Soft Comput (2018) 22:2683–2693
https://doi.org/10.1007/s00500-017-2523-9

METHODOLOGIES AND APPLICATION

B-spline collocation and self-adapting differential evolution (jDE) algorithm for a singularly perturbed convection–diffusion problem

Xu-Qiong Luo[1] · Li-Bin Liu[2] · Aijia Ouyang[3,4] · Guangqing Long[2]

Published online: 23 February 2017

Abstract Many numerical methods applied on a Shishkin mesh are very popular in solving the singularly perturbed problems. However, few approaches are used to obtain the Shishkin mesh transition parameter. Thus, in this paper, we first use the cubic B-spline collocation method on a Shishkin mesh to solve the singularly perturbed convection–diffusion problem with two small parameters. Then, we transform the Shishkin mesh transition parameter selection problem into a nonlinear unconstrained optimization problem which is solved by using the self-adapting differential evolution (jDE) algorithm. To verify the performance of our presented method, a numerical example is employed. It is shown from the experiment results that our approach is efficient. Compared with other evolutionary algorithms, the jDE algorithm performs better and with more stability.

Keywords B-spline collocation method · Self-adapting differential evolution · Singularly perturbed · Optimization problem · Shishkin mesh

Communicated by V. Loia.

✉ Li-Bin Liu
liulibin969@163.com

[1] School of Mathematics and Computing Science, Changsha University of Science and Technology, Changsha 410004, Hunan, China

[2] School of Mathematics and Statistics, Guangxi Teachers Education University, Nanning 530001, China

[3] Department of Information Engineering, Zunyi Normal College, Zunyi 563002, Guizhou, China

[4] Guangxi High School Key Laboratory of Complex System and Computational Intelligence, Nanning 530006, China

1 Introduction

In this paper, we consider the following singularly perturbed convection–diffusion problem with two small parameters

$$\begin{cases} Lu = -\varepsilon u''(x) + \mu b(x)u'(x) + c(x)u(x) = f(x), \\ \qquad\qquad\qquad\qquad\qquad\qquad x \in \Omega = (0,1), \\ u(0) = A, \quad u(1) = B, \end{cases} \tag{1}$$

where $0 < \varepsilon \ll 1$ and $0 < \mu \ll 1$. The functions $b(x)$, $c(x)$ and $f(x)$ are assumed to be sufficiently smooth satisfying

$$0 < b^* \le b(x),\ 0 < c^* \le c(x),\ x \in [0,1],$$

where b^* and c^* are two positive constants. When $\mu = 0$ or $\mu = 1$, this problem encompasses reaction–diffusion problem or convection–diffusion problem, respectively. These kinds of problems arise in transport phenomena in chemistry and biology (Bigge and Bohl 1985). The nature of the two-parameter problem was asymptotically examined by O'Malley (1967), where the ratio of μ to ε has significant role in solution. For this problem, two boundary layers occur at $x = 0$ and $x = 1$. Because of the presence of these layers, some standard numerical methods applied on a uniform mesh fail to give a satisfactory numerical solution. Thus, much attention has been focused on the use of a non-uniform mesh that is adapted to the singularly perturbed problems.

Recently, Gracica et al. (2006) used a second-order monotone numerical scheme which was combined with a piecewise-uniform Shishkin mesh to solve problem (1). Linß (2010) presented a streamline-diffusion finite element method (SDFEM) on a Shishkin mesh. Furthermore, Linß and Roos (2004) developed a first-order upwind difference scheme on a piecewise-uniform Shishkin mesh. Roos and Uzelac (2003) also proposed a SDFEM on a Shishkin mesh to

solve problem (1). Herceg (2011) presented a finite difference scheme for a class of linear singularly perturbed boundary value problems with two small parameters which was discretized on a Bakhvalov-type mesh. Kadalbajoo and Yadaw (2008) solved problem (1) by using the cubic B-spline collocation method on a piecewise-Shishkin mesh.

In a word, it can be seen from the above literature that the upwind finite difference scheme applied on a Shishkin mesh is very popular in solving the singularly perturbed convection–diffusion equation with two small parameters. As far as we know, this mesh contains two grid transition points λ_1 and λ_2 which have some different definitions in some papers. In Miller et al. (1996), the authors defined λ_1 and λ_2 as

$$\lambda_1 = \min\left(\frac{1}{4}, \frac{\sigma_1}{\mu_1}\ln N\right), \quad \lambda_2 = \min\left(\frac{1}{4}, \frac{\sigma_2}{\mu_2}\ln N\right),$$

where σ_1, σ_2 are two positive constants, μ_1 and μ_2 are defined in Miller et al. (1996), and N, our discretization parameter, is a positive even. Then, they divided the intervals $[0, \lambda_1]$ and $[1-\lambda_2, 1]$ into $N/4$ subintervals, respectively, and $[\lambda_1, 1-\lambda_2]$ is dissected into $N/2$. In practical computation, the numerical results of problem (1) are related to the choice of constants σ_1, σ_2. As far as we know, there is no any method which is used to calculate the grid parameters σ_1 and σ_2. Therefore, it is very important to study a clearly method to get the best Shishkin mesh parameters.

In recent years, various improved intelligence algorithms or hybrid intelligence algorithms have been designed to solve optimization problems (Ouyang and Yang 2016), such as PSO with neighborhood operator (Suganthan 1999), distance-based locally informed PSO (Qu et al. 2013), hybrid PSO algorithm (Ouyang et al. 2014), hybird genetic algorithm (Xu et al. 2014), hybrid chemical reaction optimization (Xu et al. 2015), parallel hybrid PSO (Ouyang et al. 2015), heterogeneous CLPSO algorithm (Lynn and Suganthan 2015), multi-population DE algorithm (Wu et al. 2016), hybrid harmony search algorithm (Ouyang et al. 2016a), hybrid cultural algorithm (Ali et al. 2016a, b), hybrid invasive weed optimization algorithm (Ouyang et al. 2016b).

As we know, differential evolution (DE) algorithm (Srorn and Price 1997) is a fast and simple method which performs well on a wide variety of problems. It is a population-based stochastic search technique, which is inherently parallel. DE algorithm is a relatively new nonlinear search and optimization approach, which is particularly well suited to solve some complicate optimization problems. Due to its advantages of simple structure, easy implementation and good computational efficiency, DE algorithm has been successfully used to solve many problems such as mechanical engineering (Abderazek et al. 2015), Signal processing (Liu and Lampinen 2005), pattern recognition (Das and Konar 2009), some problems of parameter estimation (Gong and Cai 1976). Recently, some hybrid DE algorithms (Gong et al. 2011, 2015) were also presented for some global numerical optimization.

In view of the unique advantages of differential evolution algorithm for estimating parameter, the mainly work of this paper is motivated by using jDE algorithm (Brest et al. 2006) to obtain the best mesh transition points. More specifically, we will first use the B-spline collocation technique developed in Kadalbajoo and Yadaw (2008) to study the numerical solution of problem (1). Then, we may use the double-mesh principle (Matthews et al. 2002) to estimate the absolute errors. At last, we transform the choice of mesh parameter problem into a nonlinear unconstrained optimization problem. Furthermore, we utilize the jDE algorithm to find two suitable mesh transition points and the corresponding numerical results for the problem (1).

The remainder of this paper is organized in the following way. Section 2 gives a simple introduction to the mesh selection strategy. Section 3 shows a detailed theoretical analysis of B-spline collocation method. Section 4 introduces A differential evolution algorithm to optimize the Shishkin mesh parameters. Section 5 displays the numerical experimental results and discussions in detail. Finally, the paper concludes with Sect. 6.

2 Mesh selection strategy

At first, we use the piecewise-uniform grid to divide the interval [0,1] into three subintervals:

$$\Omega_0 = [0, \lambda_1], \quad \Omega_c = [\lambda_1, 1-\lambda_2] \text{ and } \Omega_1 = [1-\lambda_2, 1],$$

where the transition parameters are given by

$$\lambda_1 = \min\left(\frac{1}{4}, \delta_1\mu\ln N\right), \quad \lambda_2 = \min\left(\frac{1}{4}, \delta_2\varepsilon\ln N\right),$$

where δ_1 and δ_2 are two positive parameters. Then, we place $N/4$, $N/2$ and $N/4$ mesh points in three subregions $[0, \lambda_1]$, $[\lambda_1, 1-\lambda_2]$ and $[1-\lambda_2, 1]$, respectively. Finally, the mesh widths can be obtained as follows:

$$\tilde{h} = \begin{cases} \frac{4\lambda_1}{N}, & \text{for the interval } [0, \lambda_1], \\ \frac{2(1-\lambda_1-\lambda_2)}{N}, & \text{for the interval } [\lambda_1, 1-\lambda_2], \\ \frac{4(1-\lambda_2)}{N}, & \text{for the interval } [1-\lambda_2, 1]. \end{cases}$$

3 B-spline collocation method

Let $\overline{\Omega}^N = \{x_0, x_1, x_2, \ldots, x_N\}$ be a Shishkin mesh defined in Sect. 2, and then, the cubic B-spline functions (Kadalbajoo and Yadaw 2008) are given as follows:

$$B_i(x) = \frac{1}{\tilde{h}^3}\begin{cases} (x - x_{i-2})^3, & \text{if } x_{i-2} \le x \le x_{i-1}, \\ \tilde{h}^3 + 3\tilde{h}^2(x - x_{i-1}) + 3\tilde{h}(x - x_{i-1})^2 - 3(x - x_{i-1})^3, & \text{if } x_{i-1} \le x \le x_i, \\ \tilde{h}^3 + 3\tilde{h}^2(x_{i+1} - x) + 3\tilde{h}(x_{i+1} - x)^2 - 3(x_{i+1} - x)^3, & \text{if } x_i \le x \le x_{i+1}, \\ (x_{i+2} - x)^3, & \text{if } x_{i+1} \le x \le x_{i+2}, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

where $i = 0, 1, 2, \ldots, N$. For the above functions (2), we introduce four additional knots $x_{-2} < x_{-1} < x_0$ and $x_{N+2} > x_{N+1} > x_N$. Obviously, each of the function $B_i(x)$ is twice continuously differentiable on the entire real line. In addition, for each x_j, $j = 0, 1, \ldots, N$, we have

$$B_i(x_j) = \begin{cases} 4, & \text{if } i = j, \\ 1, & \text{if } i - j = \pm 1, \\ 0, & \text{if } i - j = \pm 2. \end{cases} \tag{3}$$

Similarly, we can show that

$$B_i'(x_j) = \begin{cases} 0, & \text{if } i = j, \\ \pm\dfrac{3}{\tilde{h}}, & \text{if } i - j = \pm 1, \\ 0, & \text{if } i - j = \pm 2 \end{cases} \tag{4}$$

and

$$B_i''(x_j) = \begin{cases} \dfrac{-12}{\tilde{h}^2}, & \text{if } i = j, \\ \dfrac{6}{\tilde{h}^2}, & \text{if } i - j = \pm 1, \\ 0, & \text{if } i - j = \pm 2. \end{cases} \tag{5}$$

As far as we know, the dimensional of cubic B-spline function space is $N + 3$. Similar to (2), we first define two extra cubic B-spline functions B_{-1} and B_{N+1}. Then, the cubic B-spline function space can be given as follows

$$\Phi_3(\overline{\Omega}) = \mathtt{span}\{B_{-1}, B_0, B_1, \ldots, B_{N+1}\}.$$

Thus, for any cubic polynomial function $S(x)$, we have

$$S(x) = \sum_{i=-1}^{N+1} a_i B_i(x), \tag{6}$$

where a_i are unknown real coefficients.

Here, we use function $S(x)$ defined in (6) to approximate the exact solution of (1), yield

$$LS(x_i) = f(x_i), \quad 0 \le i \le N, \tag{7}$$

and

$$S(x_0) = A, \quad S(x_N) = B. \tag{8}$$

By using the values of B-spline functions B_i and of derivatives at mesh points $\overline{\Omega}^N$, we obtain the following system of $N + 1$ linear equations with $N + 3$ unknown variables

$$\begin{aligned}&(-6\varepsilon - 3\mu b_i\tilde{h} + c_i\tilde{h}^2)a_{i-1} + (12\varepsilon + 4c_i\tilde{h}^2)a_i \\ &\quad +(-6\varepsilon + 3\mu b_i\tilde{h} + c_i\tilde{h}^2)a_{i+1} = f_i\tilde{h}^2,\end{aligned} \tag{9}$$

where $0 \le i \le N$.

From the boundary conditions, we have

$$a_{-1} + 4a_0 + a_1 = A, \tag{10}$$

and

$$a_{N-1} + 4a_N + a_{N+1} = B. \tag{11}$$

Next, eliminating a_{-1} from first equation (9) and (10), we obtain

$$\begin{aligned}&(36\varepsilon + 12\mu\tilde{h}b_0)a_0 + 6\mu b_0\tilde{h}a_1 \\ &\quad = \tilde{h}^2 f_0 - A(-6\varepsilon - 3\mu b_0\tilde{h} + c_0\tilde{h}^2).\end{aligned} \tag{12}$$

Similarly, we have

$$\begin{aligned}&(-6\mu\tilde{h}b_N)a_{N-1} + (36\varepsilon - 12\mu b_N\tilde{h})a_N \\ &\quad = \tilde{h}^2 f_N - B(-6\varepsilon + 3\mu b_N\tilde{h} + c_N\tilde{h}^2).\end{aligned} \tag{13}$$

Finally, we can get the following system of $N + 1$ linear equations with $N + 1$ unknown variables

$$Tx_N = d_N, \tag{14}$$

where $x_N = (a_0, a_1, \ldots, a_N)^{\mathrm{T}}$ are the unknown real coefficients with right hand side

$$\begin{aligned}d_N = (&\tilde{h}^2 f_0 - A(-6\varepsilon - 3\mu b_0\tilde{h} + c_0\tilde{h}^2), \tilde{h}^2 f_1, \ldots, \tilde{h}^2 f_{N-1}, \\ &\tilde{h}^2 f_N - B(-6\varepsilon + 3\mu b_N\tilde{h} + c_N\tilde{h}^2))^{\mathrm{T}}\end{aligned}$$

and

$$T = \begin{bmatrix} t_{0,0} & t_{0,1} & 0 & 0 & \cdots & 0 & 0 \\ t_1 & t_2 & t_3 & 0 & \cdots & 0 & 0 \\ 0 & t_1 & t_2 & t_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & t_1 & t_2 & t_3 & 0 \\ 0 & 0 & \cdots & 0 & t_1 & t_2 & t_3 \\ 0 & 0 & \cdots & 0 & 0 & t_{N,N-1} & t_{N,N} \end{bmatrix},$$

where

$$\begin{aligned} t_{0,0} &= 36\varepsilon + 12\mu\tilde{h}b_0, \quad t_{0,1} = 6\mu b_0\tilde{h}, \\ t_1 &= -6\varepsilon - 3\mu b_i\tilde{h} + c_i\tilde{h}^2, \\ t_2 &= 12\varepsilon + 4c_i\tilde{h}^2, \quad t_3 = -6\varepsilon + 3\mu b_i\tilde{h} + c_i\tilde{h}^2, \\ t_{N,N-1} &= -6\mu\tilde{h}b_N, \quad t_{N,N} = 36\varepsilon - 12\mu b_N\tilde{h}. \end{aligned}$$

It is easy to see that the matrix T is strictly diagonally dominant. Thus, we can solve the above system equations (14) for $a_0, a_1, \ldots, a_N$. Furthermore, we obtain a_{-1} and a_{N+1} by substitute $a_0, a_1, \ldots, a_N$ into the boundary condition (10) and (11). Hence the collocation method by using a basis of cubic B-splines functions applied to problem (1) has a unique solution.

4 A differential evolution algorithm to optimize the Shishkin mesh parameters

4.1 The objective function

In general, the exact solution of problem (1) is not available, especially for the nonlinear problem. Thus, in order to estimate the absolute errors of numerical solution of problem (1), we can use the double-mesh principle developed in Matthews et al. (2002) to estimate the absolute errors. Obviously, for each ε, μ and N, the solution of (7) is a binary function about variables δ_1 and δ_2. So, we define $\mathbf{S}^N_{\varepsilon,\mu}(\delta_1, \delta_2)$ be the solution of the approximate scheme on the original Shishkin mesh. Similarly, on the mesh produced by uniformly bisecting the origin mesh, we define $\mathbf{S}^{2N}_{\varepsilon,\mu}(\delta_1, \delta_2)$. Then we can use the following formula to estimate the maximum point-wise error

$$E^N_{\varepsilon,\mu}(\delta_1, \delta_2) = \|\mathbf{S}^N_{\varepsilon,\mu}(\delta_1, \delta_2) - \mathbf{S}^{2N}_{\varepsilon,\mu}(\delta_1, \delta_2)\|_\infty. \tag{15}$$

In practical computation, one may choose suitable parameters δ_1 and δ_2 to make the value of $E^N_{\varepsilon,\mu}(\delta_1, \delta_2)$ as small as possible. Therefore, in this paper, we may transform the problem of mesh parameter calculation into the following nonlinear unconstrained optimization problem

$$\texttt{Fitness} = \min \|\mathbf{S}^N_{\varepsilon,\mu}(\delta_1, \delta_2) - \mathbf{S}^{2N}_{\varepsilon,\mu}(\delta_1, \delta_2)\|_\infty. \tag{16}$$

Obviously, the above objective function (16) is an implicit function above variables δ_1 and δ_2, and is not differentiable. So, some traditional optimization methods are not suitable to solve it. In addition, once the above objective function (16) has many local extreme points, the traditional optimization methods may not find the global optimization solution, efficiently.

4.2 A brief review of differential evolution algorithm

Differential evolution (DE) algorithm presented by Srorn and Price (1997) is an effective and practical intelligent optimization algorithm. It aims at solving an optimization problem by evolving a population of D-dimensional parameter vectors, so-called individuals, which encode the candidate solutions, i.e., $\mathbf{x}_{i,G} = (x_{1i,G}, \ldots, x_{Di,G})$, $i = 1, \ldots,$ NP toward the global optimum. Here, NP be the number of individuals in the population and $\mathbf{x}_{i,G}$ be each target vector at the generation G. First, the initial population should be chosen by uniformly randomizing individuals with the search space constrained by the minimum and maximum bounds $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$. Then, the DE algorithm can be concluded three operations: mutation, crossover and selection.

4.2.1 Mutation

After initialization, for each target vector $\mathbf{x}_{i,G}$, a mutant vector $\mathbf{v}_{i,G} = (v_{1i,G}, v_{2i,G}, \ldots, v_{Di,G})$ is generated by

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r1,G} + F(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}), \ r_1 \neq r_2 \neq r_3 \neq i, \tag{17}$$

where $r_1, r_2, r_3 \in [1, \text{NP}]$ are mutually different random indexes and $F \in [0, 2]$ is a scaling constant that controls the amplification of the difference vector $(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G})$.

4.2.2 Crossover

A trial vector $\mathbf{u}_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1})$ is obtained by the crossover operator, according to the following scheme

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}, & \text{if } r(j) \leq \text{CR or } j = rn(i), \\ x_{ji,G}, & \text{if } r(j) > \text{CR and } j \neq rn(i), \end{cases} \tag{18}$$

where $j = 1, 2, \ldots, D$, $r(j) \in [0, 1]$ is the jth evaluation of uniform random generator number. CR $\in [0, 1]$ is a user-specified constant which controls the fraction of parameter values copied from the mutant vector. $rn(i) \in (1, 2, \ldots, D)$ is a randomly chosen index which ensures that $\mathbf{u}_{i,G+1}$ gets at least one element from $\mathbf{v}_{i,G+1}$.

4.2.3 Selection

A greedy selection scheme is given by

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G+1}, & \text{if } f(\mathbf{u}_{i,G+1}) < f(\mathbf{x}_{i,G}), \\ \mathbf{x}_{i,G}, & \text{otherwise}, \end{cases} \tag{19}$$

where $j = 1, 2, \ldots, D$. In other words, if, and only if, the trial vector $\mathbf{u}_{i,G+1}$ has less or equal objective function value than the corresponding target vector $\mathbf{x}_{i,G+1}$, the trial vector $\mathbf{u}_{i,G+1}$ will replace the target vector $\mathbf{x}_{i,G+1}$ and enter the population of the next generation. Otherwise, the old value $\mathbf{x}_{i,G}$ will remain in the population for the next generation.

DE algorithm can be used to solve multi-objective, non-differentiable problems, and so on. It is very efficiently in a lot of diverse engineering applications such as neural networks (Piotrowski 2014), image processing (Ali et al. 2014), etc.

4.3 Previous work related to DE algorithm

The effectiveness of standard DE algorithm in solving a complicated optimization problem highly depends on the chosen mutation strategy and its associated parameter values. In the past few years, many DE researchers have some techniques for choosing trial vector generation strategies and their associated control parameter settings. According to Sroorn and Price (1997), DE algorithm is very sensitive to the choice of parameters F and CR. The suggested choices are $F \in [0.5, 1]$, $\mathrm{CR} \in [0.8, 1]$ and $\mathrm{NP} = 10D$. Liu and Lampinen (2005) used control parameters set to $F = 0.9$, $\mathrm{CR} = 0.9$. Ali and Törn (2004) chose $\mathrm{CR} = 0.5$ and used the following scheme to calculate F

$$F = \begin{cases} \max\left(l_{\min}, 1 - \left|\dfrac{f_{\max}}{f_{\min}}\right|\right), & \text{if } \left|\dfrac{f_{\max}}{f_{\min}}\right| < 1 \\ \max\left(l_{\min}, 1 - \left|\dfrac{f_{\min}}{f_{\max}}\right|\right), & \text{otherwise}, \end{cases} \tag{20}$$

where f_{max} and f_{min} are the maximum and minimum values of vectors $\mathbf{x}_{i,G}$, respectively. Gämperle et al. (2002) considered different parameter settings for DE on Sphere, Rosenbrock and Rastrigin functions. In their experiment results, the scaling factor F is equal to 0.6, the crossover rate CR be between [0.3, 0.9], and NP be between $[3D, 8D]$. Rönkkönen et al. (2005) suggested using F values in [0.4, 0.95] and CR values in [0, 0.2]. Recently, several researchers (Zaharie 2003; Zaharie and Petcu 2003; Abbass 2002) have developed some approaches to control parameters F and CR. Very recently, more and more researchers paid attention to the self-adaptive DE algorithm, see, e.g., Qin and Suganthan (2005), Omran et al. (2005) and Rahnamayan et al. (2008).

Table 1 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-8}$, $\mu = 10^{-10}$, $N = 32$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	1.3621e−02	4.8258e−03	1.0487e−02	1.4857e−05
δ_1	1.2895e+04	1.3191e+06		
δ_2	2.3611e+04	1.5039e+04		
CLPSO				
$E^N_{\varepsilon,\mu}$	3.2075e−02	4.8949e−02	1.9257e−01	7.1309e−03
δ_1	4.2223e+06	2.6597e+06		
δ_2	4.7928e+05	4.4887e+04		
rcGA				
$E^N_{\varepsilon,\mu}$	4.8680e−02	2.1879e−02	4.7722e−02	2.3836e−05
δ_1	5.3115	5.3178		
δ_2	1.3844e−02	4.7633e−07		
CMA-ES				
$E^N_{\varepsilon,\mu}$	4.8683e−02	4.8258e−03	1.6750e−02	3.7048e−04
δ_1	4.3176	1.3206e+06		
δ_2	4.8757	1.5039e+04		
DE				
$E^N_{\varepsilon,\mu}$	4.7561e−02	4.8258e−03	4.3273e−02	1.6990e−04
δ_1	1.0000e+03	1.4267e+06		
δ_2	1.0000e+03	1.5039e+04		
SaDE				
$E^N_{\varepsilon,\mu}$	4.8626e−02	4.8257e−03	1.9426e−02	4.4101e−04
δ_1	4.3906e+06	1.4424e+06		
δ_2	1.0000e−03	1.5039e+04		
JADE				
$E^N_{\varepsilon,\mu}$	4.6093e−02	4.8257e−03	1.8581e−02	3.9148e−04
δ_1	4.2778e+06	1.4553e+06		
δ_2	1.0000e−05	1.5039e+04		
jDE				
$E^N_{\varepsilon,\mu}$	3.0783e−03	3.0695e−03	3.0709e−03	4.7627e−12
δ_1	1.1465e+06	1.1485e+06		
δ_2	1.0000e−08	1.0000e−08		

4.4 Self-adapting differential evolution (jDE) algorithm

Based on the above literature review, the effectiveness of convectional DE algorithm in solving a numerical optimization problem depends on the selected mutation strategy and its associated parameter values. Therefore, choosing suitable control parameter values for the convection DE algorithm is a very important task. In Brest et al. (2006), by introducing two new control parameters τ_1 and τ_2 to adjust the value of F and CR, Brest et al. proposed a self-adapting differential evolution (jDE) algorithm. The new parameters $F_{i,G+1}$ and $\mathrm{CR}_{i,G+1}$ are calculated by

Table 2 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-8}$, $\mu = 10^{-10}$, $N = 64$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	6.7296e−03	1.6417e−03	4.6211e−03	4.5420e−06
δ_1	1.8477e+04	1.3188e+06		
δ_2	2.8946e+04	1.4531e+04		
CLPSO				
$E^N_{\varepsilon,\mu}$	3.2636e−01	5.9066e−03	1.3741e−01	9.9658e−03
δ_1	6.9633e+06	9.5552e+05		
δ_2	9.6130e+05	2.7412e+04		
rcGA				
$E^N_{\varepsilon,\mu}$	2.4470e−02	2.2950e−02	2.4274e−02	1.7393e−07
δ_1	6.0021	2.6613		
δ_2	6.8153e−03	2.7851e−05		
CMA-ES				
$E^N_{\varepsilon,\mu}$	2.4480e−02	1.6416e−03	9.3277e−03	1.1568e−04
δ_1	2.2054	1.4256e+06		
δ_2	4.7918	1.4530e+04		
DE				
$E^N_{\varepsilon,\mu}$	2.4473e−02	1.6416e−03	6.9691e−03	9.6462e−05
δ_1	5.1873e+06	1.4310e+06		
δ_2	1.0000	1.4530e+04		
SaDE				
$E^N_{\varepsilon,\mu}$	2.4430e−02	1.6416e−03	1.2276e−02	1.3371e−04
δ_1	4.4962e+06	1.4121e+06		
δ_2	1.0000e−03	1.4530e+04		
JADE				
$E^N_{\varepsilon,\mu}$	2.2945e−02	1.6416e−03	1.7264e−02	9.1811e−05
δ_1	4.9369e+06	1.4377e+06		
δ_2	1.0000e−05	1.4530e+04		
jDE				
$E^N_{\varepsilon,\mu}$	9.5269e−04	9.5166e−04	9.5176e−04	4.7049e−14
δ_1	1.0901e+06	1.0894e+06		
δ_2	1.0000e−08	1.0000e−08		

Table 3 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-8}$, $\mu = 10^{-10}$, $N = 128$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	3.3296e−03	5.1592e−04	2.1671e−03	1.3033e−06
δ_1	2.8535e+04	8.9882e+05		
δ_2	3.4955e+04	1.3967e+04		
CLPSO				
$E^N_{\varepsilon,\mu}$	2.7937e−01	1.3902e−04	7.5054e−02	6.1037e−03
δ_1	2.5562e+06	8.4154e+05		
δ_2	7.0927e+05	2.2786e+04		
rcGA				
$E^N_{\varepsilon,\mu}$	1.2281e−02	1.1042e−02	1.2222e−02	5.0346e−08
δ_1	0.1834	4.1875		
δ_2	0.1585	8.8955e−05		
CMA-ES				
$E^N_{\varepsilon,\mu}$	1.2278e−02	5.1577e−04	5.4701e−03	3.0022e−05
δ_1	10.7326	1.3921e+06		
δ_2	7.8300e−02	1.3966e+04		
DE				
$E^N_{\varepsilon,\mu}$	1.2275e−02	5.1579e−04	2.8678e−03	2.2889e−05
δ_1	6.4289e+06	1.3753e+06		
δ_2	1.0000	1.3966e+04		
SaDE				
$E^N_{\varepsilon,\mu}$	1.2238e−02	5.1579e−04	7.1588e−03	3.4910e−05
δ_1	6.2371e+06	1.3459e+06		
δ_2	1.0000e−03	1.3966e+04		
JADE				
$E^N_{\varepsilon,\mu}$	3.3404e−03	2.9601e−04	3.1375e−03	5.9659e−07
δ_1	1.0000e−08	1.0501e+06		
δ_2	1.0000e−08	1.0000e−08		
jDE				
$E^N_{\varepsilon,\mu}$	9.5269e−04	9.5166e−04	9.5176e−04	4.7049e−14
δ_1	1.0901e+06	1.0894e+06		
δ_2	1.0000e−08	1.0000e−08		

$$F_{i,G+1} = \begin{cases} F_l + \text{rand}_1 * F_u, & \text{if } \text{rand}_2 < \tau_1, \\ F_{i,G}, & \text{otherwise}, \end{cases} \tag{21}$$

$$\text{CR}_{i,G+1} = \begin{cases} \text{rand}_3, & \text{if } \text{rand}_4 < \tau_2, \\ \text{CR}_{i,G}, & \text{otherwise}, \end{cases} \tag{22}$$

where $\text{rand}_j \in [0, 1]$, $j = 1, 2, 3, 4$ are uniform random values, $F_l = 0.1$ and $F_u = 0.9$ are the lower and upper bounds of the F, respectively. Obviously, from Equations (21)–(22), the new F takes a value from [0.1, 1.0] in a random manner and the new CR takes a value form [0, 1].

In our paper, in order to solve the above nonlinear optimization problem (16), we will use the technique presented in (21)–(22) to obtain the control parameters F and CR. For the population size NP, we do not change it during the run.

5 Numerical experiments

In this section, the following numerical example is given to illustrate the effectiveness of the presented method

$$-\varepsilon u''(x) + \mu u'(x) + u(x) = \cos \pi x, \quad x \in (0, 1), \tag{23}$$

Table 4 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-10}$, $\mu = 10^{-12}$, $N = 32$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	4.2568e−02	9.7638e−03	2.3332e−02	1.3939e−04
δ_1	73.9763	4.3472e+06		
δ_2	7.6470e−05	1.0419e+05		
CLPSO				
$E^N_{\varepsilon,\mu}$	1.5780e−01	5.9745e−03	3.1958e−02	1.2731e−03
δ_1	9.6733e+06	8.7047e+06		
δ_2	1.0384e+06	1.6548e+05		
rcGA				
$E^N_{\varepsilon,\mu}$	4.8376e−02	3.8362e−02	4.5806e−02	6.6619e−06
δ_1	4.7877	4.2843		
δ_2	1.2620e−02	3.7088e−04		
CMA-ES				
$E^N_{\varepsilon,\mu}$	4.8683e−02	2.6562e−02	4.6874e−02	1.9014e−05
δ_1	4.6996	8.4693		
δ_2	7.3234	1.7047e−04		
DE				
$E^N_{\varepsilon,\mu}$	4.8024e−03	4.8022e−03	4.8023e−03	1.9435e−15
δ_1	1.0000e+07	1.0000e+07		
δ_2	1.4999e+05	1.4990e+05		
SaDE				
$E^N_{\varepsilon,\mu}$	4.8026e−03	4.8022e−03	4.8023e−03	9.2254e−15
δ_1	9.0371e+06	1.4714e+07		
δ_2	1.5000e+05	1.4999e+05		
JADE				
$E^N_{\varepsilon,\mu}$	4.2431e−02	4.8022e−03	1.2328e−02	2.3436e−04
δ_1	4.1239e+07	1.4450e+07		
δ_2	1.0000e−05	1.4999e+05		
jDE				
$E^N_{\varepsilon,\mu}$	3.0618e−03	3.0605e−03	3.0606e−03	7.4143e−14
δ_1	1.1471e+07	1.1468e+07		
δ_2	1.0000e−08	1.0000e−08		

Table 5 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-10}$, $\mu = 10^{-12}$, $N = 64$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	2.0990e−02	6.7257e−03	9.2508e−03	1.7309e−05
δ_1	1.1589e+02	2.2809e+05		
δ_2	6.1899e−04	2.8940e+05		
CLPSO				
$E^N_{\varepsilon,\mu}$	1.0749e−02	3.2125e−03	1.5806e−02	5.5454e−04
δ_1	4.0912e+06	6.8682e+06		
δ_2	1.2466e+06	1.0518e+05		
rcGA				
$E^N_{\varepsilon,\mu}$	2.3951e−02	8.9394e−03	2.1479e−02	9.4205e−06
δ_1	2.9716	4.2900		
δ_2	1.2167e−02	4.0508e−04		
CMA-ES				
$E^N_{\varepsilon,\mu}$	2.4480e−02	1.7836e−02	2.3766e−02	2.4997e−06
δ_1	3.9754	1.5812		
δ_2	2.1553	9.6043e−04		
DE				
$E^N_{\varepsilon,\mu}$	2.4480e−02	1.6208e−03	5.4308e−03	7.5076e−05
δ_1	5.1579e+07	1.3971e+07		
δ_2	10.0000	1.4434e+05		
SaDE				
$E^N_{\varepsilon,\mu}$	2.2951e−02	1.6208e−03	3.7539e−03	4.2362e−05
δ_1	5.0214e+07	1.4348e+07		
δ_2	1.0000e−03	1.4435e+05		
JADE				
$E^N_{\varepsilon,\mu}$	2.4278e−02	1.6208e−03	3.1314e−03	3.3044e−05
δ_1	5.1520e+07	1.4036e+07		
δ_2	1.0000e−05	1.4435e+05		
jDE				
$E^N_{\varepsilon,\mu}$	9.4684e−04	9.4674e−04	9.4676e−04	5.2722e−16
δ_1	1.0862e+07	1.0863e+07		
δ_2	1.0000e−08	1.0000e−08		

$$u(0) = 0, \quad u(1) = 0. \tag{24}$$

For two given parameters δ_1 and δ_2, let S^N and S^{2N} be the numerical solutions which are calculated on N and $2N$ mesh intervals, respectively. Then, the following formula is defined:

$$E^N_{\varepsilon,\mu} = \|\mathrm{S}^N - \mathrm{S}^{2N}\|_\infty.$$

Here, to solve the above problem (23)-(24), we first use jDE algorithm to optimize the problem (16), and obtain the optimal parameters δ_1 and δ_2 and the corresponding numerical solution. To facilitate the experiments, we use the software MATLAB2012a to program a M-file for implementing the algorithms on a PC with a 32-bit windows 7 operating system, a 4GB RAM and a 3.10 GHz-core(TM) i5-based processor.

In the experiment, in order to illustrate the advantages of the jDE algorithm to solve above optimize problem (16), we also calculate the numerical results by using (original) DE, particle swarm optimization (PSO) algorithm (Kennedy and Eberhart 1995), comprehensive learning particle swarm optimization(CLPSO) (Liang et al. 2006), real-coded genetic algorithm (rcGA) (Ono and Kobayashi 1997), covariance

Table 6 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-10}$, $\mu = 10^{-12}$, $N = 128$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	7.4395e−03	3.3383e−03	3.9973e−03	1.5204e−06
δ_1	1.0026e+02	1.5903e+02		
δ_2	2.2606e−03	1.1923e−03		
CLPSO				
$E^N_{\varepsilon,\mu}$	8.7927e−03	6.0044e−04	2.3998e−03	2.9078e−06
δ_1	9.8319e+06	8.9285e+06		
δ_2	5.5601e+05	15063e+05		
rcGA				
$E^N_{\varepsilon,\mu}$	1.2235e−02	3.3384e−03	1.1159e−02	5.0300e−06
δ_1	0.7578	5.6655		
δ_2	2.1813e−02	8.5448e−04		
CMA-ES				
$E^N_{\varepsilon,\mu}$	1.2279e−02	8.2464e−03	1.1907e−02	6.7323e−07
δ_1	4.4150	8.3150		
δ_2	8.3962	2.7153e−03		
DE				
$E^N_{\varepsilon,\mu}$	1.2279e−02	5.0164e−04	4.8202e−03	3.3324e−05
δ_1	6.2969e+07	1.2899e+07		
δ_2	10.0000	1.3771e+05		
SaDE				
$E^N_{\varepsilon,\mu}$	5.0319e−04	5.0165e−04	5.0184e−04	9.6571e−14
δ_1	1.3305e+07	1.2174e+07		
δ_2	13792e+05	1.3771e+05		
JADE				
$E^N_{\varepsilon,\mu}$	4.2431e−02	4.8022e−03	1.2328e−02	2.3436e−04
δ_1	4.1239e+07	1.4450e+07		
δ_2	1.0000e−05	1.4999e+05		
jDE				
$E^N_{\varepsilon,\mu}$	2.9318e−04	2.9313e−04	2.9314e−04	2.9318e−16
δ_1	1.0451e+07	1.0450e+07		
δ_2	1.0000e−08	1.0000e−08		

Table 7 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-12}$, $\mu = 10^{-14}$, $N = 32$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	1.8868e−02	1.3627e−02	1.3935e−02	1.3984e−06
δ_1	99.6925	46.9624		
δ_2	1.2474e−02	1.2107e−04		
CLPSO				
$E^N_{\varepsilon,\mu}$	1.5240e−01	6.0107e−03	2.4290e−02	8.0278e−04
δ_1	1.6092e+07	7.2459e+07		
δ_2	1.0041e+07	1.6032e+06		
rcGA				
$E^N_{\varepsilon,\mu}$	4.1422e−02	1.3627e−02	1.9134e−02	5.7299e−05
δ_1	6.1492e−01	7.7349e−01		
δ_2	5.2965e−02	7.3448e−03		
CMA-ES				
$E^N_{\varepsilon,\mu}$	4.8657e−02	1.3628e−02	4.3933e−02	6.6550e−05
δ_1	8.9376	4.9242		
δ_2	4.9833	9.8798e−03		
DE				
$E^N_{\varepsilon,\mu}$	4.7998e−03	4.7999e−03	4.7998e−03	3.7287e−21
δ_1	1.0000e+08	9.9975e+07		
δ_2	1.4995e+06	1.4995e+06		
SaDE				
$E^N_{\varepsilon,\mu}$	3.0645e−03	3.0596e−03	3.0599e−03	7.7248e−13
δ_1	1.1477e+08	1.1466e+08		
δ_2	1.0000e−06	1.0000e−06		
JADE				
$E^N_{\varepsilon,\mu}$	4.2431e−02	4.8022e−03	1.2328e−02	2.3436e−04
δ_1	4.1239e+07	1.4450e+07		
δ_2	1.0000e−05	1.4999e+05		
jDE				
$E^N_{\varepsilon,\mu}$	1.1535e−03	2.9313e−04	2.9314e−04	2.9318e−16
δ_1	1.0000e+08	1.0450e+07		
δ_2	1.0000e−08	1.0000e−08		

matrix adaptation evolution strategy (CMA-ES) (Hansen et al. 2003), self-adaptive DE (SaDE) algorithm (Qin et al. 2009), JADE (Zhang and Sanderson 2009) and self-adapting differential evolution (jDE) (Brest et al. 2006).

Throughout this paper, the parameter settings of each stochastic algorithm are as follows:

(1) the maximum number of generations $D = 50$, the population size NP = 50.
(2) For the PSO and CLPSO algorithms, two accelerating factors are set to 0.5 and 0.5, respectively. Inertia weight factor is set to 0.45.
(3) For the (original) DE, SaDE and JADE algorithms, crossover factor $F = 0.5$, crossover probability CR = 0.1.

In our experiment, for different values of ε, μ and N, the numerical results of 30 independent runs are summarized in Tables 1, 2, 3, 4,5, 6, 7, 8 and 9. The maximum values, minimum values of $E^N_{\varepsilon,\mu}$ and the corresponding parameters δ_1, δ_2 are also listed in Tables 1, 2, 3, 4,5, 6, 7, 8 and 9. Meanwhile, in order to compare the robustness of the each algorithm, the average computed values of $E^N_{\varepsilon,\mu}$ and variance are also given. It can be seen from Tables 1-9 that the computing precision of

Table 8 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-12}$, $\mu = 10^{-14}$, $N = 64$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	6.7360e−03	6.7361e−03	6.7360e−03	3.9809e−17
δ_1	55.0790	85.2487		
δ_2	3.2810e−02	2.6866e−02		
CLPSO				
$E^N_{\varepsilon,\mu}$	1.5763e−02	1.7155e−03	5.5820e−03	1.1675e−05
δ_1	8.1072e+07	8.5427e+07		
δ_2	4.2066e+06	1.3976e+06		
rcGA				
$E^N_{\varepsilon,\mu}$	4.8679e−02	4.7310e−02	4.8535e−02	1.0092e−07
δ_1	3.4948	4.5891		
δ_2	6.8112e−03	2.8259e−05		
CMA-ES				
$E^N_{\varepsilon,\mu}$	2.4431e−02	6.7360e−03	2.2385e−02	2.0074e−05
δ_1	1.6628	5.2934		
δ_2	4.3707	3.0277e−02		
DE				
$E^N_{\varepsilon,\mu}$	6.7340e−03	3.6146e−03	6.4221e−03	9.0600e−07
δ_1	1.0000e+06	2.1256e+08		
δ_2	1.0000e+06	1.0000e+06		
SaDE				
$E^N_{\varepsilon,\mu}$	9.4708e−04	9.4625e−04	9.4634e−04	3.6731e−14
δ_1	1.0866e+08	1.0860e+08		
δ_2	1.0000e−06	1.0000e−06		
JADE				
$E^N_{\varepsilon,\mu}$	1.1535e−02	3.0596e−03	3.3422e−03	2.3945e−06
δ_1	1.0000e−05	1.1466e+08		
δ_2	2.0265e+06	1.0000e−08		
jDE				
$E^N_{\varepsilon,\mu}$	3.0601e−03	3.0596e−03	3.0597e−03	8.1887e−15
δ_1	1.1466e+08	1.1466e+08		
δ_2	1.0000e−08	1.0000e−08		

Table 9 Numerical results calculated by using different algorithms with $\varepsilon = 10^{-12}$, $\mu = 10^{-14}$, $N = 128$

Method	Maximum error	Minimum error	Mean value	Variance
PSO				
$E^N_{\varepsilon,\mu}$	3.3384e−03	3.3383e−03	3.3383e−03	2.4850e−17
δ_1	56.4370	62.9639		
δ_2	1.8024e−04	11634e−01		
CLPSO				
$E^N_{\varepsilon,\mu}$	2.4479e−02	5.2292e−04	3.2197e−03	2.3353e−05
δ_1	2.6718e+07	9.5955e+07		
δ_2	8.8945e+06	1.4062e+06		
rcGA				
$E^N_{\varepsilon,\mu}$	9.9484e−03	3.3383e−03	4.7536e−03	4.1418e−06
δ_1	3.7910	9.8654e−01		
δ_2	4.7079e−01	9.9984e−02		
CMA-ES				
$E^N_{\varepsilon,\mu}$	1.2271e−02	4.4606e−03	1.1225e−02	3.4866e−06
δ_1	6.6554	1.4656		
δ_2	8.4283	1.3948e−01		
DE				
$E^N_{\varepsilon,\mu}$	3.3370e−03	1.1339e−03	2.8964e−03	8.0333e−07
δ_1	1.0000e+06	2.0614e+08		
δ_2	1.0000e+06	1.0000e+06		
SaDE				
$E^N_{\varepsilon,\mu}$	2.9305e−04	2.9285e−04	2.9288e−04	1.6218e−15
δ_1	1.0442e+08	1.0445e+08		
δ_2	1.0000e−06	1.0000e−06		
JADE				
$E^N_{\varepsilon,\mu}$	3.3035e−04	3.3018e−04	3.3020e−04	1.0801e−15
δ_1	9.9980e+07	9.9999e+07		
δ_2	1.0000e−05	1.0000e−05		
jDE				
$E^N_{\varepsilon,\mu}$	3.3028e−04	3.3017e−04	3.3020e−04	5.9715e−16
δ_1	9.9987e+07	9.9999e+07		
δ_2	1.0000e−05	1.0000e−05		

jDE is slightly higher than the other seven algorithms (PSO, CLPSO, rcGA, CMA-ES, DE, SaDE, JADE) by comparing the maximum, minimum and mean values. In addition, the algorithm stabilization of jDE is slightly stronger than the other seven algorithms (PSO, CLPSO, rcGA, CMA-ES, DE, SaDE and JADE) by comparing the variance values. The experimental results show that the jDE algorithm has certain competition advantages in computing precision and algorithm stabilization than the other seven algorithms (PSO, CLPSO, rcGA, CMA-ES, DE, SaDE and JADE).

The comparison of statistical data shows that the jDE algorithms give better results than rcGA, PSO, CLSPO and CMA-ES algorithm. Furthermore, jDE algorithm performs better than original DE, while original DE algorithm does not always perform better than PSO. Thus, to further illustrate the advantages of jDE algorithm, the convergence speed of the jDE, DE, SaDE, JADE and PSO algorithms are plotted in Figs. 1, 2 and 3 with $\varepsilon = 10^{-8}, \mu = 10^{-10}$, $\varepsilon = 10^{-10}, \mu = 10^{-12}, \varepsilon = 10^{-12}, \mu = 10^{-14}$ and $N = 32$, respectively. Obviously, from these figures, one can easily see that jDE algorithm is certainly better than PSO, DE, SaDE, JADE. It is shown from Figs. 1, 2 and 3 that the jDE algorithm has certain superiority in convergence velocity than the other four algorithms.

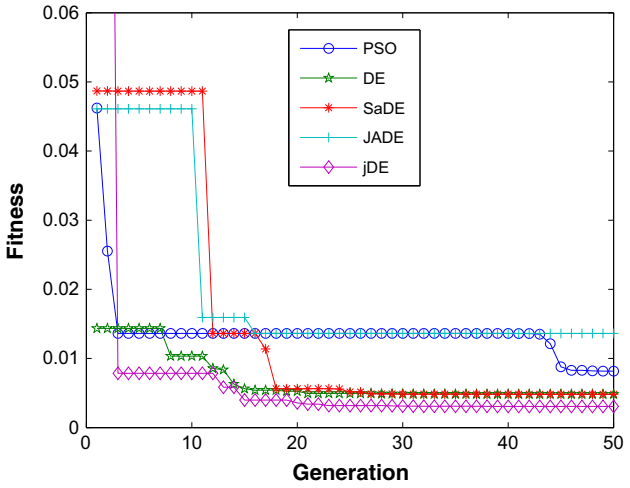

Fig. 1 Fitness of different algorithms for generation with $\varepsilon = 10^{-8}$, $\mu = 10^{-10}$ and $N = 32$

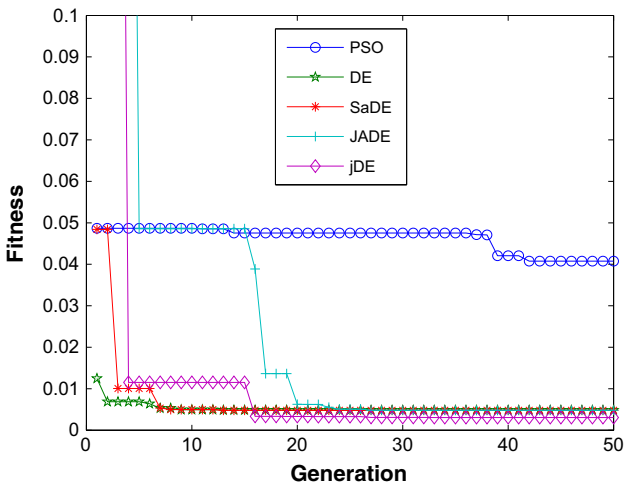

Fig. 2 Fitness of different algorithms for generation with $\varepsilon = 10^{-10}$, $\mu = 10^{-12}$ and $N = 32$

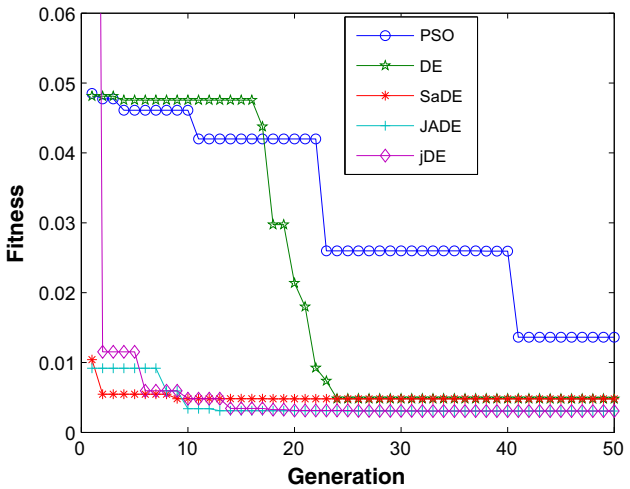

Fig. 3 Fitness of different algorithms for generation with $\varepsilon = 10^{-12}$, $\mu = 10^{-14}$ and $N = 32$

6 Conclusions

In most of the cases, the Shishkin mesh method is frequently used to solve the singularly perturbed problem. However, for the Shishkin mesh transition points, almost all of authors are arbitrary selection parameters. Thus, this work can be viewed as a preliminary step-up in finding a challenging numerical method to obtain the Shishkin mesh transition points. Specially, we transform the Shishkin mesh transition parameter selection problem into a nonlinear unconstrained optimization problem which is solved by using the self-adapting differential evolution (jDE) algorithm. The experimental results show that the jDE algorithm has certain competition advantages in computing precision, algorithm stabilization and convergence speed than the state-of-art algorithms. It is noted that the method presented in this paper can be extend to other type of singularly perturbed problems.

Acknowledgements This work was supported by the National Natural Science Foundation of China (11301044, 11401054, 61662090, 11461011), the general Project of Hunan provincial education department (14C0047), the Natural Science Foundation of Guizhou Provincial Education Department (No. KY[2016]018), the Scientific Research Fund of Hunan Provincial Education Department (No. 13C333), the Doctoral Foundation of Zunyi Normal College (No. BS[2015]13), the open fund of Key Laboratory of Guangxi High Schools for Complex System and Computational Intelligence (No. 15CI03D), Natural Science Foundation of Guangxi Education Department (No. ZD2014080).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Abbass HA (2002) The self-adaptive Pareto differential evolution algorithm. In: Proceedings of the congress evolutionary computation, Honolulu, HI, pp 831–836

Abderazek H, Ferhat D, Atanasovska I (2015) A differential evolution algorithm for tooth profile optimization with respect to balancing specific sliding coefficients of involute cylindrical spur and helical gears. Adv Mech Eng 7(9):1–11

Ali MM, Törn A (2004) Population set-based global optimization algorithms: some modifications and numerical studies. Comput Oper Res 31(10):1703–1725

Ali M, Ahn CW, Pant M (2014) Multi-level image thresholding by synergetic differential evolution. App Soft Comput 17:1–11

Ali MZ, Awad NH, Suganthan PN, Duwairi RM, Reynolds RG (2016a) A novel hybrid Cultural Algorithms framework with trajectory-based search for global numerical optimization. Inf Sci 334–335(C):219–249

Ali MZ, Suganthan PN, Reynolds RG, Al-Badarneh AF (2016b) Leveraged neighborhood restructuring in cultural algorithms for solving real-world numerical optimization problems. IEEE Trans Evol Comput 20(2):218–231

Bigge J, Bohl E (1985) Deformations of the bifurcation diagram due to discretization. Math Comput 45(172):393–403

Brest J, Greiner S, Boskovic B et al (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evolut Comput 10(6):646–657

Das S, Konar A (2009) Automatic imageg pixel clustering with an improved differential evolution. Appl Soft Comput 9(1):226–236

Gämperle R, Müller SD, Koumoutsakos P (2002) A parameter study for differential evolution. In: Grmela A, Mastorakis NE (eds) Advances in intelligent systems, fuzzy systems, evolutionary computation. WSEAS Press, Interlaken, pp 293–298

Gong W, Cai Z (1976) Parameter optimization of PEMFC model with improved multi-strategy adaptive differential evolution. Eng Appl Artif Intel 27(C):28–40

Gong W, Cai Z, Ling CX (2011) DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft Comput 15(4):645–665

Gong W, Zhou A, Cai Z (2015) A multioperator search strategy based on cheap surrogate models for evolutionary optimization. IEEE Trans Evolut Comput 19(5):746–758

Gracica JL, O'Riordan E, Pickett ML (2006) A parameter robust second order numerical method for a singularly perturbed two-parameter problem. Appl Numer Math 56(7):962–980

Hansen N, Muller SD, Koumoutsakos P (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol Comput 11(1):1–18

Herceg D (2011) Fourth-order finite-difference method for boundary value problems with two small parameters. Appl Math Comput 218(2):616–627

Kadalbajoo MK, Yadaw AS (2008) B-spline collocation mehod for a two-parameter singularly pertubed convection-diffusion boundary value problems. Appl Math Comput 201(1–2):504–513

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceeding of the IEEE international conference on neural networks (Perth, Australia). IEEE Service Ceter, Piscataway, NJ, pp 1942–1948

Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evolut Comput 10(3):281–295

Linß T (2010) A posteriori error estimation for a singularly perturbed problem with two small parameters. Int J Numer Anal Model 7(3):491–506

Linß T, Roos HG (2004) Analysis of a finite-difference scheme for a singularly perturbed problem with two small parameters. J Math Anal Appl 289(2):355–366

Liu W, Wang P, Qiao H (2012) Part-based adaptive detection of workpieces using differential evolution. Signal Process 92(2):301–307

Liu J, Lampinen J (2005) A fuzzy adaptive differential evolution algorithm. Soft Comput 9(6):448–462

Lynn N, Suganthan PN (2015) Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation. Swarm Evol Comput 24:11–24

Matthews S, O'Riordan E, Shishkin GI (2002) A numerical method for a system of singularly perturbed reaction–diffusion equations. J Comput Appl Math 145:151–166

Miller JJH, O'Riordan E, Shishkin GI (1996) Fitted numerical methods for singular perturbation problems. Error estimates in the maximum norm for linear problems in one and two dimensions. World Scientific, Singapore

O'Malley RE Jr (1967) Two-parameter singular perturbation problems for second order equations. J Math Mech 16:1143–1164

Omran MGH, Salman A, Engelbrecht AP (2005) Self-adaptive differential evolution. In: Lecture notes in artificial intelligence. Springer, Berlin, pp 192–199

Ono I, Kobayashi S (1997) A real coded genetic algorithm for function optimization using unimodal normal distributed crossover. In: International conference on genetic algorithms, East Lansing, MI, USA, pp 246–253

Ouyang A, Yang Z (2016) An efficient hybrid algorithm based on harmony search and invasive weed optimization. In: 2016 12th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD), Changsha, pp 167–172. doi:10.1109/FSKD.2016.7603169

Ouyang A, Li K, Truong TK, Sallam A, Sha EHM (2014) Hybrid particle swarm optimization for parameter estimation of Muskingum model. Neural Comput Appl 25(7–8):1785–1799

Ouyang A, Tang Z, Zhou X, Xu Y, Pan G, Li K (2015) Parallel hybrid PSO with CUDA for lD heat conduction equation. Comput Fluids 110:198–210

Ouyang A, Peng X, Liu Y, Fan L, Li K (2016a) An efficient hybrid algorithm based on HS and SFLA. Int J Pattern Recognit Artif Intell 30(5):1659012 (1–25)

Ouyang A, Peng X, Wang Q, Wang Y, Truong TK (2016b) A parallel improved iwo algorithm on gpu for solving large scale global optimization problems. J Intell Fuzzy Syst 31(2):1041–1051

Piotrowski AP (2014) Diiferential Evolution algorithms applied to neural network training suffer from stagnation. Appl Soft Comput 21:382–406

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE congress evolutionary computation, Edinburgh, Scotland, pp 1785–1791

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 13(2):398–417

Qu BY, Suganthan PN, Das S (2013) A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans Evol Comput 17(3):387–402

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Oppositionbased differential evolution. IEEE Trans Evolut Comput 12(1):64–79

Rönkkönen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: Proceedings of the IEEE congress evolutionary computation, Edinburgh, Scotland, pp 506–513

Roos HG, Uzelac Z (2003) The SDFEM for a convection–diffusion problem with two small parameters. Comput Methods Appl Math 3(3):443–458

Srorn R, Price K (1997) Differential evolution: a simple and efficient adaptive scheme for global optimization over continuous spaces. J Global Optim 11(4):341–359

Suganthan PN (1999) Particle swarm optimiser with neighbourhood operator. In: Proceedings of the 1999 congress on evolutionary computation, 1999 (CEC 99), Washington, DC

Wu G, Mallipeddi R, Suganthan PN, Wang R, Chen H (2016) Differential evolution with multi-population based ensemble of mutation strategies. Inf Sci 329:329–345

Xu Y, Li K, Hu J, Li K (2014) A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Inf Sci 270:255–287

Xu Y, Li K, He L, Zhang L (2015) A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems. IEEE Trans Parallel Distrib Syst 26(12):3208–3222

Zaharie D (2003) Control of population diversity and adaptation in differential evolution algorithms. In: Matousek R, Osmera P (eds) Proceeding of the mendel 9th international conference soft computing, Brno, Czech Republic, pp 41–46

Zaharie D, Petcu D (2003) Adaptive pareto differential evolution and its parallelization. In: Proceedings of the 5th international conference on parallel process applied mathematics, Czestochowa, Poland, pp 261–268

Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 13(5):945–958