Applied Numerical Mathematics 148 (2020) 28–44

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

A curvelet method for numerical solution of partial differential equations

Deepika Sharma [a], Kavita Goyal [a,*], Rohit Kumar Singla [b]

[a] School of Mathematics, Thapar Institute of Engineering and Technology, Patiala, India
[b] Mechanical Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India

ARTICLE INFO

Article history:
Received 4 May 2018
Received in revised form 25 June 2019
Accepted 29 August 2019
Available online 4 September 2019

Keywords:
Wavelet
Partial differential equations
Curvelet
Curvelet based numerical methods

ABSTRACT

This paper proposes a fast curvelet based finite difference method for numerical solutions of partial differential equations (PDEs). The method uses finite difference approximations for differential operators involved in the PDEs. After the approximation, the curvelet is used for the compression of the finite difference matrices and subsequently for computing the dyadic powers of these matrices required for solving the PDE in a fast and efficient manner. As a prerequisite, compression and reconstruction errors for the curvelet have been tested against different parameters. The developed method has been applied on five test problems of different nature. For each test problem the convergence of the method is examined. Moreover, to measure the performance of the proposed method the computational time taken by the proposed method is compared to that of the finite difference method. It is observed that the proposed method is computationally very efficient.

1. Introduction

Partial differential equations (PDEs) are extensively appearing for modeling real life situations associated with a wide range of fields such as financial markets, biological sciences, climate prediction etcetera. Wavelet bases were introduced as an alternative to the Fourier bases. The former has obvious and significant advantages over the later, such as absence of Gibb's effect, having a compact support etc. [27]. After looking at the usefulness of wavelets in the areas of signal and image processing [25,26], people stared using wavelets and other multi-scale representations for numerically solving PDEs [11,12, 21,29,32,31,18–20]. The most advantageous property of wavelet is that, it is possible to represent a large class of functions with fewer wavelet coefficients. This property leads to the popularity of wavelets in PDEs community. Apart from the above mentioned property, various other mathematical properties such as compact support, existence of fast wavelet transform, vanishing moments, algorithms for compression and regularization of differential operators, worked as icing on the cake [3,2]. For a large category of PDEs, wavelet methods are already designed. One can refer to the following papers for details: advection diffusion problems [28], Laplace/Poisson equations [1], reaction-diffusion equations [30], Burger's equation [23,33] and Stokes equation [16]. It can be noted that because of the advantages of wavelet bases over Fourier bases, wavelet based methods are better than Fourier methods such as FFT method for solving PDEs. Wavelet based numerical algorithms have the following advantages:

* Corresponding author.
E-mail addresses: deepika.sharma@thapar.edu (D. Sharma), kavita@thapar.edu (K. Goyal), rohit.kumar@thapar.edu (R.K. Singla).

https://doi.org/10.1016/j.apnum.2019.08.029
0168-9274/

1. The localization of wavelets, both in space and scale, governs an efficient sparse representation of differential operators (and its inverses) and functions by taking non-linear thresholding of wavelet coefficients.
2. The best marvelous characteristic of study of wavelet for numerically solving PDEs is their capability to evaluate the local regularity of the result, as by local mesh refinement, it accepts self-adaptive discretizations. Moreover, the evaluation of function spaces with respect to coefficients of wavelet function and the associated standard norm equivalence [13,22] approves preconditioning of diagonal operators in wavelet space.
3. The differential operator can be directly computed in a wavelet domain with high speed and accuracy by setting threshold values in the domain of wavelet.
4. The presence of the FWT methods yield calculations with ideal linear complexity.

Despite of the many advantages of wavelets, they have some limitations such as poor orientation sensitivity. To mitigate these limitations, curvelet had been introduced by Demanet and Candes [5]. Curvelets are attractive, because they effectively describe essential problems in which wavelet ideas are not upto mark.

1. Because of the bad orientation selectivity of wavelets, they do not present higher-dimensional irregularities efficiently. This makes curvelets interesting as they can yield an ideally adapted numerical construction to represent functions that display smooth punctuated curve.
2. Wavelets are not sufficient to detect, classify or present an intermediate dimensional arrangement with a compressed description. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with ideas from classical multiscale analysis.
3. Wavelets are not optimal for solving PDEs on complex geometries. Several constructions of wavelets on complex geometries exist. In [14,10] wavelet bases are developed in light of certain form of geometries that could be presented as separate union of smooth parametric images of a unit cube. It has numerous drawbacks from a practical viewpoint as its formation depends entirely on smooth parametrization of the standard cube. This issue is settled in [15], where finite element supported wavelet bases respecting an arbitrary initial triangularization are developed. In [17] wavelets are constructed on a sphere. In spite of vast literature available on arbitrary manifolds, the subject of wavelet based methods for numerically solving PDEs on complex geometries is yet in its emerging phase. Beauty of curvelet is that it can be designed on arbitrary manifolds.
4. Wavelets are optimal for solving elliptical PDEs, but not for hyperbolic PDEs. For hyperbolic PDEs, curvelets provide better results. The hyperbolic solution operator's curvelet representation is efficient as well as ideally sparse.

Curvelets are the basis elements which are extremely sensitive to direction and are highly anisotropic. The application of a curvelet for solving PDEs is one of the recommended future scopes in [9], the locus of this work. The most significant characteristic of the curvelet is that it can be designed on complex geometries. Therefore, it can be used for finding solution of PDEs on any type of manifolds. Recently, Ying et al. [34,9] has extended the curvelet transform to three dimensions. J. Ma is dealing with curvelet based finite difference techniques for seismic wave equation with his students [24]. The objective is to develop a fast adaptive technique for numerically figure out the wave propagation. The article [4] is the first in a projected series to show how the curvelet transform structure could be exploited.

In this work, we are marching ahead a little in the direction of use of curvelets in PDEs. We have proposed fast curvelet based finite difference method which exploits the compression property of the curvelet for sparse representation of the finite difference matrices corresponding to the differential operators involved in PDEs. The proposed method is tested on five test problems and the run time of the proposed method (obtained from **cputime** command of Matlab) is compared with that of the finite difference method (FDM). It is found that the proposed method is computationally efficient.

The paper is divided into various sections such that section 2 briefly discusses curvelets. In Section 3, behavior of the compression error with respect to different variables involved in curvelet transform has been tested for two test functions. Section 4 gives information about how curvelets are used to solve PDEs. In section 5, the numerical results of proposed method on five test problems of different nature are discussed. Section 6 gives the conclusion and a few possible future directions.

2. A short explanation of curvelet

2.1. Drawbacks of classical multiscale approaches

Our aim in this work is to rose curvelets for solving PDEs. In that area, we endeavor a sparse representation of the finite difference matrices associated with PDE. Over the past two decades, multiscale approaches such as multi-grid, fast multi-polar techniques in wavelets, finite element techniques with or without adaptive refining have been widespread. All these descriptions mean references to approximately isotropic elements in all positions as well as in scales; the template is re-scaled and handled essentially in the same way in all areas. Isotropic scaling might be useful if the function under examination has no particular highlights along the orientations chosen. Tools from hereditary multi-scale analysis such as wavelets are insufficient to detect, establish or present a compact presentation of the intermediate dimensional arrangement. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with

ideas from classical multiscale analysis. Curvelets can produce a geometrical framework that is ideally suited to describe functions that demonstrate punctuated smoothness of the curve.

2.2. Definition of curvelets

Curvelets are waveforms that have anisotropic behavior in fine layers, with an efficient support that accepts the parabolic postulate, length $\approx$ width^2 [4]. As with wavelets, there is a discrete as well as continuous transformation of the curvelet. A curvelet is classified by three quantities particularly, a parameter of orientation, say, $\theta, \theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$; a scaling parameter $j, 0 < j < 1$ and a position parameter k where $k \in R^2$. At level a, the class of curvelet is produced by dilate, translate, revolution of a fundamental component φ_j

$$\varphi_{j,k,\theta}(x) = \varphi_j(R_\theta(x-k)),$$

where $\varphi_j(x)$ is any type of spatial wavelet with spatial breadth $\sim j$ and spatial height $\sim \sqrt{j}$, (the symbol $\sim$ stands for proportional) among the minor axis facing in the horizontal position, which is described as

$$\varphi_j(x) \approx \varphi(D_j x),$$

here $D_j = \begin{pmatrix} 1/j & 0 \\ 0 & 1/\sqrt{j} \end{pmatrix}$ is the parabolic-scaling matrix and $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is a 2×2 rotation matrix with θ angle.

A significant feature of curvelets is that it obeys the fundamental law of harmonic analysis, which states that evaluating and restoring an arbitrarily function $f(x_1, x_2)$ as superpositions of that models is reasonable. It is acceptable to create stiff curvelet-frames as well as an arbitrarily function $f(x_1, x_2)$ can certainly be extended as an orthonormal set of curvelets. There is a discretization of angle/location/scale proceeding at a straightforward level of analysis, that approximately goes like $j_a = 2^{-a}, a = 0, 1, ..., \theta_{a,l} = 2\pi l.2^{-\lfloor a/2 \rfloor}, l = 0, 1, ..., 2^{\lfloor a/2 \rfloor} - 1$, and $b_k^{(a,l)} = R_{\theta_{a,l}}(k_1 2^{-j}, k_2 2^{-a/2})$, $k_1, k_2 \in Z$, the set φ_μ is a tight frame, here μ indexing the triplets $(j_a, \theta_{a,l}, k_b^{(a,l)})$. It implies

$$f = \sum_\mu < f, \varphi_\mu > \varphi_\mu, \qquad \|f\|_2^2 = \sum_\mu | < f, \varphi_\mu > |^2. \tag{1}$$

When a scale is given, curvelets φ_μ are achieved by implementing translations and orientations of the "mother" curvelet $\varphi_{a,0,0}$. In the frequency region,

$$\hat{\varphi}_{a,0,0}(\xi) = 2^{-3a/4} W(2^{-a}|\xi|) V(2^{\lfloor a/2 \rfloor}\theta).$$

In this case W, V is of compact support with continuous windows at interval $[1, 2]$ and $[-1/2, 1/2]$ respectively. Curvelets exists in the frequency region adjacent to an oriented rectangle R of width 2^{-a} and length $2^{-a/2}$ while in the spatial region, they are positioned in a parabolic drive of width $2^{a/2}$ and length 2^a with an orthogonal orientation to R. The combined localization in frequency as well as in space permits us to study curves such as the use of a "Heisenberg cell" in phase space and parabolic scales in these two realms.

2.3. Importance of curvelets over wavelets

The curvelets give optimum sparseness for "curve-punctuated continuous" function, wherever the function is continuous excluding discontinuities together with C^2 curves [6,7]. The degree of decline in the m-term estimate (reconstructing the function by employing m no. of coefficients) of the data is estimated to be sparse. A sparse description, along with better compression and additional sparseness for the remodeling of the denoising performance, increases the number of smooth functional areas. The curvelet transform is designed in such a fashion that maximum energy of the function is limited in only a few coefficients. This is measurable. There is clearly no basis for a function's coefficients with an uncertain curve of singularity to decline more quickly than in a frame of curvelet. This decline estimate is fast as compared to any other known system, included wavelets. The improved decline in the coefficient provides an optimally sparse representation, which is useful for solving the PDE.

2.4. Continuous-time curvelet transform

We begin with the set of $W(r)$ and $V(t)$ windows, which we refer to as the "radial window" and the "angular window". These two windows are non-negative, continuous and real-valued, with W getting non-negative real arguments along with carried out on $r \in [1/2, 2]$ and V using real arguments along with carried out on $t \in [-1, 1]$. The both windows will meet the requirements for admissibility, *i.e.*,

$$\sum_{j=-\infty}^{j=\infty} W^2(2^j r) = 1,\ r > 0; \qquad \sum_{l=-\infty}^{l=\infty} V^2(t-l) = 1,\ t \in R.$$

Presently, for every $j \geq j_0$, we start with the U_j frequency window, described in the Fourier domain as

$$U_j(r,\theta) = 2^{-3j/4} W(2^{-j}r) V\left(\frac{2^{\lfloor j/2 \rfloor}\theta}{2\pi}\right), \tag{2}$$

here $\lfloor \frac{j}{2} \rfloor$ is the integer portion of $\frac{j}{2}$. Therefore, the U_j support is a polar "wedge" described as the W and V support, the angular and the radial-window, used in all directions with scale-dependent window widths.

Set the waveform $\varphi_j(x)$ with the aid of its Fourier transformation $\hat{\varphi}_j(w) = U_j(w)$. We can consider φ_j as a "mother" curvelet on scale 2^{-j} means that, all curvelets on this scale, are achieved by translations as well as rotations of φ_j. Define,

- the equi-spaced series of rotated angles $\theta_l = 2\pi .2^{-\lfloor j/2 \rfloor}.l$ with $l = 0, 1, \ldots\ldots$ means that $0 \leq \theta_l < 2\pi$.
- the sequence of the $k = (k_1, k_2)$ translation parameters of the Z^2.

We define curvelets with these notations

$$\varphi_{j,l,k} = \varphi_j(R_{-\theta_{j,l}}(x - b_k^{(j,l)})), \tag{3}$$

here R_θ^{-1}, the inverse of R_θ is described as,

$$R_\theta^{-1} = R_\theta^T = R_{-\theta}.$$

The coefficient of a curvelet is merely the inner product between a curvelet $\varphi_{j,l,k}$ and an element $f \in L^2(R^2)$,

$$c(j,l,k) := < f, \varphi_{j,l,k} > = \int_{R^2} f(x)\overline{\varphi_{j,l,k}}(x)dx. \tag{4}$$

The curvelet coefficients, by using the theorem of Plancherel, can be represented as

$$c(j,l,k) = \frac{1}{(2\pi)^2}\int \hat{f}(w)\overline{\hat{\varphi}_{j,l,k}}(w)dw = \frac{1}{(2\pi)^2}\int \hat{f}(w)U_j(R_{\theta_l}w)e^{i<x_k^{(j,l)}, w>}dw.$$

We have coarse-scale elements too, as in wavelet theory. The low-pass window W_0 introduced as a function that follows the following equation

$$|W_0(r)|^2 + \sum_{j\geq 0} |W(2^{-j}r)|^2 = 1,$$

and for $k_1, k_2 \in Z$, coarse scale curvelets are these described by

$$\Phi_{j_0,k}(x) = \Phi_{j_0}(x - 2^{-j_0}k), \quad \hat{\Phi}_{j_0}(\xi) = 2^{-j_0}W_0(2^{-j_0}|\xi|). \tag{5}$$

The complete curvelet transformation consist of the directional finest scale elements $(\varphi_{j,l,k})_{j\geq j_0,l,k}$ along with the isotropic coarsest scale father wavelets $(\Phi_{j_0,k})$.

2.5. Fast discrete curvelet transform

Candes et al. developed two fast discrete curvelet transformations (FDCT) in 2006. One relies on the unevenly spaced fast-Fourier transformation (USFFT) [9] and the another one depends on the binding of specifically chosen Fourier-samplings (FDCT WRAPPING) [9]. We have used FDCT wrapping in this work, as it is the rapid curvelet transformation. After curvelet transformation, various groups of coefficients of curvelets at distinct angles and scales are constructed. The coefficients of curvelets at angle l and at scale j are expressed as a matrix $c_{j,l}$ and scale j is from finer level to coarser level and angle l begins at the upper-left edge and raises clockwise.

Assume that $f(t_1, t_2), 1 \leq t_1 \leq N_1,\ 1 \leq t_2 \leq N_2$ refers to the original function and $\hat{f}[n_1, n_2]$ defines $2D$ discrete Fourier transformation.

The FDCT WRAPPING implementation is as follows:

Step 1. $2D$ fast Fourier transform (FFT) is implemented on $f(t_1, t_2)$ to achieve $\hat{f}[n_1, n_2]$ Fourier coefficients.

Step 2. The new sampling function is provided by re-sampling $\hat{f}[n_1, n_2]$ at every combination of direction and scale l, j into the frequency-region as:

$$\hat{f}[n_1, n_2 - n_1 \tan\theta_l], \quad (n_1, n_2) \in P_j, \tag{6}$$

here $P_j = \{(n_1, n_2), n_{1,0} \leq n_1 < n_{1,0} + L_{1,j}, n_{2,0} \leq n_2 < n_{2,0} + L_{2,j}\}$ and $n_{1,0}, n_{2,0}$ both are the original locations of window function $\widetilde{U}_{j,l}[n_1, n_2]$. $L_{1,j}$, $L_{2,j}$ respectively are significant constants of 2^j and $2^{j/2}$, and are constituents of the length and width of the support interval of the window function.

Step 3. Multiply the new sampling function $\hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]$ with a window function $\widetilde{U}_{j,l}[n_1, n_2]$

$$\widetilde{f}_{j,l}[n_1, n_2] = \hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]\widetilde{U}_{j,l}[n_1, n_2], \tag{7}$$

where

$$\widetilde{U}_{j,l}[n_1, n_2] = W_j(w_1, w_2)V_j\big(S_{\theta_l}.\frac{2^{\lfloor j/2 \rfloor}w_2}{w_1}\big),$$
$$W_j(w_1, w_2) = \sqrt{\Phi^2_{j+1}(w) - \Phi^2_j(w)},$$
$$\Phi_j(w_1, w_2) = \phi(2^{-j}w_1)\phi(2^{-j}w_2),$$
$$S_{\theta_l} = \begin{pmatrix} 1 & 0 \\ -\tan\theta_l & 1 \end{pmatrix},$$
$$\tan\theta_l = l \times 2^{\lfloor -j/2 \rfloor}, \ \ l = -2^{\lfloor -j/2 \rfloor}, \cdots, 2^{\lfloor -j/2 \rfloor} - 1.$$

Step 4. To each $\widetilde{f}_{j,l}$, apply the inverse 2-D FFT and therefore we obtain the discrete coefficients $c_{j,l}$.

The FDCT Inverse Wrapping algorithm is as follows:

- For each array of the curvelet coefficients.
  a. Apply FFT on the array.
  b. Uncover the rectangular support to the initial orientation state.
  c. Translate to the original location.
- Add all the translated curvelet arrays.
- To reconstruct the function, take the inverse FFT.

It can be noted that the above discussed curvelets fall in the category of second generation curvelets. Prior to these constructions, exist the first generation curvelets developed by same authors [6]. For all our numerical experiments, we have used the second generation curvelets and the Matlab codes given in the toolbox Curvelab Toolbox [8] are utilized for performing the curvelet and inverse curvelet transforms. It can also be noted that no analytical formula for mother curvelet exists in general. By imposing certain conditions on abstractly assumed mother curvelet, the algorithm for the forward and inverse curvelet transform is designed as discussed above.

3. Reconstruction and compression error

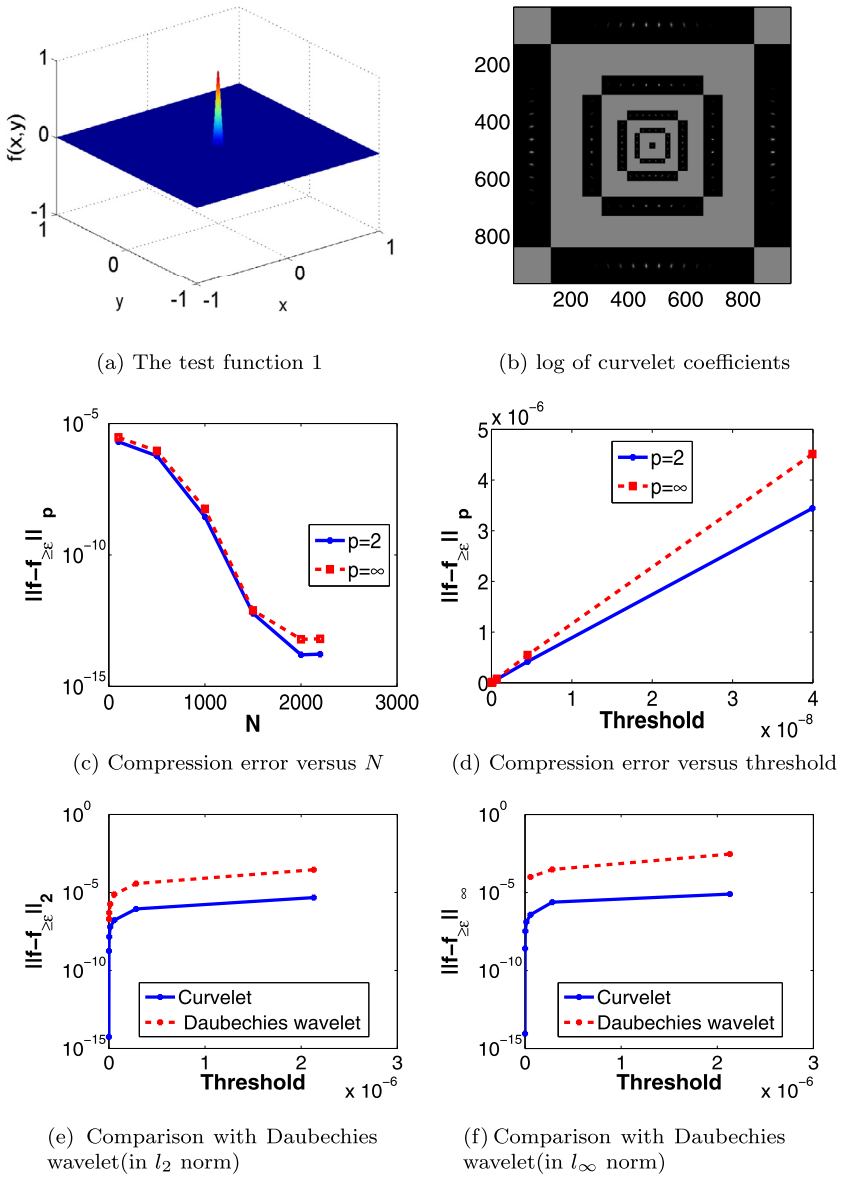
For a given function $f(x)$ and a threshold value ϵ, the curvelet expansion of function f can be decomposed into two parts as $f(x) = f_{\geq\epsilon}(x) + f_{<\epsilon}(x)$, where $f_{\geq\epsilon} = \sum_{|<f,\varphi_n>|\geq\epsilon} < f, \varphi_n > \varphi_n$ and $f_{<\epsilon} = \sum_{|<f,\varphi_n>|<\epsilon} < f, \varphi_n > \varphi_n$. The term $\|f - f_{\geq\epsilon}\|_p$ is named as the compression error. It should be noted that, $\epsilon = 0$ means no coefficients are being discarded and in that case the quantity $\|f - f_{\geq\epsilon}\|_p$ is named as the reconstruction error.

3.1. Algorithm for compression by using curvelet transform

1. Calculate the curvelet coefficients using FDCT explained in section 2, let the vector of curvelet coefficients be called as C.
2. If the threshold ϵ is given, then in the vector C, put zeros wherever the magnitude of the element is less than ϵ. Call the new vector as C_1.
3. If instead of ϵ, a compression ratio CPR is given, then the process is as follows:
   - Arrange the curvelet coefficients in descending order, let this vector be called as C_2.
   - Find out the threshold as follows: M = CPR*length of the function, and then threshold ϵ = magnitude of the Mth element of the vector C_2.
   - In the vector C, put zeros wherever the magnitude of the element is less than ϵ. Call the new vector as C_1.
4. On C_1 inverse FDCT is applied to get the compressed function.

The following two test functions are being considered for numerical experiments:

Test function 1: $f(x, y) = \exp(-1000(x^2 + y^2))$ on the domain $[-1, 1] \times [1, 1]$ is chosen as our first test function. It is shown in Fig. 1(*a*) and the corresponding curvelet coefficients (after taking log) are shown in Fig. 1(*b*). Following observations are made based on our numerical results.



(a) The test function 1

(b) log of curvelet coefficients

(c) Compression error versus N

(d) Compression error versus threshold

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

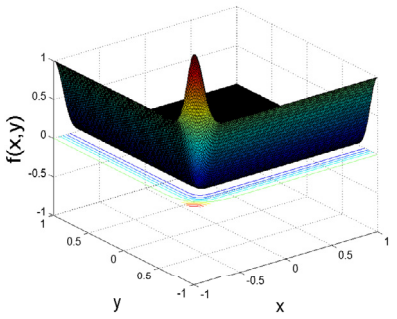
(f) Comparison with Daubechies wavelet(in l_∞ norm)

Fig. 1. Results for the test function 1. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

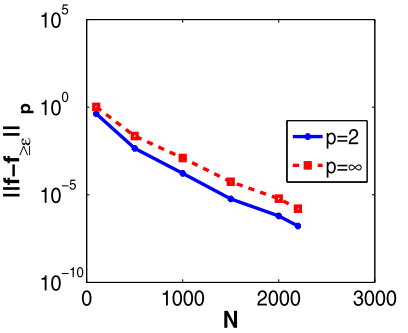
- Order of the reconstruction error is 10^{-16}.
- In Fig. 1(c), the variation of the compression error versus N is shown. One can observe a good compression.
- Fig. 1(d) plots compression error versus ϵ. An increase in the compression error with increase in values of threshold, can easily be seen. The reason for this trend is the fact that with increase in the value of threshold, more of the curvelet coefficients will be discarded.
- Fig. 1(e) and 1(f) compares the compression error for curvelet and Daubechies wavelet. It can be observed that compression in case of curvelets is much higher as compared to wavelets and hence one can conclude that curvelets are efficient than wavelets.

Test function 2: Function $f(x, y) = \exp(-50(x+1)^2 + \exp(-50(y+1)^2))$ is chosen as our second test function on the domain $[-1, 1] \times [1, 1]$ which is shown in Fig. 2(a). The following numerical results are being observed.

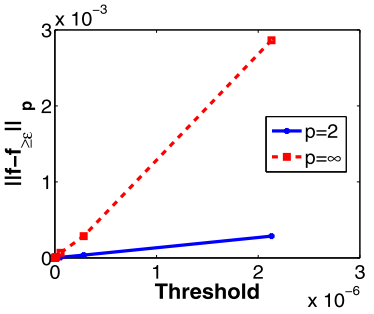
- The order of the reconstruction error is 10^{-15}.
- Fig. 2(b) plots the compression error versus N, a substantial compression can be observed. Fig. 2(c) plots compression error as a function of threshold ϵ. Here again we see that compression error increases on increasing the value of threshold.



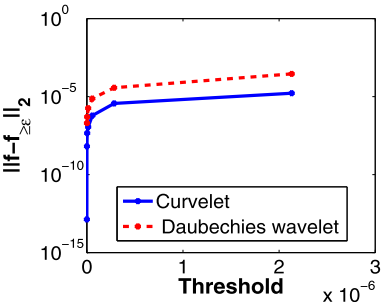
(a) The test function 2



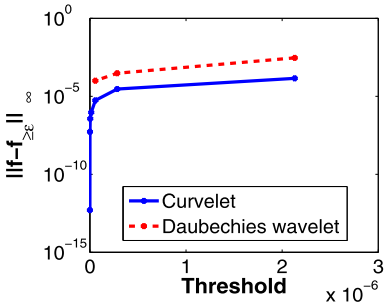
(b) Compression error versus N



(c) Compression error versus threshold



(d) Comparison with Daubechies wavelet(in l_2 norm)



(e) Comparison with Daubechies wavelet(in l_∞ norm)

Fig. 2. Results for the test function 2.

- Fig. 2(d) and 2(e) curvelets and wavelets in terms of compression error and here again curvelets perform better.

4. Solving the PDEs using curvelets

In the following text the fast curvelet based finite difference method (FCFD) to solve PDEs is explained. The factor which motivates the development of curvelet based numerical techniques for solving PDEs is that curvelet description of the finite difference matrices associated with PDEs is well organized and sparse. The main characteristic of the proposed method is to represent the differential operator in a curvelet basis φ_μ of $L^2(R^m)$ or in other words apply curvelet transform on the finite difference matrices approximating the differential operators. Having applied the curvelet transform, one can discard all the curvelet coefficients which are having magnitude less than a pre decided threshold ϵ. After that, all the computations are performed using the sparse matrices and at the end the inverse FDCT is applied to get the solution at the final time.

To explain the method in detail, a 2-D diffusion equation given as follows is considered

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x, y); 0 \leq x, y \leq 1, \tag{8}$$

with an initial profile $u(x, y, t = 0) = u_0(x, y)$ and with appropriate boundaries such as Periodic, Neumann, Dirichlet or Robin's. On applying time discretization scheme, the Eq. (8) takes the following form

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

where u^n approximated u at time $t = n\Delta t$. $\mathcal{A}$ and $\mathcal{C}$ are differential operators associated with the time discretization method. For example $\mathcal{A} = I$, $\mathcal{C} = \left(I + \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)$ for the explicit forward Euler's scheme and $\mathcal{A} = \left(I - \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)$, $\mathcal{C} = I$ for the implicit backward Euler's scheme.

After spatial discretization, we get

$$Au^n = Cu^{n-1} + \Delta t f, \;\; u^0 = u_0, \tag{9}$$

where u^n is the vector of all unknowns $u^n_{i,j}$, $i = 1, \cdots, N$; $j = 1, \cdots, M$ at time $t = n\Delta t$. Eq. (9) can be written as

$$u^n = A^{-1}(Cu^{n-1} + \Delta t f). \tag{10}$$

By using the Eq. (10) recursively, we obtain

$$u^n = (A^{-1})^n C^n u^0 + \sum_{k=0}^{n-1} (A^{-1})^k C^k \Delta t A^{-1} f. \tag{11}$$

When $A = I$, the above equation takes the following form

$$u^n = C^n u^0 + \sum_{k=0}^{n-1} C^k \Delta t f. \tag{12}$$

Now if we choose, $n = 2^m$ for some integer m which actually means that we are computing the numerical solution at times $2\Delta t, 4\Delta t, 8\Delta t, \cdots$, then

$$\begin{aligned}
\sum_{k=0}^{n-1} C^k \Delta f &= \sum_{k=0}^{2^m-1} C^k \Delta t f, \\
&= (I + C + C^2 + \cdots + C^{2^m-1})\Delta t f, \\
&= \left((I + C) + C^2(I + C) + C^4(I + C) + C^6(I + C) + \cdots C^{2^m-2}(I + C)\right)\Delta t f, \\
&= (I + C)(I + C^2 + C^4 + C^6 + \cdots C^{2^m-2})\Delta t f, \\
&= (I + C)(I + C^2)(I + C^4 + C^8 + \cdots C^{2^m-4})\Delta t f, \\
&\;\;\vdots \\
&= \prod_{k=0}^{m-1} (I + (C)^{2^k})\Delta t f.
\end{aligned} \tag{13}$$

Using Eq. (13), Eq. (12) can be written as

$$u^{2^m} = C^{2^m} u^0 + \prod_{k=0}^{m-1} (I + C^{2^k})\Delta t f. \tag{14}$$

It is clear from the Eq. (14), that the computation of the solution at time $t = 2^m \Delta t$ involves only the repeated squaring of the matrix C which leads to the following algorithm for computing the solution by compressing the dyadic powers of C using curvelets:

Algorithm for FCFD

- **Step 1:** Apply curvelet transform (FDCT) on the matrix C. It gives us a matrix of curvelet coefficients of C. Call this matrix as D.
- **Step 2:** Let $E = I$ (identity matrix).
  Iterate the following two steps Step 3 and Step 4 m times
- **Step 3:** Replace E with the new matrix which is created by applying threshold ϵ on $E + DE$ (it means discard all the elements which are having magnitude less than ϵ).

Table 1
The performance of FCFD for problem 1.

Threshold ϵ	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.2657	0.4469	0.5469
Θ	2.234	1.32	1.085

Table 2
Comparison of performance of FCFD and FEM.

Final time T (in seconds)	0.4848	3.8784	4.12
CPU time taken by FCFD (in seconds)	0.2341	0.583	0.9793
CPU time taken by FEM (in seconds)	0.87	0.92	1.32

Table 3
Order of accuracy for the test problem 1.

Number of grid points N	225	324	441	529	676
Order of accuracy α	-	1.5959	1.7728	2.4842	2.822

- **Step 4:** Replace D with a new matrix which is created by applying threshold ϵ on $D * D$.
  Note that after **Step 4** we get the curvelet coefficients of C^{2^m} and $\prod_{k=0}^{m-1}(I + (C)^{2^k})$ respectively in the form of D and E.
- **Step 5:** Apply inverse FDCT on D and E and call the new matrices as D_1 and E_1.
- **Step 6:** Solution at time $2^m \Delta t$ is then given by $u^{2^m} = D_1 u^0 + E_1(\Delta t) f$.

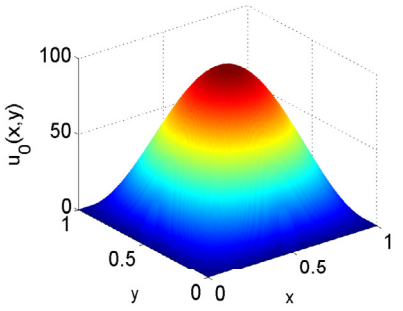
It should be noted that, if we do not choose $n = 2^m$, then an algorithm similar to the above can not be designed. Moreover, given a final time T at which we have to compute the solution of the PDE, we can always find an integer m such that $2^m \Delta t$ is approximately equal to T.

We applied the above discussed FCFD on five test problems of different nature. In each of the five test problems, we have discussed about the performance/efficiency of the method in terms of a **time compression coefficient** Θ which is described as $\Theta = \frac{CPU(threshold=0)}{CPU(threshold)}$, where $CPU(threshold = 0)$ is the run time of the FDM code and $CPU(threshold)$ is the run time of the FCFD method code. It can be noted that higher values of Θ indicate that there is a significant decrease in the CPU time of FCFD and hence the efficiency/performance of FCFD is higher.

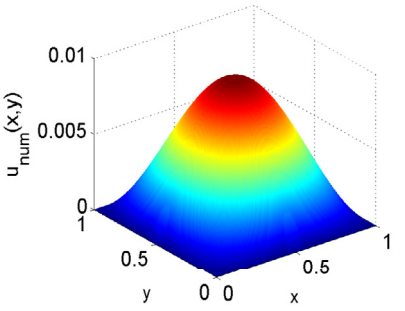
We have also computed the numerical order of accuracy, α, defined as follows: For a numerical scheme with step size h, let $\|u - u_h\| \leq kh^\alpha$, where u is the analytical solution of the problem, u_h is the numerical solution of the problem, k is a constant independent of h, then α is called the order of accuracy. Note that for the problems without analytical solutions, a reference solution with very small h can be chosen as u. Numerically, $\alpha = \frac{\log\left(\frac{\|u-u_{N1}\|}{\|u-u_{N2}\|}\right)}{\log\left(\frac{N_2}{N_1}\right)}$ where u_{N1} and u_{N2} are the numerical solutions with number of grid points equal to N_1 and N_2 respectively.

5. Results and discussions

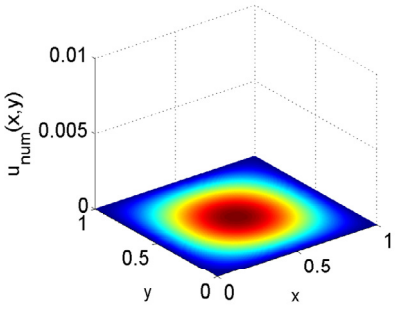
Test problem 1**:** Take $f(x, y) = 0$ in Eq. (8) along with $u(x, y, t = 0) = u_0(x, y) = 100 \sin(\pi x) \sin(\pi y)$ and the Dirichlet boundaries $u(x = 0, y, t) = u(x = 1, y, t) = u(x, y = 0, t) = u(x, y = 1, t) = 0$ are to be imposed. Fig. 3(*a*), 3(*b*), 3(*c*) show numerical solutions at time $t = 0, 0.4848, 3.8784$ respectively. We can see that with time the solution diffuses as expected. In order to have a clear view, Fig. 3(*d*) presents the numerical solution of the problem at different times by fixing $y = 0.5$. Fig. 3(*e*) shows the variation of the error (*i.e.*, $\|u_{num} - u_{ana}\|_p$) at $t = 0.4848$ versus N and the graph reports a decrease in error with an increase in value of N. Fig. 3(*f*) reports that the error increases with increase in the threshold parameter ϵ. Fig. 3(*g*) consists of two lines, the red line indicates the computational time taken by FCFD and the blue line indicates the computational time taken by FDM for computing the solution at time $t = 2^m \Delta t$. It can be observed that at all times, the red line is always below the blue line and hence the time taken by FCFD is much less. Fig. 3(*h*) shows log of curvelet coefficients of the numerical solution of the problem. Table 1 gives the variation of CPU time taken and of the time compression coefficient Θ with threshold ϵ. It can be observed that Θ decreases as the threshold decreases and hence FCFD becomes less efficient for low value of threshold. From the above analysis, one may infer that a high value of Θ can be chosen for more efficiency, but Fig. 3(*f*) should be kept in mind which indicates an increase in error with threshold ϵ. Therefore, we can conclude that the value of ϵ should be chosen wisely keeping in mind both points of view. A comparison of the run time consumed by the finite element method (FEM) with square finite elements and quadratic basis functions with that of the run time of FCFD is given in the Table 2. In this table the CPU time of FCFD and FEM is given for computing the numerical solution at the final time $T = 0.4848, 3.8784$ and 4.12 seconds. It can be observed that FCFD is taking less time. Table 3 gives the α of the proposed method.



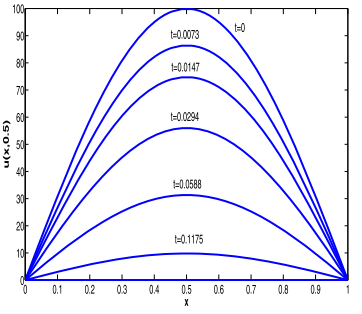
(a) The solution at time $t = 0$



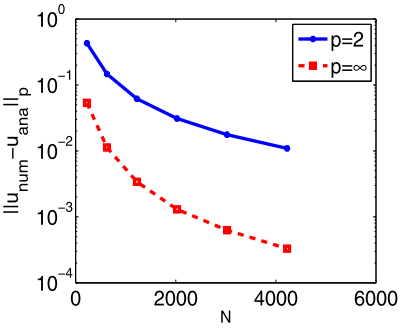
(b) The solution at time $t = 0.4848$



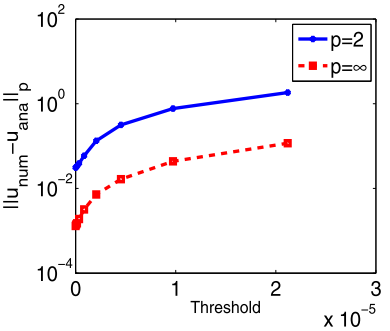
(c) The solution at time $t = 3.8784$



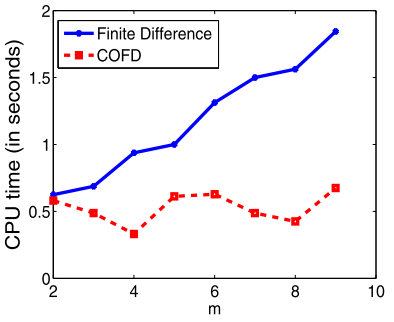
(d) $u(x, 0.5)$



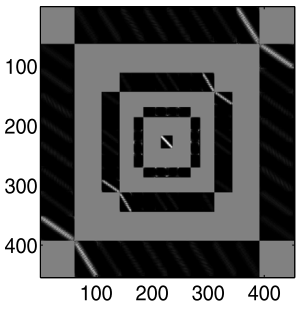
(e) $\|u_{num} - u_{ana}\|_p$ versus N



(f) $\|u_{num} - u_{ana}\|_p$ versus threshold

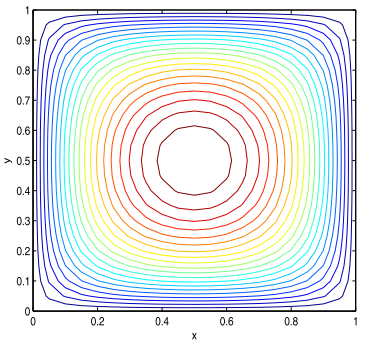


(g) CPU time for computing the solution at the time $t = 2^m \Delta t$

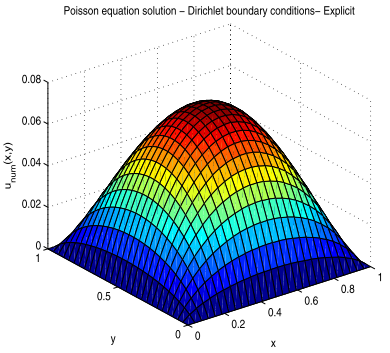


(h) log of curvelet coefficients

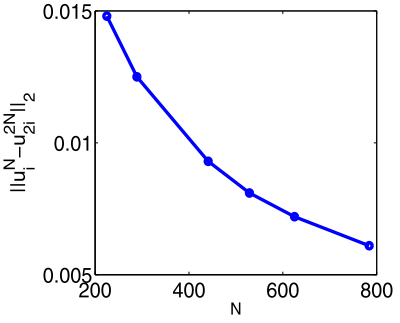
Fig. 3. Results for the test problem 1.



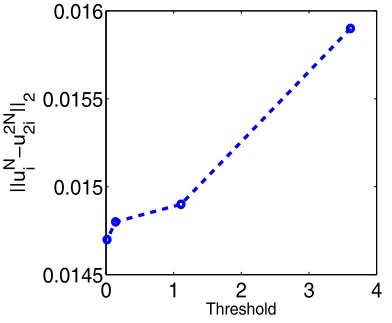
(a) Contour plot of solution



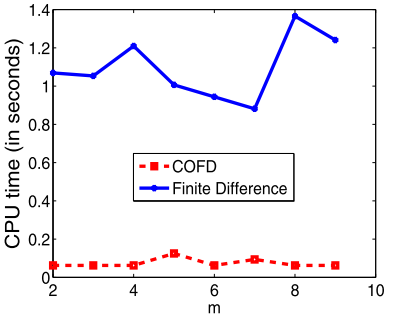
(b) Poisson equation solution



(c) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold



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

Fig. 4. Results for the test problem 2.

Test problem 2**:** The following 2-dimensional Poisson equation is considered

$$-\Delta u = 1, 0 \leq x, y \leq 1,$$

with $u(x = 0, y) = u(x, y = 0) = u(x = 1, y) = u(x, y = 1) = 0$ as the Dirichlet boundaries. On discretizing the domain with 625×625 points, Fig. 4(a) represent the contour plot of the solution. The closeness of contour lines indicates steepness. We have observed that, the contour lines which are evenly spaced and closed together indicates a uniform, steep slope. Fig. 4(b) represents the numerical solution of the problem. Fig. 4(c) represents the graph between the error *i.e.*, $\|(u_i)^M - (u_{2i})^{2M}\|_2$ where $i = 1, 2, \cdots, M$ and number of grid points. Here error is calculated according to double mesh principle. The procedure of double mesh principle is to double the no. of mesh points and then error is calculated as above, where $(u_{2i})^{2M}$ is the solution obtained on a mesh containing the same mesh points which are used in the previous mesh. We have observed from the graph that error decreases on increasing the N. Fig. 4(d) represents the graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold (chosen by the user). It shows that error increases as we increase threshold. Fig. 4(e) compares the computational time taken by FDM and FCFD method for computing the solution at time $t = 2^m \Delta t$. It is shown that CPU time taken by

Table 4
The efficiency of FCFD for problem 2.

Threshold ϵ	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.0103	0.0198	0.0254
Θ	3.038	1.581	1.232

Table 5
Order of accuracy for the test problem 2.

Number of grid points N	400	625	841	1089
Order of accuracy α	-	1.8171	2.5079	2.5226

Table 6
The efficiency of FCFD for problem 3.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.4853	0.5478	0.5573
Θ	1.197	1.061	1.052

Table 7
Order of accuracy for the test problem 3.

Number of grid points N	441	676	900	1089
Order of accuracy α	-	1.7740	2.4254	2.6200

FCFD method is less than FDM. Table 4 displays the Θ values for different values of threshold. It shows the efficiency of our algorithm. Table 5 shows the order of accuracy FCFD for the second test problem.

Test problem 3**:** Consider the following equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}; 0 \leq x, y \leq 1, t > 0, \tag{15}$$

along with $u(x, y, 0) = x(x-1)y(y-1)$ and $\frac{\partial w}{\partial t}(x, y, 0) = 0$ as the initial profiles and the Dirichlet zero boundaries. Figs. 5(a), 5(b), 5(c) show the solution of the test problem 3 at time $t = 0$, $t = 0.5$, $t = 2$ respectively. Fig. 5(d) plots the error $\|u_i^M - u_{2i}^{2M}\|_2$, where $i = 1, 2, \cdots, M$ at the final time 0.99 as a function of N. We have observed that error decreases on increasing the number of grid points. Fig. 5(e) represents the graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold. It shows that error increases on increasing threshold. Fig. 5(f) compares the CPU time consumed by FCFD and FDM to compute the solution at time $t = 2^m \Delta t$. It shows that CPU time taken by FCFD is less than FDM. Table 6 shows that as we decrease threshold, the value of Θ decreases which reveals the efficiency of our method. Table 7 shows α of the developed method.

Test problem 4**:** The next PDE is given as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}; 0 \leq x, y, z \leq 1,$$

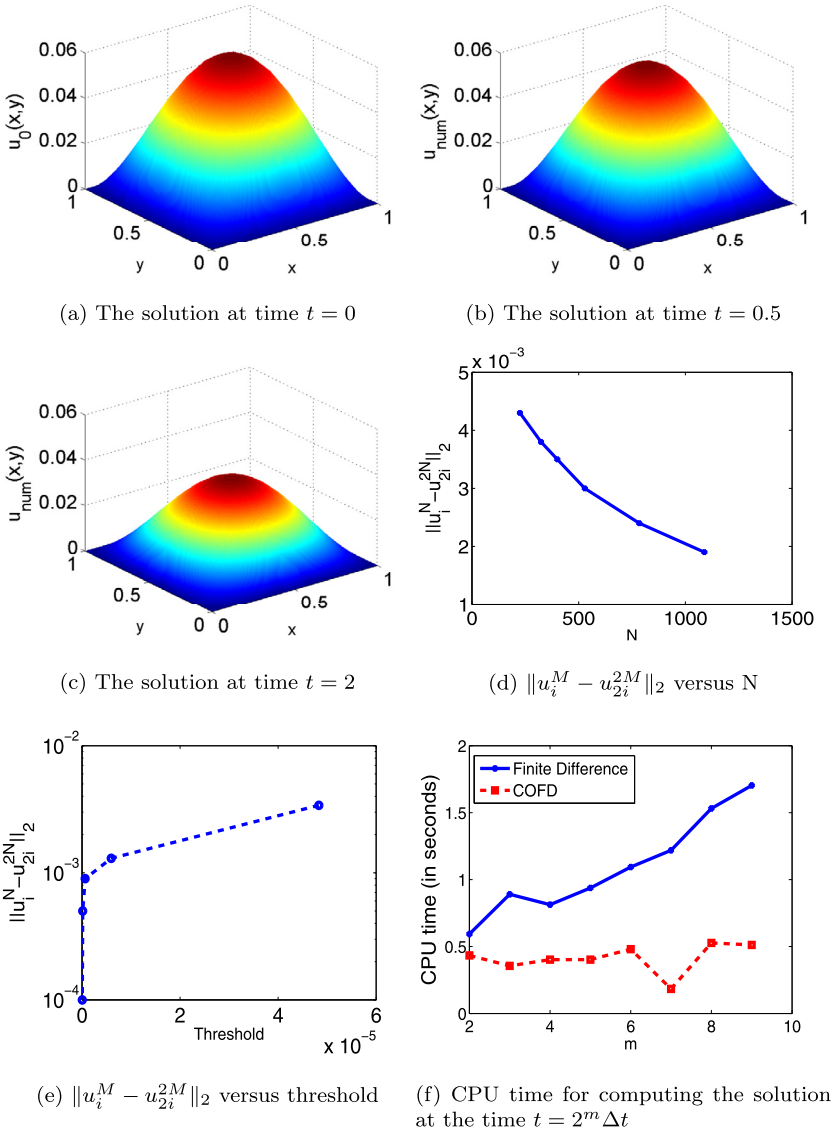
where an initial profile is $u(x, y, z, t = 0) = 100 \sin(\pi x) \sin(\pi y) \sin(\pi z)$ and Dirichlet zero boundary conditions. Figs. 6(a), 6(b), 6(c) show the solution profile for a fixed z coordinate as $z = 0.2$ at times $0, 0.1316$ and 4.213 respectively. For a better view of the solution profile, we draw the solution of the problem by making the slices at $x = \{0.3, 0.6\}$, $y = 0.5$ and $z = 0.5$. Figs. 6(d), 6(e) show such solutions at time 0 and 0.1316 respectively. Fig. 6(f) plots the graph of error versus N. Fig. 6(g) plots $\|u_{num} - u_{ana}\|_p$ as a function of threshold. We can here note that the increase in the error with respect to ϵ is insignificant. Fig. 6(h) compares the run time consumed by FCFD and the FDM to compute the solution at time $t = 2^m \Delta t$. It shows that CPU time taken by FCFD is less than FDM (Table 8). Table 9 shows the α of the developed method.

Test problem 5**:** Now we consider two dimensional Lotka-Volterra predator-prey model

$$u_t = c_{11} u_{xx} + c_{12} u_{yy} + a_1 u - r_1 u v,$$
$$v_t = c_{21} v_{xx} + c_{22} v_{yy} + a_2 v - r_2 u v, \quad 0 \leq x, y \leq 1$$

where $v(t, x, y)$ and $u(t, x, y)$ respectively represent the predator and prey population density at time t and space co-ordinates (x, y). Following parameters are chosen: $c_{12} = c_{11} = 0.1$ and $c_{22} = c_{21} = 0.01$, $\{a_1, a_2\} = \{0.47, 0.76\}$, $\{r_1, r_2\} = \{0.024, 0.023\}$. At the boundary, we assume $\overrightarrow{n}.\nabla u = 0$ and $\overrightarrow{n}.\nabla v = 0$. The initial profile is chosen as follows:

$$u(0, x, y) = \begin{cases} 10, & (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \leq 16, \\ 0, & otherwise. \end{cases} \tag{16}$$



(a) The solution at time $t = 0$

(b) The solution at time $t = 0.5$

(c) The solution at time $t = 2$

(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N

(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold

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

Fig. 5. Results for the test problem 3.

Table 8
The efficiency of FCFD for problem 4.

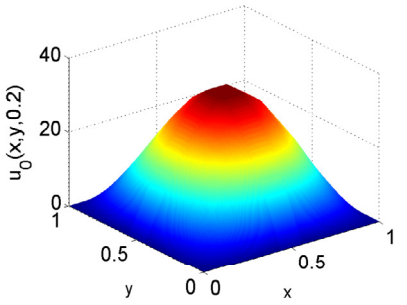
Threshold	10^{-3}	10^{-5}	10^{-6}
CPU time taken (in seconds)	39.9063	41.0938	41.5313
Θ	1.048	1.018	1.007

Table 9
Order of accuracy for the test problem 4.

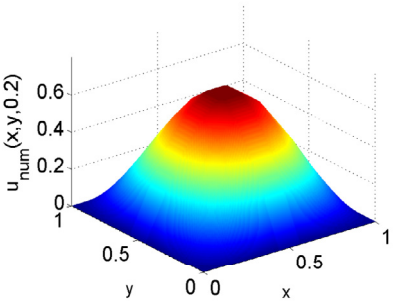
Number of grid points N	216	343	512	1000
Order of accuracy α	-	2.4822	2.3903	2.7824

$$v(0,x,y) = \begin{cases} 10, & (x-\frac{1}{2})^2 + (y-\frac{1}{2})^2 \geq 4, \\ 0, & otherwise. \end{cases} \tag{17}$$

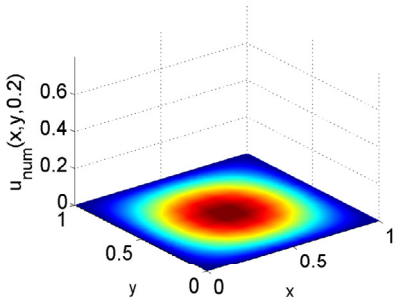
Fig. 7(*a*), 7(*b*) shows the numerical solution of prey population density at time $t = 1$ and $t = 2$ respectively. Fig. 7(*c*), 7(*d*) shows the numerical solution of predator population density at time $t = 1$ and $t = 2$ respectively. Fig. 7(*e*) shows the error (*i.e.*, $\|u_i^M - u_{2i}^{2M}\|_2$) with respect to number of grid points. Fig. 7(*f*) plots $\|u_i^M - u_{2i}^{2M}\|_2$ as a function of threshold. Fig. 7(*g*)



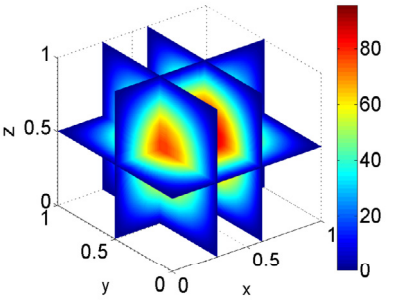
(a) The solution $u(x, y, 0.2)$ at time $t = 0$



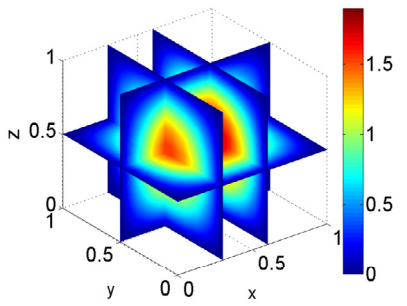
(b) The solution $u(x, y, 0.2)$ at time $t = 0.1316$



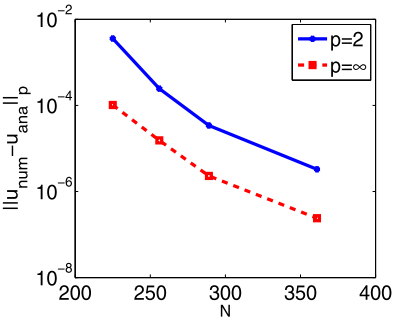
(c) The solution $u(x, y, 0.2)$ at time $t = 4.213$



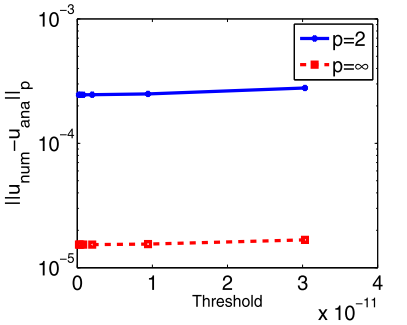
(d) The solution $u(x, y, z)$ at time $t = 0$



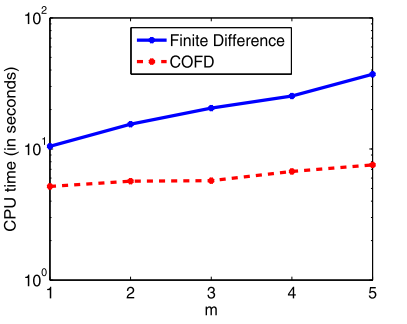
(e) The solution $u(x, y, z)$ at time $t = 0.1316$



(f) $\|u_{num} - u_{ana}\|_p$ versus N

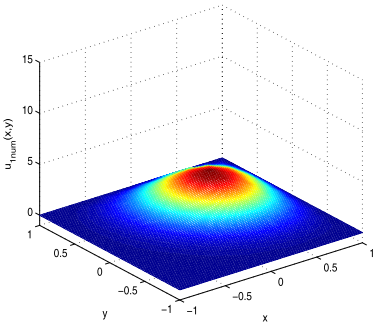


(g) $\|u_{num} - u_{ana}\|_p$ versus threshold

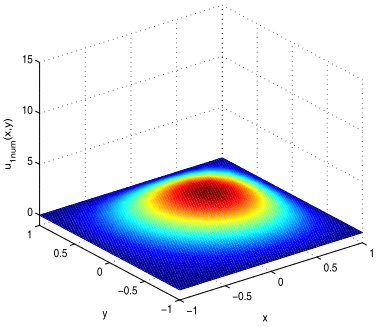


(h) CPU time for computing the solution at the time $t = 2^m \Delta t$

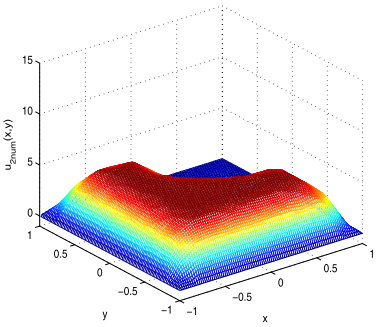
Fig. 6. Results for the test problem 4.



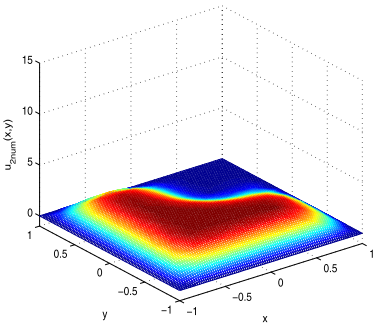
(a) The solution $u(x, y, t)$ at time $t = 1$



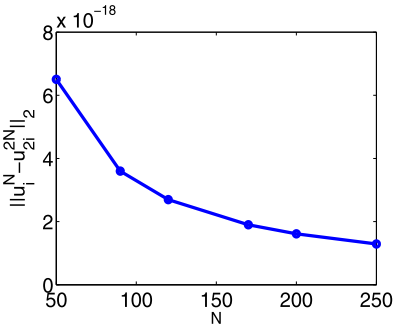
(b) The solution $u(x, y, t)$ at time $t = 2$



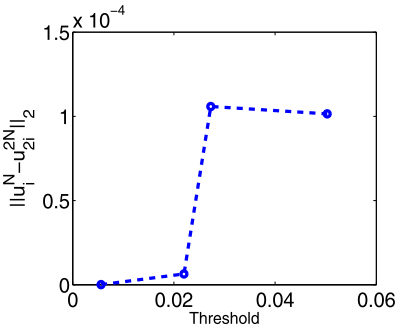
(c) The solution $v(x, y, t)$ at time $t = 1$



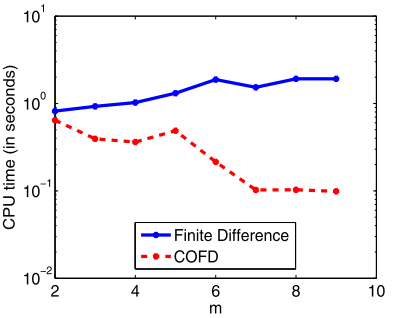
(d) The solution $v(x, y, t)$ at time $t = 2$



(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(f) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold



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

Fig. 7. Results for the test problem 5.

Table 10
The performance of FCFD for test problem 5.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.3667	0.5379	0.6379
Θ	1.894	1.301	1.089

Table 11
Order of accuracy for the test problem 5.

Number of grid points N	150	200	250	300
Order of accuracy α	-	1.6338	1.8215	2.5900

compares the CPU time taken FCFD and FDM. Table 10 gives the variation of computational time taken with Θ. Table 11 shows the α of the developed method.

6. Conclusion and future work

A fast curvelet based FDM has been devised for finding the numerical solutions of PDEs. The differential operators are approximated using finite difference matrices and curvelets are used for compressing these matrices. The method is applied on five test problems and it is found that the developed method is computationally very efficient. In future, the proposed method can be applied for solving PDEs on complex manifolds.

Acknowledgements

The first author would like to thank Council of Scientific and Industrial Research, New Delhi, India for providing financial support under Senior Research Fellowship scheme with File No. 09/677(0038)/2019-EMR-I. The second author is grateful to Science and Engineering Research Board (DST) for MTR/2017/000619 grant in support of this research work and TIET for Seed grant.

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] S. Bertoluzza, G. Naldi, J.C. Ravel, Wavelet methods for numerical solution of boundary value problems on the interval, in: C. Chui, L. Montefusco, L. Puccio (Eds.), Wavelets: Theory, Algorithms and Applications, Academic Press, New York, 1994, pp. 425–448.
[3] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms, Commun. Pure Appl. Math. 44 (1991) 141–183.
[4] E.J. Candes, L. Demanet, The curvelet representation of wave propagators is optimally sparse, Commun. Pure Appl. Math. 58 (2005) 1472–1528.
[5] E.J. Candes, D.L. Donoho, Curvelets - a surprisingly effective nonadaptive representation for objects with edges, in: Curves and Surfaces, Vanderbilt University Press, Nashville, TN, 2000, pp. 105–120.
[6] E.J. Candes, D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities, Commun. Pure Appl. Math. 57 (2004) 219–266.
[7] E.J. Candes, D.L. Donoho, Continuous curvelet transform: i. Resolution of the wavefront set, Appl. Comput. Harmon. Anal. 19 (2005) 162–197.
[8] E.J. Candes, L. Demanet, D.L. Donoho, L. Ying, Curvelab Toolbox, Version 2.1.3, CIT, 2005.
[9] E.J. Candes, L. Demanet, D.L. Donoho, L. Ying, Fast discrete curvelet transforms, Multiscale Model. Simul. 5 (2006) 861–899.
[10] A.C. Canuto, K. Urban, The wavelet element method. Part i: construction and analysis, Appl. Comput. Harmon. Anal. 6 (1999) 1–5.
[11] A. Cohen, W. Dahmen, R. DeVore, Adaptive wavelet methods for elliptic operator equations: convergence rates, Math. Comput. 70 (2001) 27–75.
[12] S. Dahlke, W. Dahmen, R. Hochmuth, R. Schneider, Stable multiscale bases and local error estimation for elliptic problems, Appl. Numer. Math. 23 (1997) 21–47.
[13] W. Dahmen, A. Kunoth, Multilevel preconditioning, Numer. Math. 63 (1992) 315–344.
[14] W. Dahmen, R. Schneider, Wavelets on manifolds i: construction and domain decomposition, SIAM J. Math. Anal. 31 (1999) 184–230.
[15] W. Dahmen, R. Stevenson, Element-by-element construction of wavelets satisfying stability and moment conditions, SIAM J. Numer. Anal. 37 (1999) 319–352.
[16] W. Dahmen, K. Urban, J. Vorloeper, Adaptive Wavelet Methods: Basic Concepts and Applications to the Stokes Problem, World Scientific, Singapore, 2002, pp. 39–80.
[17] 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.
[18] K. Goyal, M. Mehra, An adaptive meshfree diffusion wavelet method for partial differential equations on the sphere, J. Comput. Phys. 272 (2014) 747–771.
[19] K. Goyal, M. Mehra, Fast diffusion wavelet method for partial differential equations, Appl. Math. Model. 40 (2016) 5000–5025.
[20] K. Goyal, M. Mehra, An adaptive meshfree spectral graph wavelet method for partial differential equations, Appl. Numer. Math. 113 (2017) 168–185.
[21] M. Holmstrom, J. Walden, Adaptive wavelet methods for hyperbolic PDE's, J. Comput. Phys. 13 (1998) 19–49.
[22] S. Jaffard, Wavelet methods for fast resolution of elliptic problems, SIAM J. Numer. Anal. 29 (1992) 965–986.
[23] J. Liandrat, P. Tchamitchian, Resolution of the 1D Regularized Burgers Equation Using a Spatial Wavelet Approximation, NASA CR-187480, ICASE report No. 90-83, 1990.
[24] J. Ma, G. Tang, M.Y. Hussaini, A refining estimation for adaptive solution of wave equation based on curvelets, Proc. SPIE Wavelets XII 6701 (2007).
[25] S. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. Mach. Intell. 11 (1989) 674–693.
[26] S. Mallat, Multifrequency channel decompositions of images and wavelet models, IEEE Trans. Signal Process. 37 (1989) 2091–2110.
[27] S. Mallat, A Wavelet Tour of Signal Processing: The Sparse Way, Academic Press, Burlington, MA, 2009.

[28] 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.
[29] O. Roussel, K. Schneider, A. Tsigulin, H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDE's, J. Comput. Phys. 188 (2003) 493–523.
[30] K. Schneider, F. Chemie, T. Chemie, J. Frohlich, J. Frohlich, An adaptive wavelet-vaguelette algorithm for the solution of PDEs, J. Comput. Phys. 130 (1997) 90–174.
[31] D. Sharma, K. Goyal, Second-generation wavelet optimized finite difference method (SGWOFD) for solution of Burger's equation with different boundary conditions, Int. J. Wavelets Multiresolut. Inf. Process. 16 (2018) 1850032.
[32] D. Sharma, K. Goyal, Spectral graph wavelet optimized finite difference method for solution of Burger's equation with different boundary conditions, J. Differ. Equ. Appl. 25 (2019) 373–395.
[33] O.V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.
[34] L. Ying, L. Demanet, E. Candes, 3D discrete curvelet transform, Proc. SPIE Wavelets XI 5914 (2005).