Journal of Computational and Applied Mathematics 230 (2009) 803–812

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Wavelet optimized finite difference method using interpolating wavelets for self-adjoint singularly perturbed problems

Vivek Kumar [a,*], Mani Mehra [b]

[a] Tata Institute of Fundamental Research, Center for Applicable Mathematics, Bangalore - 560065, India
[b] Department of Mathematics, Indian Institute of Technology Delhi, New Delhi-110016, India

ARTICLE INFO

Article history:
Received 21 September 2006
Received in revised form 21 January 2009

Keywords:
Wavelet optimized finite difference method
Self-adjoint singularly perturbed problems
Interpolating wavelet transform
Lagrange finite difference

ABSTRACT

We design a wavelet optimized finite difference (WOFD) scheme for solving self-adjoint singularly perturbed boundary value problems. The method is based on an interpolating wavelet transform using polynomial interpolation on dyadic grids. Small dissipation of the solution is captured significantly using an adaptive grid. The adaptive feature is performed automatically by thresholding the wavelet coefficients. Numerical examples have been solved and compared with non-standard finite difference schemes in [J.M.S. Lubuma, K.C. Patidar, Uniformly convergent non-standard finite difference methods for self-adjoint singular perturbation problems, J. Comput. Appl. Math. 191 (2006) 228–238]. The proposed method outperforms the non-standard finite difference for studying singular perturbation problems for small dissipations (very small ϵ) and effective grid generation. Therefore, the proposed method is better for studying the more challenging cases of singularly perturbed problems.

1. Introduction

We consider the following class of self-adjoint singularly perturbed two-point boundary value problems in the conservation form (please see [2])

$$Lu \equiv -\epsilon(a(x)y'(x))' + b(x)y(x) = g(x), \quad \text{where } 0 \le x \le 1, \tag{1}$$

subject to

$$y(0) = \alpha, \qquad y(1) = \beta, \quad \alpha, \beta \in \mathbb{R}, \tag{2}$$

where ϵ is a small positive parameter and $a(x)$, $b(x)$ and $g(x)$ are smooth functions that satisfy

$$a(x) \ge a^* > 0, \qquad a'(x) \ge 0, \qquad b(x) \ge b^* > 0.$$

Under these conditions, the operator L admits a maximum principle [3]. Such problems have non-smooth solutions as $\epsilon \to 0$ with singularities related to boundary layers [4] and arise in various fields of science and engineering, for instance, fluid mechanics, quantum mechanics, optimal control, chemical-reactor theory, aerodynamics, reaction-diffusion process, geophysics etc. There are two kinds of approaches to deal with such problems: the first one is fitted mesh method which consists of choosing finer meshes in the layer region(s), and another is fitted operator method in which meshes remain uniform and the difference operator reflects the singularly perturbed nature of the differential operator. These kinds of

* Corresponding author. Tel.: +91 8023600062.
E-mail addresses: vivek@math.tifrbng.res.in, vivekkumar.ag@gmail.com (V. Kumar), mmehra@maths.iitd.ac.in (M. Mehra).

0377-0427/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2009.01.017

problems using one or both of the strategies have been discussed in [5,6]. Earlier authors have discussed such problems using B-spline with Shishkin mesh [7]. One of the drawbacks with the Shishkin mesh method is that it requires a priori knowledge of the location and width of the boundary layers, therefore it motivates us to look for some adaptive methods. Recently, Lubuma and Patidar [1] have given a non-standard finite difference scheme to deal with such problems using the second approach (fitted operator). Wavelet optimized finite difference [8] works by using an adaptive wavelet to generate irregular grids which is then exploited for the finite difference method (Lagrange finite difference in our case) and therefore it comes under fitted mesh methods.

In singular perturbation problems we have shocks as boundary layers. For such kinds of problems, a solution can be smooth in most of the solution domain with a small area where the solution changes quickly. When solving such problems numerically, one would like to adjust the discretization to the solution. In terms of mesh generation (first approach), we want to have many points in an area where the solution has strong variations and few points in the area where the solution has weak variations. With a very small perturbation parameter ϵ, a large N_j (total no. of mesh points at jth level) is required to obtain accurate solution. For a good resolution of the numerical solution, at least one of the collocation points should lie in the boundary layer. For example, if the problem possesses a boundary layer of width $O(\epsilon)$, then on a uniform grid with $O(N_j^{-1})$ spacing between the points we need $N_j = O(\epsilon^{-1})$, which is not practically possible when $\epsilon \ll 1$. There have been many attempts to develop numerical methods using specially designed grids that contain more points in and around the layers from time to time. Farrell et al. [6] developed a successful upwind central difference scheme on a piecewise uniform mesh. More literature can be found in [4–7].

Wavelets have been making their presence felt in many pure and applied areas of science and engineering [8,9]. Wavelets detect information at different scales and at different locations throughout the computational domain. Wavelets can provide a bases in which the basis functions are constructed by dilating and translating a fixed function known as the mother wavelet (first generation wavelets). The mother wavelet can be seen as a high pass filter in the frequency domain. One of the key strength of the wavelet methods is data compression. An efficient basis is one in which a given set of data can be represented with as few basis elements as possible. Suppose we have wavelet representation of a function

$$\sum_k c_k^j \varphi_k^j(x) + \sum_{j,k} d_k^j \psi_k^j(x),$$

where $\varphi_k^j(x)$ are scaling functions and $\psi_k^j(x)$ are wavelets. The coefficients of the scaling functions c_k^j, deal with smoother part of the function, while the wavelet coefficients d_k^j contain information of the function's behavior on successive finer scales. The most common way of compressing such a representation is thresholding. We generally delete all wavelet coefficients of magnitude less than some threshold, say τ. If the total number of coefficients in the original representation are N_j, we have N_s significant coefficients left after the thresholding. Note that by thresholding a wavelet representation, we have a way to find a adaptive feature and we can also use this representation to compute function values at any point.

We discuss interpolating wavelet transform and adaptive mesh generation in Section 2. Lagrange finite difference has been given in Section 3 and numerical results and comparisons have been given in Section 4.

2. Interpolating wavelet transform

Here we briefly describe the interpolating wavelets of Donoho and Harten [10,11]. Interpolating wavelets are constructed on a set of dyadic grids on the line

$$\mathbb{A}^j = \{x_k^j \in \mathbb{R} : x_k^j = 2^{-j}k, k \in \mathbb{Z},\ j \in \mathbb{Z}\}, \tag{3}$$

where x_k^j are the grid (collocation) points and j is the level of resolution. Since $x_k^{j-1} = x_{2k}^j$, it follows that $\mathbb{A}^{j-1} \subset \mathbb{A}^j$. Essentially, interpolating wavelets can be formally introduced through the interpolating subdivision scheme of Deslauriers and Dubuc [12], which consider the problem of building an interpolant $f^j(x)$ on a grid $\mathbb{A}^{j+1}$ for a given data sequence $f(x_k^j)$. The algorithm proceeds by interpolating the data $f(x_k^j)$ to the points on the grid $\mathbb{A}^{j+1}$ which do not belong to $\mathbb{A}^j$. The even numbered grid points x_{2k}^j exist in V_j and the corresponding function values are kept unchanged. Values at odd numbered grid points x_{2k+1}^j are computed from the polynomial interpolation from the values at the even numbered grid points. The interpolation is achieved by constructing local polynomials, $P_{2N-1}(x)$ of order $2N-1$, which uses $2N$ closest points. For example, to find the value of the interpolant at location x_{2k+1}^{j+1} we construct the polynomial of order $2N-1$ based on the values of the function at locations $x_{k+1}^j (l = -N+1, \dots, N)$ and evaluate it at location x_{2k+1}^{j+1}. Evaluating this polynomial at points x_{2k+1}^{j+1} and substituting the values of polynomial coefficients expressed in terms of values $f(x_k^j)$, we can easily get

$$f^j(x_{2k+1}^{j+1}) = \sum_{l=-N+1}^{N} w_{k,l}^j f(x_{k+1}^j). \tag{4}$$

The main attraction of the interpolating subdivision scheme is that the values of these weights are the same for evenly spaced grids and the procedure can be easily extended to the nonuniform grids, which will result in location dependent weights. The adaption to the boundaries is simple. We use the closest point inside the boundary on the coarser grid to define the interpolating polynomial.

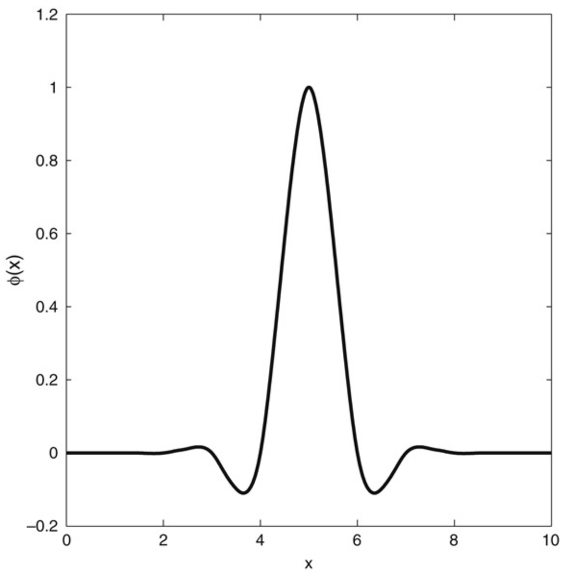


Fig. 1. Scaling function $\phi(x)$.

The interpolating scaling function $\phi_k^j(x)$ can be formally defined by setting $f(x_l^j) = \delta_{l,k}$, where $\delta_{l,k}$ is the Kronecker delta and then performing the interpolating subdivision scheme up to an arbitrary high level of resolution J. Now using the linear superposition, it is easy to show that

$$f^j(x) = \sum_k c_k^j \phi_k^j(x), \tag{5}$$

where $c_k^j = f(x_k^j)$. An example of an interpolating scaling function $\phi(x)$ is shown in Fig. 1. It is easy to show that the interpolating function has the following properties:

- compact support of $[-2N+1, 2N-1]$;
- $\phi(x)$ is cardinal interpolating; i.e $\phi(x) = \delta_{k,0}$;
- linear combination of $\phi_k^j(x)$ reproduce the polynomials up to degree $2N-1$;
- $\phi(x)$ satisfies a refinement relation $\phi_k^j = \sum_{l \in \mathbb{A}^{j+1}} h_{k,l}^j \phi_l^{j+1}$
- $\phi(x)$ is the autocorrelation of the Daubechies scaling function of order $2N$ [13].

Used recursively, the interpolating subdivision scheme generates function values on a finer grid for given values on coarse grid. On the contrary, when we go from a finer grid to a coarser grid, one could just throw away half of the grid points at each level, but at the same time some information is lost. Instead at each level, for odd numbered grid points, we compute the differences between the known function values and the function values predicted by the interpolation from the coarser grid which are known as wavelet coefficients d_k^j. Repeating these recursively we have an algorithm for wavelet transform as follows.

The forward interpolating wavelet transform can be written as

$$d_k^j = \frac{1}{2}\left(c_{2k+1}^{j+1} - \sum_l w_{k,l}^j c_{2k+2l}^{j+1}\right), \tag{6}$$

$$c_k^j = c_{2k}^{j+1}, \tag{7}$$

while the inverse wavelet interpolation transform is given by

$$c_{2k}^{j+1} = c_k^j, \tag{8}$$

$$c_{2k+1}^{j+1} = 2d_k^j + \sum_l w_{k,l}^j c_{k+l}^j, \tag{9}$$

where $w_{k,l}^j$ are the interpolating coefficients as introduced in (4). The algorithm for constructing interpolating wavelets on an interval and on an uniform grid are same except the modification of weights at the boundaries.

2.1. Adaptive mesh generation

Adaptive mesh is generated in the following steps:

step 1: First we solve the given problem on a uniform mesh for the initial solution profile.

step 2: We apply the wavelet transform to the solution profile and calculate the wavelet coefficients. Wavelet coefficients will be smaller where the solution is smooth and larger at the location of solution with high gradient like boundary layers in our case.

step 3: We remove the mesh points where $|d_k^j| < \tau$ and keep the remaining mesh points.

step 4: We apply the Lagrange finite difference as given in the next section on the newly generated non-uniform adaptive mesh in step 3.

3. Lagrange finite difference

Using adaptive mesh points from the interpolating wavelet, derivatives on non-uniform grid are approximated using Lagrangian interpolating polynomial through p points [8]. We consider only odd $p \geq 3$ because it makes the algorithm simpler. Let $w = \frac{p-1}{2}$ and define

$$u_I(x) = \sum_{k=i-w}^{i+w} u(x_k) \frac{P_{w,i,k}(x)}{P_{w,i,k}(x_k)}, \tag{10}$$

where

$$P_{w,i,k}(x) = \prod_{l=i-w, l\neq k}^{i+w} (x - x_l).$$

It follows that $u_I(x_i) = u(x_i)$ for $i = 0, 1, 2 \dots N_J - 1, N_J$ i.e u_I interpolates u at the grid points. Differentiation of $u_I(x) d$ times yields

$$u_I^d(x) = \sum_{k=i-w}^{i+w} u(x_k) \frac{P_{w,i,k}^d(x)}{P_{w,i,k}(x_k)}. \tag{11}$$

The derivatives $u_I^d(x)$ can then be approximated at all the grid points by

$$u_I^d = D_p^d u,$$

where the differentiation matrix D_p^d is defined by

$$[D_p^d]_{i,k} = \frac{P_{w,i,k}^d(x_i)}{P_{w,i,k}(x_k)}; \quad d = 1, 2.$$

First and second derivatives are

$$P_{w,i,k}^{(1)}(x) = \sum_{l=i-w, l\neq k}^{i+w} \prod_{m=i-w, m\neq k,l}^{i+w} (x - x_m), \tag{12}$$

and

$$P_{w,i,k}^{(2)}(x) = \sum_{l=i-w, l\neq k}^{i+w} \sum_{m=i-w, m\neq k,l}^{i+w} \prod_{n=i-w, n\neq k,l,m}^{i+w} (x - x_n). \tag{13}$$

4. Numerical experiments and discussion

In this section we discuss some test cases to demonstrate the efficiency of the proposed method.

Test case 1: First we consider the test case of Reference [1], where in Eq. (1) $a(x)$, $b(x)$ and $g(x)$ are given as

$$a(x) = 1 + x^2, \qquad b(x) = 1 + x(1-x),$$
$$g(x) = 1 + x(1-x) - \exp(-x/\sqrt{\epsilon})[x(2x^2 - 3x + 1) - 2\sqrt{\epsilon}(2x^2 - x(1+\sqrt{\epsilon}) + 1)]$$
$$+ \exp(-(1-x)/\sqrt{\epsilon})[x^2(2x-1) + 2\sqrt{\epsilon}(2x^2 + x\sqrt{\epsilon} + 1)].$$

Its exact solution is given as

$$y(x) = 1 + (x-1)\exp(-x/\sqrt{\epsilon}) - x\exp(-(1-x)/\sqrt{\epsilon}).$$

This test case has boundary layers of width $O(\sqrt{\epsilon})$ at both the boundary points. The Table 1 shows the maximum error obtained by the proposed method for test case 1, which has also been solved in [1] using non-standard finite difference technique. On comparing Table 1 with Table 2 we find that our results are comparable with results in [1] for large ϵ and

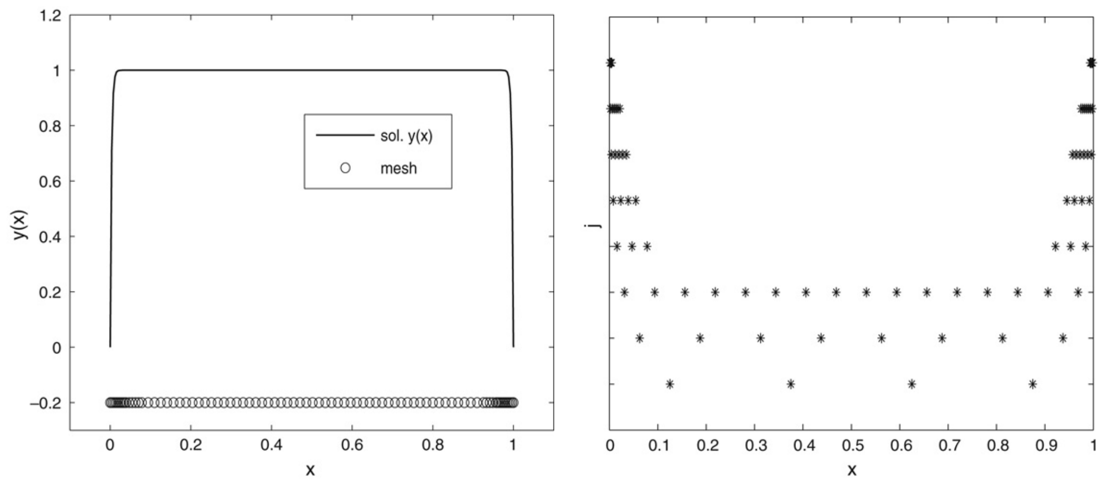
Table 1
Maximum error for test case 1 ($\tau = 10^{-4}$, $N = 2$).

$\epsilon = 10^{-k}$	$N_j = 2^4$	$N_j = 2^5$	$N_j = 2^6$	$N_j = 2^7$	$N_j = 2^8$
$k = 1$	$.72 \times 10^{-2}$	$.18 \times 10^{-2}$	$.22 \times 10^{-2}$	$.11 \times 10^{-2}$	$.54 \times 10^{-3}$
N_s[a]$\rightarrow$	16	22	27	42	74
$k = 2$	$.24 \times 10^{-1}$	$.83 \times 10^{-2}$	$.26 \times 10^{-2}$	$.42 \times 10^{-2}$	$.16 \times 10^{-2}$
	16	30	37	44	74
$k = 3$	$.77 \times 10^{-1}$	$.37 \times 10^{-1}$	$.14 \times 10^{-1}$	$.29 \times 10^{-2}$	$.32 \times 10^{-2}$
	16	28	41	55	80
$k = 4$	$.40 \times 10^{-1}$	$.83 \times 10^{-1}$	$.60 \times 10^{-1}$	$.25 \times 10^{-1}$	$.90 \times 10^{-2}$
	16	25	36	54	88
$k = 5$	$.46 \times 10^{-2}$	$.19 \times 10^{-1}$	$.63 \times 10^{-1}$	$.82 \times 10^{-1}$	$.45 \times 10^{-1}$
	16	24	33	50	85
$k = 6$	$.46 \times 10^{-3}$	$.21 \times 10^{-2}$	$.86 \times 10^{-2}$	$.33 \times 10^{-1}$	$.78 \times 10^{-1}$
	16	24	32	48	81
$k = 7$	$.46 \times 10^{-4}$	$.20 \times 10^{-3}$	$.86 \times 10^{-3}$	$.35 \times 10^{-2}$	$.14 \times 10^{-1}$
	16	24	32	48	80
$k = 8$	$.46 \times 10^{-5}$	$.20 \times 10^{-4}$	$.86 \times 10^{-4}$	$.35 \times 10^{-3}$	$.14 \times 10^{-2}$
	16	24	32	48	80
$k = 11$	$.46 \times 10^{-8}$	$.20 \times 10^{-7}$	$.86 \times 10^{-7}$	$.35 \times 10^{-6}$	$.14 \times 10^{-5}$
	16	24	32	48	80
$k = 12$	$.46 \times 10^{-9}$	$.20 \times 10^{-8}$	$.86 \times 10^{-8}$	$.35 \times 10^{-7}$	$.14 \times 10^{-6}$
	16	24	32	48	80

[a] Number of grid points after adaptivity.

Table 2
Maximum error for test case 1 as computed in [1].

$\epsilon = 10^{-k}$	$N_j = 2^4$	$N_j = 2^5$	$N_j = 2^6$	$N_j = 2^7$	$N_j = 2^8$
$k = 1$	$.94 \times 10^{-3}$	$.24 \times 10^{-3}$	$.60 \times 10^{-4}$	$.15 \times 10^{-4}$	$.37 \times 10^{-5}$
$k = 2$	$.57 \times 10^{-2}$	$.15 \times 10^{-2}$	$.38 \times 10^{-3}$	$.96 \times 10^{-4}$	$.24 \times 10^{-4}$
$k = 3$	$.28 \times 10^{-1}$	$.11 \times 10^{-1}$	$.29 \times 10^{-2}$	$.73 \times 10^{-3}$	$.18 \times 10^{-3}$
$k = 4$	$.71 \times 10^{-2}$	$.29 \times 10^{-1}$	$.21 \times 10^{-1}$	$.61 \times 10^{-2}$	$.17 \times 10^{-2}$
$k = 5$	$.53 \times 10^{-2}$	$.13 \times 10^{-2}$	$.16 \times 10^{-1}$	$.31 \times 10^{-1}$	$.15 \times 10^{-1}$
$k = 6$	$.53 \times 10^{-2}$	$.13 \times 10^{-2}$	$.33 \times 10^{-3}$	$.31 \times 10^{-2}$	$.25 \times 10^{-1}$
$k = 7$	$.53 \times 10^{-2}$	$.13 \times 10^{-2}$	$.33 \times 10^{-3}$	$.82 \times 10^{-4}$	$.15 \times 10^{-3}$
$k = 8$	$.53 \times 10^{-2}$	$.13 \times 10^{-2}$	$.33 \times 10^{-3}$	$.83 \times 10^{-4}$	$.21 \times 10^{-4}$
$k = 11$	$.53 \times 10^{-2}$	$.13 \times 10^{-2}$	$.33 \times 10^{-3}$	$.83 \times 10^{-4}$	$.21 \times 10^{-4}$



(a) Numerical approximation at $N_j = 2^8$, $\epsilon = 10^{-5}$, $N_s = 85$.

(b) Distribution of wavelet coefficients for $\epsilon = 10^{-4}$.

Fig. 2. Test case 1.

become far better for smaller ϵ (very thin boundary layers). Fig. 2(a) shows the solution for test case 1. It is clear from this figure. that mesh points are more concentrated near the boundaries (where the boundary layers occur). Fig. 2(b) shows the wavelet coefficients at different resolutions. As the resolution is increased, non zero wavelet coefficients can be seen only at the location of solution with high gradient.

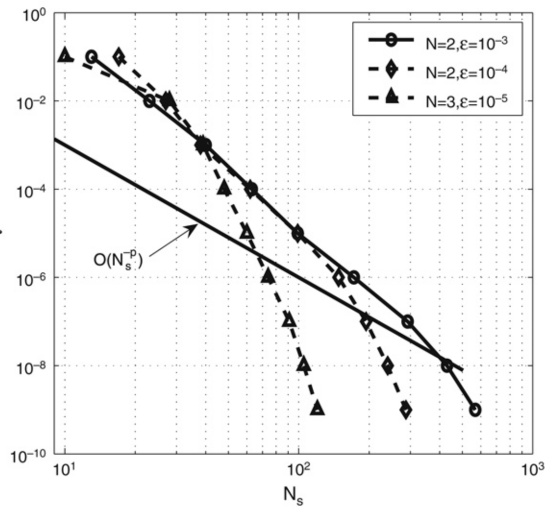
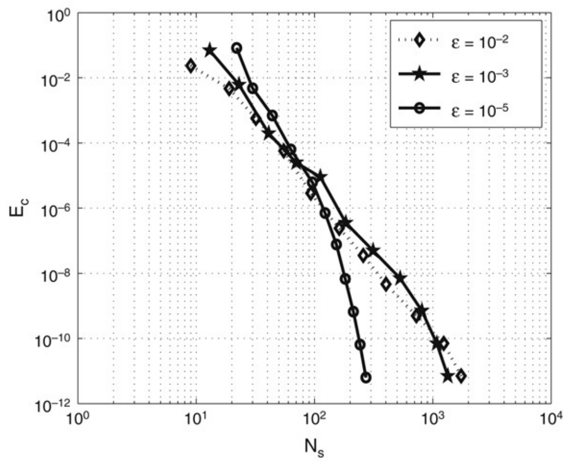
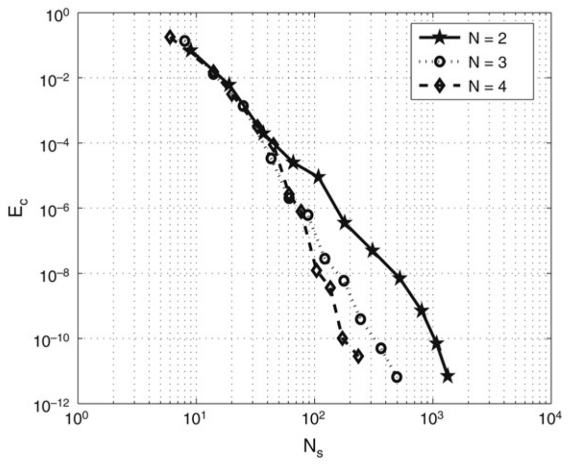


Fig. 3. Relation between τ and N_s for various ϵ and N for test case 1. (-) shows the line with slope $O(N_s^{-p})$ for $\epsilon = 10^{-3}$.



(a) Pointwise error as a function of N_s for various ϵ.



(b) Pointwise error as function of N_s for various N and $\epsilon = 10^{-3}$.

Fig. 4. Test case 1.

We denote $y_\tau^j(x)$ be the approximation of $y(x)$ at jth level for prescribed threshold τ using interpolating wavelet. Let us define maximum pointwise error as

$$E_c = \|y_\tau^j - y\|_\infty, \tag{14}$$

where $\|.\|_\infty$ is the maximum norm. Fig. 3 gives the relation between τ and N_s and as we decrease the tolerance τ, the significant coefficients increase that verifies the theoretical results given in [10]

$$\tau \le c_1 N_s^{-p}, \tag{15}$$

where $p = 2N - 1$ is the degree of interpolating polynomial.

Fig. 4 gives the maximum pointwise error for test case 1 at different N_s for various values of ϵ (perturbation parameter) and N. It verifies that the interpolating wavelet representation satisfies the estimate given by

$$\|y_\tau^j - y\|_\infty \le c_2 N_s^{-p}. \tag{16}$$

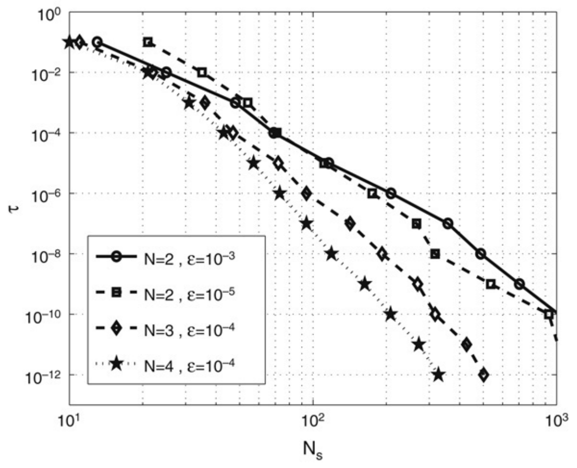
The efficiency of the adaptive algorithm can be measured by the compression coefficient $\mathcal{C} = \frac{N_j}{N_s}$. The higher the compression coefficient the more efficient the adaptive algorithm. In test case 1, the highest compression coefficient ($\mathcal{C} = 3.24$) is achieved at $\epsilon = 10^{-1}, j = 8$.

Test case 2: Now we consider the problem [2] with

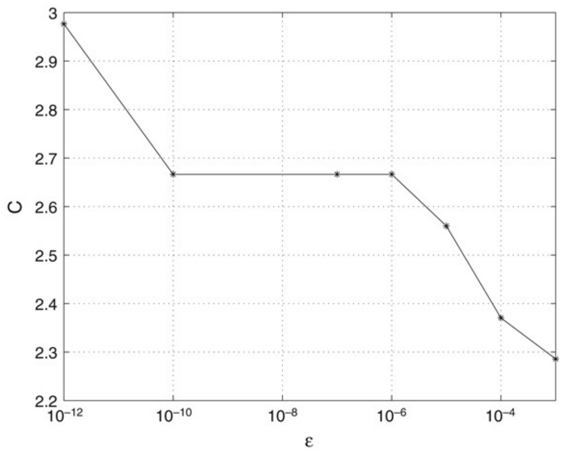
$$a(x) = 1, \qquad b(x) = 1, \qquad g(x) = -\cos^2(\pi x) - 2\epsilon\pi^2\cos(2\pi x),$$

Table 3
Maximum error for test case 2 ($\tau = 10^{-4}$, $N = 2$).

$\epsilon = 10^{-k}$	$N_j = 2^4$	$N_j = 2^5$	$N_j = 2^6$	$N_j = 2^7$	$N_j = 2^8$
$k = 3$	$.55 \times 10^{-1}$	$.27 \times 10^{-1}$	$.10 \times 10^{-1}$	$.23 \times 10^{-2}$	$.27 \times 10^{-2}$
$N_s \rightarrow$	16	32	45	56	62
$k = 4$	$.28 \times 10^{-1}$	$.58 \times 10^{-1}$	$.44 \times 10^{-1}$	$.19 \times 10^{-1}$	$.69 \times 10^{-2}$
	16	32	43	54	66
$k = 5$	$.31 \times 10^{-2}$	$.13 \times 10^{-1}$	$.42 \times 10^{-1}$	$.59 \times 10^{-1}$	$.34 \times 10^{-1}$
	16	32	41	50	61
$k = 6$	$.31 \times 10^{-3}$	$.13 \times 10^{-2}$	$.55 \times 10^{-2}$	$.21 \times 10^{-1}$	$.53 \times 10^{-1}$
	16	32	40	48	57
$k = 7$	$.31 \times 10^{-4}$	$.13 \times 10^{-3}$	$.56 \times 10^{-3}$	$.22 \times 10^{-2}$	$.90 \times 10^{-2}$
	16	32	40	48	56
$k = 10$	$.31 \times 10^{-7}$	$.13 \times 10^{-6}$	$.56 \times 10^{-6}$	$.22 \times 10^{-5}$	$.91 \times 10^{-5}$
	16	32	40	48	56
$k = 12$	$.31 \times 10^{-9}$	$.13 \times 10^{-8}$	$.56 \times 10^{-8}$	$.22 \times 10^{-7}$	$.91 \times 10^{-7}$
	16	32	38	43	48



(a) Relation between τ and N_s for various ϵ and N.



(b) Relation between $\mathcal{C}$ and ϵ.

Fig. 5. Test case 2.

and the exact solution is given by

$$y(x) = \frac{\exp(-1(1-x)/\sqrt{\epsilon}) + \exp(-x/\sqrt{\epsilon})}{1 + \exp(-1/\sqrt{\epsilon})} - \cos^2(\pi x).$$

Table 3 gives the maximum error for test case 2. It further strengthens the claim that the proposed method gives excellent results even for very small ϵ. Fig. 5(a) gives the relation between τ and N_s and Fig. 5(b) gives the relation between $\mathcal{C}$ and ϵ. Here we observe that as ϵ is decreasing (which is the more challenging value of ϵ), the compression coefficient is increasing. Therefore, for smaller ϵ, the wavelet adaptive algorithms are more efficient. Fig. 6(a) and (b) show the pointwise error for test case 2 for various ϵ and N respectively. Test case 2 again satisfies the estimates (15) and (16).

Test case 3: This test case [1] is with

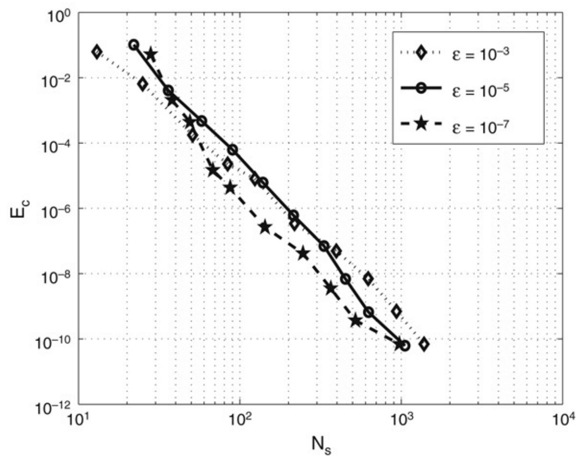
$$a(x) = 1, \qquad b(x) = \frac{4[1 + \sqrt{\epsilon}(x+1)]}{(x+1)^4},$$

$$g(x) = -\frac{4}{(x+1)^4}\left[(1 + \sqrt{\epsilon}(x+1)4\pi^2\epsilon)\cos\left(\frac{4\pi x}{x+1}\right) - 2\pi\epsilon(x+1)\sin\left(\frac{4\pi x}{x+1}\right) + \frac{3(1+\sqrt{\epsilon}(x+1))}{(1-\exp(-1/\epsilon))}\right].$$

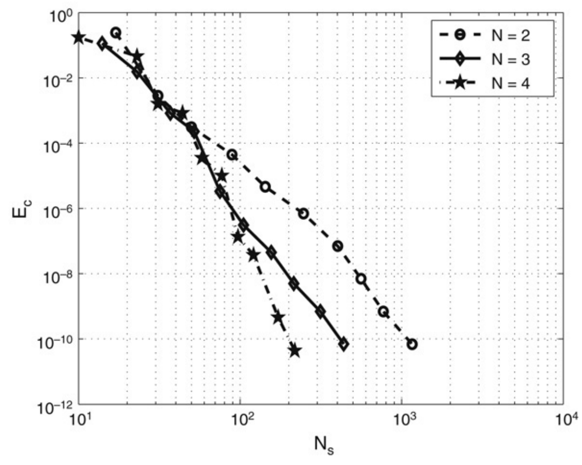
The exact solution is given by

$$y(x) = -\cos\left(\frac{4\pi x}{x+1}\right) + \frac{3[\exp(-2x/\sqrt{\epsilon}(x+1)) - \exp(-1/\sqrt{\epsilon})]}{1 - \exp(-1/\sqrt{\epsilon})}.$$

Here the relation between τ and N_s is plotted in Fig. 7 and pointwise error is shown in Fig. 8.



(a) Pointwise error as a function of N_s for various ϵ.



(b) Pointwise error as a function of N_s for various N and $\epsilon = 10^{-5}$.

Fig. 6. Test case 2.

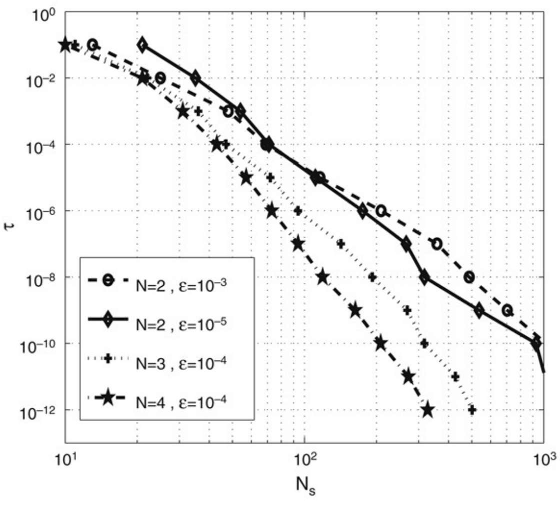
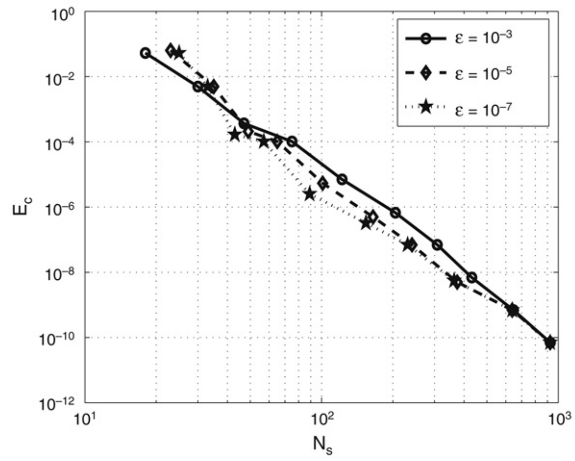
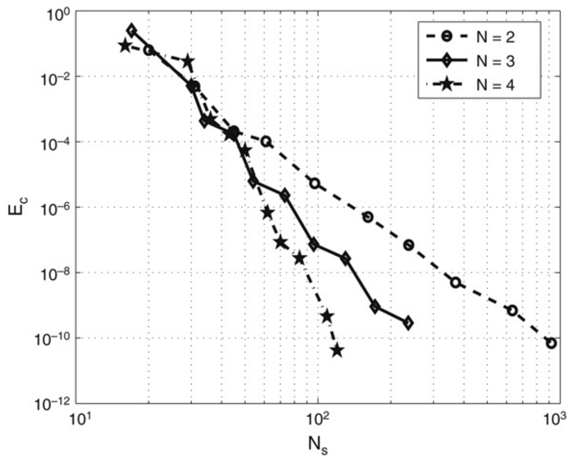


Fig. 7. Relation between τ and N_s for various ϵ and N for test case 3.

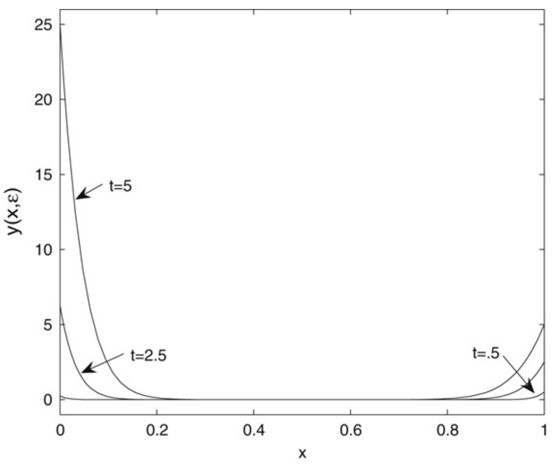


(a) Pointwise error as a function of N_s for various ϵ.

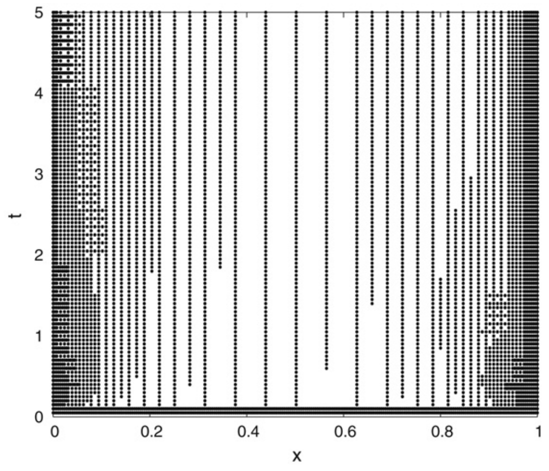


(b) Pointwise error as a function of N_s for various N and $\epsilon = 10^{-4}$.

Fig. 8. Test case 3.

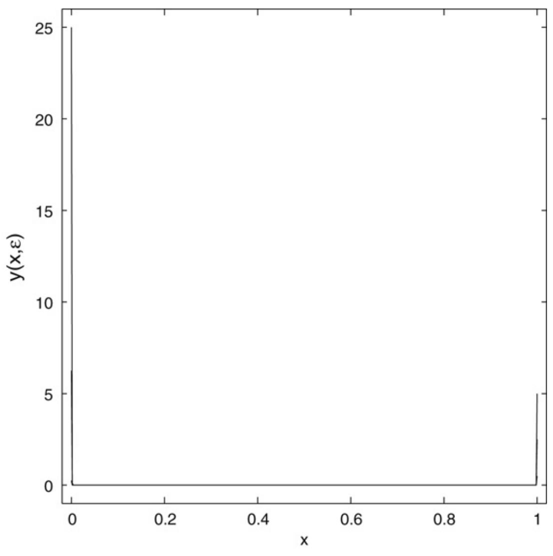


(a) Numerical solution at various times.

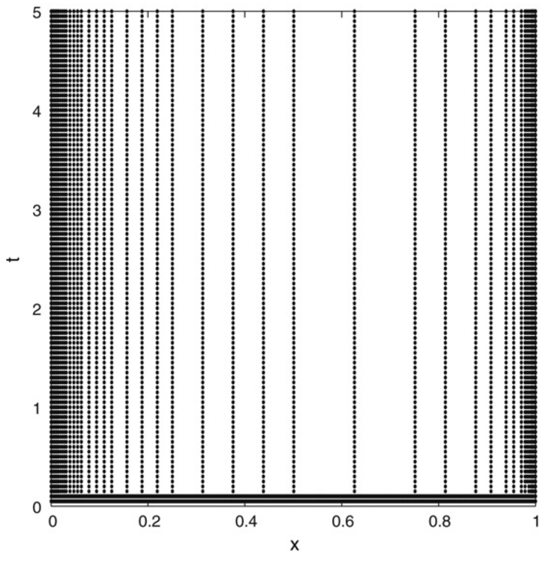


(b) Adaptive mesh as $t \in (0, 5)$.

Fig. 9. Test case 4 for $\epsilon = 10^{-3}$.



(a) Numerical solution at various times.



(b) Adaptive mesh as $t \in (0, 5)$.

Fig. 10. Test case 4 for $\epsilon = 10^{-8}$.

Test case 4: We consider the one-dimensional parabolic reaction-diffusion problem as

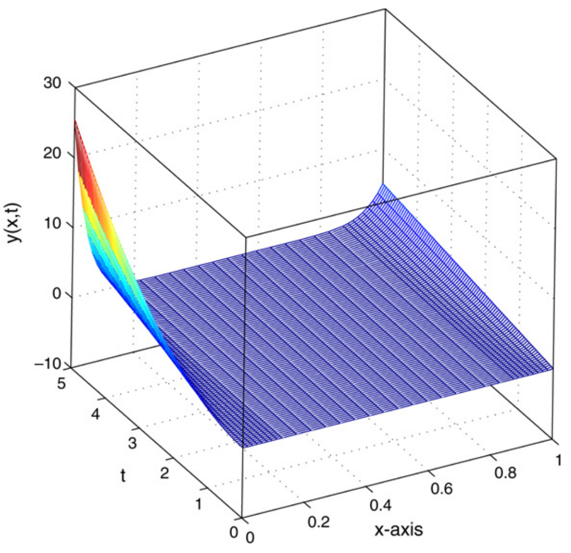
$$\frac{\partial y(x,t)}{\partial t} = \epsilon \frac{\delta^2 y(x,t)}{\delta x^2}; \quad (x,t) \in (0,1) \times (0,T], \qquad (17)$$
$$y(x,0) = 0; \qquad y(0,t) = t^2 \quad \text{and} \quad y(1,t) = t.$$

In this case, the exact solution is not known and the boundary layers width moves with respect to time and becomes very thin as time increases. Fig. 9 gives the solution at different times and the corresponding adaptive mesh.

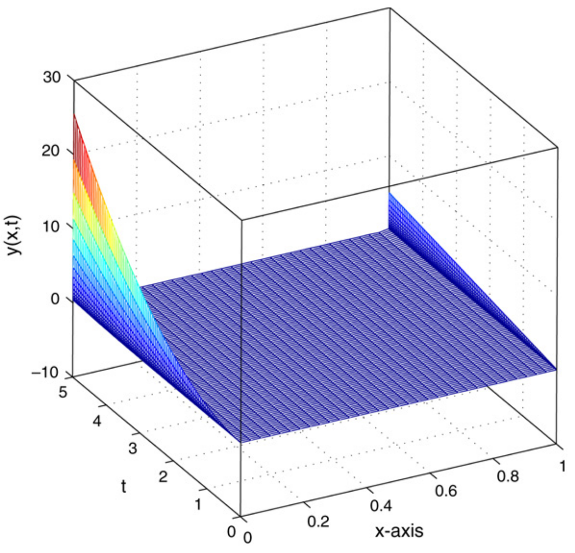
In Fig. 10, we plot the solution and corresponding adaptive grid and observe that for a very small $\epsilon = 10^{-8}$ the boundary layer becomes very thin in a narrow region and the corresponding adaptive grid also follows the narrow region of boundary layer. Fig. 11 gives the numerical solution by WOFD at time interval $(0, 5)$ for different ϵ.

5. Conclusion

An adaptive wavelet optimized finite difference method for solving a self-adjoint singularly perturbed two point boundary value problem has been proposed. Numerical results are presented and compared with the exact solution. The



(a) $\epsilon = 10^{-3}$ for $N_j = 32$.



(b) $\epsilon = 10^{-8}$ for $N_j = 64$.

Fig. 11. Test case 4 for $\Delta t = .05$.

adaptivity is achieved using the interpolating wavelet bases. The novelty of this work lies in the thresholding scheme used to obtain the automatic adaptive feature of wavelet in contrast to the conventional adaptive methods for singularly perturbed problems where we require pre-knowledge of the solution. Finally, the WOFD method seems to be more efficient even for a very small dissipation and is easy to implement.

Acknowledgements

We would like to thank the referees for their useful comments. The first author also acknowledges Prof. A.S. Vasudeva Murthy for his valuable comments in the preparation of this paper.

References

[1] J.M.S. Lubuma, K.C. Patidar, Uniformly convergent non-standard finite difference methods for self-adjoint singular perturbation problems, J. Comput. Appl. Math. 191 (2006) 228–238.
[2] E.P. Doolan, J.J.H. Miller, W.H.A. Schilders, Uniform Numerical Methods for Problems with Initial and Boundary Layers, Boole Press, Dublin, 1980.
[3] M.H. Protter, H.F. Weinberger, Maximum Principles in Differential Equations, Prantice-Hall, Inc., Englewood Cliffs, NJ, 1967.
[4] H.G. Roos, M. Stynes, L. Tobiska, Numerical Methods for Singularly Perturbed Differential Equations, Springer, 1996.
[5] J.J.H. Miller, E. O'Riordan, I.G. Shishkin, Fitted Numerical Methods for Singular Perturbation Problems, World Scientific, 1996.
[6] P.A. Farrell, A.F. Hegarty, J.J.H. Miller, E. O'Riordan, I.G. Shishkin, Robust Computational Techniques for Boundary Layers, Chapman and Hall, CRC, 2000.
[7] M.K. Kadalbajoo, V.K. Aggarwal, Fitted mesh B-spline collocation method for solving self adjoint singularly perturbed boundary value problems, Appl. Math. Comput. 161 (3) (2005) 973–987.
[8] L. Jameson, A wavelet-optimized, very high order adaptive grid and numerical method, SIAM J. Sci. Comput. 19 (1998) 1980–2013.
[9] V. Kumar, M. Mehra, Cubic spline adaptive wavelet scheme to solve singularly perturbed reaction diffusion problems, Int. J. Wavelets Multiresoult. Inf. Process. 5 (2) (2007) 317–331.
[10] D.L. Donoho, Interpolating wavelet transform, Tech. Report 408, Dept. of Stat., Stanford Uni., CA, 1992.
[11] A. Harten, Adaptive multiresolution schemes for shock computations, J. Comput. Phys. 115 (1994) 319–338.
[12] G. Deslauriers, S. Dubuc, Symmetric iterative interpolation process, Constr. Approx. 5 (49) (1989).
[13] N. Saito, G. Beylkin, Multiresolution representation using the autocorrelation functions of compactly supported wavelets, IEEE Trans. Signal Process. 41 (3584) (1993).