel Core 2 Duo processor. The operating system used is Ubuntu 10.04.4 LTS. For solving the third test problem at high times we used 32 blade servers each with $2 \times 6$ core Intel(R) Xeon(R) CPU X5670 @ 2.93 GHz and 16 GB RAM (provided by IIT Delhi private cloud).

#### 4.1. Test problem 1

We consider the spherical diffusion equation

\[
\frac{\partial u(x, t)}{\partial t} = \nu \nabla^2 u(x, t) + f, \quad x \in X,
\]

where $f$ is a localized source chosen in such a way that the solution of the diffusion equation is given by

\[
u(x, t) = 2\epsilon^{-\frac{1}{2}} e^{-\frac{1}{4}\epsilon^{-\frac{1}{2}} \left(\sin(\theta_1) \sin(\theta_2) \cos(\phi) - \sin(\theta_0) \sin(\phi)\right)^2}.
\]

where $(1, \theta_1, \theta_2)$ is the spherical polar representation of the point $x \in X$. The initial condition is obtained from the analytical solution, and the problem is solved for the parameters $\nu = 1/(4\pi^2)$, $\theta_0^2 = 0$, $\phi_0^2 = 0$. Table 2 shows the node arrangement modifications at different times while computing the solution for different values of $\epsilon$. It is clear from Table 2 that for large value of $\epsilon$, the value of $N(\epsilon)$ is small and hence the computational cost will be small. But for large value of $\epsilon$ the error will be large. Hence we need to choose the value of $\epsilon$ optimally which makes a balance between the error and the computational cost. Fig. 14 shows the solution of (37) and the corresponding adaptive node arrangement at different times. Figs. 15 and 16 verifies the convergence of the method with respect to $N(\epsilon)$ and the threshold $\epsilon$ respectively (note that $u_{\text{num}}$ stands for the numerical solution on the adaptive node arrangement computed using AMDWMS and $u_{\text{ana}}$ stands for the analytic solution of the problem). It can be observed from these figures that the expected order of accuracy is achieved.

In order to verify the efficiency of AMDWMS, we compare the CPU time taken by the AMDWMS (we call it CPU($\epsilon$)) with the CPU time taken by simple collocation method using RBFs (we call it CPU$(\epsilon = 0)$, it should be noted that simple collocation method does not use adaptivity and the fast computation of the powers of the matrices). For its measurement we define time compression coefficient as $\Theta = \frac{\text{CPU}(\epsilon = 0)}{\text{CPU}(\epsilon = \epsilon)}$. Larger is the value of the time compression coefficient, more efficient is the adaptive algorithm. Table 3 gives variation of CPU($\epsilon$) with $\epsilon$ for CPU$(\epsilon = 0) = 11.4$. It can be observed that for $\epsilon = 0.5$, $\Theta = 21.11$ and hence AMDWMS is performing quite good for this value of $\epsilon$ but again we have to choose a value of $\epsilon$ which makes a balance between the efficiency of the method and the error introduced.

#### 4.2. Test problem 2

Consider the problem of computing a moving steep front along the direction $\vec{k} = (1, 1, 1)^T$ which is governed by the differential equation

\[
\frac{\partial u}{\partial t} + u \vec{k} \cdot \nabla u = \zeta \nabla^2 u, \quad x \in X,
\]

\[
\begin{array}{c|c|c}
\text{Time (t)} & \epsilon = 0.5 & \epsilon = 0.1 \\
\hline
N(\epsilon = 0) & N(\epsilon) & N(\epsilon = 0) & N(\epsilon) \\
0 & 512 & 81 & 512 & 209 \\
0.25 & 64 & 6 & 209 & 212 \\
0.50 & 212 & 149 & 466 & 456 \\
0.75 & 149 & 512 & 496 & 474 \\
0.25 & 6 & 17 & 6 & 14 \\
1.25 & 6 & 17 & 6 & 14 \\
1.5 & 6 & 17 & 6 & 14
\end{array}
\]
Fig. 14. Solution of the PDE and the corresponding adaptive node arrangement at $t = 0, 0.75, 1.5$ for the test problem 1.

Fig. 15. $\|u_{\text{num}} - u_{\text{ana}}\|_p$ versus $N(\epsilon)$ for the test problem 1.

with $u(x, 0) = (1 + e^{a+b+c+1})^{-1}$, $x = (a, b, c)$. Tables 4 and 5 show the node arrangement modifications at different times while computing the solution for different values of $\epsilon$ for $\zeta = 0.01$ and $\zeta = 0.5$ respectively. It can be observed from these tables that $N(\epsilon)$ increases with increase in the value of viscosity coefficient $\zeta$. Fig. 17 shows the solution of (38) and
Fig. 16. $\|u_{\text{num}} - u_{\text{ana}}\|_p$ versus $\epsilon$ for the test problem 1.

Table 4
Node arrangement modifications while solving test problem 2 for different values of $\epsilon$ with $\zeta = 0.01$.

<table>
<thead>
<tr>
<th>Time ($t$)</th>
<th>$\epsilon = 0.5$</th>
<th>$\epsilon = 0.1$</th>
<th>$\epsilon = 0.01$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N(\epsilon = 0)$</td>
<td>$N(\epsilon)$</td>
<td>$N(\epsilon = 0)$</td>
</tr>
<tr>
<td>0</td>
<td>512</td>
<td>73</td>
<td>512</td>
</tr>
<tr>
<td>0.25</td>
<td>73</td>
<td>62</td>
<td>186</td>
</tr>
<tr>
<td>0.50</td>
<td>62</td>
<td>48</td>
<td>173</td>
</tr>
<tr>
<td>1.0</td>
<td>48</td>
<td>12</td>
<td>115</td>
</tr>
<tr>
<td>1.25</td>
<td>12</td>
<td>12</td>
<td>82</td>
</tr>
<tr>
<td>1.5</td>
<td>12</td>
<td>12</td>
<td>65</td>
</tr>
</tbody>
</table>

Table 5
Node arrangement modifications while solving test problem 2 for different values of $\epsilon$ with $\zeta = 0.5$.

<table>
<thead>
<tr>
<th>Time ($t$)</th>
<th>$\epsilon = 0.5$</th>
<th>$\epsilon = 0.1$</th>
<th>$\epsilon = 0.01$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N(\epsilon = 0)$</td>
<td>$N(\epsilon)$</td>
<td>$N(\epsilon = 0)$</td>
</tr>
<tr>
<td>0</td>
<td>512</td>
<td>112</td>
<td>512</td>
</tr>
<tr>
<td>0.25</td>
<td>112</td>
<td>82</td>
<td>232</td>
</tr>
<tr>
<td>0.50</td>
<td>82</td>
<td>65</td>
<td>193</td>
</tr>
<tr>
<td>0.75</td>
<td>65</td>
<td>52</td>
<td>181</td>
</tr>
<tr>
<td>1.0</td>
<td>52</td>
<td>43</td>
<td>162</td>
</tr>
<tr>
<td>1.25</td>
<td>43</td>
<td>35</td>
<td>146</td>
</tr>
<tr>
<td>1.5</td>
<td>35</td>
<td>34</td>
<td>69</td>
</tr>
</tbody>
</table>

the corresponding adaptive node arrangement at different times. It can be seen that the front moves along the direction $(1, 1, 1)^T$. The area with solution value 0 becomes bigger, and area with solution value 1 becomes smaller as time increases. At $t = 1.5$, the solution is mostly 0 on the sphere and is 1 only on a small area. It is seen that our method adapts the node arrangement extremely well in the regions with large solution gradients.

Tables 6 and 7 show the variation of CPU($\epsilon$) with $\epsilon$ for $\zeta = 0.01$ and $\zeta = 0.5$ respectively (CPU($\epsilon = 0$) = 13.5). Fig. 18 shows the variation of $\Theta$ with the parameter $\zeta$. It can be observed that as the value of $\zeta$ increases, value of $\Theta$ decreases. Hence AMDWMS performs better for smaller value of $\zeta$.

4.3. Test problem 3

In 1952, A.M. Turing [39] settled the basis for explaining biological patterns using two interacting chemicals that, under certain conditions, can generate stable patterns from an initial near-homogeneity. This phenomenon has now been shown to occur in chemistry and biology. The Turing patterns are governed by a system of nonlinear reaction–diffusion equations. We solve the following system [40]

$$
\frac{\partial u}{\partial t} = D\delta \nabla^2 u + \alpha u(1 - r_1 v^2) + v(1 - r_2 u),
$$

$$
\frac{\partial v}{\partial t} = \delta \nabla^2 v + \beta v \left( 1 + \frac{\alpha r_1}{\beta} u v \right) + u(\gamma + r_2 v).
$$

(39)
At $t = 0$ we consider $u = v = 0$, except on a narrow band as shown in Fig. 19(a). The stable patterns can be either stripes or spots, depending on the parameters $r_1$ and $r_2$. The parameter $r_1$ favours stripes while $r_2$ favours spots. We fix the parameters $D = 0.516$, $\alpha = 0.899$, $\beta = -0.91$ and $\gamma = -\alpha$.

As case 1, we take $r_1 = 0.02$, $r_2 = 0.2$, $\delta = 0.0171$. The evolution of the spotted pattern and the corresponding adaptive node arrangement is shown in Fig. 19. It is evident that AMDWMS is able to track the emergence of spots over the entire

**Table 6**

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>CPU($\epsilon$)</th>
<th>$\Theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5819</td>
<td>23.2</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>1.2054</td>
<td>11.2</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>2.5140</td>
<td>5.37</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>2.7551</td>
<td>4.9</td>
</tr>
</tbody>
</table>

**Table 7**

<table>
<thead>
<tr>
<th>$\epsilon$</th>
<th>CPU($\epsilon$)</th>
<th>$\Theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.7234</td>
<td>18.661</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>1.5302</td>
<td>8.822</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>2.998</td>
<td>4.503</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>3.215</td>
<td>4.199</td>
</tr>
</tbody>
</table>

Fig. 17. Solution of the PDE and the corresponding adaptive node arrangement at $t = 0, 0.75, 1.5$ for the test problem 2 with $\zeta = 0.01$. 

---

**Fig. 18.** Variation of $\Theta$ with $\zeta$ for the test problem 2.

**Fig. 19.** Evolution of the solution and dynamically adapted node arrangement for the Turing problem for $r_1 = 0.02$, $r_2 = 0.2$, $\delta = 0.0171$, $\epsilon = 10^{-4}$ (only one component $u$ is shown).
Fig. 20. $x$–$y$ maps of the solution and the corresponding dynamically adapted node arrangement for case 1 of the Turing problem.

Fig. 21. Evolution of the solution and dynamically adapted node arrangement for the Turing problem for $r_1 = 3.5$, $r_2 = 0$, $\delta = 0.0021$, $\epsilon = 10^{-4}$ (only one component $u$ is shown).

This demonstrates that the proposed method is able to efficiently and accurately capture the emergence of the localized structures at multiscales which characterize the solution of nonlinear PDEs. To have a better view of the configuration of the spots, $x$–$y$ maps of the solution and the corresponding node arrangement is shown in Fig. 20. Fig. 22 shows the
variation of $\Theta$ with respect to the threshold $\epsilon$. It can be observed that at $\epsilon = 10^{-1}$ there is a 92% reduction in the CPU time.

As case 2, we take $r_1 = 3.5$, $r_2 = 0$, $\delta = 0.0021$. The evolution of the striped pattern and the corresponding adaptive node arrangement is shown in Fig. 21. The adaptivity of node arrangement is not very clearly visible because of large variations in the solution. Fig. 23 shows the variation of $\Theta$ with respect to the threshold $\epsilon$.

5. Conclusion and future work

An adaptive meshfree diffusion wavelet method is developed to solve PDEs on the sphere. The RBFs are used to approximate the differential operators and the convergence of these approximations to $\nabla^2$ and $\vec{\nabla}$ is verified for two test functions. The diffusion wavelet is used for the adaptation of node arrangement as well as for the fast computation of the powers of the matrices involved in the computation of the numerical solution. The efficiency and the convergence of the AMDWMS is verified. The CPU time taken by the AMDWMS is compared with the CPU time taken by the RBF based collocation method and it turns out that AMDWMS performs much better. To best of our knowledge the useful properties of diffusion wavelet are for the first time exploited to solve PDEs on an adaptive node arrangement on the sphere in a meshfree setting.

In future we will use AMDWMS to solve problems of industrial and academic interest which cannot be easily treated with classical mesh based methods. For example, the simulation of manufacturing processes such as extrusion and modelling, where it is necessary to deal with extremely large deformations of the mesh. Moreover, we will use AMDWMS for turbulence modelling.

Acknowledgements

The first author would like to thank Council of Scientific and Industrial Research for providing Ph.D. scholarship. We would also like to acknowledge prof. Nicholas Kevlahan for his valuable comments. We are grateful to the anonymous reviewers for their expert suggestions.

References
