Memetic Comp. (2012) 4:303–316
DOI 10.1007/s12293-012-0096-9

REGULAR RESEARCH PAPER

Fitness based Differential Evolution

Harish Sharma · Jagdish Chand Bansal · K. V. Arya

Received: 29 February 2012 / Accepted: 19 October 2012 / Published online: 6 November 2012

Abstract Differential Evolution (DE) is a well known and simple population based probabilistic approach for global optimization. It has reportedly outperformed a few Evolutionary Algorithms and other search heuristics like Particle Swarm Optimization when tested over both benchmark and real world problems. But, DE, like other probabilistic optimization algorithms, sometimes exhibits premature convergence and stagnates at suboptimal point. In order to avoid stagnation behavior while maintaining a good convergence speed, a new position update process is introduced, named fitness based position update process in DE. In the proposed strategy, position of the solutions are updated in two phases. In the first phase all the solutions update their positions using the basic DE and in the second phase, all the solutions update their positions based on their fitness. In this way, a better solution participates more times in the position update process. The position update equation is inspired from the Artificial Bee Colony algorithm. The proposed strategy is named as Fitness Based Differential Evolution ($FBDE$). To prove efficiency and efficacy of $FBDE$, it is tested over 22 benchmark optimization problems. A comparative analysis has also been carried out among proposed FBDE, basic DE, Simulated Annealing Differential Evolution and Scale Factor Local Search Differential Evolution. Further, $FBDE$ is also applied to solve a well known electrical engineering problem called Model Order Reduction problem for Single Input Single Output Systems.

H. Sharma · J. C. Bansal (✉) · K. V. Arya
ABV-Indian Institute of Information Technology and Management, Gwalior, India
e-mail: jcbansal@gmail.com

H. Sharma
e-mail: harish.sharma0107@gmail.com

K. V. Arya
e-mail: kvarya@gmail.com

Keywords Evolutionary optimization · Differential Evolution · Fitness based position update · Model order reduction

1 Introduction

Differential Evolution (DE) scheme is relatively a simple, fast and population based stochastic search technique, proposed by Storn and Price [32]. DE falls under the category of Evolutionary Algorithms (EAs). But in some sense it differs significantly from EAs, e.g. trial vector generation process (explained in Sect. 2) uses the information of distance and direction from current population to generate a new trial vector. Furthermore, in EAs, crossover is applied first to generate a trial vector, which is then used within the mutation operation to produce one offspring while, in DE, Mutation is applied first and then crossover [10].

Researchers are continuously working to improve the performance of DE. Some of the recently developed versions of DE with appropriate applications can be found in [3]. In the literature [40], it has been shown that for many times DE performs better than the Genetic Algorithm (GA) [14] or the Particle Swarm Optimization (PSO) [17]. DE has successfully been applied to various areas of science and technology, such as chemical engineering [20], signal processing [7], mechanical engineering design [35], machine intelligence, and pattern recognition [28]. Recently, machine intelligence and cybernetics are most well-liked field in which DE algorithm has become a popular strategy.

There are two fundamental processes which drive the evolution of a DE population: the variation process, which enables exploring different areas of the search space, and the selection process, which ensures the exploitation of the previous experience. However, it has been shown that DE may

occasionally stop proceeding towards the global optimum even though the population has not converged to a local optimum [19]. Therefore, to maintain the proper balance between exploration and exploitation behavior of DE, a new position update process is introduced based on the fitness of the solution. The position update takes place in two phases in the proposed strategy. In the first phase, the basic DE is used to generate the new solutions and in the second phase each solution is updated based on its fitness. The proposed update process is inspired from onlooker bee phase of Artificial Bee Colony algorithm (ABC) [16]. In this process, better candidate gets more chance to update its position. Further, the solution updates only in single dimension in each chance, hence generates the new solution in the neighborhood of the old one and in this way exploits the search space.

Rest of the paper is organized as follows: Section 2 describes brief overview of the basic Differential Evolution algorithm. Fitness Based Differential Evolution algorithm ($FBDE$) is proposed and established in Sect. 3. In Sect. 4 experiments are carried and a comparative study among the proposed strategy, the basic DE and its variants is done. In Sect. 5, a real-world optimization problem, model order reduction problem for single input single output ($SISO$) systems is solved using $FBDE$. Finally, in Sect. 6, paper is concluded.

2 Brief overview of Differential Evolution algorithm

DE has several strategies based on method of selecting the target vector, number of difference vectors used and the type of crossover [32]. In this paper $DE/rand/1/bin$ scheme is used where DE stands for differential evolution, 'rand' specifies that the target vector is selected randomly, '1' is for number of differential vectors and 'bin' notation is for *binomial* crossover. The popularity of Differential Evolution is due to its applicability to a wider class of problems and ease of implementation. Differential Evolution consists of the properties of both evolutionary algorithms and swarm intelligence. The detailed description of DE is as follows:

Like other population based search algorithms, in DE a population of potential solutions (individuals) searches the solution. In a D-dimensional search space, an individual is represented by a D-dimensional vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_D})$, $i = 1, 2, \ldots, NP$ where NP is the population size (number of individuals).

In DE, there are three operators: mutation, crossover and selection. Initially, a population is generated randomly with uniform distribution then the mutation, crossover and selection operators are applied to generate a new population. Trial vector generation is a crucial step in DE process. The two operators mutation and crossover are used to generate the trial vectors. The selection operator is used to select the best trial vector for the next generation. DE operators are explained briefly in following subsections.

2.1 Mutation

A trial vector is generated by the DE mutation operator for each individual of the current population. For generating the trial vector, a target vector is mutated with a weighted differential. An offspring is produced in the crossover operation using the newly generated trial vector. If G is the index for generation counter, the mutation operator for generating a trial vector $u_i(G)$ from the parent vector $x_i(G)$ is defined as follows:

- Select a target vector $x_{i_1}(G)$ from the population such that $i \neq i_1$.
- Again, randomly select two individuals, x_{i_2} and x_{i_3}, from the population such that i, i_1, i_2 and i_3 all are distinct to each other.
- Then the target vector is mutated for calculating the trial vector as follows:

$$u_i(G) = x_{i_1}(G) + \underbrace{F \times \overbrace{(x_{i_2}(G) - x_{i_3}(G))}^{\text{Variation Component}}}_{\text{Step size}} \tag{1}$$

where $F \in [0, 1]$ is the mutation scale factor which is used in controlling the amplification of the differential variation [10].

2.2 Crossover

Offspring $x'_i(G)$ is generated using the crossover of parent vector $x_i(G)$ and the trial vector $u_i(G)$ as follows:

$$x'_{ij}(G) = \begin{cases} u_{ij}(G), & \text{if } j \in J \\ x_{ij}(G), & \text{otherwise} \end{cases} \tag{2}$$

where J is the set of cross over points or the points that will go under perturbation, $x_{ij}(G)$ is the jth element of the vector $x_i(G)$.

Different methods may be used to determine the set J of in which binomial crossover and exponential crossover are the most frequently used [10]. In this paper, the DE and its variants are implemented using the binomial crossover. In this crossover, the crossover points are randomly selected from the set of possible crossover points, $\{1, 2, \ldots, D\}$, where D is the problem dimension. Algorithm 1 shows the steps of binomial crossover to generate crossover points [10]. In this algorithm, CR is the probability that the considered crossover point will be included. The larger the value of CR, the more crossover points will be selected. Here, J is a set of crossover points, CR is crossover probability, $U(1, D)$ is a uniformly distributed random integer between 1 and D.

Algorithm 1 Binomial Crossover:

$J = \phi$
$j^* \sim U(1, D)$;
$J \leftarrow J \cup j^*$;
for each $j \in 1...D$ **do**
  if $U(0,1) < CR$ and $j \neq j^*$ **then**
    $J \leftarrow J \cup j$;
  end if
end for

2.3 Selection

There are two functions of the selection operator: First it selects the individual for the mutation operation to generate the trial vector and second, it selects the best, between the parent and the offspring based on their fitness value for the next generation. If fitness of parent is greater than the offspring then parent is selected otherwise offspring is selected:

$$x_i(G+1) = \begin{cases} x'_i(G), & \text{if } f(x'_i(G)) > f(x_i(G)). \\ x_i(G), & \text{otherwise.} \end{cases} \quad (3)$$

This ensures that the population's average fitness does not deteriorate.

The Pseudo-code for Differential Evolutionary strategy, is described in Algorithm 2 [10].

Here, F (scale factor) and CR (crossover probability) are the control parameters and influence the performance of the DE. P is the population vector.

Algorithm 2 Differential Evolutionary Algorithm:

Initialize the control parameters F and CR;
Create and initialize the population $P(0)$ of NP individuals;
while stopping condition(s) not true **do**
  for each individual $x_i(G) \in P(G)$ **do**
    Evaluate the fitness, $f(x_i(G))$;
    Create the trial vector $u_i(G)$ by applying the mutation operator;
    Create an offspring $x'_i(G)$ by applying the crossover operator;
    if $f(x'_i(G))$ is better than $f(x_i(G))$ **then**
      Add $x'_i(G)$ to $P(G+1)$;
    else
      Add $x_i(G)$ to $P(G+1)$;
    end if
  end for
end while
Return the individual with the best fitness as the solution;

3 Fitness based Differential Evolution

3.1 A few drawbacks of DE

The inherent drawback with most of the population based stochastic algorithms is premature convergence. DE is not an exception. Any population based algorithm is regarded as an efficient algorithm if its performance and efficiency (ability to give solution faster) over a large set of problems is better. In other words, if a population based algorithm is capable of balancing between exploration and exploitation of the search space, then it is expected that the algorithm perform better and regarded as an efficient algorithm. From this point of view, basic DE is not an efficient algorithm [24]. Also some studies proved that stagnation is another inherent drawback with DE, i.e. DE sometimes stop proceeding towards the global optima even though the population has not converged to local optima or any other point [19]. Mezura-Montes et al. [24] compared the different variants of DE for global optimization and found that DE shows poor performance and remains inefficient in exploring the search space, especially for multimodal functions. Price et al. [33] also drawn the same conclusions. The problems of premature convergence and stagnation is a matter of serious consideration for designing a comparatively efficient DE algorithm (as it is not possible to design a fully efficient population based stochastic algorithm).

3.2 The proposed strategy

Exploration of the whole search space and exploitation of the near optimal solution region may be balanced by maintaining the diversity in early and later iterations of any random number based search algorithm. It is clear from the Eqs. (1) and (2) that DE explores the search space based on the value of CR and F. In DE, exploration and exploitation of the search space depend on the value of CR and F i.e. for high value of CR and F exploration will be high and for low value, exploitation. In this paper, we are proposing a new position update process which balances the exploration and exploitation of the search space. The position update process is inspired from the Artificial Bee Colony (ABC) algorithm's onlooker bee phase [16]. In employed bee phase of ABC, all the employed bees search the food source and calculate their fitness using Eq. (4):

$$fitness_i = \begin{cases} 1/(1+f_i), & \text{if } f_i \geq 0 \\ 1 + abs(f_i), & \text{if } f_i < 0 \end{cases} \quad (4)$$

and then in the onlooker bee phase, Onlooker bees analyze the available information and select a solution with a probability, $prob_i$, related to its fitness. The probability $prob_i$ may be calculated using Eq. (5) (there may be some other but must be a function of fitness):

$$prob_i(G) = \frac{0.9 \times fitness_i(G)}{maxfit(G)} + 0.1, \quad (5)$$

where G is the iteration counter, $fitness_i(G)$ is the fitness value of ith solution and $maxfit(G)$ is the maximum fitness of the solutions in Gth iteration. Position update equation of ABC is shown in Eq. (6):

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (6)$$

Algorithm 3 Proposed Position Update Process:

```
for each individual, x_i do
  if prob_i > rand(0,1) then
    v_ij = x_ij + φ_ij(x_ij − x_kj),
    Calculate fitness of v_i,
    Apply greedy selection between v_i and x_i using equation (3),
  end if
end for
```

where $k \in \{1, 2, \ldots, NP\}$ and $j \in \{1, 2, \ldots, D\}$ are randomly chosen indices, k must be different from i, ϕ_{ij} is a random number between $[-1, 1]$ and x_{kj} is a random individual in the current population. In the basic ABC, at any given time, only one dimension is updated in employed or onlooker bee phase. In onlooker bee phase this update takes place based on a probability which is a function of fitness. The proposed strategy $FBDE$, in this paper, is inspired from ABC's onlooker bee phase discussed above. In $FBDE$, Algorithm 3 is applied after basic DE operators. The insertion of Algorithm 3 makes $FBDE$ more capable of exploitation in the better search regions. It is expected because in $FBDE$ after applying basic DE operators, better candidate solutions are offered to update themselves more times than worse candidates. The pseudo-code of the proposed position update process which works after DE operators is shown in Algorithm 3. The Pseudo-code for the proposed $FBDE$ algorithm is shown in Algorithm 4.

4 Experimental results and discussion

4.1 Test problems under consideration

In order to see the effect of the new position update process on DE, 22 different global optimization problems (f_1–f_{22}) are selected (listed in Table 1). These problems are minimization problems and have different degrees of complexity and multimodality. Test problems f_1–f_{12} and f_{19}–f_{20} are taken from [2] and test problems f_{13}–f_{18} are taken from [39] with the associated offset values.

4.2 Experimental setting

To prove the efficiency of $FBDE$ algorithm, it is compared with three variants of DE, namely, $DE/rand/bin/1$ (usually known as basic DE) [32], Simulated Annealing Differential Evolution ($SADE$) [42] and Scale Factor Local Search Differential Evolution ($SFLSDE$) [27]. In SADE algorithm, simulated annealing (SA) updating strategy is incorporated with the basic DE which helps to escape from the local optima, and achieve the balance between exploration and exploitation. In SADE, the greedy updating method is replaced by the SA updating method. Each individual contains a set of F values instead of single value within the

Algorithm 4 Fitness Based Differential Evolution($FBDE$):

```
Initialize the control parameters, F and CR;
Initialize generation counter G = 0;
Create and initialize the population P(0) of NP individuals;
while stopping condition(s) not true do
  for each individual x_i(G) ∈ P(G) do
    Evaluate the fitness f(x_i(G));
    Create the trial vector u_i(G) by applying the scale factor (mutation operator) F using
    equation (1);
    Create an offspring x'_i(G) by applying the crossover operator using equation (2);
    Calculate the fitness of the newly generated offspring using equation (4);
    if f(x'_i(G)) is better than f(x_i(G)) then
      Add x'_i(G) to P(G + 1);
    else
      Add x_i(G) to P(G + 1);
    end if
  end for
  t = 1, i = 1
  while t ≤ NP do
    if prob_i(G) > rand(0,1) then
      { prob_i is the probability of an individual x_i described by equation (5)}
      v_ij = x_ij + φ_ij(x_ij − x_kj),
      {j is randomly selected index}
      Calculate fitness of v_i;
      Apply greedy selection between v_i and x_i using equation (3);
      t = t + 1
    end if
    i = i + 1
    if i > NP then
      i = 1
    end if
  end while
end while
Return the individual with the best fitness as the solution;
```

Table 1 Test problems

Test problem	Objective function	Search range	Optimum value	D	Acceptable error
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-5.12, 5.12]$	$f(\mathbf{0}) = 0$	30	1.0E−05
Griewank	$f_2(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})$	$[-600, 600]$	$f(\mathbf{0}) = 0$	30	1.0E−05
Rosenbrock	$f_3(x) = \sum_{i=1}^{D}(100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2)$	$[-30, 30]$	$f(\mathbf{1}) = 0$	30	1.0E−02
Rastrigin	$f_4(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)]$	$[-5.12, 5.12]$	$f(\mathbf{0}) = 0$	30	1.0E−05
Ackley	$f_5(x) = -20 + e + exp(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D}{x_i}^3}) - exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)x_i)$	$[-1, 1]$	$f(\mathbf{0}) = 0$	30	1.0E−05
Michalewicz	$f_6(x) = -\sum_{i=1}^{D}\sin x_i(\sin(\frac{i{x_i}^2}{\pi})^{20})$	$[0, \pi]$	$f_{min} = -9.66015$	10	1.0E−05
Cosine Mixture	$f_7(x) = \sum_{i=1}^{D}{x_i}^2 - 0.1(\sum_{i=1}^{D}\cos 5\pi x_i) + 0.1D$	$[-1, 1]$	$f(\mathbf{0}) = -D \times 0.1$	30	1.0E−05
Step function	$f_8(x) = \sum_{i=1}^{D}(\lfloor x_i + 0.5\rfloor)^2$	$[-100, 100]$	$f(-0.5 \le x \le 0.5) = 0$	30	1.0E−05
Quartic function	$f_9(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]$	$f(\mathbf{0}) = 0$	30	1.0E−05
Inverted cosine wave	$f_{10}(x) = -\sum_{i=1}^{D-1}\left(\exp\left(\frac{-(x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1})}{8}\right) \times \mathrm{I}\right)$ where, $\mathrm{I} = cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0.5x_i x_{i+1}}\right)$	$[-5, 5]$	$f(\mathbf{0}) = -D + 1$	10	1.0E−05
Kowalik function	$f_{11}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]$	$f(0.192833, 0.190836, 0.123117, 0.135766) = 0.000307486$	4	1.0E−05
2D Tripod function	$f_{12}(x) = p(x_2)(1 + p(x_1)) + \lvert(x_1 + 50p(x_2)(1 - 2p(x_1)))\rvert + \lvert(x_2 + 50(1 - 2p(x_2)))\rvert$	$[-100, 100]$	$f(0, -50) = 0$	2	1.0E−04
Shifted Rosenbrock	$f_{13}(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias}, z = x - o + 1,$ $x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = 390$	10	1.0E−01
Shifted Sphere	$f_{14}(x) = \sum_{i=1}^{D} z_i^2 + f_{bias}, z = x - o, x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0E−05
Shifted Rastrigin	$f_{15}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias},$ $z = (\mathrm{x} - \mathrm{o}), x = (x_1, x_2, \ldots, x_D), \mathrm{o} = (o_1, o_2, \ldots, o_D)$	$[-5, 5]$	$f(o) = f_{bias} = -330$	10	1.0E−02
Shifted Schwefel	$f_{16}(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} z_j)^2 + f_{bias}, z = x - o, x = [x_1, x_2, \ldots, x_D],$ $o = [o_1, o_2, \ldots, o_D]$	$[-100, 100]$	$f(o) = f_{bias} = -450$	10	1.0E−05
Shifted Griewank	$f_{17}(x) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{bias}, z = (x - o),$ $x = [x_1, x_2, \ldots, x_D], o = [o_1, o_2, \ldots, o_D]$	$[-600, 600]$	$f(o) = f_{bias} = -180$	10	1.0E−05
Shifted Ackley	$f_{18}(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20$ $+ e + f_{bias}, z = (x - o), x = (x_1, x_2, \ldots, x_D), o = (o_1, o_2, \ldots, o_D)$	$[-32, 32]$	$f(o) = f_{bias} = -140$	10	1.0E−05
Goldstein-Price	$f_{19}(x) = (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))(30 +$ $(2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-2, 2]$	$f(0, -1) = 3$	2	1.0E−14
Six-hump camel back	$f_{20}(x) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$	$[-5, 5]$	$f(-0.0898, 0.7126) = -1.0316$	2	1.0E−05
Sinusoidal problem	$f_{21}(x) = -[A\prod_{i=1}^{D}\sin(x_i - z) + \prod_{i=1}^{D} sin(B(x_i - z))], A = 2.5, B = 5, z = 30$	$[0, 180]$	$f(90 + z) = -(A + 1)$	10	1.0E−02
Moved axis parallel hyper-ellipsoid	$f_{22}(x) = \sum_{i=1}^{D} 5i \times x_i^2$	$[-5.12, 5.12]$	$f(x) = 0; x(i) = 5 * i,$ $i = 1 : D$	30	1.0E−15

range [0.1, 1]. The value of CR are changed by a probability or remains unchanged. CR is assigned to each individual but in an identical fashion. $SFLSDE$ is a self-adaptive scheme with the two local search algorithms: Scale factor hill-climb and Scale factor golden section search. These local search algorithms are used for detecting the value of scale factor F corresponding to an offspring with a better performance. Therefore, the local search algorithms support in the global search (exploration process) and in generating offspring with high performance.

The comparative analysis has been carried out through reliability [due to success rate (SR)], efficiency [due to average number of function evaluations (AFE)] and accuracy [due to mean error (ME)]. After calculating SR, AFE and ME, statistical analyses based on t test, Acceleration Rate (AR) [34], Boxplot and Performance Index [9] have been carried out. In order to show the superiority of proposed algorithm from different point of view, these intensive statistical analysis have been carried out.

To test DE or DE variants over test problems, following experimental setting is adopted:

- Parameters for the basic DE are $CR = 0.8, F = 0.5$ [11,32,38].
- The value of F and CR for $SADE$ and $SFLSDE$ are kept same as suggested by their respective authors [27,42].
- Population size $NP = 50$.
- The stopping criteria is either maximum number of function evaluations (which is set to be 2.0×10^5) is reached or the corresponding acceptable error (mentioned in Table 1) has been achieved.
- The number of simulations/run = 100.
- In order to investigate the effect of the parameter CR on the performance of $FBDE$, its sensitivity with different values of CR in the range [0.1, 1], is examined in Fig. 1. This figure shows the graph between different values of CR and corresponding sum of average number of function evaluations for 22 problems in meeting the termination criteria for $FBDE$. It is clear from Fig. 1 that $FBDE$ is very sensitive for CR and the value 0.3 gives comparatively better results. Therefore $CR = 0.3$ is selected for the experiments in this paper.

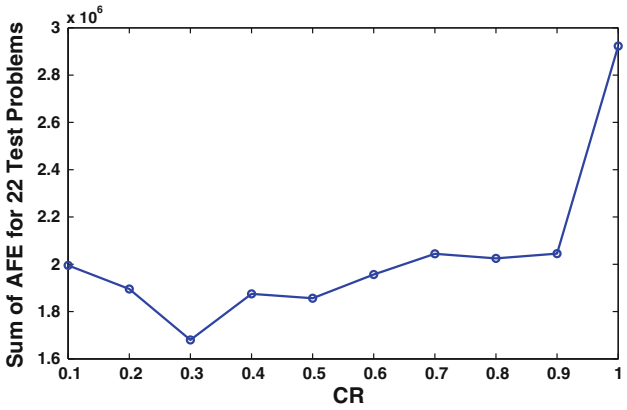

Fig. 1 Effect of parameter CR on average function evaluations

4.3 Results and discussion

In this subsection, a comparison among $FBDE$, $DE/rand/bin/1$, $SADE$ and $SFLSDE$ is carried out. Numerical results with experimental settings of Sect. 4.2, are given in Table 2. In Table 2, success rate (SR) which is the measure of reliability, mean error (ME) which is a measure of accuracy, average function evaluations (AFE) which is a measure of efficiency and standard deviation (SD) are reported. Table 2 shows that most of the time $FBDE$ improves the reliability, efficiency and accuracy. Some more intensive statistical analyses based on t test, Acceleration Rate (AR) [34], Boxplot and Performance Index (PI) [9] have been carried out for results of basic $FBDE$, DE, $SADE$ and $SFLSDE$.

4.3.1 Statistical analysis

The t test is quite popular among researchers in the field of evolutionary computing. In this paper student's t test is applied according to the description given in [6] for a confidence level of 0.95. Table 3 shows the results of the t test for the null hypothesis that there is no difference in the mean number of function evaluations of 100 runs using $FBDE$, DE, $SADE$ and $SFLSDE$. Note that here '+' indicates the significant difference (or the null hypothesis is rejected) at a 0.05 level of significance, '−' implies that there is no significant difference while '=' indicates that comparison is not possible. In Table 3, $FBDE$ is compared with the DE, $SADE$ and $SFLSDE$. The last row of Table 3, establishes the superiority of $FBDE$ over DE, $SADE$ and $SFLSDE$.

Further, a comparison is made on the basis of convergence speed of the considered algorithms by measuring the average function evaluations (AFEs). A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is the average over 100 runs. In order to compare convergence speeds, we use AR which is defined as follows, based on the AFEs for the two algorithms $ALGO$ and $FBDE$:

$$AR = \frac{AFE_{ALGO}}{AFE_{FBDE}}, \tag{7}$$

where, $ALGO \in \{DE, SADE, SFLSDE\}$ and $AR > 1$ means $FBDE$ converges faster. Table 4 shows a clear comparison between $FBDE$-DE, $FBDE$-$SADE$ and

Table 2 Comparison of the results of $FBDE$, DE, $SADE$ and $SFLSDE$

Test problem	Algorithm	SD	ME	AFE	SR
f_1	FBDE	6.98E−07	9.21E−06	20,722	100
	DE	8.24E−07	9.06E−06	22,444	100
	SADE	9.13E−07	8.97E−06	22,824	100
	SFLSDE	6.86E−07	9.22E−06	51,686.74	100
f_2	FBDE	8.60E−07	9.18E−06	42,471.25	100
	DE	4.81E−03	2.37E−03	68,869	78
	SADE	2.15E−03	5.26E−04	43,190	94
	SFLSDE	7.07E−07	9.12E−06	75,301.68	100
f_3	FBDE	1.64E+01	2.99E+01	20,0035.1	0
	DE	2.87E+01	3.76E+01	19,8906.5	1
	SADE	3.52E+00	7.05E+00	199,972	2
	SFLSDE	2.74E+01	3.04E+01	199,973.94	0
f_4	FBDE	7.89E−07	9.19E−06	130,816.83	100
	DE	4.35E+00	1.44E+01	200,050	0
	SADE	3.87E+00	5.37E+00	199,859	4
	SFLSDE	3.38E+00	2.43E+01	199,962.52	0
f_5	FBDE	4.77E−07	9.51E−06	39,100.32	100
	DE	5.09E−07	9.42E−06	42,970.5	100
	SADE	4.71E−07	9.47E−06	43,522	100
	SFLSDE	4.21E−07	9.55E−06	97,538.56	100
f_6	FBDE	7.13E−03	1.31E−03	27,538.8	96
	DE	5.47E−02	5.09E−02	167,464.5	23
	SADE	5.32E−02	2.06E−02	99,597	72
	SFLSDE	2.04E−02	5.42E−03	73,381.35	85
f_7	FBDE	8.81E−07	9.06E−06	32,078.01	100
	DE	3.77E−02	1.04E−02	35,630.5	93
	SADE	8.64E−07	8.88E−06	23,097	100
	SFLSDE	7.80E−07	9.16E−06	51,046.39	100
f_8	FBDE	0.00E+00	0.00E+00	19,731.95	100
	DE	4.49E−01	9.00E−02	26,860	94
	SADE	1.00E−06	1.00E−06	15,181	100
	SFLSDE	1.00E−06	1.00E−06	33,567.17	100
f_9	FBDE	3.93E−01	9.57E+00	200,045.62	0
	DE	3.11E−01	9.13E+00	200,050	0
	SADE	3.49E−01	8.66E+00	200,000	0
	SFLSDE	3.56E−01	9.34E+00	199,961.53	0
f_{10}	FBDE	1.99E−06	7.48E−06	103,199.62	100
	DE	6.30E−01	9.63E−01	180,142	14
	SADE	7.21E−01	4.60E−01	142,282	46
	SFLSDE	6.93E−01	5.77E−01	156,354.59	31
f_{11}	FBDE	4.95E−05	9.71E−05	47,104.69	97
	DE	2.00E−03	4.42E−04	54,410.5	75
	SADE	1.81E−04	5.97E−04	185,152	11
	SFLSDE	1.57E−04	6.20E−04	190,930.5	8
f_{12}	FBDE	2.18E−01	5.00E−02	17,640.59	95
	DE	2.55E−01	7.00E−02	18,004	93
	SADE	2.30E−07	6.58E−07	14,705	100
	SFLSDE	1.40E−01	2.00E−02	18,773.5	98

Table 2 continued

Test problem	Algorithm	SD	ME	AFE	SR
f_{13}	FBDE	3.53E−01	2.00E−01	126,758.1	74
	DE	1.78E+00	2.42E+00	191,330.5	5
	SADE	4.57E−03	9.46E−02	60,960	100
	SFLSDE	7.31E−03	9.29E−02	107,819.14	100
f_{14}	FBDE	1.57E−06	8.07E−06	9,270.67	100
	DE	1.68E−06	7.85E−06	10,358	100
	SADE	1.60E−06	7.97E−06	15,681	100
	SFLSDE	1.40E−06	8.11E−06	24,678.66	100
f_{15}	FBDE	1.57E+01	1.10E+02	200,034.9	0
	DE	1.25E+01	7.80E+01	200,050	0
	SADE	1.54E+01	1.05E+02	200,000	0
	SFLSDE	1.24E+01	1.08E+02	199,959.98	0
f_{16}	FBDE	2.01E+03	1.03E+04	200,034.01	0
	DE	4.14E+03	1.06E+04	200,050	0
	SADE	5.41E+03	1.94E+04	200,000	0
	SFLSDE	5.25E+03	2.00E+04	199,959.78	0
f_{17}	FBDE	2.06E−06	7.49E−06	29,421.97	100
	DE	1.30E−02	1.40E−02	160,446	26
	SADE	1.49E−06	8.04E−06	83,632	100
	SFLSDE	1.46E−06	8.01E−06	91,227.93	100
f_{18}	FBDE	8.51E−07	9.07E−06	21,833.79	100
	DE	1.08E−06	8.70E−06	15,577.5	100
	SADE	1.13E−06	8.66E−06	23,258	100
	SFLSDE	8.87E−07	9.11E−06	36,540.41	100
f_{19}	FBDE	4.21E−15	5.12E−15	10,651.27	100
	DE	4.23E−15	4.61E−15	3,806.5	100
	SADE	4.84E−14	5.67E−14	118,225	43
	SFLSDE	4.84E−14	5.54E−14	116,232.94	45
f_{20}	FBDE	1.49E−05	1.53E−05	84,391.2	57
	DE	1.44E−05	1.67E−05	102,772	49
	SADE	1.48E−05	1.63E−05	99993	51
	SFLSDE	1.44E−05	1.49E−05	86656.62	58
f_{21}	FBDE	1.71E−03	7.96E−03	56940.16	100
	DE	2.30E−01	5.73E−01	198890	2
	SADE	1.53E−01	8.78E−01	200000	0
	SFLSDE	1.05E−01	4.05E−01	199960.08	0
f_{22}	FBDE	5.75E−17	9.31E−16	60081.25	100
	DE	9.03E−17	8.94E−16	59160.5	100
	SADE	8.48E−17	8.89E−16	61167	100
	SFLSDE	7.50E−17	9.10E−16	137125.49	100

$FBDE$-$SFLSDE$ in terms of AR. It is clear from Table 4 that, for most of the test problems, convergence speed of $FBDE$ is faster among all the considered algorithms.

For the purpose of comparison in terms of performance, boxplot analysis is carried out for all the considered algorithms. The empirical distribution of data is efficiently represented graphically by the boxplot analysis tool [41]. Analysis of univariate expression, where the variability of measurements may be affected many parameters, is effectively done by the boxplot tool. Degree of dispersion and skewness in the data are easily analyzed by measuring the spacings between the different parts of the box. The boxplots for comparison among $FBDE$, DE, $SADE$ and $SFLSDE$ based on AFE are shown in Fig. 2. It is clear

Table 3 Results of the Student's *t* test

Test problems	FBDE vs. DE	FBDE vs. SADE	FBDE vs. SFLSDE
f_1	+	+	+
f_2	+	+	+
f_3	=	=	=
f_4	+	+	+
f_5	+	+	+
f_6	+	+	+
f_7	+	−	+
f_8	+	−	+
f_9	=	=	=
f_{10}	+	+	+
f_{11}	+	+	+
f_{12}	+	−	+
f_{13}	+	−	−
f_{14}	+	+	+
f_{15}	=	=	=
f_{16}	=	=	=
f_{17}	+	+	+
f_{18}	−	+	+
f_{19}	−	+	+
f_{20}	+	+	+
f_{21}	+	+	+
f_{22}	−	+	+
Total number of + sign	15	14	17

from this figure that $FBDE$ is best among all considered strategies as Interquartile Range and Median are low for $FBDE$.

In order to compare the consolidated performance of $FBDE$ with DE and its variant ($SADE$ and $SFLSDE$), the value of performance index PI [9] is computed. This index gives a weighted importance to the success rate, the mean error as well as the average number of function evaluations. The value of this performance index for a computational algorithm under comparison is given by Eq. (8).

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i) \tag{8}$$

where $\alpha_1^i = \frac{Sr^i}{Tr^i}$; $\alpha_2^i = \begin{cases} \frac{Mf^i}{Af^i}, & \text{if } Sr^i > 0. \\ 0, & \text{if } Sr^i = 0. \end{cases}$; and $\alpha_3^i = \frac{Mo^i}{Ao^i}$

$i = 1, 2, \ldots, N_p$

- Sr^i = Number of successful runs of ith problem.
- Tr^i = Total number of runs of ith problem.

Table 4 Acceleration rate (AR) of $FBDE$ compare to the basic DE, $SADE$ and $SFLSDE$

Test problems	DE	SADE	SFLSDE
f_1	1.083100087	1.101438085	2.494293022
f_2	1.621543986	1.016923213	1.77300362
f_3	0.99435799	0.999684555	0.999694254
f_4	1.529237484	1.527777427	1.528568763
f_5	1.098980776	1.11308552	2.494571911
f_6	6.081038389	3.616606388	2.664653144
f_7	1.110745336	0.720025962	1.591320347
f_8	1.361244074	0.769361366	1.701158274
f_9	1.000021895	0.999771952	0.999579646
f_{10}	1.745568443	1.378706627	1.515069435
f_{11}	1.155097295	3.930648944	4.053322504
f_{12}	1.020600785	0.833588899	1.064221775
f_{13}	1.509414389	0.480916012	0.850589745
f_{14}	1.1172871	1.691463508	2.66201472
f_{15}	1.000075487	0.99982553	0.999625465
f_{16}	1.000079936	0.999829979	0.999628913
f_{17}	5.453271824	2.842501709	3.100673748
f_{18}	0.71345836	1.065229628	1.67357156
f_{19}	0.357375224	11.09961535	10.91258977
f_{20}	1.2178047	1.184874726	1.026844268
f_{21}	3.492965246	3.512459396	3.511758309
f_{22}	0.984674919	1.018071362	2.282334172

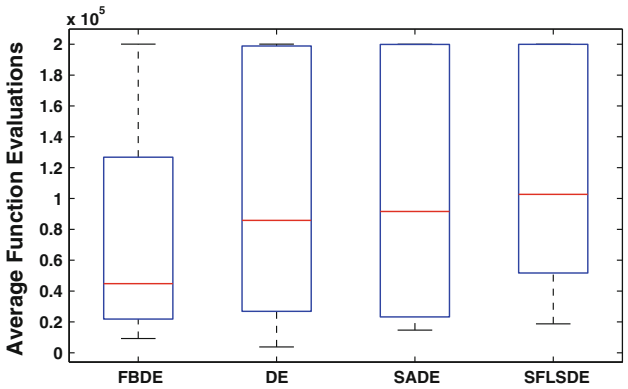

Fig. 2 *Boxplot graph* for average function evaluation: (*1*) $FBDE$, (*2*) DE, (*3*) $SADE$, (*4*) $SFLSDE$

- Mf^i = Minimum of average number of function evaluations used by all the algorithms in obtaining the solution of ith problem.
- Af^i = Average number of function evaluations used by an algorithm in obtaining the solution of ith problem.
- Mo^i = Minimum of mean error obtained by all the algorithms for the ith problem.
- Ao^i = Mean error obtained by an algorithm for the ith problem.
- N_p = Total number of problems analyzed.

k_1, k_2 and k_3 ($k_1 + k_2 + k_3 = 1$ and $k_1, k_2, k_3 \leq 1$) are the weights assigned to success rate, average number of function evaluations and mean error, respectively. From above definition, it is clear that PI is a function of k_1, k_2 and k_3. Since $k_1 + k_2 + k_3 = 1$ one of $k_i, i = 1, 2, 3$ could be eliminated to reduce the number of dependent variables from the expression of PI. We adopt the same methodology as given in [9] i.e. equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows:

1. $k_1 = W, k_2 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
2. $k_2 = W, k_1 = k_3 = \frac{1-W}{2}, 0 \leq W \leq 1;$
3. $k_3 = W, k_1 = k_2 = \frac{1-W}{2}, 0 \leq W \leq 1$

The graphs corresponding to each of the cases (1), (2) and (3) are shown in Fig. 3a–c respectively.

In these figures the horizontal axis represents the weight W and the vertical axis represents the performance index PI.

In case (1), the average number of function evaluations and the mean error are given equal weights. In case (2) the success rate and the mean error are given equal weights and in case (3) the average number of function evaluations and the success rate are given equal weights. PIs of all four algorithms ($FBDE$, DE, $SADE$ and $SFLSDE$) are superimposed in the Fig. 3a–c for comparison. It is observed that for $FBDE$, the value of PI is more than all the remaining three algorithms i.e. DE, $SADE$ and $SFLSDE$.

In order to prove wide applicability of $FBDE$, the next section presents application of $FBDE$ to solve model order reduction problem for single input single output system.

5 Application of $FBDE$ in Model Order Reduction (MOR) problem

Model Order Reduction (MOR) problem is studied in the branch of systems and control theory. In a real world situation, usually we get a system of very high order which is inappropriate for representing some properties that are important for effective use of the system. Model Order Reduction (MOR) problem deals with reduction of complexity of a dynamical system, while preserving their input-output behavior. Although many conventional approaches [4,5,8,13,18,21] of model order reduction guarantee the stability of the reduced order model but sometimes the model may turn out to be non-minimum phase. Therefore to obtain better reduced order models, the use of some kind of optimization is necessary by itself and in combination with other techniques. Error minimization is one of the popular techniques for model order reduction of continuous time systems. In this technique, lower order model is obtained by minimizing an error function constructed from the time responses

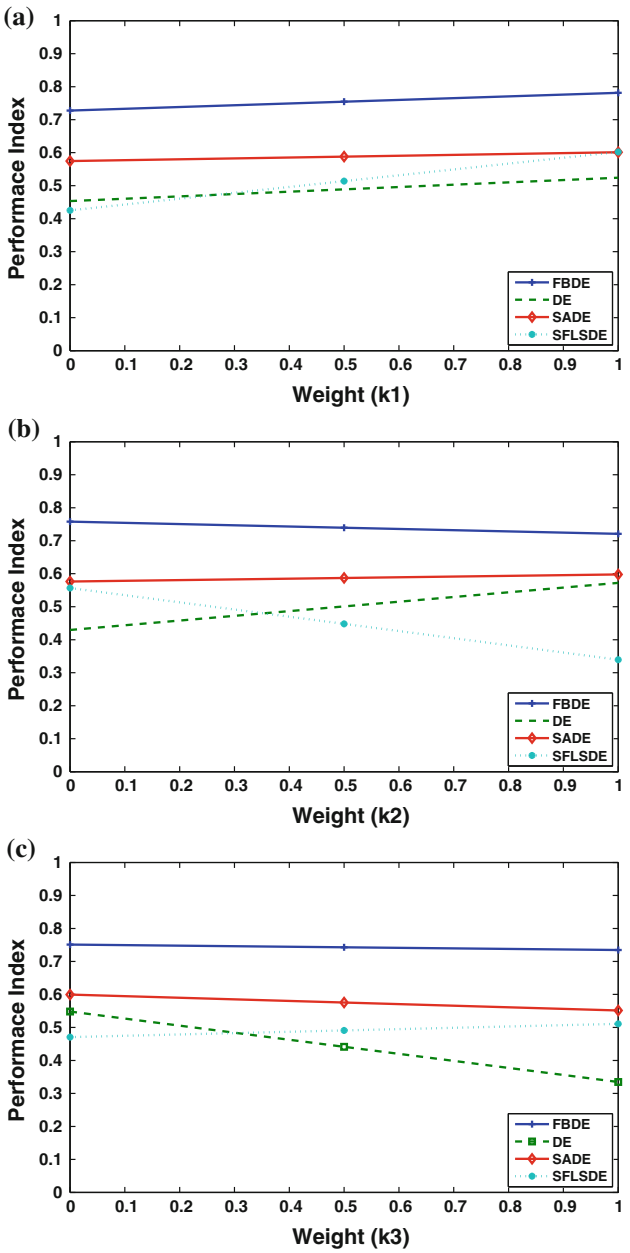

Fig. 3 Performance index; **a** for case (1), **b** for case (2) and **c** for case (3)

(or alternatively frequency responses) of the system and reduced order model.

5.1 MOR as an optimization problem

Consider an nth order linear time invariant dynamic SISO system given by

$$G(s) = \frac{N(s)}{D(s)} = \frac{\sum_{i=0}^{n-1} a_i s^i}{\sum_{i=0}^{n} b_i s^i} \tag{9}$$

where a_i and b_i are known constants.

Table 5 List of MOR problem examples

S. No.	Source	Original model
1	Shamash [36]	$G_1(s) = \dfrac{18s^7 + 514s^6 + 5982s^5 + 36380s^4 + 122664s^3 + 222088s^2 + 185760s + 40320}{s^8 + 36s^7 + 546s^6 + 4536s^5 + 22449s^4 + 67284s^3 + 118124s^2 + 109584s + 40320}$
2	Lucas [22]	$G_2(s) = \dfrac{8169.13s^3 + 50664.97s^2 + 9984.32s + 500}{100s^4 + 10520s^3 + 52101s^2 + 10105s + 500}$

Table 6 Comparison of the Methods for example 1

Method of order reduction	Reduced models; $R_1(s)$	ISE	IRE
Original	$G_1(s)$	–	21.740
FBDE	$\dfrac{17.3217s + 5.3660}{s^2 + 7.0240s + 5.3660}$	0.8×10^{-3}	21.74
DE	$\dfrac{20s + 5.6158}{s^2 + 9.2566s + 5.6158}$	0.3729×10^{-1}	21.908
Pade approximation	$\dfrac{15.1s + 4.821}{s^2 + 5.993s + 4.821}$	1.6177	19.426
Routh approximation	$\dfrac{1.99s + 0.4318}{s^2 + 1.174s + 0.4318}$	1.9313	1.8705
Gutman et al. [29]	$\dfrac{4[133747200s + 203212800]}{85049280s^2 + 552303360s + 812851200}$	8.8160	4.3426
Hutton and Friedland [15]	$\dfrac{1.98955s + 0.43184}{s^2 + 1.17368s + 0.43184}$	18.3848	1.9868
Krishnamurthy and Sheshadri [18]	$\dfrac{155658.6152s + 40320}{65520s^2 + 75600s + 40320}$	17.5345	2.8871
Mittal et al. [1]	$\dfrac{7.0908s + 1.9906}{s^2 + 3s + 2}$	6.9159	9.7906
Mukherjee and Mishra [26]	$\dfrac{7.0903s + 1.9907}{s^2 + 3s + 2}$	6.9165	9.7893
Mukherjee et al. [25]	$\dfrac{11.3909s + 4.4357}{s^2 + 4.2122s + 4.4357}$	2.1629	18.1060
Pal [30]	$\dfrac{151776.576s + 40320}{65520s^2 + 75600s + 40320}$	17.6566	2.7581
Prasad and Pal [31]	$\dfrac{17.98561s + 500}{s^2 + 13.24571s + 500}$	18.4299	34.1223
Shamash [36]	$\dfrac{6.7786s + 2}{s^2 + 3s + 2}$	7.3183	8.9823

The problem is to find a rth order reduced model in the transfer function form $R(s)$, where $r < n$ represented by Eq. (10), such that the reduced model retains the important characteristics of the original system and approximates its step response as closely as possible for the same type of inputs with minimum Integral Square Error.

$$R(s) = \frac{N_r(s)}{D_r(s)} = \frac{\sum_{i=0}^{r-1} a_i' s^i}{\sum_{i=0}^{r} b_i' s^i} \tag{10}$$

where a_i' and b_i' are unknown constants.

Mathematically, the Integral Square Error of step responses of the original and the reduced system can be expressed by error index J [12],

$$J = \int_0^\infty [y(t) - y_r(t)]^2 \, dt. \tag{11}$$

where $y(t)$ is the unit step response of the original system and $y_r(t)$ is the unit step response of the reduced system. This error index J is the function of unknown coefficients a_i'

Table 7 Comparison of the methods for example 2

Method of order reduction	Reduced models; $R_2(s)$	ISE	IRE
Original	$G_2(s)$	–	34.069
FBDE	$\frac{85.33529245s + 462.3004006}{s^2 + 113.6582937s + 462.3004006}$	$0.17826566 \times 10^{-2}$	34.06884
DE	$\frac{220.8190s + 35011.744}{s^2 + 1229.4502s + 35011.744}$	0.4437568×10^{-2}	34.069218
Singh [37]	$\frac{93.7562s + 1}{s^2 + 100.10s + 10}$	0.8964×10^{-2}	43.957
Pade approximation	$\frac{23.18s + 2.36}{s^2 + 23.75s + 2.36}$	0.46005×10^{-2}	11.362
Routh approximation	$\frac{0.1936s + 0.009694}{s^2 + 0.1959s + 0.009694}$	2.3808	0.12041
Gutman et al. [29]	$\frac{0.19163s + 0.00959}{s^2 + 0.19395s + 0.00959}$	2.4056	0.11939
Chen et al. [5]	$\frac{0.38201s + 0.05758}{s^2 + 0.58185s + 0.05758}$	1.2934	0.17488
Marshall [23]	$\frac{83.3333s + 499.9998}{s^2 + 105s + 500}$	0.193×10^{-2}	35.450

and b_i'. The aim is to determine the coefficients a_i' and b_i' of reduced order model so that the error index J is minimized.

5.2 Modified objective function for MOR

In this paper, minimization is carried out based on both ISE and IRE. The low order model is obtained by minimizing an error function, constructed from minimization of the Integral Square Error (ISE) between the transient responses of original higher order model and the reduced low order model pertaining to a unit step input as well as minimization of the difference between the high order model's impulse response energy (IRE) and the reduced low order IRE.

The impulse response energy (IRE) for the original and the various reduced order models is given by:

$$IRE = \int_0^\infty g(t)^2\, dt. \tag{12}$$

where, $g(t)$ is the impulse response of the system.

Therefore, in this paper, both, ISE and IRE, are used to construct the objective function for minimizing the ISE and difference between IRE of high order model and reduced order model. The following modified objective function is constructed to carry out the results.

$$objective\_value = |ISE| + \frac{|IRE_R - IRE_O|}{IRE_R + IRE_O} \tag{13}$$

where ISE is an integral squared error of difference between the responses given by the Eq. (11), IRE_O is the impulse response energy of the original high order model and IRE_R is the impulse response energy of the reduced order model.

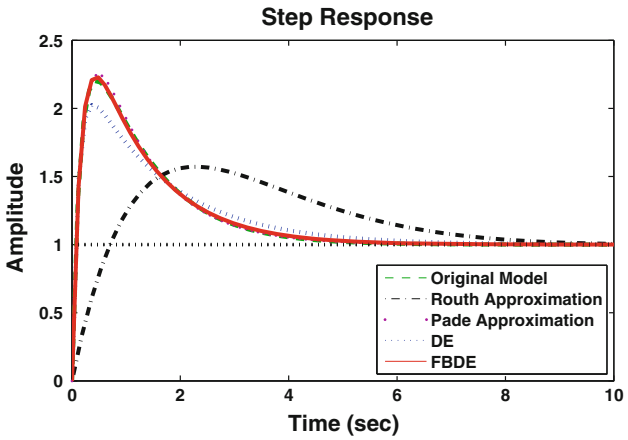

Fig. 4 Comparison of step responses for example 1

The advantage of this modified objective function is that it minimizes ISE as well as the differences of IRE of both the models (high order and reduced order).

5.3 Experimental results and numerical examples

Total two examples are taken into consideration in this section (see Table 5).

The best solution obtained out of 100 runs is reported as the global optimal solution. The reported solutions are in the form of step and impulse responses. The results obtained by $FBDE$ are compared with that of DE and other stochastic as well as deterministic methods.

Tables 6 and 7 present the original and the reduced systems for examples 1 and 2 respectively. In these tables results

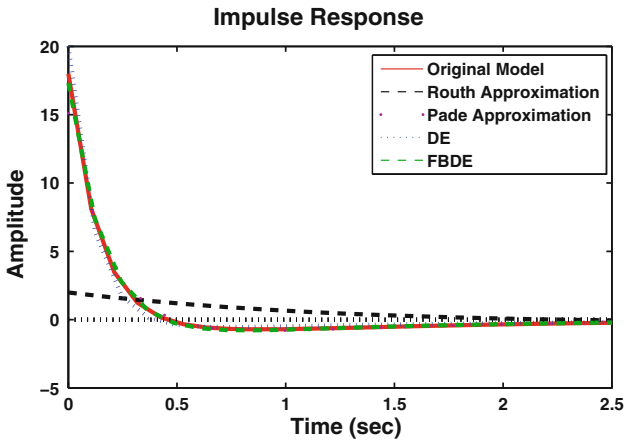

Fig. 5 Comparison of impulse responses for example 1

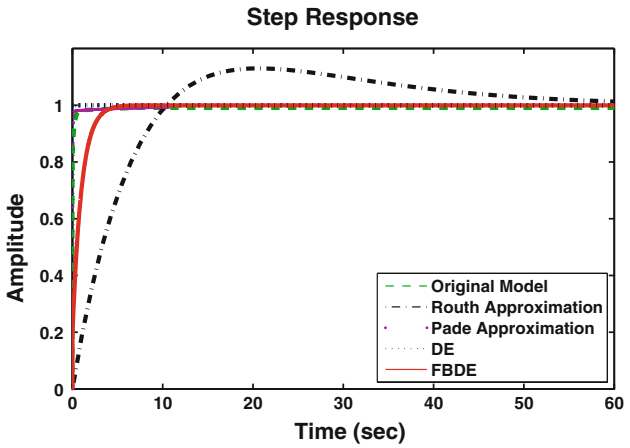

Fig. 6 Comparison of step responses for example 2

obtained by $FBDE$ are compared with that of the basic DE, Pade approximation method, Routh approximation method and other earlier reported results. Corresponding unit step responses of the original and the reduced systems using $FBDE$, DE, Pade Approximation and Routh Approximation are shown in Figs. 4 and 6 respectively. The impulse responses of the original and the reduced systems using $FBDE$, DE, Pade Approximation and Routh Approximation are shown in Figs. 5 and 7, respectively.

It can be observed from Tables 6 and 7 that for examples 1 and 2, ISE obtained by $FBDE$ are relatively less than that of other methods. Also for these examples, IRE of the reduced models obtained by $FBDE$ is most close to that of the originals. It may also be seen that the steady state responses of the original and the reduced order models by $FBDE$ are exactly matching while the transient response matching is also very close as compared to other methods. Thus these examples establish the superiority of $FBDE$ over other methods for this problem. Overall, $FBDE$ performance is superior than the basic DE and other deterministic as well as probabilistic methods.

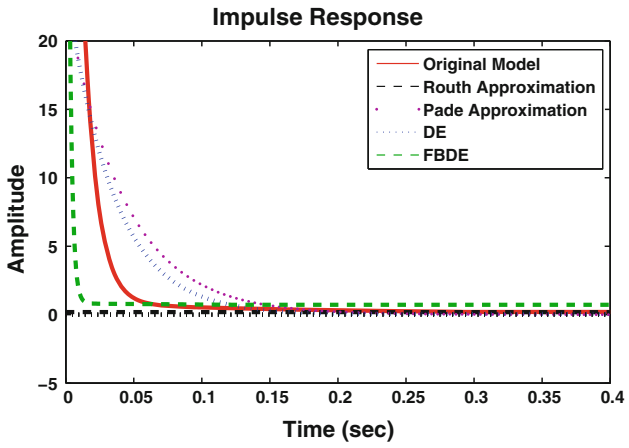

Fig. 7 Comparison of impulse responses for example 2

6 Conclusion

In this paper, $FBDE$ is proposed, analyzed and validated with the help of test problems and an engineering optimization problem. With the introduction of a new position update process, inspired from onlooker bee phase of ABC, $FBDE$ has improved the performance as compare to DE, $SADE$ and $SFLSDE$. Through intensive statistical analysis, improvement is shown in terms of reliability, efficiency and accuracy. Overall, authors recommend $FBDE$ as a better candidate in the field of nature inspired algorithms for function optimization due to its ability to exploit the better search regions in an efficient way.

The future scope of this work is the implementation of the proposed strategy to other nature inspired algorithms.

References

1. Prasad R, Mittal AK, Sharma SP (2004) Reduction of linear dynamic systems using an error minimization technique. J Inst Eng India 84:201–206
2. Ali MM, Khompatraporn C, Zabinsky ZB (2005) A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. J Glob Optim 31:635–672
3. Chakraborty UK (2008) Advances in differential evolution. Springer, Berlin
4. Chen CF, Shieh LS (1968) A novel approach to linear model simplification. Int J Control 8:561–570
5. Chen TC, Chang CY, Han KW (1979) Reduction of transfer functions by the stability-equation method. J Franklin Inst 308:389–404
6. Croarkin C, Tobias P (2010) Nist/sematech e-handbook of statistical methods. Retrieved 1 Mar 2010
7. Das S, Konar A (2006) Two-dimensional IIR filter design with modern search heuristics: a comparative study. Int J Comput Intell Appl 6:329–355
8. Davison JE (1966) A method for simplifying linear dynamic systems. IEEE Trans Autom Control AC-11 1:93–101
9. Thakur M, Deep K (2007) A new crossover operator for real coded genetic algorithms. Appl Math Comput 188(1):895–911

10. Engelbrecht AP (2007) Computational intelligence: an introduction. Wiley, Hoboken
11. Gamperle R, Muller SD, Koumoutsakos A (2002) A parameter study for differential evolution. Adv Intell Syst Fuzzy Syst Evol Comput 10:293–298
12. Gopal M (2002) Control systems: principles and design. Tata McGraw-Hill, New Delhi
13. Gustafson RD (1966) A paper and pencil control system design. Trans ASME J Basic Eng 329–336
14. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan press, Ann Arbor
15. Hutton M, Friedland B (1975) Routh approximations for reducing order of linear, time-invariant systems. IEEE Trans Autom Control 20:329–337
16. Karaboga D, Akay B (2011) A modified artificial bee colony (abc) algorithm for constrained optimization problems. Appl Soft Comput 11:3021–3031
17. Kennedy J, Eberhart R (1995) Particle swarm optimization. IEEE international conference on neural networks proceedings, vol 4. IEEE, pp 1942–1948
18. Krishnamurthy V, Seshadri V (1978) Model reduction using the routh stability criterion. IEEE Trans Autom Control 23:729–731
19. Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. Proceedings of MENDEL 6th international Mendel conference on Soft Computing, pp 76–83
20. Liu PK, Wang FS (2008) Inverse problems of biological systems using multi-objective optimization. J Chin Inst Chem Eng 39:399–406
21. Lucas TN (1983) Factor division: a useful algorithm in model reduction. Control Theory and Applications, IEE Proceedings D, vol 130, IET, pp 362–364
22. Lucas TN (1986) Continued-fraction expansion about two or more points: a flexible approach to linear system reduction. J Franklin Inst 321:49–60
23. Marshall S (1983) Comments on viability of methods for generating stable reduced order models. IEEE Trans Autom Control 28:630–631
24. Mezura-Montes E, Velázquez-Reyes J, Coello Coello CA (2006) A comparative study of differential evolution variants for global optimization, Proceedings of the 8th annual conference on genetic and evolutionary computation, ACM, pp 485–492
25. Mukherjee S et al (2005) Model order reduction using response-matching technique. J Franklin Inst 342:503–519
26. Mukherjee S, Mishra RN (1987) Order reduction of linear systems using an error minimization technique. J Franklin Inst 323:23–32
27. Neri F, Tirronen V (2009) Scale factor local search in differential evolution. Memet Comput 1:153–171
28. Omran MGH, Engelbrecht AP, Salman A (2005) Differential evolution methods for unsupervised image classification, Evolutionary Computation, The 2005 IEEE Congress on, vol 2. IEEE, pp 966–973
29. Mannerfelt CF, Gutman PO, Molander P (1982) Contributions to the model reduction problem. IEEE Trans Autom Control AC-27 2:454–455
30. Pal J (1979) Stable reduced-order padã â© approximants using the routh-hurwitz array. Electron Lett 15:225–226
31. Prasad R, Pal J (1991) Stable reduction of linear systems by continued fractions. Inst Eng India Electr Eng Div 72:113–113
32. Price KV (1996) Differential evolution: a fast and simple numerical optimizer, Fuzzy Information Processing Society, NAFIPS. 1996 Biennial Conference of the North American. IEEE, pp 524–527
33. Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
34. Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12:64–79
35. Rogalsky T, Kocabiyik S, Derksen RW (2000) Differential evolution in aerodynamic optimization. Can Aeronaut Space J 46:183–190
36. Shamash Y (1975) Linear system reduction using pade approximation to allow retention of dominant modes. Int J Control 21:257–272
37. Singh N (2007) Reduced order modeling and controller design, Ph.D. thesis, Indian Institute of Technology Roorkee, India
38. Storn R, Price K (1997) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. J Glob Optim 11:341–359
39. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, KanGAL, Report, pp 341–357
40. Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Evolutionary Computation CEC 2004. Congress on, vol 2. IEEE, pp 1980–1987
41. Williamson DF, Parker RA, Kendrick JS (1989) The box plot: a simple visual method to interpret data. Ann Intern Med 110:916
42. Yan JY, Ling Q, Sun DM (2006) A differential evolution with simulated annealing updating method. International Conference on Machine Learning and Cybernetics, IEEE, pp 2103–2106