Applied Mathematical Modelling 40 (2016) 5000–5025

Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Fast diffusion wavelet method for partial differential equations

Kavita Goyal [a], Mani Mehra [a,*]

Department of Mathematics, IIT Delhi, Hauz Khas New Delhi, 110016, India

ARTICLE INFO

Article history:
Received 13 May 2014
Revised 2 March 2015
Accepted 12 October 2015
Available online 18 December 2015

Keywords:
Multiresolution analysis (MRA)
Wavelet based numerical methods
Partial differential equations

ABSTRACT

A fast diffusion wavelet method for solving partial differential equations (PDEs) is developed. Classes of operators which can be used for the construction of diffusion wavelet include approximation of second order differential operators. The efficiency of the method is that the same diffusion operator is used for the construction of diffusion wavelet as well as for approximation of second order differential operator. As a part of the wavelet method the behavior of compression error with respect to different parameters involved in the construction of diffusion wavelet is tested for two test functions. Furthermore, the diffusion wavelet is used for the compression of operators and hence for the fast and efficient computing of the dyadic powers of the diffusion operator T which are required for solving the PDE. We have considered PDEs with Dirichlet and periodic boundary conditions on one, two, and three dimensional domains. For each test problem, the CPU time taken by the fast diffusion wavelet method is compared with the CPU time taken by the finite difference method. We have also verified the convergence of the proposed method.

1. Introduction

Partial differential equations are widely used for realistic representation of real world problems such as fluctuations in stock markets, epidemiological models, climate modeling etc. 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, has motivated their use for numerical solutions of PDEs. Wavelet methods have been developed for most kinds of linear PDEs such as Laplace/Poisson equations [1] and advection diffusion problems [2]. For non linear PDEs also, there exist a large spectrum of wavelet methods, which have been applied to Burgers equation [3,4], reaction–diffusion equations [5] and Stokes equation [6].

There are two main issues that arise while solving the PDEs with wavelet based numerical methods.

1. The first issue is to solve PDEs on general manifolds. Many techniques have been developed to construct wavelets on general manifolds. For example, in [7,8] wavelet bases are constructed on a specific type of manifolds which can be represented as disjoint union of smooth parametric images of a standard cube. The construction is based solely on smooth parameterization of the unit cube, which have several disadvantages from a practical point of view. This problem, is resolved in [9], where a finite element based wavelet bases with respect to an arbitrary initial triangularization are constructed. Wavelets are constructed in particular on the sphere in [10]. The second generation wavelet

* Corresponding author. Tel.: +91 11 26591487.
E-mail addresses: kavita@maths.iitd.ac.in (K. Goyal), mmehra@maths.iitd.ac.in (M. Mehra).

http://dx.doi.org/10.1016/j.apm.2015.10.054
S0307-904X(15)00752-0/

was developed by Swelden and his collaborators using lifting scheme in [11], which has many practical advantages over the traditional wavelet. Despite of vast literature of wavelets on general manifolds, the wavelet theory for numerical solutions of PDEs on general manifold is still in its nascent stage. In 2008, a dynamic adaptive numerical method for solving PDEs on the sphere was developed in [12] using second generation wavelet.

2. The second issue is to deal with boundary conditions (except the periodic boundary conditions which can be easily handled with conventional wavelet methods). Various approaches to deal with general boundary conditions are discussed in [4,13–18].

The diffusion wavelet was introduced by Coifman and co-workers [19] and its beauty is that it can be constructed on any type of manifold. Since then it has been used in several fields of Engineering [20,21]. The use of diffusion wavelet for solution of PDEs is one of the future directions suggested in [19], which is the focus of our work. Diffusion wavelet can handle above two issues as follows.

1. Since the diffusion wavelet can be constructed on general manifolds, this can be used to solve PDEs on general manifolds. Being our first attempt in use of diffusion wavelet for PDEs, we restrict our attention to Euclidean domains in this paper. In this way, fast diffusion wavelet method (FDWM) developed in this paper is a first step in direction of usage of diffusion wavelet for solution of PDEs on general manifolds.
2. Any type of boundary conditions can be dealt with this method because of the following reasons:
   - If we are solving PDE on the domain $\Omega \subset \mathbb{R}^n$, then the diffusion wavelet is constructed for the space $\mathcal{L}_2(\Omega)$ and not for $\mathcal{L}_2(\mathbb{R}^n)$, hence we do not need to restrict the diffusion wavelet from $\mathcal{L}_2(\mathbb{R}^n)$ onto $\mathcal{L}_2(\Omega)$ which is a common way to achieve wavelet approximation on bounded domains.
   - In FDWM the Neumann and periodic boundary conditions will be incorporated in the operator T and Dirichlet's boundary conditions will be incorporated in $\mathbf{f}$ (the right hand side vector of algebraic system of the equations $A\mathbf{u} = \mathbf{f}$ obtained by discretizing the PDE $Lu = f$, where L is any differential operator).

   We have applied FDWM to test problems with Dirichlet and periodic boundary conditions, see Section 4.

The paper is organized as follows. Section 2 gives a brief overview of the construction of the diffusion wavelet. Multiresolution analysis required for the construction of diffusion wavelet is also discussed in this section. In Section 3 FDWM for the numerical solution of PDEs is developed. Moreover, in this section we have also tested the behavior of compression error with respect to different parameters involved in the construction of diffusion wavelet for two test functions. Section 4 contains the numerical results where we have applied FDWM to seven test problems. Test problems are of different natures (for example they vary in the types of boundary conditions, domain of the solutions, or in nature of the problem (linear and non linear)). The test problem 6, where domain of the solution is region outside a butterfly in a square can not be handled with the traditional wavelet. Section 5 concludes the paper and gives a brief idea of future direction.

2. Construction of diffusion wavelet

Diffusion wavelet is constructed for any general manifold X, multiresolution analysis (MRA) is built using a diffusion operator T on $\mathcal{L}_2(X)$.

2.1. Diffusion operator T used for the construction of diffusion wavelet

The operator $T : \mathcal{L}_2(X) \to \mathcal{L}_2(X)$ which is used in the construction of diffusion wavelet should satisfy following conditions:

- T is local, i.e. $T(\delta_k)$, where δ_k is a mollification of the Dirac δ-function at $k \in X^J$ ($X^J = \{x_1, x_2, \ldots, x_N\}$ is discretization of X using N points), has small support.
- High powers of T have low numerical rank Ran_τ, where Ran_τ is defined as

Definition 2.1. Let H be a Hilbert space and $\mathcal{V} \subseteq H$, then a subset $\{\xi_i\}_{i \in \kappa}$ of $\mathcal{V}$ τ-spans $\mathcal{V}$ if for each $v \in \mathcal{V}$; $\| P_{\langle\{\xi_i\}_{i\in\kappa}\rangle} v - v \|_H \leq \tau$. The τ dimension of $\mathcal{V}$ and $Ran_\tau(T)$ are defined as

$$\dim_\tau(\mathcal{V}) = inf\{\dim(V) : \text{ V is an } \tau - \text{span of } \mathcal{V}\} \quad Ran_\tau(T) = \dim_\tau(range(T)).$$

- T should be self adjoint.

2.2. Multiresolution analysis (MRA)

MRA is a natural technique for the construction of wavelet. The goal of MRA is to express an arbitrary function $f \in \mathcal{L}_2(X)$ at various levels [22]. For a classical wavelet, MRA of $\mathcal{L}_2(X)$ is a sequence $\{\mathcal{V}^j\}_{j\in\mathbb{Z}}$ satisfying following axioms:

(1) $\{0\} \subset \cdots \subset \mathcal{V}^{-1} \subset \mathcal{V}^0 \subset \mathcal{V}^1 \subset \cdots \subset \mathcal{L}_2(X)$.
(2) $\overline{\bigcup_{j\in\mathbb{Z}} \mathcal{V}^j} = \mathcal{L}_2(X)$.
(3) Invariance to dilations, *i.e.,* $f \in \mathcal{V}^j$ iff $f(2(.)) \in \mathcal{V}^{j+1}$.
(4) Invariance to translations, *i.e.,* $\{\phi_k^0 \text{ (scaling function)} = \phi(x-k) | k \in \mathbb{Z}\}$ is an orthonormal basis for $\mathcal{V}^0$.

2.2.1. Construction of MRA for diffusion wavelet

A precision $\tau > 0$ is fixed. We consider the manifold X and T is an operator on $\mathcal{L}_2(X)$ which satisfies all the requirements given in Section 2.1. 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 i.e. columns of $[B_1]_{B_2}$ are the coordinates of the vectors in B_1 in the coordinates B_2.

- **Construction of the space $\mathcal{V}^J$:** Let $\Phi^J = \{\delta_k\}_{k\in X^J}$ (here δ_k is a $N \times 1$ vector having 1 at kth place and 0 otherwise), then space $\mathcal{V}^J$ is defined as:

$$\mathcal{V}^J = \overline{\text{span}\{\Phi^J\}}.$$

- **Construction of the space $\mathcal{V}^{J-1}$:** Columns of $[T]_{\Phi^J}^{\Phi^J}$ are set of functions $\tilde{\Phi}^{J-1} = \{T\delta_k\}_{k\in X^J}$. A local multiscale orthogonalization procedure [19] is used to orthonormalize these columns to get a basis $\Phi^{J-1} = \{\phi_k^{J-1}\}_{k\in X^{J-1}}$ written with respect to the basis Φ^J, of the range of T upto a precision of τ.

$$[T]_{\Phi^J}^{\Phi^J} \xrightarrow{\text{Orthonormalization}} [\Phi^{J-1}]_{\Phi^J}.$$

From the basis Φ^{J-1} we get the space $\mathcal{V}^{J-1} = \overline{\text{span}\{\Phi^{J-1}\}} = \overline{\text{range}_\tau(T)}$, where $\text{range}_\tau(T)$ is a space which is τ close to the range of the operator T. It is clear that $|X^{J-1}| \le |X^J|$.

- **Construction of the space $\mathcal{V}^{J-2}$:** Using the matrices $[T]_{\Phi^J}^{\Phi^J}$ and $[\Phi^{J-1}]_{\Phi^J}$, the matrix $[T]_{\Phi^J}^{\Phi^{J-1}}$ is obtained as follows:

$$[T]_{\Phi^J}^{\Phi^{J-1}} = [T]_{\Phi^J}^{\Phi^J}[\Phi^{J-1}]_{\Phi^J}.$$

Having calculated $[T]_{\Phi^J}^{\Phi^{J-1}}$, $[T^2]_{\Phi^{J-1}}^{\Phi^{J-1}}$ is calculated as follows

$$[T^2]_{\Phi^{J-1}}^{\Phi^{J-1}} = [\Phi^{J-1}]_{\Phi^J}[T^2]_{\Phi^J}^{\Phi^J}[\Phi^{J-1}]_{\Phi^J}' = [T]_{\Phi^J}^{\Phi^{J-1}}([T]_{\Phi^J}^{\Phi^{J-1}})^*.$$

The last equality holds because the operator T is self-adjoint. Columns of the matrix $[T^2]_{\Phi^{J-1}}^{\Phi^{J-1}}$ are set of the functions $\tilde{\Phi}^{J-2} = \{T^2\phi_k^{J-1}\}_{k\in X^{J-1}}$. Again the local multiscale orthogonalization procedure is used to orthonormalize these columns to get a basis $\Phi^{J-2} = \{\phi_k^{J-2}\}_{k\in X^{J-2}}$ written with respect to the basis Φ^{J-1}, of the range of T^{2+1} upto a precision of τ. From the basis Φ^{J-2} we get the space $\mathcal{V}^{J-2} = \overline{\text{span}\{\Phi^{J-2}\}} = \overline{\text{range}_\tau(T^{2+1})}$.

- **Construction of the space $\mathcal{V}^{J-j}$:** After $j-1$ steps in above fashion, we will have a representation of $T^{2^{j-1}}$ with respect to the basis $\Phi^{J-(j-1)} = \{\phi_k^{J-(j-1)}\}_{k\in X^{J-(j-1)}}$, encoded in the matrix $[T^{2^{j-1}}]_{\Phi^{J-(j-1)}}^{\Phi^{J-(j-1)}}$. The columns of $[T^{2^{j-1}}]_{\Phi^{J-(j-1)}}^{\Phi^{J-(j-1)}}$ are the set of the functions $\tilde{\Phi}^{J-j} = \{T^{2^{j-1}}\phi_k^{J-(j-1)}\}_{k\in X^{J-(j-1)}}$. The local multiscale orthogonalization procedure on this set will yield an orthonormal basis $\Phi^{J-j} = \{\phi_k^{J-j}\}_{k\in X^{J-j}}$ for $\text{range}_\tau(T^{2^{j-1}+2^{j-2}+\cdots+2+1})$ and hence we get the space $\mathcal{V}^{J-j} = \overline{\text{span}\{\Phi^{J-j}\}} = \overline{\text{range}_\tau(T^{2^{j-1}+2^{j-2}+\cdots+2+1})}$.
- Note that the dyadic powers of T dilates the basis elements and the orthogonalization step of the resulting functions down sample them to an orthonormal basis of the range of that power of T, which constitutes the next scaling function space.
- For the operator T, it is clear that

$$\cdots \subseteq \overline{\text{range}_\tau(T^{1+2+\cdots+2^{j-1}})} \subseteq \cdots \subseteq \overline{\text{range}_\tau(T)} \subseteq \overline{\text{span}\{\Phi^J\}} \subseteq \cdots \subseteq \mathcal{L}_2(X),$$

so that we have

$$\cdots \subseteq \mathcal{V}^{J-j} \subseteq \cdots \subseteq \mathcal{V}^{J-1} \subseteq \mathcal{V}^J \subseteq \cdots \subseteq \mathcal{L}_2(X),$$

which is analogous to the axiom (1) of MRA.
- Clearly $\overline{\bigcup_{j\in\mathbb{Z}} \mathcal{V}^j} = \mathcal{L}_2(X)$, which is analogous to axiom (2) of MRA as defined in Section 2.2.
- 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. Now 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 $\mathcal{V}^{j+1}$ (Note that a function f is dilation of the function g, it means that the operator T is operated on g to obtain f.).
- Also Φ^J τ-spans the space $\mathcal{V}^J$, which is analogous to axiom (4) of MRA.

(Note on numerical stability of the orthogonalization algorithm: Modified Gram–Schmidt algorithm is used for orthogonalization. The algorithm do not assume any regularity on the family $\tilde{\Phi}$, but there is no guarantee for good bounds on numerical stability. However, as stated in [19], generally the algorithm gives very satisfying results.)

In this way an MRA is constructed for the space $\mathcal{L}_2(X)$ i.e. we have constructed the approximation spaces $\mathcal{V}^j$s. The detail spaces $\{\mathcal{W}^j\}$s are constructed in such a way that $\mathcal{V}^j = \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$. The sets Ψ^js which τ-span the spaces $\mathcal{W}^j$s are

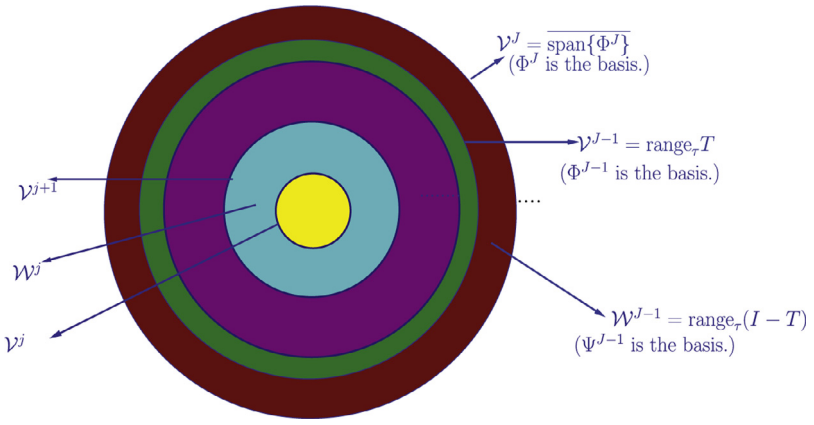


Fig. 2.1. Diffusion approximation and wavelet spaces.

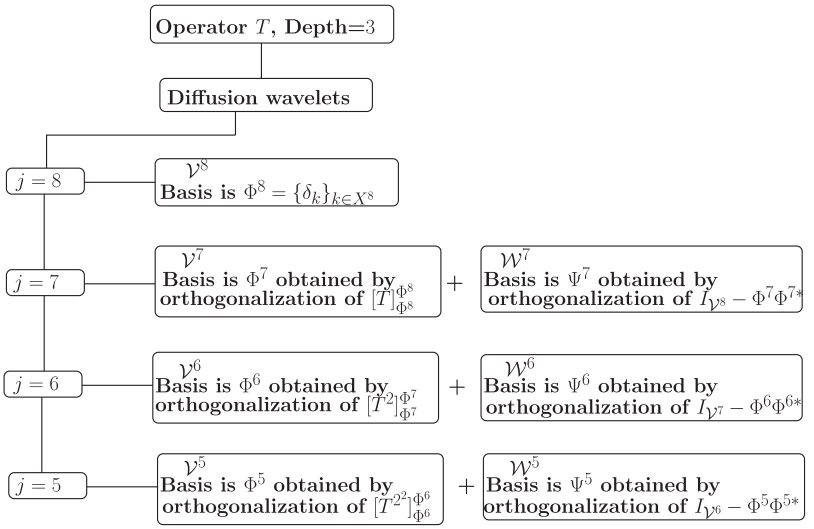


Fig. 2.2. Construction of the diffusion wavelet on [0, 1].

constructed simply by continuing the orthogonalization procedure till the domain (instead of the range) of $[T^{2^j}]^{\Psi^j}_{\Psi^j}$ is exhausted. In practice, to preserve numerical accuracy, this is achieved by starting with the columns of $I_{\mathcal{V}^{j+1}} - \Phi^j\Phi^{j*}$. Fig. 2.1 diagrammatically shows the approximation and wavelet spaces in case of diffusion wavelet.

Note that we can either construct the MRA (i.e. the spaces $\mathcal{V}^j$s) till the range of $[T^{2^j}]^{\Phi^j}_{\Phi^j}$ 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 $\dim_\tau(\mathcal{V}^j)$ 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. If J_0 is the coarsest level then

$$\mathcal{V}^J = \mathcal{W}^{J-1} \oplus \mathcal{V}^{J-1} = \mathcal{W}^{J-1} \oplus \mathcal{W}^{J-2} \oplus \mathcal{V}^{J-2} = \cdots = \mathcal{V}^{J_0} \oplus \bigoplus_{j=J_0}^{J-1} \mathcal{W}^j.$$

Let $X = [0, 1]$, discretize this domain using $N = 2^8 = 256$ points and call it X^8 (we are using dyadic grid for simplicity). Take the basis $\Phi^8 = \{\delta_k\}_{k \in X^8}$. Fig. 2.2 explains the construction of the diffusion wavelet in this particular case for depth = 3. There will be 4 levels, at each level except the first level there will be an approximation space and a wavelet space. The first level will have only the approximation space.

Note that in moving from the space $\mathcal{V}^{j+1}$ to the spaces $\mathcal{V}^j$ and $\mathcal{W}^j$; $\dim(\mathcal{V}^j) \neq \dim(\mathcal{W}^j) \neq 2^j$, which is the case with Daubechies wavelet.

Table 2.1 shows the complete construction of the diffusion wavelet on [0, 1] for the diffusion operator T which is constructed according the construction given in [23] and $\tau = 10^{-9}$. (Note that the approximation spaces $\mathcal{V}^j$s are constructed till the range of $[T^{2^j}]^{\Phi^j}_{\Phi^j}$ is exhausted without fixing any coarsest level). We can observe from Table 2.1 that $\dim_\tau(\mathcal{V}^7) = 232 \neq 2^7$ and $\dim_\tau(\mathcal{W}^7) = 24 \neq 2^7$, however $\dim_\tau(\mathcal{V}^7) + \dim_\tau(\mathcal{W}^7) = \dim_\tau(\mathcal{V}^8) = 256$.

Table 2.1
Diffusion wavelet on [0, 1] using T which is constructed according to the construction given in [23] and $\tau = 10^{-9}$.

Level (j)	$\dim_\tau(\mathcal{V}^j)$	$\dim_\tau(\mathcal{W}^j)$
8	256	0
7	232	24
6	43	189
5	16	27
4	12	4
3	9	3
2	7	2
1	5	2
0	4	1
−1	3	1
−2	2	1
−3	2	0
−4	1	1

Table 2.2
Diffusion wavelet on [0, 1] using T which is finite difference approximation of a second order differential operator.

Level (j)	$\dim_\tau(\mathcal{V}^j)$ at $\tau = 10^{-9}$	$\dim_\tau(\mathcal{V}^j)$ at $\tau = 10^{-2}$	$\dim_\tau(\mathcal{W}^j)$ at $\tau = 10^{-9}$	$\dim_\tau(\mathcal{W}^j)$ at $\tau = 10^{-2}$
8	256	256	0	0
7	255	253	1	3
6	255	252	0	1
5	255	250	0	2
4	252	206	3	44
3	212	150	40	56
2	159	95	53	55
1	102	67	57	28
0	70	43	32	24
−1	49	30	21	13
−2	34	17	15	13
−3	24	10	10	7
−4	17	9	7	1

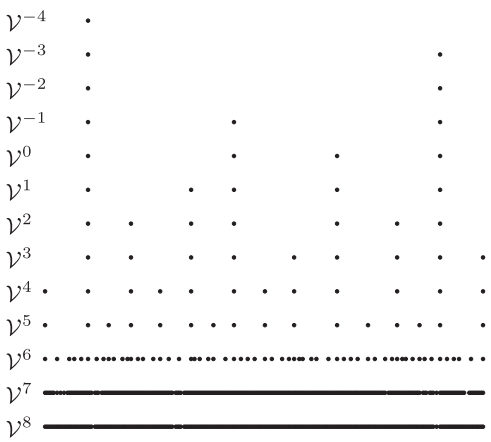


Fig. 2.3. Positions of the grid points for different $\mathcal{V}^j$s.

Table 2.2 shows the construction of diffusion wavelet for the diffusion operator T which is a finite difference approximation of a second order differential operator. In this table it can be seen that for $\tau = 10^{-9}$, $\dim_\tau(\mathcal{V}^7) = 255$ and $\dim_\tau(\mathcal{W}^7) = 1$ and for $\tau = 10^{-2}$, $\dim_\tau(\mathcal{V}^7) = 253$ and $\dim_\tau(\mathcal{W}^7) = 3$. So in case of diffusion wavelet the rate of decay in the rank of dyadic powers of T and the precision τ used in the construction will decide $\dim_\tau(\mathcal{V}^j)$ and $\dim_\tau(\mathcal{W}^j)$.

If we are working on the interval [0, 1] with $N = 256$, then there will be 256 grid points corresponding to the finest space $\mathcal{V}^8$ and number of grid points will keep on reducing as we will move to the space $\mathcal{V}^7$ and then to $\mathcal{V}^6$ and so on. The rate of decay will depend on the kind of the operator T used in the construction of diffusion wavelet and the precision τ. Fig. 2.3 represents the position of the grid points at different levels for the operator T which is constructed using the

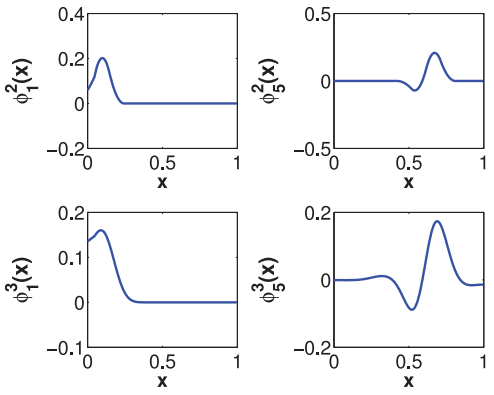


Fig. 2.4. Diffusion scaling functions.

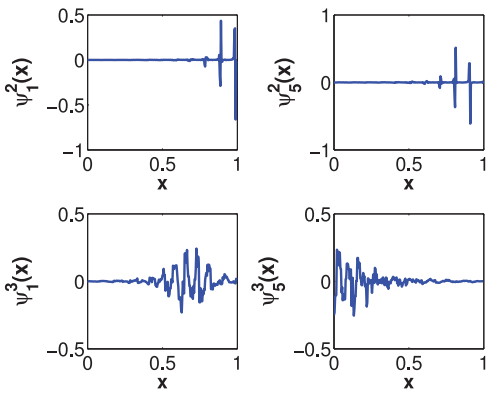


Fig. 2.5. Diffusion wavelet functions.

construction given in [23] and at $\tau = 10^{-9}$. For same T and τ, Fig. 2.4 and Fig. 2.5 respectively shows some diffusion scaling and wavelet functions on the interval [0, 1].

2.3. Diffusion scaling function and inverse diffusion scaling function transform

For any $f \in \mathcal{L}_2(X)$ we have

$$P_{\mathcal{V}^j} f(x) = \sum_{k \in X^j} c_k^j \phi_k^j(x),$$

The coefficients $\{c_k^j\}_{k \in X^j}$ are called the scaling function coefficients of the function $f(x)$ at the level j. Because of the orthonormality of $\{\phi_k^j\}_{k \in X^j}$ we have $c_k^j = \langle f, \phi_k^j \rangle$. Let $\mathbf{c}^j = \{c_k^j\}_{k \in X^j}$ so that $\mathbf{c}^j = \langle f, \Phi^j \rangle$. The set $\langle f, \Phi^j \rangle$ is nothing but $[\Phi^j]'_{\Phi^J}\mathbf{f}$, where $\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_N)]'$, i.e.,

$$\mathbf{c}^j = [\Phi^j]'_{\Phi^J}\mathbf{f}, \tag{2.1}$$

and the matrix $[\Phi^j]_{\Phi^J}$ can be calculated as follows

$$[\Phi^j]_{\Phi^J} = [\Phi^{j+1}]_{\Phi^J}[\Phi^j]_{\Phi^{j+1}} = \cdots = [\Phi^{J-1}]_{\Phi^J}[\Phi^{J-2}]_{\Phi^{J-1}} \cdots [\Phi^{j+1}]_{\Phi^{j+2}}[\Phi^j]_{\Phi^{j+1}}. \tag{2.2}$$

Eq. (2.1) is called as diffusion scaling function transform (DST). From Eq. (2.1)

$$\mathbf{f} = ([\Phi^j]'_{\Phi^J})^{-1}\mathbf{c}^j = [\Phi^j]_{\Phi^J}\mathbf{c}^j. \tag{2.3}$$

Eq. (2.3) is called the inverse diffusion scaling function transform (IDST). The important point to note is that the computational cost of IDST is same as computational cost of computing the transpose of the matrix $[\Phi^j]_{\Phi^J}$ (which is much less than the cost of computing the inverse of it). Note that because Φ^j for $j = J$ is delta basis of the space $\mathcal{V}^J$, so $\mathbf{f}$ is nothing but $\mathbf{c}^J$. So no diffusion scaling function transformation is required for obtaining the scaling function coefficients of the function f in the space $\mathcal{V}^j$ for $j = J$. Fig. 2.6 shows how to apply DST and IDST.

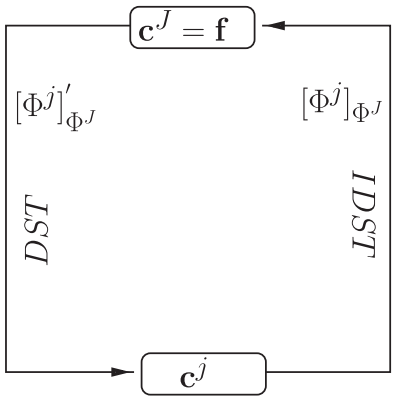


Fig. 2.6. DST and IDST.

2.4. Diffusion wavelet and inverse diffusion wavelet transform

For any function $f \in \mathcal{L}_2(X)$, $P_{\mathcal{V}^j} f = P_{\mathcal{V}^{j-1}} f + P_{\mathcal{W}^{j-1}} f$. So we can write

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

where Y^{j-1} is the index set. Given the set $\mathbf{c}^j$ we want to obtain the sets $\mathbf{c}^{j-1} = \{c_k^{j-1}\}_{k \in X^{j-1}}$ and $\mathbf{d}^{j-1} = \{d_k^{j-1}\}_{k \in Y^{j-1}}$. The set $\mathbf{c}^{j-1}$ can be obtained as follows:

$$\mathbf{c}^{j-1} = \langle f, \Phi^{j-1} \rangle = [\Phi^{j-1}]'_{\Phi^J} \mathbf{f} = [\Phi^{j-1}]'_{\Phi^j} [\Phi^j]'_{\Phi^J} \mathbf{f} = [\Phi^{j-1}]'_{\Phi^j} \mathbf{c}^j.$$

The set $\mathbf{d}^{j-1} = \langle f, \Psi^{j-1} \rangle$ can be obtained as follows:

$$\mathbf{d}^{j-1} = \langle f, \Psi^{j-1} \rangle = [\Psi^{j-1}]'_{\Phi^J} \mathbf{f} = [\Psi^{j-1}]'_{\Phi^j} [\Phi^j]'_{\Phi^J} \mathbf{f} = [\Psi^{j-1}]'_{\Phi^j} \mathbf{c}^j.$$

Hence we have the relations

$$\mathbf{c}^{j-1} = [\Phi^{j-1}]'_{\Phi^j} \mathbf{c}^j,$$
$$\mathbf{d}^{j-1} = [\Psi^{j-1}]'_{\Phi^j} \mathbf{c}^j,$$

which defines a way to obtain $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ from $\mathbf{c}^j$. We call it as **partial diffusion wavelet transform** (PDWT). Now for the coarsest level J_0 and the finest level J, we can decompose the space $\mathcal{V}^J$ as

$$\mathcal{V}^J = \mathcal{V}^{J_0} \oplus \bigoplus_{j=J_0}^{J-1} \mathcal{W}^j.$$

So we can write

$$(P_{\mathcal{V}^J} f)(x) = \sum_{k \in X^{J_0}} c_k^{J_0} \phi_k^{J_0}(x) + \sum_{j=J_0}^{J-1} \sum_{k \in Y^j} d_k^j \psi_k^j(x). \tag{2.4}$$

PDWT can be applied on $\mathbf{c}^j$ for $j = J.J-1, \ldots, J_0+1$ to obtain the **full diffusion wavelet transform** (FDWT) which will give all the coefficients for the expansion given by Eq. (2.4), see Fig. 2.7.

Next we want to construct the set $\mathbf{c}^j$ (i.e. the coordinates of the function $P_{\mathcal{V}^j} f$ in the basis Φ^j of the space $\mathcal{V}^j$) from the sets $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ (coefficients of the projection of the function f in the space $\mathcal{V}^{j-1}$ and in the space $\mathcal{W}^{j-1}$ respectively).

Since Φ^{j-1} is the basis of the space $\mathcal{V}^{j-1}$ and Ψ^{j-1} is the basis of the space $\mathcal{W}^{j-1}$, $\Phi^{j-1} \cup \Psi^{j-1}$ (call it Φ^{j1}) is the basis of $\mathcal{V}^j$ (because $\mathcal{V}^j = \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$) and the matrix $[\Phi^{j1}]_{\Phi^j}$ is given as

$$[\Phi^{j1}]_{\Phi^j} = \left(\begin{array}{c|c} [\Phi^{j-1}]_{\Phi^j} & 0 \\ \hline 0 & [\Psi^{j-1}]_{\Phi^j} \end{array} \right).$$

The coordinates of $P_{\mathcal{V}^j} f$ in the basis Φ^{j1} of the space $\mathcal{V}^j$ are $\begin{pmatrix} \mathbf{c}^{j-1} \\ \mathbf{d}^{j-1} \end{pmatrix}$ and hence the coordinates of $P_{\mathcal{V}^j} f$ in the basis Φ^j of the space $\mathcal{V}^j$ are given by

$$\mathbf{c}^j = \left(\begin{array}{c|c} [\Phi^{j-1}]_{\Phi^j} & 0 \\ \hline 0 & [\Psi^{j-1}]_{\Phi^j} \end{array} \right) \begin{pmatrix} \mathbf{c}^{j-1} \\ \mathbf{d}^{j-1} \end{pmatrix},$$

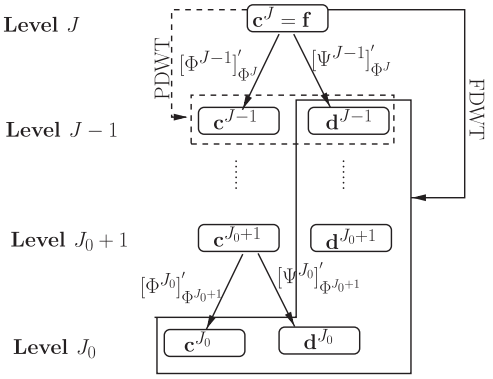


Fig. 2.7. PWDT and FDWT.

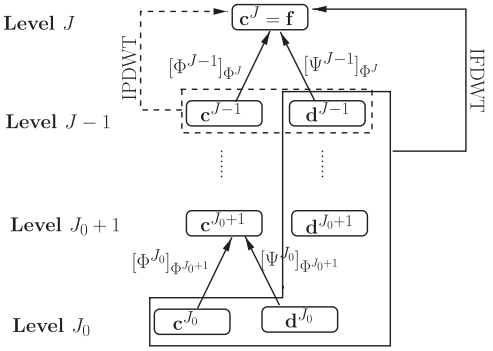


Fig. 2.8. IPDWT and IFDWT.

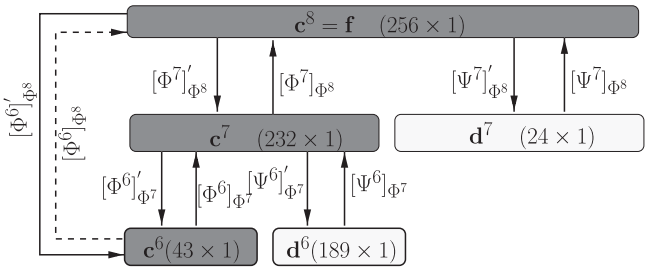


Fig. 2.9. Matrices involved in DST, IDST, PDWT, FDWT, IPDWT, IFDWT.

or

$$\mathbf{c}^j = [\Phi^{j-1}]_{\Phi^j}\mathbf{c}^{j-1} + [\Psi^{j-1}]_{\Phi^j}\mathbf{d}^{j-1}. \tag{2.5}$$

Eq. (2.5) is called **inverse partial diffusion wavelet transform** (IPDWT). Now suppose $j = J_0$ is the coarsest level then the **inverse full diffusion wavelet transform** (IFDWT) is obtained by applying IPDWT recursively. Fig. 2.8 gives us a clear understanding of IPDWT and IFDWT.

For the diffusion wavelet constructed on the interval [0, 1] with T from [23], $\tau = 10^{-9}$ and $N = 256$, Fig. 2.9 shows the matrices involved in DST, IDST, PDWT, FDWT, IPDWT and IFDWT. The matrices $[\Phi_7]'_{\Phi_8}$ and $[\Phi_6]'_{\Phi_8}$ are involved in DST and their transpose matrices ($[\Phi_7]_{\Phi_8}$ and $[\Phi_6]_{\Phi_8}$) in IDST. For PDWT from $j = 8$ to $j = 7$, we need the matrices $[\Phi_7]'_{\Phi_8}$, $[\Psi_7]'_{\Phi_8}$ and their transpose matrices ($[\Phi_7]_{\Phi_8}$, $[\Psi_7]_{\Phi_8}$) are required for IPWDT. For the FDWT from $j = 8$ to $j = 6$, the matrices $[\Phi_7]'_{\Phi_8}$, $[\Phi_6]'_{\Phi_7}$, $[\Psi_7]'_{\Phi_8}$ and $[\Psi_6]'_{\Phi_7}$ are required and their transpose matrices ($[\Phi_7]_{\Phi_8}$, $[\Phi_6]_{\Phi_7}$, $[\Psi_7]_{\Phi_8}$ and $[\Psi_6]_{\Phi_7}$) are required for IFWDT.

2.5. Vanishing moments

In Euclidean setting, vanishing moments are usually defined via orthogonality relations to the subspaces of polynomials upto a certain degree. For example, in case of Daubechies wavelet [24], the statement that there are M vanishing moments

implies that

$$\int_{-\infty}^{\infty} x^p \psi(x)dx = 0, \quad x \in \mathbb{R}, \quad p = 0, 1, \ldots, M-1.$$

Here in case of diffusion wavelet the vanishing moment of the scaling function ϕ_k^j [19] is defined as the number of functions in $\mathcal{W}^{j-1}$ to which $T^{2^j}\phi_k^j$ is orthogonal upto a precision τ.

Now since the size of $\mathcal{W}^{j-1}$ are determined by the precision τ and the rate of decay of the rank of the diffusion operator T used for the construction of diffusion wavelet. Hence the vanishing moments of the scaling functions are controlled by τ and the operator T used for the construction of the wavelet.

3. Fast diffusion wavelet method (FDWM)

In this section we will explain FDWM to solve the PDEs using diffusion wavelet.

General approach: Given a manifold X and a diffusion operator T on $\mathcal{L}_2(X)$ such that high powers of T have low numerical rank, a multiresolution analysis can be constructed on $\mathcal{L}_2(X)$ which leads to the construction of diffusion wavelet. Classes of operators which can be used for the construction of diffusion wavelet include approximation of second order differential operators. Given a differential equation on the manifold, we can approximate the differential operator involved in it to get the operator T which can be used for the construction of diffusion wavelet on that manifold (Note that we have used the finite different approximations of the operator involved). Then this wavelet is used for the computation of dyadic powers of the operator T which will be used to solve the given PDE numerically. This work may be considered as an extension of work done in [25], where Daubechies wavelet is used to solve PDEs on [0, 1] with periodic boundary conditions. The generalizations to deal with general boundary problems are yet to be developed. FDWM is a generalization of this work using diffusion wavelet.

3.1. Reconstruction and compression error

For any $f(x) \in \mathcal{L}_2(X)$ and a given threshold ϵ, Eq. (2.4) can be written as; $P_{\mathcal{V}^J} f(x) = f_{\geq\epsilon}(x) + f_{<\epsilon}(x)$, where

$$f_{\geq\epsilon}(x) = \sum_{k\in X^{J_0}} c_k^{J_0}\phi_k^{J_0}(x) + \sum_{j=J_0}^{J-1}\sum_{|d_k^j|\geq\epsilon} d_k^j\psi_k^j(x) \text{ and } f_{<\epsilon}(x) = \sum_{j=J_0}^{J-1}\sum_{|d_k^j|<\epsilon} d_k^j\psi_k^j(x).$$

The number of significant coefficients $N(\epsilon)$ is defined as; $N(\epsilon) = \#X^{J_0} + \sum_{j=J_0}^{J-1}\#\{d_k^j | k \in Y^j \text{ and } |\, d_k^j\,| \geq \epsilon\}$. $\|f - f_{\geq\epsilon}\|_p$ is called compression error and for $\epsilon = 0$ there is no compression and hence it is reconstruction error. We prove the following result.

Theorem 3.1. *For sufficiently smooth function f,*

$$\|f - f_{\geq\epsilon}\|_\infty \leq C\epsilon, \tag{3.1}$$

where C is a constant.

Proof. For the coefficients $\Phi = \{\{c_k^{J_0}\}_k, \{\{d_k^j\}_k\}_{j=J_0 J_0+1,\ldots}\}$, we define a norm as

$$\|\Phi\|_{b_{p,q}^\sigma} = \|c_k^{J_0}\|_{l^p} + \left(\sum_{j\geq J_0}\left(2^{js}\left(\sum_k |d_k^j|^p\right)^{\frac{1}{p}}\right)^q\right)^{\frac{1}{q}},$$

where $s = \sigma + \frac{1}{2} - \frac{1}{p}$. Also we define $\Phi_{\geq\epsilon} = \{\{c_{k,\geq\epsilon}^{J_0}\}_k, \{\{d_{k,\geq\epsilon}^j\}_k\}_{j=J_0 J_0+1,\ldots}\}$, with $c_{k,\geq\epsilon}^{J_0} = c_k^{J_0}$ and $d_{k,\geq\epsilon}^j = d_k^j \cdot 1_{|d_k^j|\geq\epsilon}$. The coefficients $\Phi_{\geq\epsilon}$ will reconstruct $f_{\geq\epsilon}$.

It should be noted that there exists a J such that $d_{k,\geq\epsilon}^j = 0, \forall j \geq J, k \in Y^j$ and this J can be defined as the unique real root of $2^{-j(\sigma+\frac{1}{2}-\frac{1}{p})}\|\Phi\|_{b_{p,q}^\sigma} = \epsilon 2^{-j/2}$, which implies that $J = (\sigma - \frac{1}{p})^{-1}\log(\frac{\|\Phi\|_{b_{p,q}^\sigma}}{\epsilon})$. Having said that such a J exists, we can write

$$\|f - f_{\geq\epsilon}\|_\infty \leq \sum_{j=J_0}^{J}\left\|\sum_k (d_k^j - d_{k,\geq\epsilon}^j)\psi_k^j\right\|_\infty + \sum_{j>J}\left\|\sum_k d_k^j\psi_k^j\right\|_\infty. \tag{3.2}$$

Now two points should be noted

$$\|d_k^j\|_\infty \leq 2^{-j(\sigma+\frac{1}{2}-\frac{1}{p})}\|\Phi\|_{b_{p,q}^\sigma}, \tag{3.3}$$

and

$$|d_k^j - d_{k,\geq\epsilon}^j| \leq \epsilon 2^{-\frac{j}{2}}. \tag{3.4}$$

Table 3.1
Difference in decay of dimension of $\mathcal{V}^j$s for different τs.

$\mathcal{V}^j$	$\dim_\tau(\mathcal{V}^j)$ for $\tau = 10^{-9}$	$\dim_\tau(\mathcal{V}^j)$ for $\tau = 10^{-2}$
$\mathcal{V}^{11}$	2048	2048
$\mathcal{V}^{10}$	2047	278
$\mathcal{V}^9$	2046	34
$\mathcal{V}^8$	886	15
$\mathcal{V}^7$	55	12
$\mathcal{V}^6$	16	8
$\mathcal{V}^5$	12	6

If M_1 is the maximum number of wavelets at any level j, touching at least a single point in the domain, M_2 is the maximum l_∞ norm of ψ_k^j, then using Eq. (3.4), the first term of Eq. (3.2) is bounded as

$$\sum_{j=J_0}^{J}\left\|\sum_k (d_k^j - d_{k,\geq\epsilon}^j)\psi_k^j\right\|_\infty \leq \sum_{j=J_0}^{J} M_1 M_2 \epsilon 2^{-j/2} 2^{j/2} \leq M_1 M_2 \epsilon J.$$

The second term of Eq. (3.2) is bounded as

$$\begin{aligned}\sum_{j>J}\left\|\sum_k d_k^j \psi_k^j\right\|_\infty &\leq \sum_{j>J} M_1 M_2 \|d_k^j\|_\infty 2^{j/2},\\ &\leq \sum_{j>J} M_1 M_2 2^{-j(\sigma+\frac{1}{2}-\frac{1}{p})}\|\Phi\|_{b_{p,q}^\sigma} 2^{j/2},\\ &\leq M_1 M_2 \|\Phi\|_{b_{p,q}^\sigma}\frac{2^{-J(\sigma-\frac{1}{p})}}{(1-2^{-(\sigma-1/p)})},\\ &= M_1 M_2 \frac{\epsilon}{(1-2^{-(\sigma-1/p)})}.\end{aligned}$$

Using these results Eq. (3.2) gives us

$$\|f - f_{\geq\epsilon}\|_\infty \leq M_1 M_2 \epsilon J + M_1 M_2 \frac{\epsilon}{(1-2^{-(\sigma-1/p)})},$$

or

$$\|f - f_{\geq\epsilon}\|_\infty \leq M_1 M_2 \epsilon (J + \frac{1}{(1-2^{-(\sigma-1/p)})}),$$

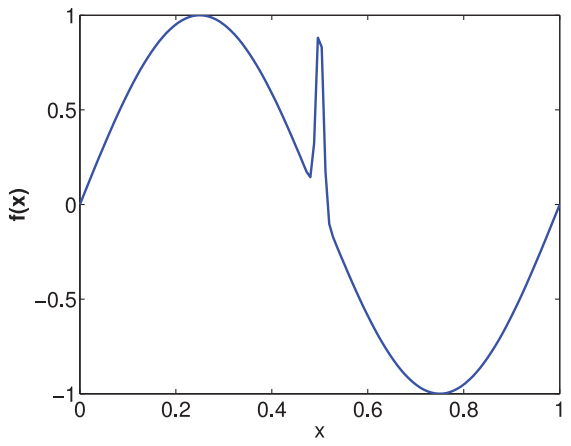
or

$$\|f - f_{\geq\epsilon}\|_\infty \leq C\epsilon.$$

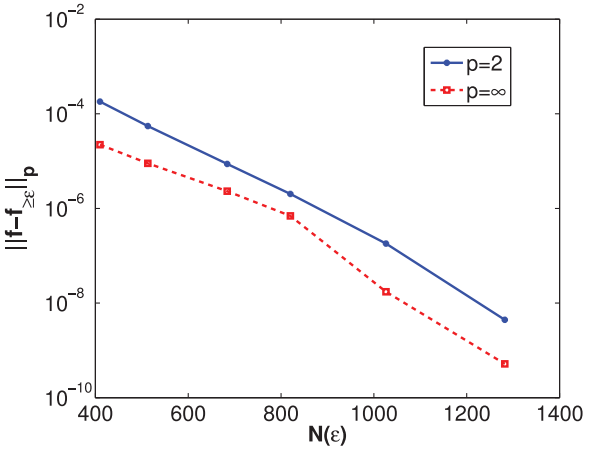
□

- **Test function 1:** $\sin(2\pi x) + e^{-\alpha(x-.5)^2}$, with $\alpha = 10^4$ on the interval [0, 1] shown in Fig. 3.1(a). For $N = 2^{11} = 2048$ we call the finest level as $J = 11$. Following results are obtained:
  - Reconstruction error is of the order 10^{-14}.
  - Fig. 3.1(b) shows the relation between the number of significant coefficients i.e. $N(\epsilon)$ and compression error i.e. $\|f - f_{\geq\epsilon}\|_p$. A good compression can be observed from the graph. Fig. 3.1(c) shows the graph between the compression error $\|f - f_{\geq\epsilon}\|_p$ and the threshold ϵ. It can be observed that (3.1) is verified.
  - Fig. 3.1(d) shows the relation between the compression error $\|f - f_{\geq\epsilon}\|_p$ and the precision τ used in the construction of the wavelet (75% wavelet coefficients are used for reconstruction). It is clear from the graph that the compression error increases if we increase τ.
  - Fig. 3.1(e) shows the graph between $N(\epsilon)$ and J. It can be observed that $N(\epsilon)$ increases with increase in the value of J but a stage comes where further increase in J does not effect $N(\epsilon)$. The smallest such value of J will be the optimal value of J. For example for test function 1, the optimal value of J is 11.

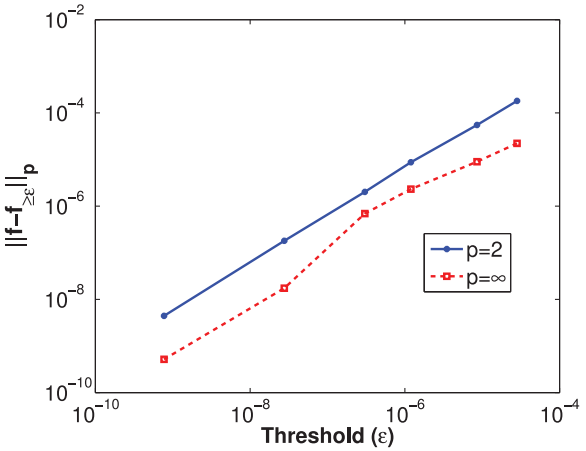
  It is to be noted that if we choose τ very small say for example 10^{-9}, compression error will be small but the decay in the dimension of $\mathcal{V}^j$s in this case will be slow as compared to the decay with $\tau = 10^{-2}$ say. This thing can be observed from Table 3.1. The slow decay in dimension of $\mathcal{V}^j$s means the computational cost will be high. So we have to choose τ optimally.
- **Test function 2:** $f(x) = -\tanh(\frac{x+x_0}{2\nu}) + e^{-64^2(x-x_0)^2}$, with $x_0 = \frac{1}{3}$ and $\nu = 10^{-3}$ shown in Fig. 3.2(a), we obtained the following results:



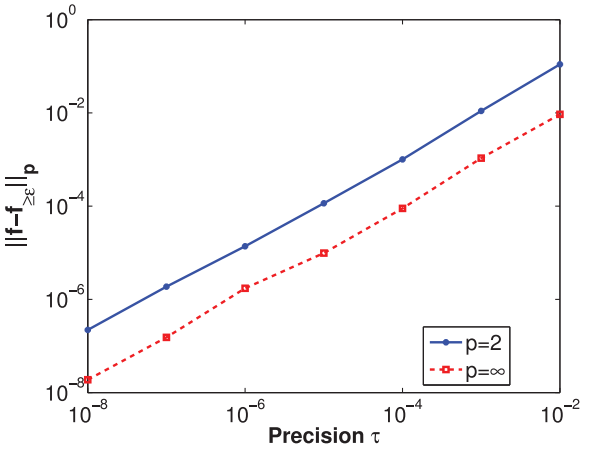
(a) The test function 1.



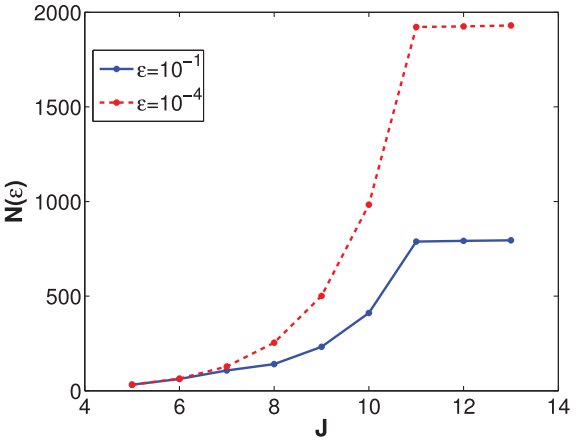
(b) Compression error versus number of significant coefficients $N(\epsilon)$.



(c) Compression error versus threshold ϵ.

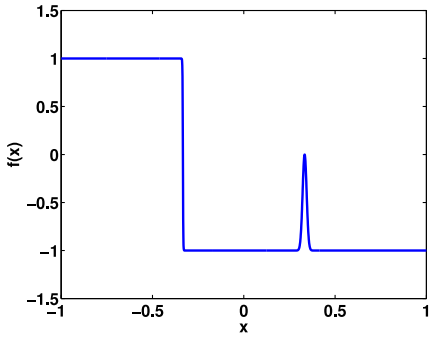


(d) Compression error versus the precision τ.

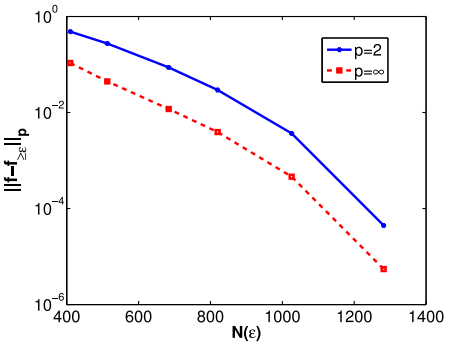


(e) $N(\epsilon)$ versus J.

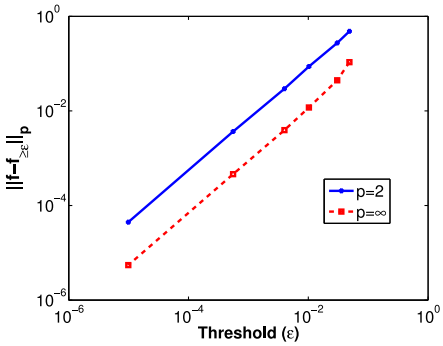
Fig. 3.1. Results for test function 1.



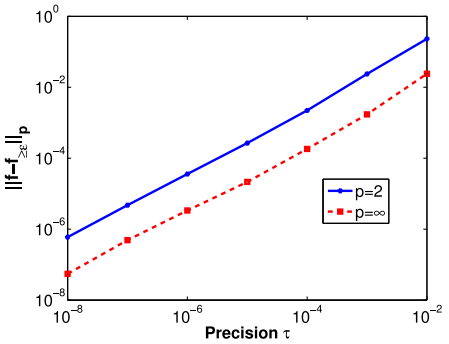
(a) The test function 2.



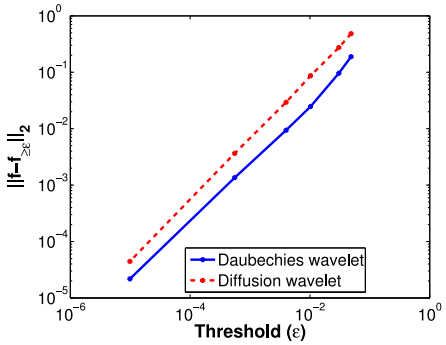
(b) Compression error versus number of significant coefficients $N(\epsilon)$.



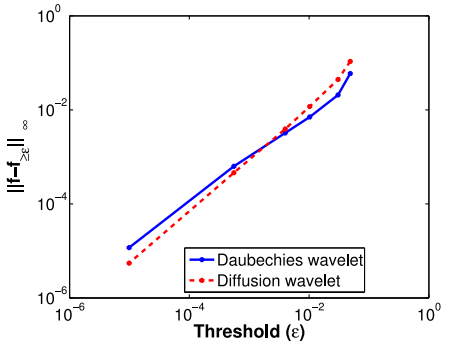
(c) Compression error versus threshold ϵ.



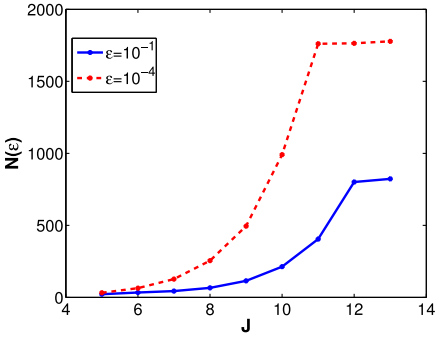
(d) Compression error versus the precision τ.



(e) Comparison with Daubechies wavelet (in l_2 norm).



(f) Comparison with Daubechies wavelet (in l_∞ norm).



(g) $N(\epsilon)$ versus J.

Fig. 3.2. Results for test function 2.

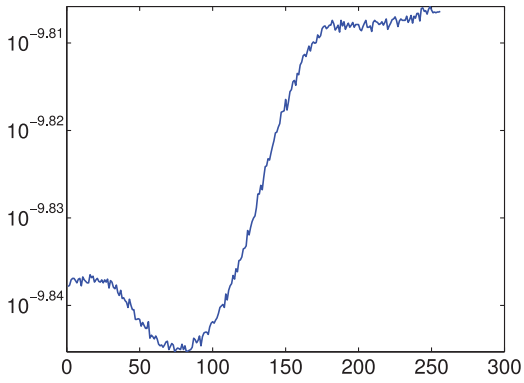


Fig. 3.3. Point wise error between analytic and numerical value of $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$.

- Reconstruction error is of the order 10^{-14}.
- Fig. 3.2(b) shows the compression error as a function of number of significant coefficients. Fig. 3.2(c) shows the graph between the compression error and threshold ϵ.
- Fig. 3.2(d) shows the relation between the compression error and the precision τ (75% coefficients are used for reconstruction).
- Fig. 3.2(e) and Fig. 3.2(f) compares the compression error in case of diffusion wavelet and Daubechies wavelet of order 4.
- Fig. 3.2(g) shows the graph between $N(\epsilon)$ and J.

3.2. Efficient computation of $\{T^{2^m}, m > 0\}$

Suppose we are given a function $f \in \mathcal{L}_2(X)$ and we want to compute $T^{2^m}f$. Note that $T^{2^m}f \approx [T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$. We have the result $[T^{2^m}]_{\Phi^J}^{\Phi^{J-m}} = [T^{2^{m-1}}]_{\Phi^{J-(m-1)}}^{\Phi^{J-m}}[T^{2^{m-2}}]_{\Phi^{J-(m-2)}}^{\Phi^{J-(m-1)}}\cdots[T^2]_{\Phi^{J-1}}^{\Phi^{J-2}}[T]_{\Phi^J}^{\Phi^{J-1}}[T]_{\Phi^J}^{\Phi^J}$. Also, we know that $[T^{2^m}]_{\Phi^J}^{\Phi^{J-m}}\mathbf{f}$ will give us the coordinates of $[T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$ in the basis Φ^{J-m} of the space $\mathcal{V}^{J-m}$, and these coordinates are nothing but scaling function coefficients of $[T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$ in the space $\mathcal{V}^{J-m}$ i.e. $\mathbf{c}^{J-m}$. From $\mathbf{c}^{J-m}$ we can compute $\mathbf{c}^J$ using IDST. $\mathbf{c}^J$ is nothing but vectors of coefficients of $[T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$ in the basis Φ^J which is $[T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$ itself. Algorithm to compute $[T^{2^m}]_{\Phi^J}^{\Phi^J}\mathbf{f}$ is:

$T^{2^m}f \approx \mathbf{c}^J$ = **ALGORITHM**(T,f,m)

- $g = [T]_{\Phi^J}^{\Phi^J}\mathbf{f}$.
- **For** $k = 0, 1, \cdots m-1$
- $g = [T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}g$
- **end**

} This gives us $\mathbf{c}^{J-m}$

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

We will call the length of $\mathbf{c}^{J-m}$ as the **number of significant coefficients** required to obtain $T^{2^m}f$. We chose $f(x) = x$ and computed $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$ (T is the diffusion operator constructed using the construction of [23]) both analytically (Note that $[T^{2^{12}}]_{\Phi^8}^{\Phi^8} = ([T]_{\Phi^8}^{\Phi^8})^{2^{12}}$ and analytical computation means the matrix $[T]_{\Phi^8}^{\Phi^8}$ is multiplied 12 times and finally with $\mathbf{f}$) and numerically with the help of the algorithm explained above. Fig. 3.3 shows the point wise error between analytic and numerical value of $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$. The error in l_2 and l_∞ norm are 2.393×10^{-9} and 1.563×10^{-10} respectively.

In this case, if we calculate $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$ analytically (i.e. without exploiting the fact that higher powers of T have low numerical rank), then we have to multiply a 256×256 matrix 12 times and finally the multiplication of a 256×256 matrix with a 256×1 matrix. While calculating $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$ using diffusion wavelet we are exploiting that higher powers of T have low numerical rank. From Table 3.2 it can be observed that we need to multiply twelve matrices to get $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$ and ten of these are of the size less than 20% of the size of T. The CPU time taken for the numerical calculation of $[T^{2^{12}}]_{\Phi^8}^{\Phi^8}\mathbf{f}$ is 2% of the CPU time taken for its analytic calculation.

Table 3.2
Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$.

k	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	256 × 256	256 × 256
0	251 × 256	45 × 256
1	241 × 251	16 × 45
2	60 × 241	12 × 16
3	16 × 60	8 × 12
4	12 × 16	6 × 8
5	9 × 12	4 × 6
6	7 × 9	3 × 4
7	5 × 7	2 × 3
8	4 × 5	1 × 2
9	3 × 4	1 × 1
10	2 × 3	1 × 1
11	2 × 2	1 × 1

3.3. FDWM for parabolic differential equation

Consider the one dimensional parabolic differential equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(a(x)\frac{\partial u}{\partial x}\right) + f(x), \quad x \in [0,1], \tag{3.5}$$

with an initial condition $u(x,0) = u_0(x)$ and with suitable boundary conditions which can be Dirichlet, Neumann, periodic or mixed boundary conditions. After time discretization Eq. (3.5) becomes

$$\mathcal{A}u^n = \mathcal{C}u^{n-1} + \Delta t f, \quad u^0 = u_0, \tag{3.6}$$

where u^n is an approximation of u at time $t = n\Delta t$, and $\mathcal{A}$ and $\mathcal{C}$ are differential operators linked to the chosen time discretization scheme. For example (after setting $a(x) = \nu$ in Eq. (3.5)) with the explicit forward Euler's scheme we get

$$\mathcal{A} = I \quad \mathcal{C} = \left(I + \nu\Delta t\frac{\partial^2}{\partial x^2}\right), \tag{3.7}$$

with the implicit backward Euler scheme we get

$$\mathcal{A} = \left(I - \nu\Delta t\frac{\partial^2}{\partial x^2}\right) \quad \mathcal{C} = I, \tag{3.8}$$

and with the implicit second-order Crank Nicolson scheme

$$\mathcal{A} = \left(I - \nu\Delta t\frac{\partial^2}{\partial x^2}\right) \quad \mathcal{C} = \left(I + \nu\Delta t\frac{\partial^2}{\partial x^2}\right). \tag{3.9}$$

In all the cases $\mathcal{A}$ is an invertible elliptic operator. Now after the space discretization the problem will move into a finite dimensional space. In this finite dimensional space the problem becomes

$$A\mathbf{u}^n = C\mathbf{u}^{n-1} + \Delta t\mathbf{f}, \quad \mathbf{u}^0 = \mathbf{u}_0, \tag{3.10}$$

where $\mathbf{u}^n$ is the vector of all the unknowns $u_j^n, \quad j = 1, \ldots, N$ at time $t = n\Delta t$ and $\mathcal{A}$ and $\mathcal{C}$ are replaced by respectively with A and C which are finite matrices. Eq. (3.10) can be written as

$$\mathbf{u}^n = A^{-1}\left(\frac{\|C\|}{\|C\|}C\mathbf{u}^{n-1} + \Delta t\mathbf{f}\right) = A^{-1}\left(\|C\|B\mathbf{u}^{n-1} + \Delta t\mathbf{f}\right), \tag{3.11}$$

where $B = \frac{C}{\|C\|}$. Using Eq. (3.11) recursively we can obtain

$$\mathbf{u}^n = \|C\|^n(A^{-1})^nB^n\mathbf{u}^0 + \sum_{k=0}^{n-1}\|C\|^k(A^{-1})^kB^k\Delta tA^{-1}\mathbf{f}. \tag{3.12}$$

In particular when $A = I$, Eq. (3.12) can be written as

$$\mathbf{u}^n = \|C\|^nB^n\mathbf{u}^0 + \sum_{k=0}^{n-1}\|C\|^kB^k\Delta tA^{-1}\mathbf{f}. \tag{3.13}$$

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

$$\begin{aligned}\sum_{k=0}^{n-1} \|C\|^k B^k \Delta t \mathbf{f} &= \sum_{k=0}^{2^m-1} \|C\|^k B^k \Delta t \mathbf{f},\\ &= (I + \|C\|B + \|C\|^2 B^2 + \|C\|^3 B^3 + \cdots + \|C\|^{2^m-1} B^{2^m-1})\Delta t \mathbf{f},\\ &= \prod_{k=0}^{m-1} (I + \|C\|^{2^k} B^{2^k})\Delta t \mathbf{f}.\end{aligned}$$

Hence Eq. (3.13) can be written as

$$\mathbf{u}^{2^m} = \|C\|^{2^m} B^{2^m} \mathbf{u}^0 + \prod_{k=0}^{m-1} (I + \|C\|^{2^k} B^{2^k})\Delta t \mathbf{f}. \tag{3.14}$$

The matrix B satisfies all the assumptions given in Section 2.1, so it is suitable for the construction of diffusion wavelet (Note that there are several choices for the operator T used in the construction of diffusion wavelet. One of the choice is that $I - T$ (I is the identity operator of size of T) should converge to Laplace operator [19]). We construct diffusion wavelet using the operator B and then the dyadic powers of B required in Eq. (3.14) can be computed efficiently as explained in Section 3.2 and using this the differential equation can be solved very efficiently.

When the matrix $A \neq I$, the matrix A is of the form $A = I - T$, where T is suitable for the construction of the diffusion wavelet. After constructing the diffusion wavelet using the operator T, we can efficiently compute the dyadic powers of T. Hence A^{-1} can be computed as follows

$$A^{-1} = (I - T)^{-1} = \sum_{k=0}^{\infty} T^k = \lim_{K\to\infty} \prod_{k=0}^{K} (I + T^{2^k}). \tag{3.15}$$

The above method can be easily extended to two-dimensional case where the spatial domain is a square and to three-dimensional case where the domain is a cube.

3.4. FDWM for elliptic differential equation

Consider the Poisson differential equation with Dirichlet's boundary condition

$$\nabla^2 u = f(x), \quad 0 \le x \le 1, \quad u(0) = \alpha, \quad u(1) = \beta. \tag{3.16}$$

Discretize Eq. (3.16) using central finite difference method, to get

$$\begin{aligned}\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} &= f_i, \quad i = 2, 3, \ldots, N-1,\\ u_1 = \alpha \quad u_N &= \beta.\end{aligned} \tag{3.17}$$

Eq. (3.17) can be written in the matrix form as follows:

$$B\mathbf{u} = \mathbf{f}, \tag{3.18}$$

where $\mathbf{f} = (\Delta x)^2[f_1 - \frac{u_1}{\Delta x^2}, f_2, \ldots, f_{N-2}, f_{N-1} - \frac{u_N}{\Delta x^2}]'$. Suppose $\lambda_{\max}$ be the eigenvalue of the matrix B with maximum absolute value. Divide Eq. (3.18) with $\lambda_{\max}$

$$\frac{1}{\lambda_{\max}} B\mathbf{u} = \frac{1}{\lambda_{\max}} \mathbf{f} \text{ or } A\mathbf{u} = \mathbf{f}, \tag{3.19}$$

where $A = \frac{1}{\lambda_{\max}} B$ and $\mathbf{f} = \frac{1}{\lambda_{\max}} \mathbf{f}$. Consider the matrix $T = I - A$, where I is identity matrix. This matrix T satisfies the conditions given in Section 2.1 and hence can be used to construct the diffusion wavelet on [0, 1].

Having found the operator T, Eq. (3.19) can be written as

$$(I - T)\mathbf{u} = \mathbf{f} \Rightarrow \mathbf{u} = (I - T)^{-1}\mathbf{f}. \tag{3.20}$$

Eq. (3.20) implies that

$$\mathbf{u} = (I - T)^{-1}\mathbf{f} = \sum_{k=0}^{\infty} T^k \mathbf{f} = \lim_{K\to\infty} \prod_{k=0}^{K} (I + T^{2^k})\mathbf{f}. \tag{3.21}$$

Because the operator T is such that high powers of it have low numerical rank, we can terminate the series of Eq. (3.21) after some terms depending on the precision required. Now we can evaluate the $T^{2^j}, j \ge 0$ efficiently using the algorithm explained in Section 3.2.

Table 4.1
Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 1.

k	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[B]_{\Phi^9}^{\Phi^9}$, 511 × 511	511 × 511
0	510 × 511	510 × 511
1	510 × 510	491 × 510
2	510 × 510	419 × 491
3	487 × 510	329 × 419
4	422 × 487	230 × 329
5	332 × 422	175 × 230
6	247 × 332	125 × 175
7	181 × 247	83 × 125
8	128 × 181	59 × 83
9	91 × 128	42 × 59
10	65 × 91	29 × 42
11	46 × 65	24 × 29
12	32 × 46	20 × 24
13	22 × 32	18 × 20
14	16 × 22	16 × 18

4. Numerical results

Here we discuss the numerical results obtained using FDWM on six test problems. To decide the optimal value of J, we observe the variation of $N(\epsilon)$ with J. Also note that LU factorization is used for the solution of the system of linear algebraic equations arising from discretization by finite difference method.

Test problem 1: Take $a(x) = \frac{1}{9}$ and $f(x) = 0$ in Eq. (3.5) with following initial and boundary conditions

$$u(x,0) = 2\sin(3\pi x), \quad u(0,t) = 0 \text{ and } u(1,t) = 0, \quad t \geq 0.$$

Discretization of the PDE using the scheme (3.7) in time and central finite difference method in space will give us

$$\mathbf{u}^{n+1} = C\mathbf{u}^n + \mathbf{f},$$

where $\mathbf{f} = \left[\lambda u_1^n, 0, \ldots, 0, \lambda u_N^n\right]'$, with both $u_1^n = u_N^n = 0$ (boundary conditions). Here the boundary conditions are incorporated in the vector $\mathbf{f}$. Since $\mathbf{f} = 0$ we get

$$\mathbf{u}^n = \|C\|^n B^n \mathbf{u}^0 = B^n(\|C\|^n \mathbf{u}^0),$$

where $B = \frac{C}{\|C\|}$ is a matrix of size $(N-2) \times (N-2)$.

If $N = 513$, then the matrix B will be a 511 × 511 matrix. If we want to compute $\mathbf{u}^{2^{15}}$ which is solution u at the time $t = 2^{15}\Delta t$, then we need to find $B^{2^{15}}$. Number of operations required to multiply two matrices (one of size $m \times p$ and the other one of size $p \times n$) is $O(mpn)$. So if we naively calculate $B^{2^{15}}$ then the number of operations will be $O(512^3) + \cdots +$ 15times $+ \cdots + O(512^3)$. Computing $B^{2^{15}}(\|A\|^n \mathbf{u}^0)$ using diffusion wavelet requires the multiplication of the matrices given in Table 4.1. Multiplication of these matrices will give us $\mathbf{c}^{-6}$ (size 16 × 1) which is the vector of scaling function coefficients of $\mathbf{g} = B^{2^{15}}(\|A\|^{2^{15}} \mathbf{u}^0)$ in the space $\mathcal{V}^{-6}$, on which we will apply IDST i.e. multiplication with $[\Phi^{-6}]_{\Phi^9}$ (size 511 × 16) to obtain $\mathbf{c}^9$ which is an approximation of $\mathbf{g}$. Following are the steps required for the calculation of $\mathbf{u}^{2^{15}}$:

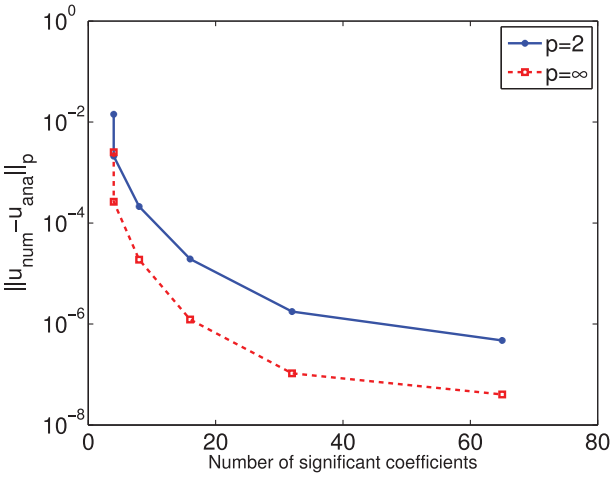
- Let $\mathbf{g} = B^{2^{15}}(\|C\|^{15}\mathbf{u}^0)$ and $\mathbf{U}^0 = \|C\|^{2^{15}}\mathbf{u}^0$.
- $\mathbf{c}^9$= **ALGORITHM**(B,$\mathbf{U}^0$,**15**).
- $\mathbf{u}^{2^{15}} = \mathbf{g} \approx \mathbf{c}^9$.

Fig. 4.1 (a) shows the error ($\|u_{num} - u_{ana}\|_p$, where u_{num} stands for the numerical solution of the differential equation and u_{ana} stands for the analytic solution of the differential equation) at the time $t = 2^{15}\Delta t = 9$, as a function of number of significant coefficients. Fig. 4.1(b) shows the graph between $\|u_{num} - u_{ana}\|_p$ and τ. Fig. 4.1(c) shows the analytic and numerical solution of the Eq. (3.5) at the time $t = 9$. From Fig. 4.1(d) we can observe the diffusion in above case with time.

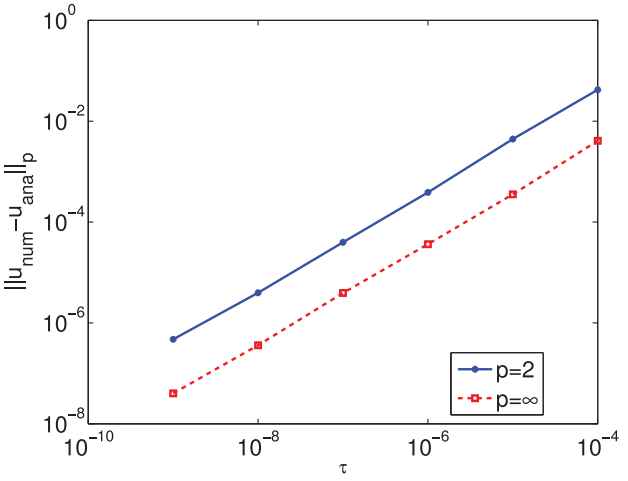
Fig. 4.2 compares the CPU time taken by the method FDWM developed in this paper with the CPU time taken by finite difference method for computing the solution at the time $t = 2^m\Delta t$. We can observe from the Fig. 4.2 that at the time $t = 2^{13}\Delta t = 2.25$, the CPU time taken by FDWM is 3.75% of the CPU time taken by finite difference method. Fig. 4.3 shows the number of elements in B^{2^k} with magnitude larger than τ versus k. It can be observed from the figure that the matrix B^{2^k} becomes more and more dense with increase in the value of k for both central FDM (central finite difference method) and spectral method. But in case of FDWM the matrix B^{2^k} becomes sparser as k increases.

Test problem 2: Here, we take

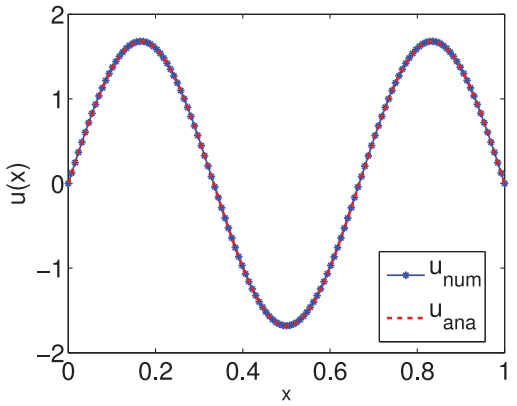
$$a(x) = 0.5 + 0.25\sin(2\pi x), \quad f(x) = -\pi^2\cos(2\pi x)^2 + \pi^2(0.5 + 0.25\sin(2\pi x))\sin(2\pi x),$$



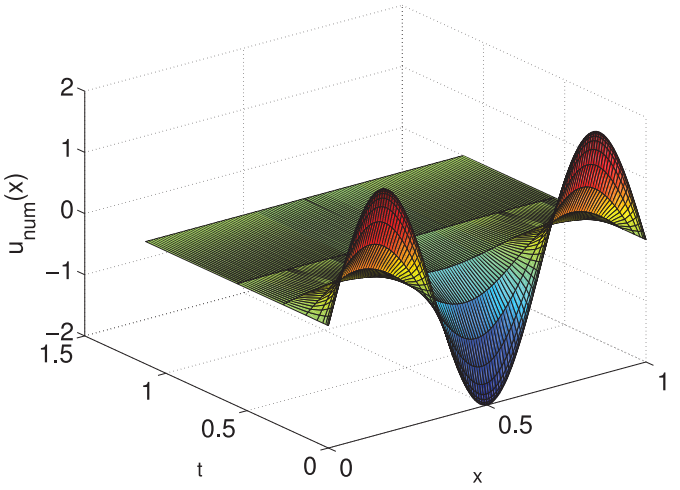
(a) $\|u_{num} - u_{ana}\|_p$ versus number of significant coefficients.



(b) Compression error versus τ.



(c) Numerical and analytic solution at the time $t = 9$.



(d) Diffusion with time.

Fig. 4.1. Results for test problem 1.

in Eq. (3.5) with initial and boundary condition

$$u(x,0) = u_0(x) = \sin(4\pi x) \quad u(0,t) = u(1,t).$$

Discretizing Eq. (3.5) using FTCS

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{a(x_j)(u_{j+1}^n - u_j^n) - a(x_{j-1})(u_j^n - u_{j-1}^n)}{\Delta x^2} + f(x_j),\ j = 1, 2, \ldots, N-1, \tag{4.1}$$

with $a(x_0) = a(x_{N-1}) = a_{N-1}$ and $u(x_0) = u(x_{N-1}) = u_{N-1}$ (because of periodicity of $a(x)$ and $u(x)$). Eq. (4.1) can be written as

$$\mathbf{u}^{n+1} = C\mathbf{u}^n + \mathbf{f}, \tag{4.2}$$

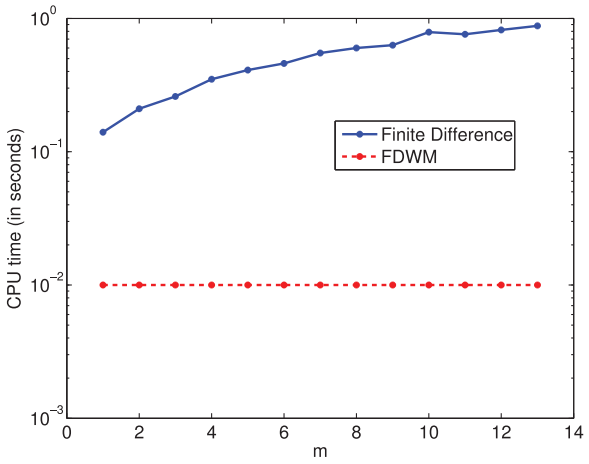


Fig. 4.2. CPU time for computing the solution at the time $t = 2^m \Delta t$ for test problem 1.

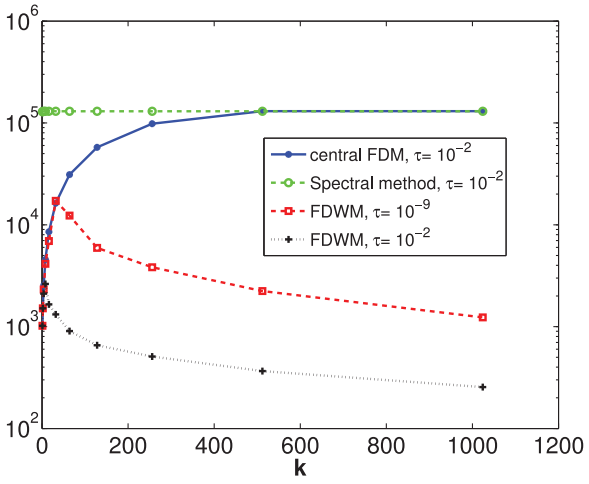


Fig. 4.3. Number of non zeros elements in B^{2^k} with magnitude larger than τ versus k for test problem 1.

where

$$C = \begin{bmatrix} 1-\lambda(a_1+a_{N-1}) & \lambda a_1 & \cdots & 0 & \lambda a_{N-1} \\ \lambda a_1 & 1-\lambda(a_1+a_2) & \cdots & \cdots & 0 \\ 0 & \lambda a_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1-\lambda(a_{N-2}+a_{N-1}) & \lambda a_{N-1} \\ \lambda a_{N-1} & 0 & \cdots & \lambda a_{N-2} & 1-\lambda(a_{N-1}+a_{N-2}) \end{bmatrix},$$

and

$$\mathbf{f} = \Delta t[f_1, f_2, f_3, \cdots, f_{N-2}, f_{N-1}]'.$$

Note that here the boundary conditions are incorporated in the matrix A. Eq. (4.2) can be further written as

$$\mathbf{u}^{n+1} = \frac{\|C\|}{\|C\|} A\mathbf{u}^n + \mathbf{f} = \|C\| B\mathbf{u}^n + \mathbf{f}. \tag{4.3}$$

If $n = 2^m$ for some m then Eq. (4.3) can be written as

$$\mathbf{u}^{2^m} = B^{2^m}(\|C\|^{2^m}\mathbf{u}^0) + \prod_{k=0}^{m-1}(I + \|C\|^{2^k}B^{2^k})\mathbf{f}.$$

Table 4.2
Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 2.

k	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[B]_{\Phi^9}^{\Phi^9}$, 512×512	512×512
0	512×512	512×512
1	512×512	500×512
2	512×512	413×500
3	512×512	302×413
4	506×512	193×302
5	426×506	133×193
6	318×426	85×133
7	203×318	58×85
8	141×203	35×58
9	99×141	20×35
10	69×99	15×20
11	49×69	14×15
12	35×49	12×14
13	25×35	12×12
14	17×25	12×12

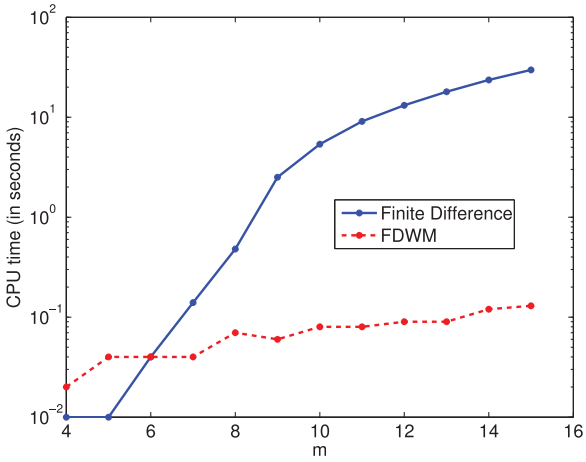


Fig. 4.4. CPU time for computing the solution at the time $t = 2^m \Delta t$ for test problem 2.

The dyadic powers of B are calculated using the algorithm described in Section 3.2. Table 4.2 shows the sizes of different dyadic powers of B (size 512×512) in this case. Following are the steps required for the calculation of $\mathbf{u}^{2^{15}}$:

- Let $\mathbf{g} = B^{2^{15}}(\|C\|^{15}\mathbf{u}^0)$ and $\mathbf{U}^0 = \|C\|^{15}\mathbf{u}^0$.
- $\mathbf{c}^9$= **ALGORITHM**(B,$\mathbf{U}^0$,**15**).
- $\mathbf{g} \approx \mathbf{c}^9$.

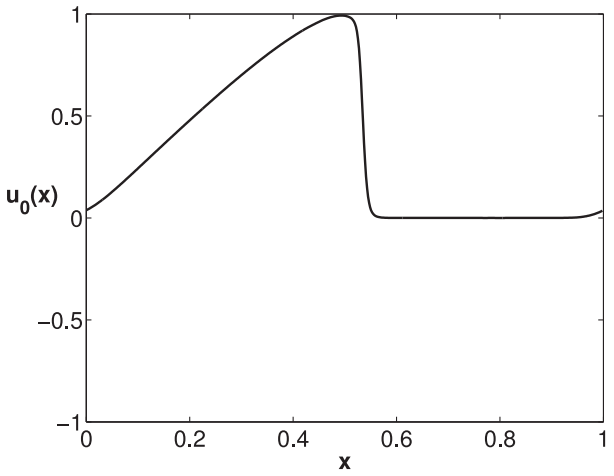
- Let $\mathbf{h} = \prod_{k=0}^{15-1}(I + \|C\|^{2^k}B^{2^k})\mathbf{f}$ and $\mathbf{h1} = \mathbf{f}$.
- **For** $k = 0, 1, \cdots, 15-1$
  h2= **ALGORITHM**(B,**h1**,k),
  h1= **h1**+$\|C\|^{2^k}$**h2**,
  End
- **h1**= $\mathbf{c}^9$ (scaling function coefficients of **h**).
- $\mathbf{h} \approx$ **h1**.

- $\mathbf{u}^{2^{15}} \approx \mathbf{g} + \mathbf{h}$.

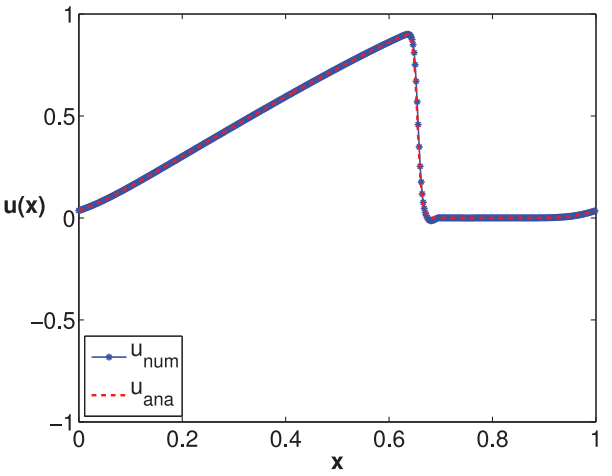
Fig. 4.4 compares the CPU time taken by FDWM with the CPU time taken by finite difference method for computing the solution at the time $t = 2^m \Delta t$. We can observe from Fig. 4.4 that at the time $t = 2^{15}\Delta t = 0.0312$, the CPU time taken by FDWM is 0.4% of the CPU time taken by finite difference method.

Test problem 3: Consider Burger's equation

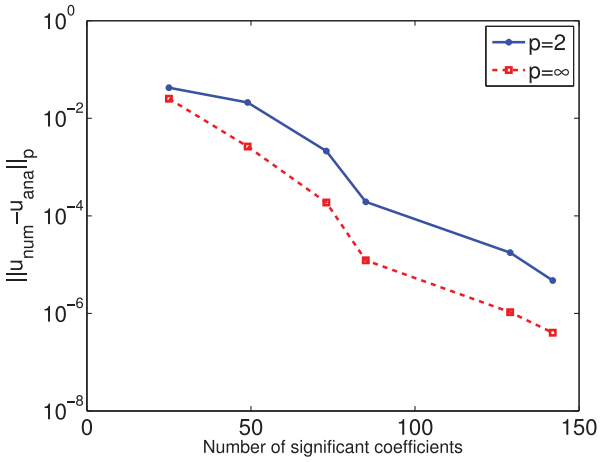
$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}, \tag{4.4}$$



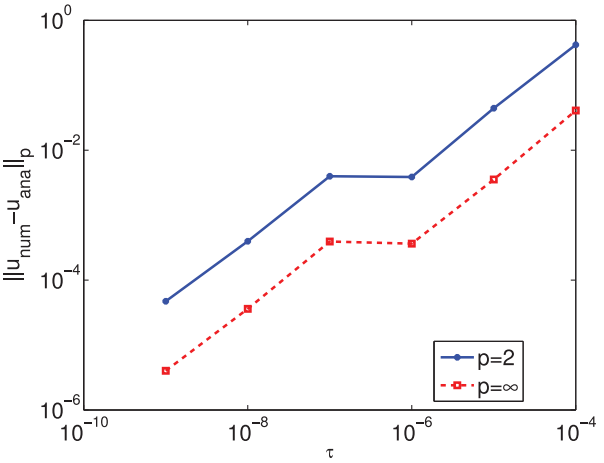
(a) $u_0(x)$.



(b) Numerical and analytic solution at $t = 0.5$.



(c) $\|u_{num} - u_{ana}\|_p$ versus number of significant coefficients.



(d) Compression error versus τ.

Fig. 4.5. Results for test problem 3.

Table 4.3
Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 3.

k	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$	k	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[T]_{\Phi^9}^{\Phi^9}$, 511 × 511	511 × 511	7	85 × 102	139 × 187
0	495 × 511	511 × 511	8	69 × 85	105 × 139
1	424 × 495	511 × 511	9	59 × 69	83 × 105
2	319 × 424	489 × 511	10	55 × 59	63 × 83
3	256 × 219	426 × 489	11	54 × 55	56 × 63
4	171 × 256	336 × 426	12	42 × 54	40 × 56
5	128 × 171	256 × 336	13	34 × 42	32 × 40
6	102 × 128	187 × 256	14	22 × 34	26 × 32

with periodic boundary condition and with the non-smooth initial condition

$$u_0(x) = \begin{cases} \sin(2\pi x) & \text{if } x \le 1/2 \\ 0 & \text{if } x > 1/2, \end{cases}$$

shown in Fig. 4.5(a). Here we used second order Crank Nicolson scheme for time discretization and hence $A \neq I$. The Table 4.3 shows the size of dyadic powers of the matrix B (size of B is 511 × 511 in our case). Fig. 4.5(c) shows the error (i.e. $\|u_{num} - u_{ana}\|_p$ at $t = 0.5$, as a function of number of significant coefficients. Fig. 4.5(d) shows the graph between

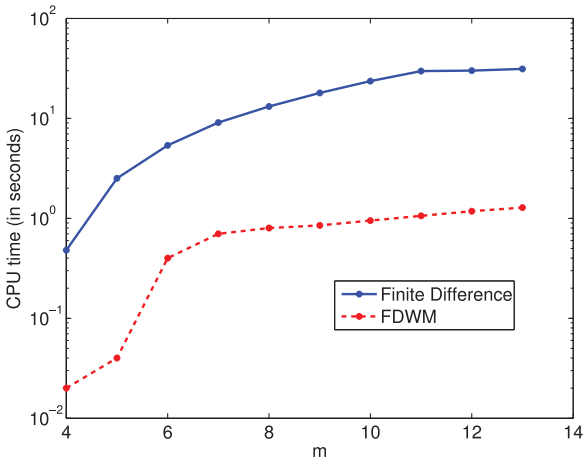
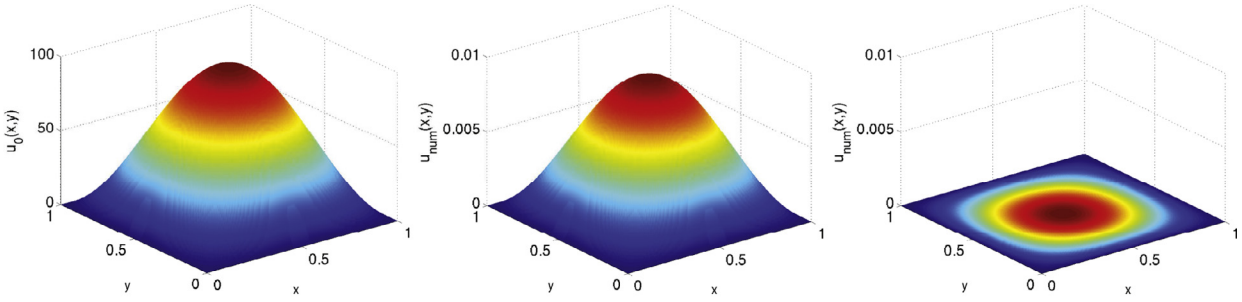


Fig. 4.6. CPU time for computing the solution at the time $t = 2^m \Delta t$ for test problem 3.



(a) The solution at the time $t = 0$. (b) The solution at the time $t = 0.4702$. (c) The solution at the time $t = 3.7616$.

Fig. 4.7. The solution for test problem 4 at different times.

$\|u_{num} - u_{ana}\|_p$ and τ. Fig. 4.5(b) shows the analytic and numerical solution of Eq. (4.4) at the time $t = 0.5$. Fig. 4.6 compares the CPU time taken by the method FDWM developed in this paper with the CPU time taken by finite difference method for computing the solution at the time $t = 2^m \Delta t$.

Test problem 4: We take the two dimensional diffusion equation given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \quad 0 \le x, y \le 1, \tag{4.5}$$

with the initial condition $u(x, y, t = 0) = u_0(x, y) = 100 \sin(\pi x) \sin(\pi y)$ and boundary conditions $u(x = 0, y, t) = u(x = 1, y, t) = u(x, y = 0, t) = u(x, y = 1, t) = 0$. Fig. 4.7 shows the solution of the problem at the time $t = 0$, $t = 0.4702$ and $t = 3.7616$. Fig. 4.8(a) shows the error (i.e. $\|u_{num} - u_{ana}\|_p$) at time $t = 0.4702$ as a function of number of significant coefficients. Fig. 4.8(b) shows the graph between $\|u_{num} - u_{ana}\|_p$ and τ. Fig. 4.8(c) shows the solution of the problem at $y = 0.5$ at different time steps, diffusion process is clearly visible from this figure. Fig. 4.8(d) compares the CPU time taken by FDWM with the CPU time taken by finite difference method for computing the solution at the time $t = 2^m \Delta t$. We can observe from Fig. 4.8(d) that at the time $t = 2^{12} \Delta t = 0.4702$, the CPU time taken by FDWM is 2% of the CPU time taken by finite difference method. If B is the matrix involved, then Table 4.4 shows the size of dyadic powers of the matrix B (size of B is 1024×1024 in our case).

Test problem 5: We take the three dimensional diffusion equation given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \quad 0 \le x, y, z \le 1, \tag{4.6}$$

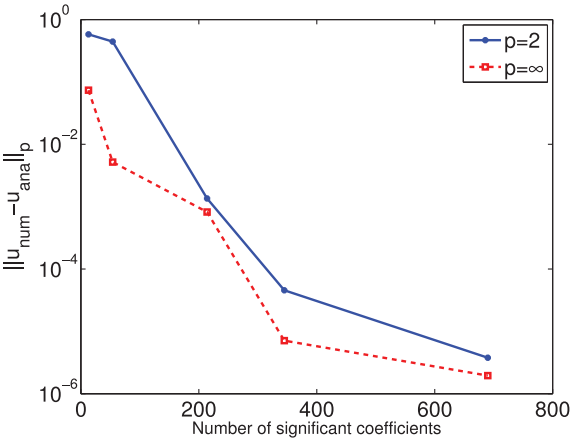
with the initial condition

$$u(x, y, z, t = 0) = u_0(x, y, z) = 100 \sin(\pi x) \sin(\pi y) \sin(\pi z),$$

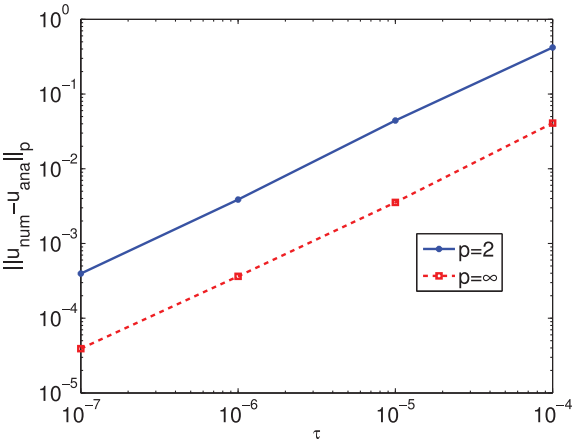
and boundary conditions

$$u(x = 0, 1, y, z, t) = u(x, y = 0, 1, z, t) = u(x, y, z = 0, 1, t) = 0.$$

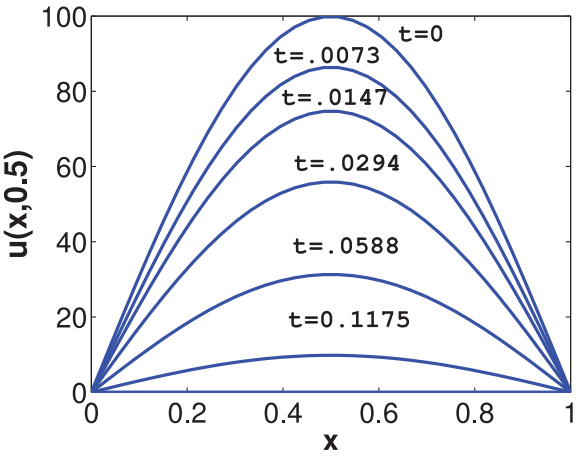
Fig. 4.9(a) shows the error (i.e. $\|u_{num} - u_{ana}\|_p$) at the time $t = 0.128$ as a function of number of significant coefficients. Fig. 4.9(b) shows the graph between $\|u_{num} - u_{ana}\|_p$ and τ. Fig. 4.9(c) compares the CPU time taken by FDWM with the CPU



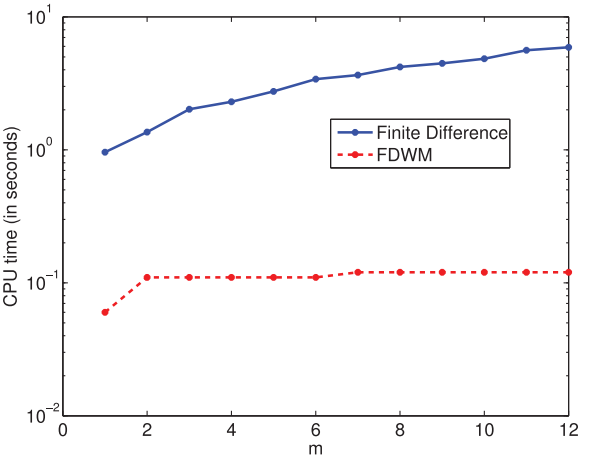
(a) $\|u_{num} - u_{ana}\|_p$ versus number of significant coefficients.



(b) $\|u_{num} - u_{ana}\|_p$ versus τ.



(c) $u(x, 0.5)$.

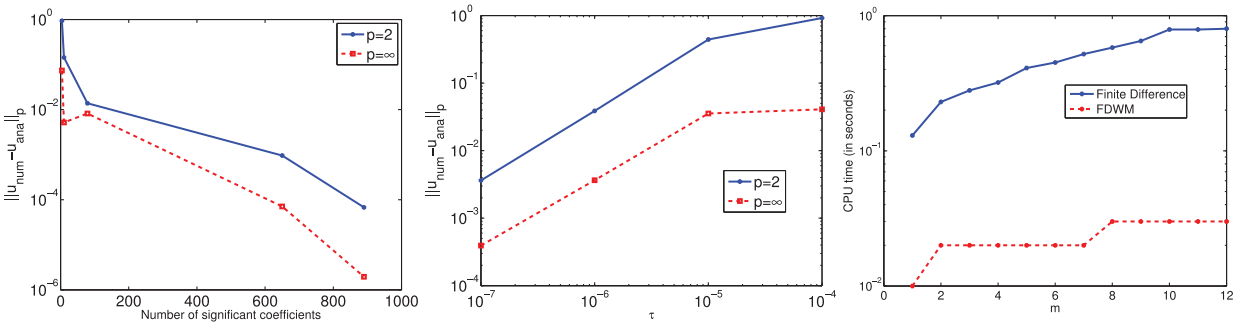


(d) CPU time for computing the solution at the time $t = 2^m \Delta t$.

Fig. 4.8. Results for test problem 4.

Table 4.4
Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 4.

k	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[B]_{\Phi^{10}}^{\Phi^{10}}$, 1024 × 1024	1024 × 1024
0	1024 × 1024	1024 × 983
1	1024 × 1023	983 × 814
2	1023 × 980	814 × 395
3	980 × 823	395 × 172
4	823 × 473	172 × 79
5	473 × 214	79 × 42
6	214 × 102	42 × 23
7	102 × 51	23 × 14
8	51 × 24	14 × 7
9	24 × 11	7 × 6
10	11 × 6	6 × 5

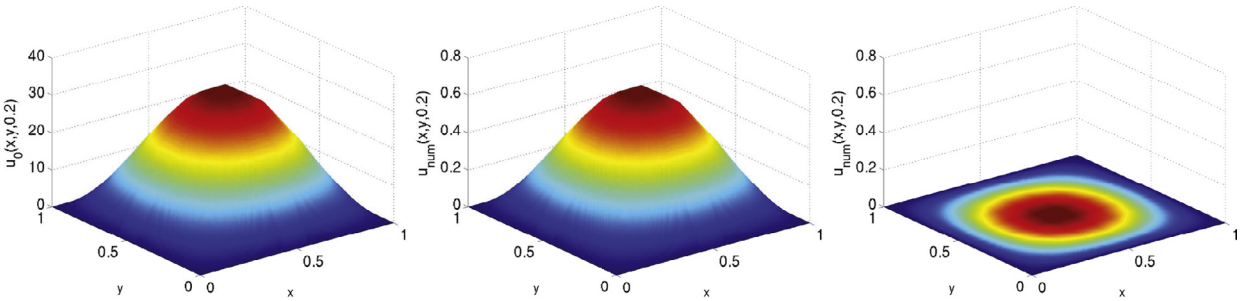


(a) $\|u_{num}-u_{ana}\|_p$ versus number of significant coefficients.

(b) $\|u_{num}-u_{ana}\|_p$ versus τ.

(c) CPU time for computing the solution at the time $t=2^m\Delta t$.

Fig. 4.9. Results for test problem 5.

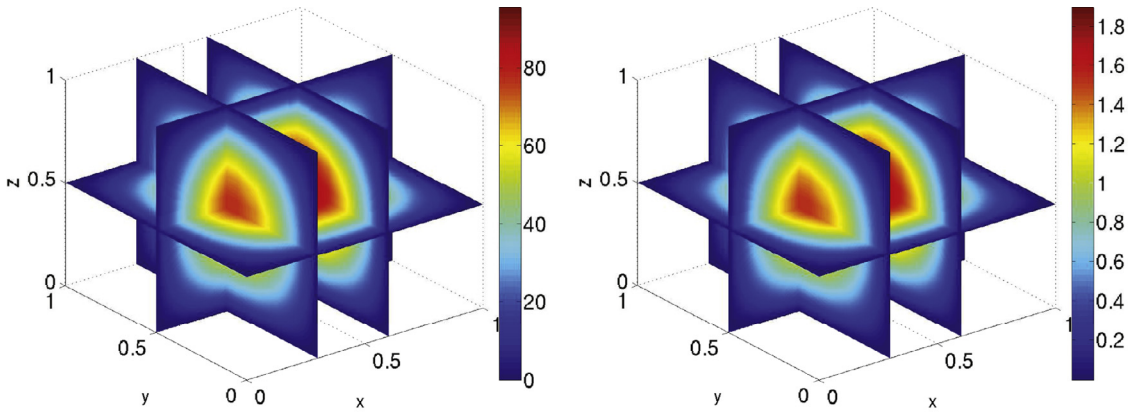


(a) $u(x,y,0.2)$ at the time $t=0$.

(b) $u(x,y,0.2)$ at the time $t=0.128$.

(c) $u(x,y,0.2)$ at the time $t=4.096$.

Fig. 4.10. The solution u(x,y,0.2) for test problem 5 at different times.



(a) $u(x,y,z)$ at the time $t=0$.

(b) $u(x,y,z)$ at the time $t=0.128$.

Fig. 4.11. The solution $u(x, y, z)$ for test problem 5 at different times.

time taken by finite difference method for computing the solution at the time $t=2^m\Delta t$. We can observe from Fig. 4.9(c) that at the time $t=2^{12}\Delta t=4.096$, the CPU time taken by FDWM is 3% of the CPU time taken by finite difference method.

Fig. 4.10 shows the solution of the problem at $z=0.2$ for $t=0$, $t=0.128$ and $t=4.096$.

Fig. 4.11 shows the solution of the problem for the slices made at $x=\{0.3,0.6\}$, $y=0.5$ and $z=0.5$ at the time $t=0$, $t=0.128$. We also computed the solution at the time $t=4.096$ and it is completely diffused as expected.

If B is the matrix involved, then Table 4.5 shows the size of dyadic powers of the matrix B (size of B is 512 × 512 in our case).

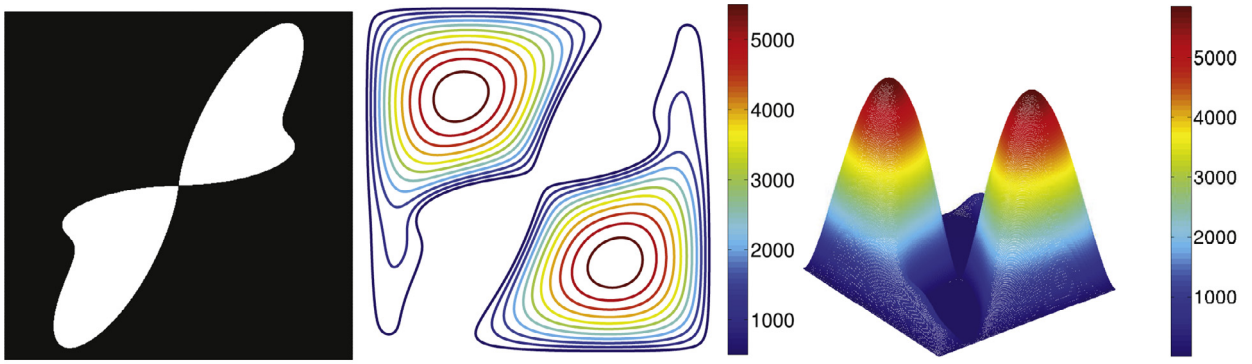
Test problem 6: In Eq. (3.16) take $f(x)=5$ with $u(0)=0$ and $u(1)=1$. Table 4.6 shows sizes of matrices $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for different k for $N=513$. From Table 4.6 we can observe that the decay in size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ is slow for $\tau=10^{-9}$ as

Table 4.5
Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 5.

k	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[B^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[B]_{\Phi^9}^{\Phi^9}$, 512×512	512×512
0	512×512	512×508
1	512×512	508×463
2	512×508	463×217
3	508×455	217×55
4	455×254	55×19
5	254×79	19×8
6	79×23	8×4
7	23×10	4×4
8	10×4	4×2
9	4×1	2×1
10	1×1	1×1

Table 4.6
Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 6.

k	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[T]_{\Phi^9}^{\Phi^9}$, 512×512	512×512
0	510×512	512×512
1	509×510	510×512
2	487×509	494×510
3	424×487	421×494
4	332×424	321×421
5	249×332	255×321
6	181×249	170×255
7	129×181	126×170
8	93×129	85×126
9	65×93	64×85
10	46×65	42×64



(a) Domain of the solution. (b) Contour plot of the solution. (c) Mesh plot of the solution.

Fig. 4.12. Results for test problem 7.

compared to $\tau = 10^{-2}$ as expected (discussed in Section 3.1). Moreover, we can also see the reduction in the size of the matrices $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$.

Test problem 7: We considered the Poisson equation ($\nabla^2 u = 1$) with homogeneous Dirichlet's boundary condition on the domain shown in Fig. 4.12(a) which is exterior of a butterfly in a square. We have taken 512 points for the discretization of the domain. Fig. 4.12(b) and (c) shows the contour and mesh plots of the solution of the problem. Table 4.7 shows the decay in size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$. The CPU time taken by FDWM is 5% of the CPU time taken by finite difference method.

Table 4.7
Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for test problem 7.

k	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-9}$	Size of $[T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}}$ for $\tau = 10^{-2}$
	$[T]_{\Phi^9}^{\Phi^9}$, 512×512	512×512
0	501×512	512×512
1	491×501	477×512
2	472×491	399×477
3	414×472	387×399
4	323×414	311×387
5	245×323	201×311
6	173×245	150×201
7	111×173	125×150
8	87×111	75×125
9	60×87	58×75
10	39×60	27×58

5. Conclusion and future work

In this paper a fast diffusion wavelet method is developed for numerical solution of PDEs. The operator T obtained by discretizing the differential operators present in PDE using finite difference method is used for the construction of diffusion wavelet. The compression error with respect to different parameters involved in the construction of diffusion wavelet is tested for two test functions. This diffusion wavelet is used for computing dyadic powers of the operator T which are used for fast computation of the numerical solution of PDE. The convergence of the FDWM is verified and the CPU time taken by FDWM is compared with the CPU time taken by finite difference method. To the best of our knowledge the useful properties of diffusion wavelet constructed in [19] are for the first time exploited for the numerical solution of PDEs. Generalizing this method to a general manifold will be direction of our future work.

Acknowledgments

The first author would like to thank Council of Scientific and Industrial Research for providing PhD scholarship. The authors also thank Prof. Nicholas Kevlahan and Dr. Siddhartha P. Chakrabarty for their valuable comments.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.apm.2015.10.054

References

[1] A. Barinka, T. Barsch, P. Charton, A. Cohen, S. Dahlke, W. Dahmen, K. Urban, Adaptive wavelet schemes for elliptic problems - implementation and numerical experiments, SIAM J. Sci. Comput. 23 (1999) 910–939.
[2] M. Mehra, B.V.R. Kumar, Time-accurate solution of advection–diffusion problems by wavelet-Taylor-Galerkin method, Commun. Numer. Methods Eng. 21 (2005) 313–326.
[3] J. Liandrat, P. Tchamitchian, Resolution of the 1D regularized Burgers equation using a spatial wavelet approximation, NASA Contactor Report 187480: ICASE report No. 90-83, 1990.
[4] O.V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.
[5] K. Schneider, F. Chemie, T. Chemie, J. Fröhlich, J. Frohlich, An adaptive wavelet-vaguelette algorithm for the solution of PDEs, J. Comput. Phys. 130 (1997) 90–174.
[6] W. Dahmen, K. Urban, J. Vorloeper, Adaptive wavelet methods: basic concepts and applications to the stokes problem, World Scientific, Singapore, 2002, pp. 39–80.
[7] W. Dahmen, R. Schneider, Wavelets on manifolds i: construction and domain decomposition, SIAM J. Math. Anal. 31 (1999) 184–230.
[8] A. C.Canuto, K.Urban, The wavelet element method. part i: construction and analysis, Appl. Comput. Harm. Anal. 6 (1999) 1–52.
[9] W. Dahmen, R. Stevenson, Element-by-element construction of wavelets satisfying stability and moment conditions, SIAM J. Numer. Anal. 37 (1999) 319–352.
[10] W. Freeden, M. Schreiner, Orthogonal and non-orthogonal multiresolution analysis, scale discrete and exact fully discrete wavelet transform on the sphere, Constr. Approx. 14 (1997) 493–515.
[11] W. Sweldens, The lifting scheme: a construction of second generation wavelets, SIAM J. Math. Anal. 29 (1998) 511–546.
[12] M. Mehra, N.K.-R. Kevlahan, An adaptive wavelet collocation method for the solution of partial differential equations on the sphere, J. Comput. Phys. 227 (2008) 5610–5632.
[13] O.V. Vasilyev, S. Paolucci, M. Sen, A multilevel wavelet collocation method for solving partial differential equations in a finite domain, J. Comput. Phys. 120 (1995) 33–47.
[14] S. Bertoluzza, G. Naldi, A wavelet collocation method for numerical solution of partial differential equations, Appl. Comput. Harmonic Anal. 3 (1996) 1–9.
[15] K. Urban, Wavelet methods for elliptic partial differential equations, Oxford University press, 2009.
[16] A.K. W. Dahmen, K. Urban, Biorthogonal spline wavelets on the interval-spline and moment conditions, Appl. Comput. Harmonic Anal. 6 (1999) 132–196.
[17] M. Primbs, New stable biorhogonal spline-wavelets on the interval, Result Math. 57 (2010) 121–162.

[18] D. Cerna, V. Finek, Construction of optimally conditioned cubic spline wavelets on the interval, Adv. Comput. Math. 34 (2011) 219–252.
[19] R.R. Coifman, M. Maggioni, Diffusion wavelets, Appl. Comput. Harmonic Anal. 21 (2006) 53–94.
[20] J. Haupt, W.U. Bajwa, M. Rabbat, R. Nowak, Compressed sensing for networked data, Signal Process. Mag. IEEE 25 (2008) 92–101.
[21] M. Sridhar, M. Maggioni., Value function approximation with diffusion wavelets and laplacian eigenfunctions, Adv. Neural Inf. Process. Syst. (2005) 843–850.
[22] I. Daubechies, Ten lectures on wavelets, SIAM, Philadelphia, 1992.
[23] R.R. Coifman, S. Lafon, Diffusion maps, Appl. Comput. Harmonic Anal. 21 (2006) 5–30.
[24] O.M. Nilsen, Wavelets in scientific computing, Ph.D. thesis, Technical University of Denamark, Lyngby, 1998.
[25] B. Enguist, S. Osher, S. Zhong, Fast wavelet based algorithms for linear evolution equations, SIAM J. Sci. Comput. 15 (1994) 755–775.