Computers and Mathematics with Applications 68 (2014) 568–577

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

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

A fast adaptive diffusion wavelet method for Burger's equation

Kavita Goyal, Mani Mehra [*]

Indian Institute of Technology Delhi, India

ARTICLE INFO

Article history:
Received 3 October 2013
Received in revised form 28 May 2014
Accepted 1 June 2014
Available online 3 July 2014

Keywords:
Multiresolution analysis
Burger's equation
Adaptive grid

ABSTRACT

A fast adaptive diffusion wavelet method is developed for solving the Burger's equation. The diffusion wavelet is developed in 2006 (Coifman and Maggioni, 2006) and its most important feature is that it can be constructed on any kind of manifold. Classes of operators which can be used for construction of the diffusion wavelet include second order finite difference differentiation matrices. The efficiency of the method is that the same operator is used for the construction of the diffusion wavelet as well as for the discretization of the differential operator involved in the Burger's equation. The diffusion wavelet is used for the construction of an adaptive grid as well as for the fast computation of the dyadic powers of the finite difference matrices involved in the numerical solution of Burger's equation. In this paper, we have considered one dimensional and two dimensional Burger's equation with Dirichlet and periodic boundary conditions. For each test problem the CPU time taken by fast adaptive diffusion wavelet method is compared with the CPU time taken by finite difference method and observed that the proposed method takes lesser CPU time. We have also verified the convergence of the given method.

1. Introduction

Burger's equation models the situations where both typical non-linearity and diffusion occur [1–4]. The equation is intensively used to test the numerical schemes. Various numerical methods have been developed for solving Burger's equation, for example finite element schemes in [5], finite difference schemes in [6], spectral methods in [7,8] and distributed functional approaches in [9,10].

While solving a partial differential equation (PDE) numerically, using an adaptive grid [11–13] has obvious advantages over using a static grid. Wavelets are widely used for numerical solutions of PDEs (and in particular the Burger's equation) on adaptive grids. In [14] one dimensional Burger's equation with periodic boundary conditions is solved on a static grid using Daubechies wavelet and in [15] it is solved using quasi wavelets [16]. An adaptive grid is generated using spline wavelets to solve one-dimensional Burger's equation with periodic boundary conditions in [17]. In [18] second generation wavelet is used to solve one dimensional Burger's equation on an adaptive grid. But the wavelet theory for numerical solution of PDEs on general manifold is a relatively new field, although there are many non wavelet numerical techniques available [19–22]. One of the work done in this direction is a dynamic adaptive numerical method for solving PDEs on the sphere using second

[*] 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.camwa.2014.06.007
0898-1221/© 2014 Elsevier Ltd. All rights reserved.

generation spherical wavelet [23]. Fast adaptive diffusion wavelet method (FADWM) developed in this paper can be seen as a first step in this direction.

The diffusion wavelet is introduced by Coifman and his collaborators in 2006 [24]. The most important feature of the diffusion wavelet is that it can be constructed on general manifolds. This wavelet has not been used for numerical solutions of PDEs and to best of our knowledge ours is the first attempt. In FADWM, diffusion wavelet is used for making an adaptive grid and for the fast computation of the dyadic powers of the finite difference matrices involved in the numerical solution of the Burger's equation.

The paper is organized as follows: Section 2 gives a brief description of the diffusion wavelet. FADWM is developed in Section 3. Section 4 contains the numerical results. Section 5 concludes the paper and gives a brief idea of the future work.

2. A brief description of diffusion wavelet

Diffusion wavelet [24] is constructed on any general manifold X. Multiresolution analysis (MRA) is built using a diffusion operator T on $\mathcal{L}_2(X)$ which is local, self adjoint and whose high powers have low numerical rank. For example $I-T$ with I as an identity operator on $\mathcal{L}_2(X)$ could be the Laplace–Beltrami operator. For any $f \in \mathcal{L}_2(X)$ we have $P_{\mathcal{V}^j} f(x) = \sum_{k\in X^j} c_k^j \phi_k^j(x)$. Computing the scaling function coefficients $\{c_k^j\}_{k\in X^j}$ using the values of $\{f(x_k)\}_{x_k \in X^j}$ is called diffusion scaling function transform (DST). Computing the function f from the coefficients $\{c_k^j\}_{k\in X^j}$ is called the inverse diffusion scaling function transform (IDST).

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 $\mathbf{c}^j$, computing the sets $\mathbf{c}^{j-1} = \{c_k^{j-1}\}_{k\in X^{j-1}}$ and $\mathbf{d}^{j-1} = \{d_k^{j-1}\}_{k\in Y^{j-1}}$ is termed 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 $\mathcal{V}^J = \mathcal{V}^{J_0} \bigoplus_{j=J_0}^{J-1} \mathcal{W}^j$. Therefore

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

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 in (1). Constructing the set $\mathbf{c}^j$ from the sets $\mathbf{c}^{j-1}$ and $\mathbf{d}^{j-1}$ is called inverse partial diffusion wavelet transform (IPDWT). Inverse full diffusion wavelet transform (IFDWT) is obtained by applying IPDWT recursively. For details one can see [24].

3. Fast adaptive diffusion wavelet method (FADWM)

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

Suppose that we are given a function $f \in \mathcal{L}_2(X)$ and we want to compute $T^{2^m} f \approx [T^{2^m}]_{\Phi^J}^{\Phi^J} \mathbf{f}$. Using $[T^{2^m}]_{\Phi^J}^{\Phi^{J-m}} = [T^{2^{m-1}}]_{\Phi^{J-(m-1)}}^{\Phi^{J-m}} [T^{2^{m-2}}]_{\Phi^{J-(m-2)}}^{\Phi^{J-(m-1)}} \cdots [T]_{\Phi^J}^{\Phi^{J-1}} [T]_{\Phi^J}^{\Phi^J}$, we can compute $[T^{2^m}]_{\Phi^J}^{\Phi^{J-m}} \mathbf{f}$ which is the vector of coordinates of $[T^{2^m}]_{\Phi^J}^{\Phi^J} \mathbf{f}$ in the basis Φ^{J-m} of $\mathcal{V}^{J-m}$, 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 the vector 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($T, f, m$)
1) $g = [T]_{\Phi^J}^{\Phi^J} \mathbf{f}$.
   For $k = 0, 1, \ldots m-1$  }
2) $g = [T^{2^k}]_{\Phi^{J-k}}^{\Phi^{J-(k+1)}} g$   } This gives us $\mathbf{c}^{J-m}$.
   end                          }
3) $\mathbf{c}^{J-m} \xrightarrow{\text{IDST}} \mathbf{c}^J$.
4) $\mathbf{c}^J = [T^{2^m}]_{\Phi^J}^{\Phi^J} \mathbf{f}$.
```

We took $f(x) = x$ and computed $[T^{2^{12}}]_{\Phi^8}^{\Phi^8} \mathbf{f}$ (T is the diffusion operator constructed using the construction of [25]) both analytically (the matrix $[T]_{\Phi^8}^{\Phi^8}$ is multiplied 2^{12} times and finally with $\mathbf{f}$) and using the above algorithm. The errors in l_2 and l_∞ norms are 2.393×10^{-9} and 1.563×10^{-10} respectively. The CPU time taken for computing $[T^{2^{12}}]_{\Phi^8}^{\Phi^8} \mathbf{f}$ using the above algorithm is 2% of the CPU time taken for its analytic computation.

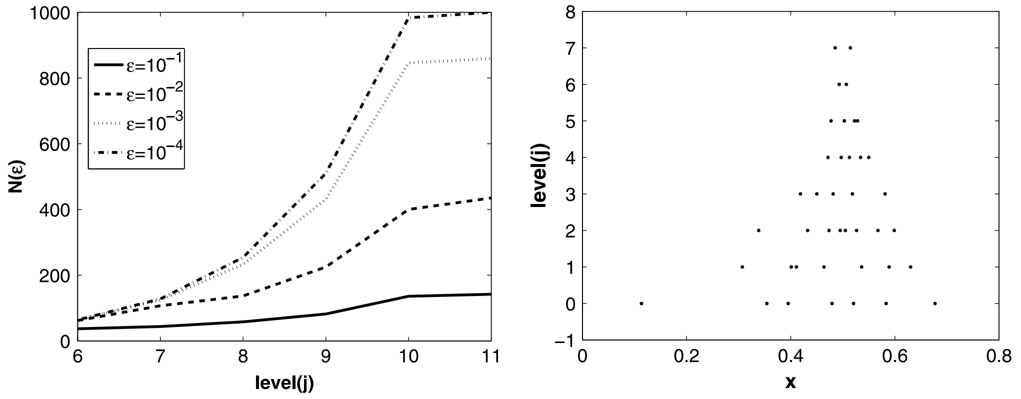


Fig. 1. Variation of $N(\epsilon)$ vs. j and position of the grid points with $|d_k^j| \geq \epsilon$.

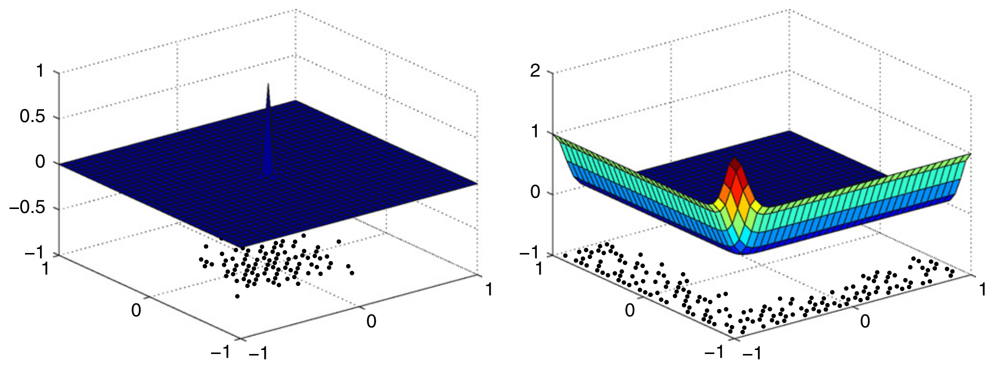


Fig. 2. Position of the grid points with $|d_k^j| \geq \epsilon$ for $f(x, y) = \exp(-1000(x^2 + y^2))$ and $f(x, y) = exp(-50(x + 1)^2) + exp(-50(y + 1)^2)$.

3.2. Adaptive grid

The property of wavelet that the wavelet coefficients d_k^j have large magnitude only near the point of discontinuity makes it suitable to detect where in the numerical solution of a PDE the shocks are located and hence an adaptive grid can be generated. For any $f(x) \in \mathcal{L}_2(X)$ and a given threshold ϵ, (1) can be written as $P_{\mathcal{V}^J} f(x) = f_{\geq\epsilon}(x) + f_{<\epsilon}(x)$, where

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

The number of significant coefficients $N(\epsilon)$ is defined as $N(\epsilon) = \#X^{J_0} + \sum_{j=J_0}^{J-1} \#\{d_k^j | k \in Y^j \text{ and } |d_k^j| \geq \epsilon\}$. $\|f - f_{\geq\epsilon}\|_p$ is called the compression error and Donoho in [26] proved that for sufficiently smooth f

$$\|f - f_{\geq\epsilon}\|_\infty < C\epsilon, \tag{2}$$

where C is a constant. For the sawtooth function on [0, 1] with discontinuity at $x = 0.5$, Fig. 1 shows the variation of $N(\epsilon)$ vs. j for different values of ϵ and the positions of the grid points where $d_k^j \geq 10^{-2}$. In two-dimensional case Fig. 2 shows the position of the grid points with $|d_k^j| \geq \epsilon = 10^{-2}$ for $f(x, y) = exp(-1000(x^2 + y^2))$ and $f(x, y) = exp(-50(x + 1)^2) + exp(-50(y + 1)^2)$. Now, we demonstrate the construction of the adaptive grid. Suppose that X^c is the current coarse grid and $\{f(x_j)\}_{j\in X^c}$ is known. Use interpolation to compute $\{f(x_j)\}_{j\in X^J}$ from $\{f(x_j)\}_{j\in X^c}$. Apply FDWT on this expanded $f(x)$ to obtain the diffusion scaling and wavelet coefficients. In the process of generating the new coarse grid X^c from the finest grid X^J, all the points corresponding to the diffusion scaling functions will be kept intact. A point corresponding to an active diffusion wavelet i.e. the point where $|d_k^j| \geq \epsilon$ is kept intact.

It is important to note that while solving a PDE, we will not adapt the grid at each time step. To ensure the accuracy, the grid points corresponding to the diffusion wavelets which can possibly become significant during the period of the time when the grid remains unchanged should also be included. Thus if x_l is the point in the finest grid X^J corresponding to active diffusion wavelet (call it an active grid point), then x_l as well as the points x_m such that $|m - l| \leq L$ (for some fixed positive integer L) are included in the new coarse grid X^c. The diffusion wavelet ψ_m^j is said to be in the adjacent zone of ψ_l^j if $|m - l| \leq L$. Figs. 3 and 4 explain the construction of the adaptive grid in one and two dimensional cases respectively.

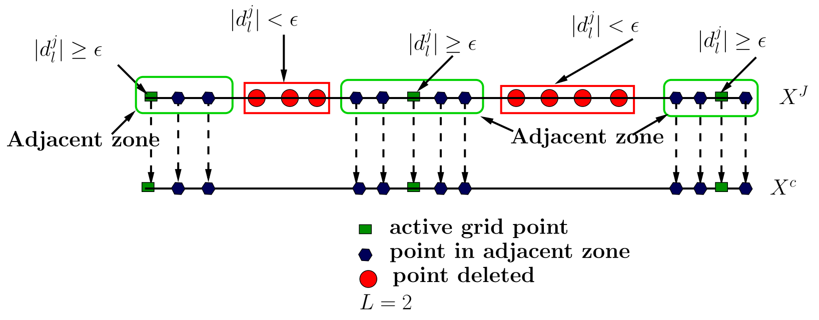


Fig. 3. Generating the adaptive grid in one-dimensional case.

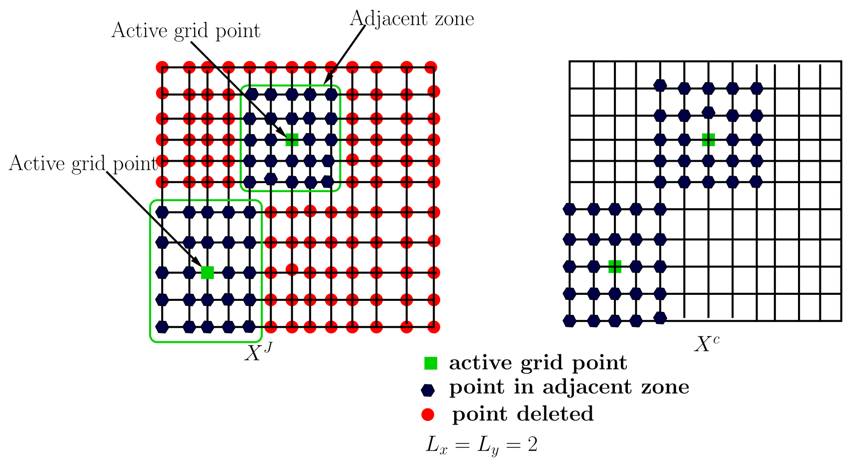


Fig. 4. Generating the adaptive grid in two-dimensional case.

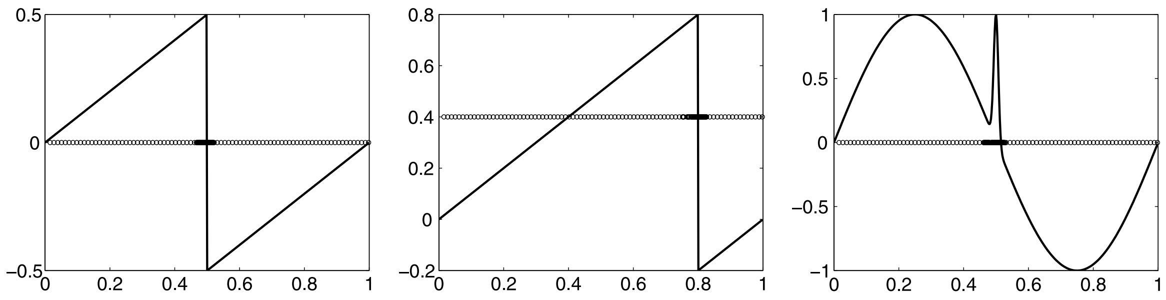


Fig. 5. The adaptive grid for (a) Sawtooth function with discontinuity at $x = 0.5$ (b) Sawtooth function with discontinuity at $x = 0.8$ (c) $f(x) = sin(2\pi x) + exp(-10^4(x-0.5)^2)$.

For $\epsilon = 10^{-2}$, $L = 2$ and $\lambda = 7$, Fig. 5 shows the adaptive grid created using the above explained technique for different functions.

3.3. FADWM for non-linear PDEs

To demonstrate FADWM we will consider the second-order PDE of the type

$$\frac{\partial u}{\partial t} = F\left(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}\right), \quad t \geq 0, x \in [0, 1], \tag{3}$$

with $u(x, 0) = u_0(x)$ and with suitable boundary conditions. F is a non-linear operator. For discretization of (3) we use Lagrange interpolating polynomial through p points $u_I(x) = \sum_{k=i-w}^{i+w} u(x_k)\frac{P_{w,i,k}(x)}{P_{w,i,k}(x_k)}$, $w = \frac{(p-1)}{2}$, $P_{w,i,k} = \prod_{\substack{l=i-w \\ l\neq k}}^{i+w}(x - x_l)$. For sufficiently smooth u, we have the estimate $\left|\frac{d^d}{dx^d}u_I(x_k) - \frac{d^d}{dx^d}u(x_k)\right| = O(h^{p-1})$, $d = 1, 2$, where $h = \max\{h_i = x_i - x_{i-1}\}$.

Table 1
Grid modifications while solving test problem 1 for different ν.

Time (t)	$\nu = 0.005$		$\nu = 0.01$		$\nu = 0.05$	
	$N(\epsilon = 0)$	$N(\epsilon)$	$N(\epsilon = 0)$	$N(\epsilon)$	$N(\epsilon = 0)$	$N(\epsilon)$
0	512	122	512	393	512	502
0.0625	122	141	393	395	502	496
0.1250	141	134	395	391	496	494
0.1875	134	125	391	393	494	497
0.25	125	133	393	398	497	494
0.3125	133	128	398	399	494	494
0.3750	128	128	399	398	494	490
0.4375	128	129	398	394	490	491

With this space discretization we get $\frac{d}{dt}\mathbf{u}(t) = L\mathbf{u}(t) + N(\mathbf{u}(t))$, where $L\mathbf{u}(t)$ and $N(\mathbf{u}(t))$ represent the linear and non-linear part respectively. Now using Crank Nicolson scheme for discretization in time [27], we get

$$\mathbf{u}^n = A^{-1}\left(B\mathbf{u}^{n-1} + \mathbf{f}(\mathbf{u}^n, \mathbf{u}^{n-1})\right), \quad \mathbf{u}^0 = \mathbf{u}_0, \tag{4}$$

where $A = I - \frac{\Delta t}{2}L$, $B = I + \frac{\Delta t}{2}L$, and $\mathbf{f}(\mathbf{u}^n, \mathbf{u}^{n-1}) = N\left(\frac{\mathbf{u}^n + \mathbf{u}^{n-1}}{2}\right)$. An iterative process is required to solve (4), i.e., $\mathbf{u}^n_{(0)} = \mathbf{u}^{n-1}$, and iterate until the required convergence, i.e., $\mathbf{u}^n_{(q+1)} = A^{-1}\left(B\mathbf{u}^{n-1} + \mathbf{f}(\mathbf{u}^n_{(q)}, \mathbf{u}^{n-1})\right), \quad q = 0, 1, \ldots$. 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 as explained in Section 3.1. Hence A^{-1} can be computed as follows:

$$A^{-1} = (I - T)^{-1} = \sum_{k=0}^{\infty} T^k = \lim_{K\to\infty} \prod_{k=0}^{K} (I + T^{2^k}). \tag{5}$$

The same diffusion wavelet is used for generating an adaptive grid. Hence in FADWM we will use the diffusion wavelet for two purposes, one for the fast computation of A^{-1} and second for making the adaptive grid. The algorithm for FADWM is as follows:

```
Construct the new coarse grid using $\mathbf{u}^{n-1}$.
Compute the matrices $A$ and $B$ for the new coarse grid.
Compute $\mathbf{A}^{-1}$ using (5).
Compute $\mathbf{u}^n$ using (4) on the new coarse grid.
Perform several time steps on the new coarse grid.
```

Any type of boundary conditions can be dealt with this method because of the following reasons: (1) if we are solving PDE on the domain $\Omega \in \mathbb{R}^n$, then the diffusion wavelet is constructed for the space $\mathcal{L}_2(\Omega)$ and not for $\mathcal{L}_2(\mathbb{R}^n)$; (2) in FADWM the Neumann and periodic boundary conditions will be incorporated in the operator T and Dirichlet's boundary conditions will be incorporated in the right hand side vector $\mathbf{f}$. It should be noted that the order of accuracy of the proposed method before introducing adaptivity is $O(N^{-(p-1)} + \Delta t^2)$ (in numerical experiments we have used $p = 3$). As clear from (2), for a threshold ϵ, the error introduced during adaptivity is $O(\epsilon)$. Therefore, the order of accuracy of the final scheme is $O((N(\epsilon))^{-(p-1)} + \Delta t^2 + \epsilon)$.

4. Numerical results

FADWM is used to solve three test problems. In the first two test problems (3) is considered with $F(u, x, t, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}) = -u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}$ and $\nu = 0.005$. Third test problem is a two-dimensional coupled Burger's equation.

Test problem 1: the initial condition is $u_0(x) = \sin(2\pi x)$ and the boundary condition is periodic, i.e., $u(x, t) = u(x + 1, t)$. The analytic solution [3] of this problem is a stationary wave which develops a steep gradient at $x = 0.5$. We used FADWM to solve this PDE. Table 1 shows the grid modifications at different times for different values of ν. It is clear from the table that $N(\epsilon)$ decreases with the decrease in the value of ν and hence the method performs better for low values of ν. Fig. 6 shows the solution and the corresponding adaptive grid at different times. Fig. 7 shows the point wise error at different times and it is observed that the error is maximum near the variation in the solution as expected. Hence more points will be added in this area and it will lead to an automatic adaptive grid generation. Fig. 8(a), (b) verifies the convergence of the method with respect to $N(\epsilon)$ and the threshold ϵ respectively (note that u_{num} stands for the numerical solution computed using FADWM and u_{ana} stands for the analytic solution of the problem). From these graphs it can be observed that the expected order of

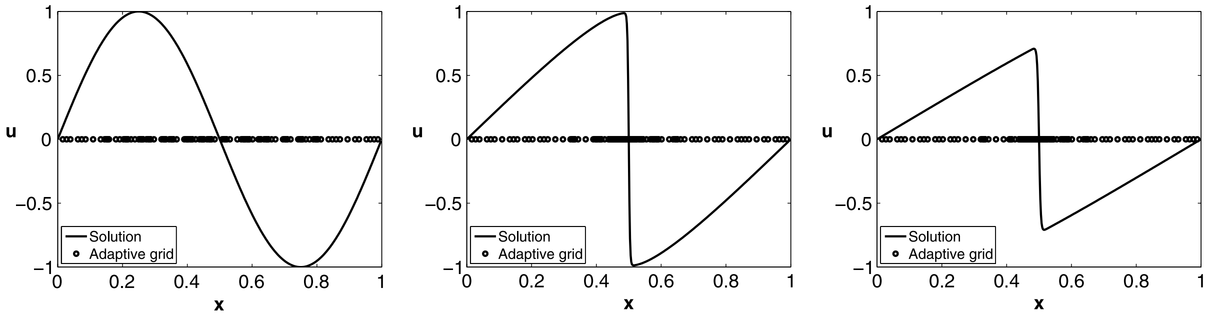


Fig. 6. Solution and the corresponding adaptive grid at $t = 0, 0.25, 0.5$.

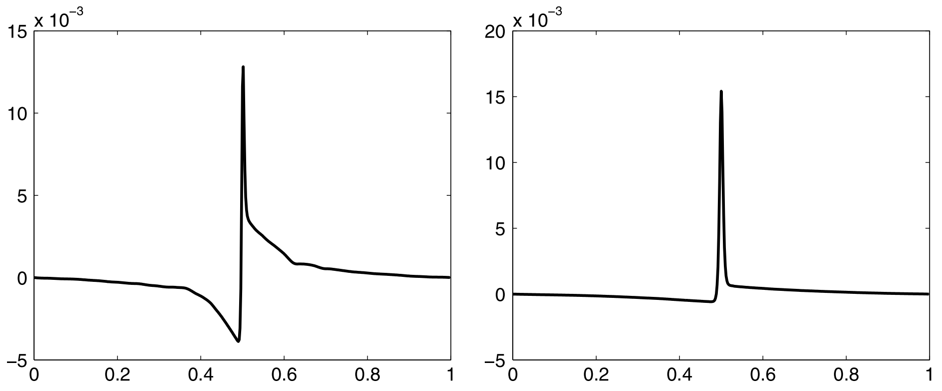
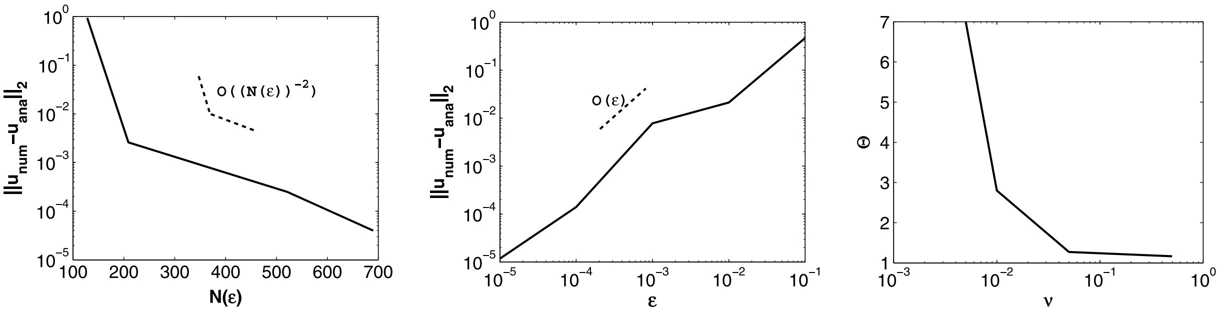


Fig. 7. Point wise error at $t = 0.25, 0.5$.



(a) $\|u_{num} - u_{ana}\|_2$ vs. $N(\epsilon)$.

(b) $\|u_{num} - u_{ana}\|_2$ vs. ϵ.

(c) Θ vs. ν.

Fig. 8. Results for test problem 1.

Table 2
The performance of FADWM for test problem 1.

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
CPU(ϵ)	0.1	0.1	0.35	0.42
Θ	7	7	2	1.667

convergence has been achieved. It is clear that error will decrease with the decrease in value of ϵ but the computational cost will increase. Therefore ϵ is to be chosen wisely which makes a balance between the error and the computational cost. In order to observe the efficiency of FADWM the CPU time taken by the FADWM (we will call it CPU(ϵ)) is compared with the CPU time taken by the finite difference method without adaptivity and without fast computation of powers of the finite difference matrices (we will call it CPU($\epsilon = 0$)). For its measurement we define the time compression coefficient as $\Theta = \frac{\text{CPU}(\epsilon=0)}{\text{CPU}(\epsilon)}$. The larger the value of Θ, the more efficient is the adaptive algorithm. Table 2 gives variation of CPU(ϵ) with ϵ for $\nu = 0.005$ (CPU($\epsilon = 0$) $= 0.7$). It should be noted that for $\epsilon = 10^{-1}$ and $\epsilon = 10^{-2}$ the value of Θ is same. But for $\epsilon = 10^{-2}$, the error will be less, so one should use $\epsilon = 10^{-2}$. Fig. 8(c) shows the variation of Θ as a function of ν. It can be observed that lower is the value of ν, higher is the value of Θ, i.e., the method performs better for lower values of ν.

Table 3
Grid modifications while solving test problem 2.

Time (t)	0	0.0625	0.1250	0.1875	0.25	0.3125	0.3750	0.4375
$N(\epsilon = 0)$	512	33	33	38	48	58	57	52
$N(\epsilon)$	33	33	38	48	58	57	52	60

Table 4
The performance of FADWM for test problem 2.

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
CPU(ϵ)	0.1	0.2	0.22	0.24
Θ	4	2	1.8182	1.6667

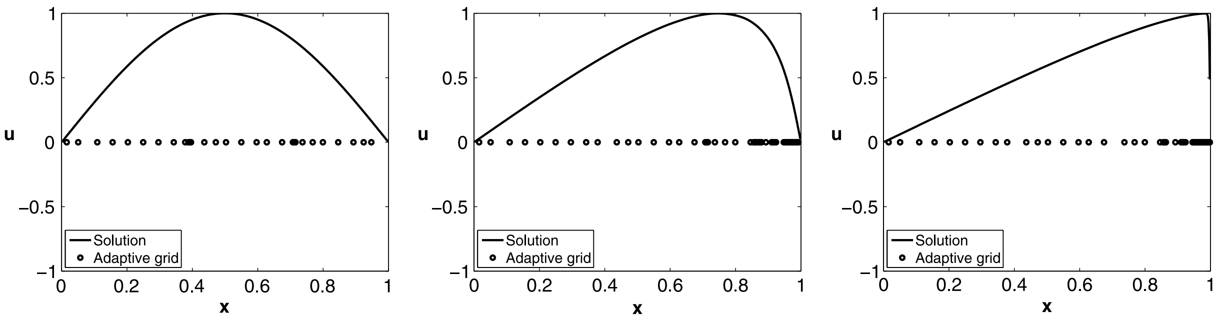


Fig. 9. Solution and the corresponding adaptive grid modification at $t = 0, 0.25, 0.5$.

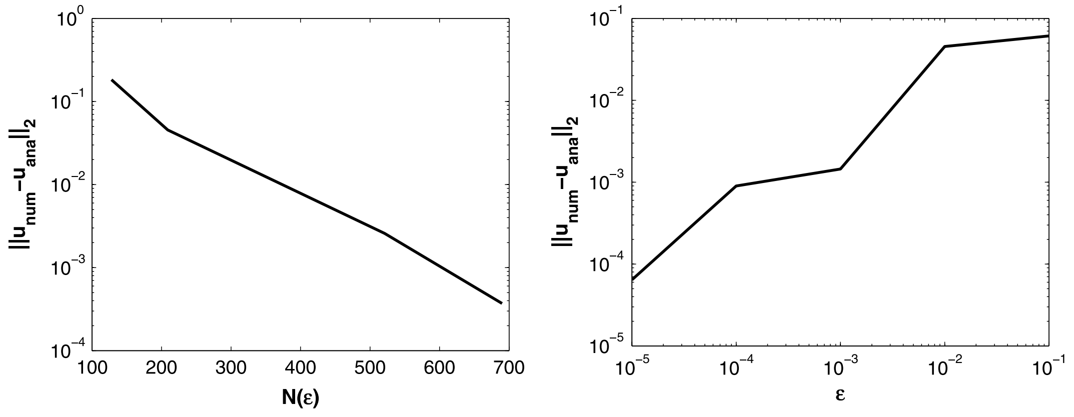


Fig. 10. $\|u_{num} - u_{ana}\|_2$ vs. $N(\epsilon)$ and ϵ.

Test problem 2: consider the initial condition $u_0(x) = sin(\pi x)$ and homogeneous Dirichlet's conditions $u(0, t) = u(1, t) = 0$. Table 3 shows the grid modifications at different times. Fig. 9 shows the solution and the corresponding adaptive grid at different times. Fig. 10 verifies the convergence of the method with respect to $N(\epsilon)$ and ϵ. Table 4 shows values of Θ for different values of ϵ (CPU($\epsilon = 0$) = 0.4).

This test problem is solved by Zhang et al. in [9] using a scheme based on properties of distributed approximating functional. We have compared our results with their results (shown in Table 5). It can be observed from the table that the maximum absolute errors with Zhang et al. and FADWM are almost same. The number of grid points used in DAF for $Re = 10$ is 10 and for $Re = 100$, it is 25. Whereas in the case of FADWM average number of grid points for $Re = 100$ is 40 and for $Re = 10$ it is 50, i.e., FADWM is more efficient for high Reynold's number (it was expected as clear from Table 1).

Test problem 3: we consider the two-dimensional coupled non-linear Burger's equation defined as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0, \tag{6}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) = 0, \quad 0 \le x, y \le 1, \tag{7}$$

Table 5
Comparison between Zhang et al. and FADWM.

Time	$\|u_{num} - u_{ana}\|_\infty$			
	$Re = 10$		$Re = 100$	
	Zhang et al.	FADWM	Zhang et al.	FADWM
$t = 0.05$	6.14×10^{-4}	8.22×10^{-4}	3.22×10^{-3}	2.05×10^{-3}
$t = 0.25$	7.63×10^{-4}	6.12×10^{-4}	5.98×10^{-3}	7.31×10^{-3}
$t = 0.75$	1.66×10^{-4}	5.07×10^{-4}	1.29×10^{-3}	6.00×10^{-3}
$t = 1.50$	7.70×10^{-5}	8.50×10^{-4}	2.57×10^{-5}	5.90×10^{-4}

Table 6
Grid modifications while solving two dimensional Burger's equation.

$time(t)$	0	0.10	0.20	0.30	0.40	0.50	0.60
$N(\epsilon = 0)$	400	44	74	82	77	72	64
$N(\epsilon)$	142	63	77	79	76	69	58

Table 7
The performance of FADWM in the case of two dimensional Burger's equation.

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
CPU(ϵ)	0.35	0.75	1.375	1.625
Θ	7.1429	3.3333	1.8182	1.5385

subject to suitable initial and boundary conditions [28]. $u(x, y, t)$ and $v(x, y, t)$ are the velocity components along x-axis and y-axis respectively, Re is the Reynold's number. After discretization we have

$$u_x = \boldsymbol{\mathcal{D}}_x^{(1)}\mathbf{u}(t) + a_u, \qquad u_y = \boldsymbol{\mathcal{D}}_y^{(1)}\mathbf{u}(t) + b_u, \qquad u_{xx} = \boldsymbol{\mathcal{D}}_x^{(2)}\mathbf{u}(t) + c_u, \qquad u_{yy} = \boldsymbol{\mathcal{D}}_y^{(2)}\mathbf{u}(t) + d_u. \tag{8}$$

Use (8) to discretize (6)

$$\begin{aligned}&\frac{d}{dt}\mathbf{u}(t) - \mathbf{N}_x(\mathbf{u}(t))\mathbf{u}(t) - \mathbf{N}_y(\mathbf{u}(t))\mathbf{v}(t) - \frac{1}{Re}\left(\boldsymbol{\mathcal{D}}_x^{(2)} + \boldsymbol{\mathcal{D}}_y^{(2)}\right)\mathbf{u}(t)\\ &\quad + (\text{diag}(a_u) + \text{diag}(c_u) + \text{diag}(d_u))\,\mathbf{u}(t) + (\text{diag}(b_u))\mathbf{v}(t) = 0,\end{aligned} \tag{9}$$

where $\mathbf{N}_x(\mathbf{u}(t)) = -\text{diag}(\boldsymbol{\mathcal{D}}_x^{(1)}\mathbf{u}(t))$, and diag($a$) denotes a matrix with the vector a as its diagonal. Let us call

$$A_u = (\text{diag}(a_u) + \text{diag}(c_u) + \text{diag}(d_u)), \qquad B_u = \text{diag}(b_u), \mathbf{L} = \frac{1}{Re}\left(\boldsymbol{\mathcal{D}}_x^{(2)} + \boldsymbol{\mathcal{D}}_y^{(2)}\right),$$

so that (9) can be written as

$$\frac{d}{dt}\mathbf{u}(t) + (-\mathbf{N}_x(\mathbf{u}(t)) + A_u)\,\mathbf{u}(t) + \left(-\mathbf{N}_y(\mathbf{u}(t)) + B_u\right)\mathbf{v}(t) - \mathbf{L}(\mathbf{u}(t)) = 0. \tag{10}$$

Now applying Crank Nicolson on (10), we get

$$\begin{aligned}\mathbf{u}^n =& \mathcal{A}_u^{-1}\left[\mathcal{B}_u\mathbf{u}^{n-1} + \Delta t\mathbf{N}_x\left(\frac{\mathbf{u}^n + \mathbf{u}^{n-1}}{2}\right)\left(\frac{\mathbf{u}^n + \mathbf{u}^{n-1}}{2}\right)\right.\\ &\left. + \Delta t\mathbf{N}_y\left(\frac{\mathbf{u}^n + \mathbf{u}^{n-1}}{2}\right)\left(\frac{\mathbf{v}^n + \mathbf{v}^{n-1}}{2}\right) + \Delta tB_u\left(\frac{\mathbf{v}^n + \mathbf{v}^{n-1}}{2}\right)\right],\end{aligned} \tag{11}$$

where

$$\mathcal{A}_u = \mathbf{I} + \Delta t\frac{A_u}{2} - \Delta t\frac{L}{2}, \qquad \mathcal{B}_u = \mathbf{I} - \Delta t\frac{A_u}{2} + \Delta t\frac{L}{2}.$$

On similar lines we can obtain an equation for $\mathbf{v}^n$. Table 6 shows the grid modifications at different times. Fig. 11 shows the solution and the corresponding adaptive grid at different times. Fig. 12 shows $u(x, y = 0.5, t)$, $v(x, y = 0.5, t)$ at different times and the corresponding adaptive grid. Fig. 13 verifies the convergence of the method with respect to $N(\epsilon)$ and the threshold ϵ. Table 7 shows values of CPU(ϵ) for different values of ϵ for the case of two dimensional Burger's equation, CPU($\epsilon = 0$) = 2.5. In order to compare our results with the results of Wei et al. in [10], we have solved (6) and (7) with the initial conditions $u(x, y, 0) = \sin(\pi x)\sin(\pi y)$, $v(x, y, 0) = (\sin(\pi x) + \sin(2\pi x))(\sin(\pi y) + \sin(2\pi y))$ and homogeneous boundary condition for both variables. We used $Re = 1, 100$ and computed the solution at $t = 0.01$. The number of grid points used by FADWM are 52 for $Re = 1$ and 36 for $Re = 100$, which again shows that the method performs for high Reynold's number. Our results are quite in agreement with the results of Wei et al. (see Table 8).

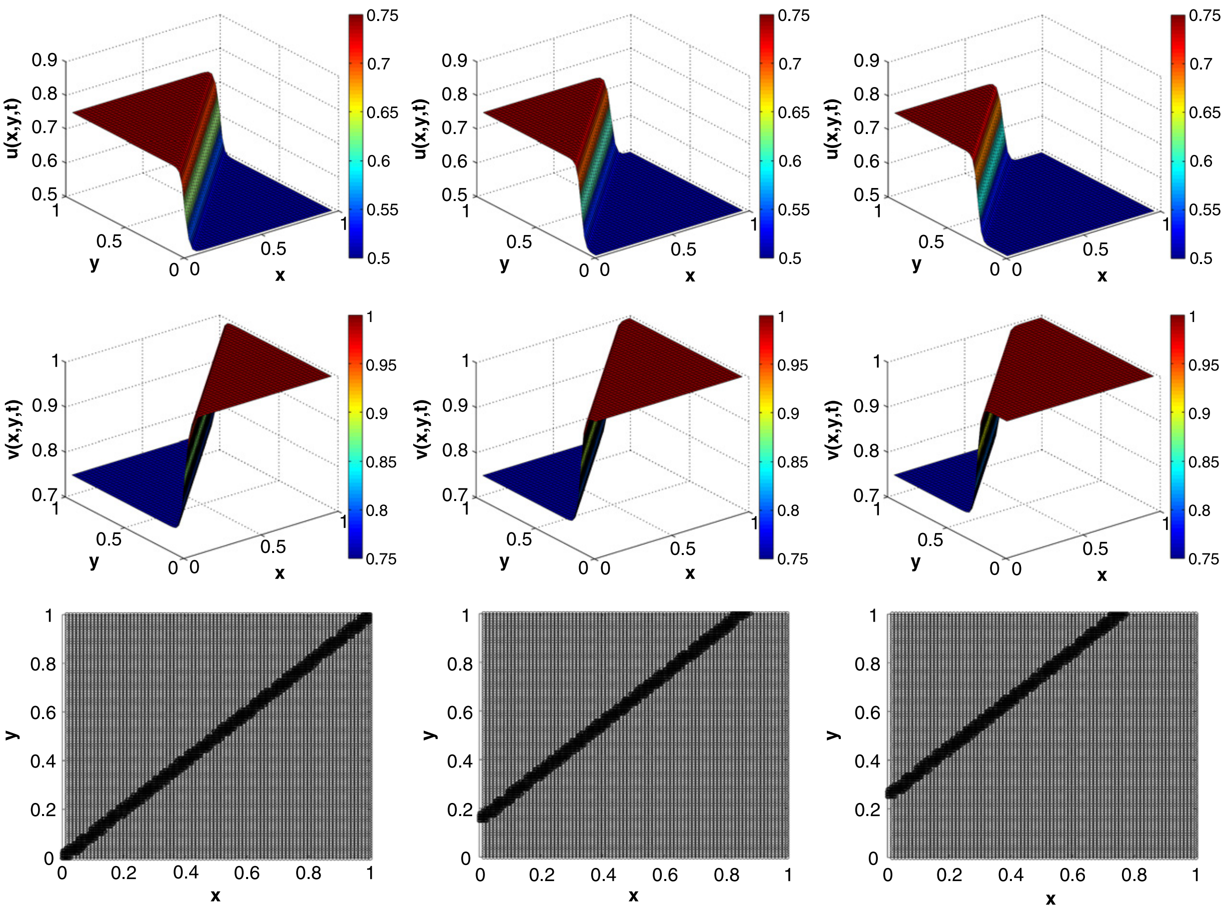


Fig. 11. Solution and the corresponding adaptive grid at $t = 0, 0.5, 1$.

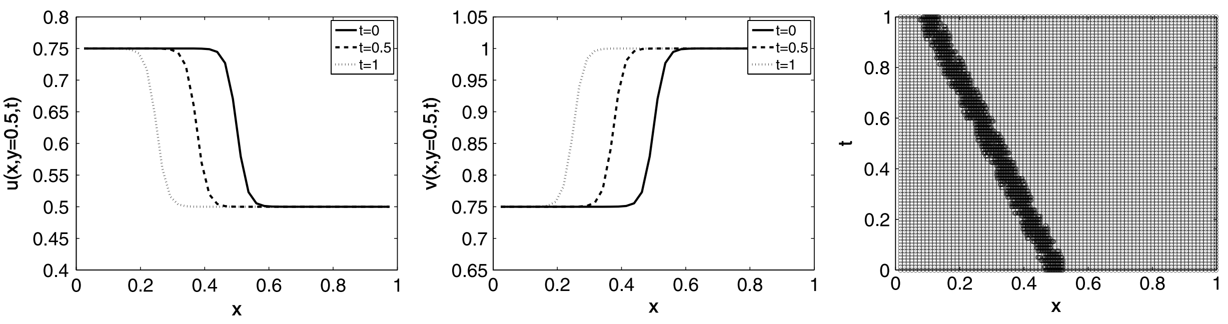


Fig. 12. $u(x, y = 0.5, t)$, $v(x, y = 0.5, t)$ and the corresponding adaptive grid.

Table 8
Comparison of FADWM with results of Wei et al. [10].

(x, y)	(0.1, 0.1)	(0.2, 0.8)	(0.4, 0.4)	(0.7, 0.1)	(0.9, 0.9)
Wei et al.					
$(\Delta x = \Delta y = \frac{1}{20})$	$u = 0.07250$	$u = 0.27758$	$u = 0.72169$	$u = 0.20478$	$u = 0.07946$
$(\Delta t = \frac{1}{2000})$	$v = 0.43116$	$v = -0.12436$	$v = 1.65256$	$v = 0.06681$	$v = 0.01342$
FADWM					
$(N(\epsilon) = 52)$	$u = 0.07231$	$u = 0.27757$	$u = 0.72156$	$u = 0.20253$	$u = 0.07939$
$(\Delta t = \frac{1}{2000})$	$v = 0.43091$	$v = -0.12447$	$v = 1.65321$	$v = 0.06672$	$v = 0.01341$

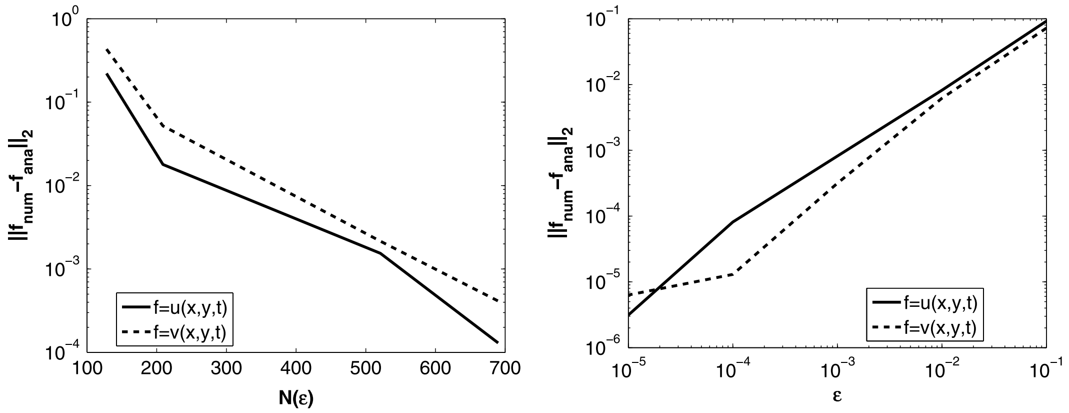


Fig. 13. $\|u_{num} - u_{ana}\|_2$, $\|v_{num} - v_{ana}\|_2$ vs. $N(\epsilon)$ and ϵ.

5. Conclusion and future work

In this paper, FADWM is developed for the numerical solution of Burger's equation. The diffusion operator obtained by discretizing the Burger's equation is used for the construction of diffusion wavelet. The diffusion wavelet is used for constructing an adaptive grid as well as for the fast computation of the powers of the finite difference matrices involved in the computation of the numerical solution. The efficiency and convergence of FADWM are verified for all the test problems. The CPU time taken by the FADWM is compared with the CPU time taken by the finite difference method and it turns out that FADWM is performing much better. To the best of our knowledge, useful properties of diffusion wavelet constructed in [24] are for the first time exploited for generating the adaptive grid. In future we will use FADWM for turbulence modelling and generalize it to general manifolds.

Acknowledgement

The first author would like to thank Council of Scientific and Industrial Research for providing the PhD scholarship.

References

[1] H. Bateman, Some recent researches in motion of fluids, Mon. Weather Rev. 43 (1915) 163–170.
[2] J.M. Burger, A mathematical model illustrating the theory of turbulence, Adv. Appl. Mech. 1 (1948) 177–199.
[3] G. Whitham, Linear and Nonlinear Waves, Wiley and Sons, 1974.
[4] S. Pudasaini, Some exact solutions for debris and avalanche flows, Phys. Fluids, 23.
[5] E. Varoglu, W. Finn, Space–time finite elements incorporating characterstics for burger's equation, Int. J. Numer. Methods Eng. 16 (1980) 171–184.
[6] S. Biringen, A. Satti, Comaprison of several finite-difference methods, J. Aircr. 27 (1990) 90–92.
[7] P. Bar-Yoseph, E. Moses, U. Zrahia, A.L. Yarin, Space–time spectral element methods for one dimensional non linear advection diffusion problems, J. Comput. Phys. 119 (1995) 62–75.
[8] G. Mansell, W. Merryfield, B. Shizgal, U. Weinert, A comparison of differential quadrature methods for solution of partial differential equations, Comput. Methods Appl. Mech. Engrg. 104 (1993) 295–316.
[9] D.S. Zhang, G.W. Wei, D.J. Kouri, Burger's equation with high Reynolds number, Phys. Fluids 9 (1997) 1853–1855.
[10] G.W. Wei, D.S. Zhang, D.J. Kouri, D.K. Hoffman, Distributed approximating functional approach to burger's equation in one and two space dimensions, Comput. Phys. Commun. 111 (1998) 93–109.
[11] K. Xia, M. Zhan, D. Wan, G.W. Wei, Adaptively deformed mesh based interface method for elliptic equations with discontinuous coefficients, J. Comput. Phys. 231 (2012) 1440–1461.
[12] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[13] L. Jameson, A wavelet-optimized, very high order adaptive grid and order numerical method, SIAM J. Sci. Comput. 19 (1998) 1980–2013.
[14] A. Garba, A wavelet collocation method for numerical solution of Burgers' equation, (Tech. Rep.) International Center for Theorytical Physics, 1996.
[15] D.C. Wan, G.W. Wei, The study of quasi wavelets based numerical method applied to Burger's equation, Appl. Math. Mech. 21 (2000) 1099–1110.
[16] G.W. Wei, Quasi wavelets and quasi interpolating wavelets, Chem. Phys. Lett. 296 (1998) 215–222.
[17] 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.
[18] O. V. Vasilyev, C. Bowman, Second generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165 (2000) 660–693.
[19] B. Merriman, S.J. Ruuth, Diffusion generated motion of curves on surfaces, J. Comput. Phys. 225 (2007) 2267–2282.
[20] C.B. Macdonald, S.J. Ruuth, The implicit closest point method for the numerical solutions of partial differential equations on the surfaces, SIAM J. Sci. Comput. 31 (2009) 4330–4350.
[21] M. Bertalmio, L.T. Cheng, O. Osher, G. Sapiro, Variational problems and partial differential equations on implicit surfaces, J. Comput. Phys. 174 (2001) 759–780.
[22] M. Bergdorf, G.-H. Cottet, P. Koumoutsakos, Multilevel adaptive particle methods for convection–diffusion equations, Multiscale Model. Simul. 4 (2005) 328–357.
[23] 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.
[24] R.R. Coifman, M. Maggioni, Diffusion wavelets, Appl. Comput. Harmon. Anal. 21 (2006) 53–94.
[25] R.R. Coifman, S. Lafon, Diffusion maps, Appl. Comput. Harmon. Anal. 21 (2006) 5–30.
[26] D.L. Donoho, Interpolating wavelet transforms, Tech. Rep. 408, Department of Statistics, Stanford University, 1992.
[27] O.M. Nilsen, Wavelets in scientific computing (Ph.D. thesis), Technical University of Denamark, Lyngby, 1998.
[28] A.R. Bahadir, A fully implicit finite-difference scheme for two-dimensional Burgers' equations, Appl. Math. Comput. 137 (2003) 131–137.