An International Journal of Optimization and Control: Theories & Applications
ISSN:2146-0957 eISSN:2146-5703
Vol.10, No.1, pp.118-136 (2020)
http://doi.org/10.11121/ijocta.01.2020.00797

RESEARCH ARTICLE

An improved differential evolution algorithm with a restart technique to solve systems of nonlinear equations

Jeerayut Wetweerapong and Pikul Puphasuk*

Deparment of Mathematics, Faculty of Science, Khon Kaen University, Khon Kaen, 40002, Thailand
wjeera@kku.ac.th, ppikul@kku.ac.th

ARTICLE INFO

Article History:
Received 06 March 2019
Accepted 16 December 2019
Available 31 January 2020

Keywords:
Systems of nonlinear equations
Global optimization
Differential evolution algorithm
Restart technique

AMS Classification 2010:
65K10

ABSTRACT

In this research, an improved differential evolution algorithm with a restart technique (DE-R) is designed for solutions of systems of nonlinear equations which often occurs in solving complex computational problems involving variables of nonlinear models. DE-R adds a new strategy for mutation operation and a restart technique to prevent premature convergence and stagnation during the evolutionary search to the basic DE algorithm. The proposed method is evaluated on various real world and synthetic problems and compared with the recently developed methods in the literature. Experiment results show that DE-R can successfully solve all the test problems with fast convergence speed and give high quality solutions. It also outperforms the compared methods.

1. Introduction

Difficult nonlinear problems arise in a variety of fields in science and engineering, which demands for the effective computational methods and leads to the development of various intelligence methods [1, 2] such as genetic algorithm, particle swarm optimization, and differential evolution algorithm. These methods are recognized as the alternative approaches to the analytical ones and have found applications in many areas. For example, they have been applied to data clustering problems in data science [3, 4], the weight training of artificial neural networks [5,6], and numerical treatments of nonlinear systems [7–11].
This research focuses on solving nonlinear systems of equations which is common and important both in theoretical and application aspects [12–16]. It has been used as a key part of solving complex problems involving decision variables of nonlinear models. New systems of nonlinear equations often emerge from computational problems and can range from moderate problems with a few variables to the hard ones with many variables. For example, physical models that are expressed as nonlinear partial differential equations become large systems of nonlinear equations when discretized [17]. The systems may also have strongly dependent variables or a large number of local solutions, which makes them much more difficult to solve. So the analytical solution approach aiming to get the closed form solution is usually impractical or impossible.
There are two practical approaches to solving a system of nonlinear equations: the local methods that directly solve the original system and the global methods that transform the system into an optimization problem (with box constraints of the variables) and solve that equivalent optimization problem instead. The local methods consist of iterative procedures that require an approximate solution as a starting vector point and use the local information (the derivatives or gradients) from the equational functions of the system to compute a new better approximate solution point for the next iteration. They include all various Newton-type methods [12] and can produce solutions with good computational speed and solution quality if

*Corresponding Author

the functions satisfy some analytical properties and an approximate solution sufficiently close to a real solution is given. Since these critical requirements of the local methods are often not fullfilled, the derivative-free global solution methods are needed.

1.1. Related work

There are a few global methods proposed for solving nonlinear systems in the literature. In 1998, Karr et al. [18] presented a hybrid scheme using a genetic algorithm to locate initial guesses of solutions, which are then supplied to a Newton method. Their results on one selected test problem of finding the nodes and weights for Gauss-Legendre quadrature showed that the genetic algorithm can effectively locate an initial guess that allows the Newton method to converge to an accurate solution. Later, Grosan and Abraham [19] applied a multi-objective optimization approach to solve nonlinear systems in 2008. They used a genetic algorithm and considered the nondominated solutions stored in an external set. Several problems are tested and the obtained solutions of various qualities both in number of different solutions and accuracy are reported. In 2009, Hirsch et al. [20] used the continuous GRASP optimization method, a multi-start local search procedure, to find all roots of nonlinear systems. In order to find different solutions, the objective function is adaptively modified to create an area of repulsion (or penalty region) around solutions that have already been found, and the continuous GRASP is run multiple times. The method showed promising results on four selected nonlinear systems from the literature. Jaberipour et al. [21] used a particle swarm optimization algorithm to solve nonlinear systems in 2011. They proposed a new way of updating each particle and a mechanism to replace some of the worst particles. Several test problems including both nonlinear systems of a single equation and a few equations are tested, and the solutions and their accuracies are reported. In 2012, Pourjafari and Mojallali [22] proposed a hybrid scheme of a two-phase root finder for a nonlinear system using an invasive weed optimization algorithm and a clustering technique. They also aimed to locate all roots of the system. The approach gives successful results on several constructed nonlinear equations in single variable that have many local solutions and on three real world problems of small size that have a few different solutions. Oliveira and Petraglia [23] proposed a stochastic optimization method known as fuzzy adaptive simulated annealing (fuzzy ASA) to find many solutions of a nonlinear system in 2013. The fuzzy ASA is run several times to explore different regions during different activations. The method is stopped when the solutions with the predefined accuracy are found. Several test problems are tested and the obtained high accuracy solutions are reported. In 2016, Raja et al. [24] presented a memetic algorithm (GA-SQP), a hybrid of a genetic algorithm and a local search method based on a sequential quadratic programming (SQP) technique, for solving nonlinear systems. Several variants of GA-SQPs are proposed and tested on six different application problems. The results showed that the hybrid approaches give higher precision solutions and their proposed methods outperform several methods: simulated annealing (SA), pattern-search (PS), Nelder–Mead (NM) and Levenberg-Marquardt (LM) algorithm.
Recently in 2018, Zhang et al. [25] proposed a modified cuckoo search algorithm (CSA) for solving nonlinear equations and nonlinear systems. Four application systems are used to evaluate the CSA performance. By setting high precision tolerance as the termination condition, solutions with high accuracies are obtained and reported. They have shown that the CSA gives more accurate solutions than those obtained by GA-SQPs. And also in 2018, Raja et al. [10] have presented the particle swarm optimization hybrid with Nelder-Mead method (PSO-NMM) to solve nonlinear benchmark models. PSO-NMM exploits the strength of PSO as an efficient global search method to find good initial solutions and then applies the Nelder-Mead simplex method to refine the solutions for rapid local convergence. They have shown that for moderate to difficult nonlinear system problems, the hybridization of NMM can enhance the convergence and give quality solutions with high precision.

1.2. Innovative contribution

From the development of computational intelligence and evolutionary optimization methods to solve various complex real world and synthetic test problems of nonlinear systems, it is clear that the global methods become the essential tools and need more researchers attentions. This approach will also make solving nonlinear systems an important field of global optimization since it can supply plenty of challenging test problems. In addition, research contributions in this direction still lack common ground of how to compare and establish the obtained results.
In this research, we aim to apply the differential

evolution algorithm (DE) to solve nonlinear systems. DE is a popular and efficient global optimization method introduced by Storn and Price during the years 1995-1997 [26, 27] and it is surprising that no explicit research contributions on using DE to solve the problems are found among our extensive related literature reviews. We propose an improved differential evolution algorithm with a restart technique (DE-R) for solving the problems and offer a basic ground of applying DE in this direction. The main features of the proposed algorithm can be summarized as follows:

- The mixing strategy of the xbest-mutation to the basic DE mutation that utilizes the current best vector solution to enhance the search and the convergence to an optimal solution.
- The incorporating of a restart technique to prevent premature convergence and stagnation during the evolutionary search.
- Its performance both in term of the convergence speed and the achievement of high precision solutions are tested on several nonlinear benchmark problems of varying difficulty.

The remainder of the paper is organized as follows. The basic differential algorithm is presented in the next section and the proposed algorithm is described in Section 3. Section 4 gives details of the experimental design and lists all the benchmark problems. In Section 5, the performances of the DE-R method are compared with those of the basic DE algorithms based on the setting of the value to reach, and with those of the PSO and PSO-NMM methods [10] based on the setting of the maximum number of function evaluations. Finally, the conclusion are given in the last section.

2. The differential evolution algorithm (DE)

The basic or *classic* DE algorithm [26,27] for solving a continuous optimization problem is a stochastic search method using a population of real vectors with four main operations: initialization, mutation, crossover and selection. The pseudo code of the basic DE is illustrated in Table 1. Its main features are the differential mutation, the combined binomial crossover and the greedy selection. First, the initial population is generated uniformly in feasible region. For each generation and each target vector x_i, three different random population vectors x_{r1}, x_{r2}, x_{r3}, which are also different from the target vector, are used to generate a mutant vector $xm = x_{r1} + F(x_{r2} - x_{r3})$ by adding the scaled difference of two vectors to another one with the scaling factor F. Then, some components of the target vector are exchanged with those of the mutant vector according to the crossover rate CR to produce the trial vector. The trial vector will replace the target vector in the selection process if it is fitter.

For more than two decades, DE has been shown to be one of the most efficient methods for continuous optimization problems [28–30]. However, DE's performances depend on five main factors: the population size NP, the control parameters F and CR, the dimension D, the objective function f to be solved, and the amount of computations allowed [31,32]. There are numerous research contributions on modifying DE to improve the performances in solving practical problems. These include the main approaches of adapting the control parameters and adjusting the basic mutation operation [33–42].

3. An improved differential evolution algorithm with a restart technique (DE-R)

We propose an improved differential evolution algorithm with a restart technique (DE-R) as a general purpose method for solving nonlinear systems through their equivalent transformed objective functions. Let the form of a nonlinear system be

$$\begin{cases} f_1(x_1, x_2, ..., x_n) &= 0 \\ f_2(x_1, x_2, ..., x_n) &= 0 \\ & \vdots \\ f_m(x_1, x_2, ..., x_n) &= 0 \end{cases}$$

where $f_i : [LB, UB]^n \subseteq R^n \to R$ for $i = 1, ..., m$ are nonlinear equations (including linear functions) and $x = (x_1, x_2, ..., x_n)$ is a real vector. We want to find a solution x^* such that $f_i(x^*) = 0$ simultaneously for all i. The problem is transformed into the corresponding optimization problem by defining the objective function f as the sum of the absolutes or the squares of all f_i. For the smoothness of f, we use the sum of the squares. So the objective function f is as follows:

$$f(x) = \sum_{i=1}^{m} f_i^2(x). \qquad (1)$$

Table 1. Pseudo code of the DE algorithm.

The DE algorithm

(1) **Inputs:**
Objective function to be minimize (f), problem dimension (D), and population size (NP).

(2) **Initialization:**
(2.1) Randomly initialize all (row) vectors of the population matrix $P = [x_{ij}]$ of size $NP \times D$ with real values between the lower and upper bounds LB and UB.
(2.2) Calculate the fitness (the objective function value) for each population vector and record the best vector *xbest* and the best value *fbest*.

(3) **Mutation:**
For each target population vector x_i, construct the mutant vector xm by

$$xm = x_{r1} + F(x_{r2} - x_{r3})$$

where $r1$, $r2$ and $r3$ are randomly generated distinct indices (in the range of 1 to NP) which are also different from the target index i, and F is a scaling factor.

(4) **Crossover:**
Construct the trial vector (the candidate vector) xc by replacing some components of x_i with the corresponding components of xm as follows:

$$xc_j = \begin{cases} xm_j & ; rand() < CR \text{ or } j = IC, \\ x_{ij} & ; \text{otherwise} \end{cases}$$

where $rand()$ is a uniform random number in $[0, 1]$, CR is a crossover rate, and IC is a randomly fixed index from 1 to D.

(5) **Selection:**
Apply the greedy selection and check for an update of *xbest*.
(5.1) Greedy selection:
If $f(xc) < f(x_i)$, then replace the target vector x_i with xc.
(5.2) Updating *xbest*:
If $f(xc) < fbest$, then update *xbest* with xc and update *fbest* with $f(xc)$.

(6) **Stopping condition:**
Repeat all the steps (3) - (5) until *fbest* is less than the value to reach (VTR) or the maximum number of function evaluations ($maxnf$) is reached. Then report the obtained best solution.

To be able to optimize the objective function by using small number of function evaluations, we tend to use small population size. But optimizing with small populations will tend to loss diversity and lead to premature convergence or stagnation easily [31, 32, 43, 44]. We can also accelerate the convergence by utilizing the information of the population *xbest* but this again increases the chance of those convergence problems. DE-R utilizes the *xbest* information and at the same time prevents premature convergence and stagnation by incorporating an xbest-mutation and a restart technique.

The pseudo code and the flowchart of the proposed DE-R method are presented in Table 2 and Figure 1, respectively. After initialization, the DE-R creates a mutant vector by using the mixing scheme of two mutation operations: the basic mutation

$$xm = x_{r1} + F(x_{r2} - x_{r3}) \tag{2}$$

and the xbest-mutation [28] that uses the *xbest* information

$$xm = xbest + F_1(x_{r1} - x_{r2}) + F_2(x_{r3} - x_{r4}) \tag{3}$$

where x_{r1}, x_{r2}, x_{r3}, and x_{r4} are different random vectors from the population and different from the target vector x_i. Each mutation operation is randomly chosen and applied with the proportion of $50\% : 50\%$. And instead of using a fixed value for scaling factors, the DE-R algorithm uses random values in the range of $[0.5, 0.7]$ for F, F_1 and F_2. These basic proportion and range of scaling factors are chosen from the preliminary experiments. For the crossover operation, the fixed value of crossover rate $CR = 0.9$ is used to generate the trial vector. Then the same greedy selection as in basic DE is applied. To prevent premature convergence or stagnation, the restart technique randomly restart PR of the population vectors by replacing them with the new generated vectors as in the initialization for every NRS generations. This incorporated restart operation periodically supplies small amount of new contents to the evolving population. Note that the *xbest* vector is kept, updated and used in the xbest-mutation along the entire optimization process.

Table 2. Pseudo code of the improved DE algorithm with a restart (DE-R).

The improved DE algorithm with a restart (DE-R)

(1) **Inputs and control parameters:**
Objective function to be minimize : f
Problem dimension: D
Lower and upper bounds of the problem: LB, UB
Population size: $NP = 50$
The value to reach: $VTR = 10^{-20}$
Maximum number of function evaluations: $maxnf$
Scaling factors: F, F_1, F_2 in the range of $[0.5, 0.7]$
Crossover rate: $CR = 0.9$
Mixing rate of basic mutation and xbest-mutation: 0.5
The period to apply a restart (number of generations): $NRS = 200$
Restart rate (the percentage to restart the population vectors): $PR = 0.2$

(2) **Initialization:**
(2.1) Randomly initialize all (row) vectors of the population matrix $P = [x_{ij}]$ of size $NP \times D$ with real values between the lower and upper bounds LB and UB.
(2.2) Calculate the fitness (the objective function value) for each population vector and record the best vector $xbest$ and the best value $fbest$.

(3) **Mutation:**
(3.1) For each target population vector x_i, generate a uniform random number u in $[0, 1]$. If $u < 0.5$, apply the DE basic mutation in (3.2); otherwise, apply the xbest- mutation in (3.3).
(3.2) Basic mutation:
Randomly generate distinct indices $r1$, $r2$ and $r3$ (in the range of 1 to NP) which are also different from the target index i. Construct the mutant vector xm by

$$xm = x_{r1} + F(x_{r2} - x_{r3})$$

where F is randomly generated in $[0.5, 0.7]$.
(3.3) The xbest-mutation:
Randomly generate distinct indices $r1$, $r2$, $r3$, and $r4$ (in the range of 1 to NP) which are also different from the target index i. Construct the mutant vector xm by

$$xm = xbest + F_1(x_{r1} - x_{r2}) + F_2(x_{r3} - x_{r4})$$

where $xbest$ is the current best solution and F_1, F_2 are randomly generated in $[0.5, 0.7]$.

(4) **Crossover:**
Construct the trial vector (the candidate vector) xc by replacing some components of x_i with the corresponding components of xm as follows:

$$xc_j = \begin{cases} xm_j & ; rand() < CR \text{ or } j = IC, \\ x_{ij} & ; \text{otherwise} \end{cases}$$

where xm is the mutant vector from the step (3), CR is the crossover rate, $rand()$ is a uniform random number in $[0, 1]$, and IC is a randomly fixed index from 1 to D.

(5) **Selection:**
Apply the greedy selection and check for an update of $xbest$.
(5.1) Greedy selection:
If $f(xc) < f(x_i)$, then replace the target vector x_i with xc.
(5.2) Updating $xbest$:
If $f(xc) < fbest$, then update $xbest$ with xc and update $fbest$ with $f(xc)$.

(6) **Restart:**
Apply the restart technique every NRS generations by randomly choosing $PR \times NP$ distinct population vectors to be re-initialized as in the initialization where PR is the restart rate.

(7) **Stopping condition:**
Repeat all the steps (3) - (6) until $fbest$ is less than VTR or $maxnf$ is reached. Then report the obtained best solution.

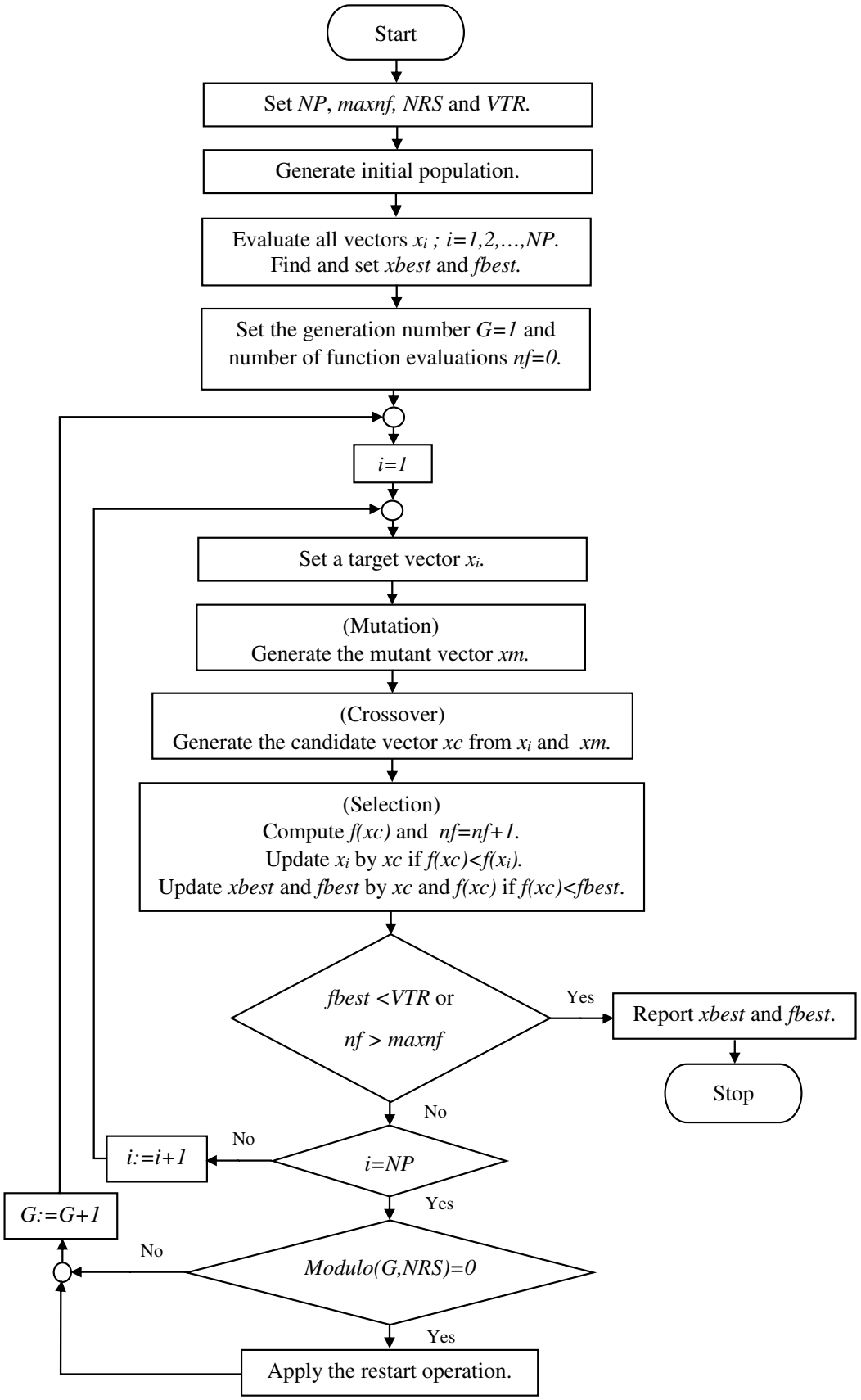

Figure 1. Flowchart of the proposed DE-R method.

4. Experimental design

To evaluate the performance of the DE-R algorithm, two experiments with different settings and measurements are performed. The first experiment assesses the proposed DE-R algorithm against the basic DE algorithms on 10 nonlinear benchmark models by setting the value to reach (VTR) while the second experiment compares the DE-R algorithm to the PSO and PSO-NMM methods [10] by setting the maximum number of function evaluations ($maxnf$).

4.1. Experiment 1: Performance comparison of the DE-R with the basic DE algorithms

The DE-R algorithm and the basic DE algorithms are tested on 10 selected nonlinear systems consisting of 6 real world problems (case study 1-6) and 4 synthetic problems (case study 7-10). Their definitions, parameters, variable bounds and the solutions (for the case of synthetic problems) are listed as follows.

Case study 1: *Neurophysiology application* [23, 45]. The system consists of the following six equations in six variables

$$\begin{cases} f_1(x) = & x_1^2 + x_3^2 - 1 = 0 \\ f_2(x) = & x_2^2 + x_4^2 - 1 = 0 \\ f_3(x) = & x_5 x_3^3 + x_6 x_4^3 - c_1 = 0 \\ f_4(x) = & x_5 x_1^3 + x_6 x_2^3 - c_2 = 0 \\ f_5(x) = & x_5 x_1 x_3^2 + x_6 x_4^2 x_2 - c_3 = 0 \\ f_6(x) = & x_5 x_1^2 x_3 + x_6 x_2^2 x_4 - c_4 = 0 \end{cases}$$

where $c_i = 0$ for all i as in [23] and $-10 \le x_i \le 10$.

Case study 2: *Robot kinematics application* [46]. This problem concerns the indirect-position problem which is to find the desired position and orientation of the robot hand and the relative joint displacements. This problem can be reduced to the following system of eight equations in eight variables

$$\begin{cases} f_1(x) = & 4.731 \times 10^{-3} x_1 x_3 - 0.3578 x_2 x_3 \\ & -0.1238 x_1 + x_7 - 1.637 \times 10^{-3} x_2 \\ & -0.9338 x_4 - 0.3571 = 0 \\ f_2(x) = & 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 \\ & -0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0 \\ f_3(x) = & x_6 x_8 + 0.3578 x_1 + 4.731 \times 10^{-3} x_2 = 0 \\ f_4(x) = & -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0 \\ f_5(x) = & x_1^2 + x_2^2 - 1 = 0 \\ f_6(x) = & x_3^2 + x_4^2 - 1 = 0 \\ f_7(x) = & x_5^2 + x_6^2 - 1 = 0 \\ f_8(x) = & x_7^2 + x_8^2 - 1 = 0 \end{cases}$$

where $-1 \le x_i \le 1$.

Case study 3: *Automative steering application* [20, 47]. This problem describes the kinematic synthesis of a trailing six-member mechanism for automotive steering. The system contains three equations in three unknown as follows

$$\begin{cases} f_i(x) = & [E_i(x_2 \sin\phi_i - x_3) - F_i(x_2 \sin\psi_i - x_3)]^2 \\ & +[F_i(1 + x_2 \cos\psi_i) - E_i(x_2 \cos\phi_i - 1)]^2 \\ & -[(1 + x_2 \cos\psi_i)(x_2 \sin\phi_i - x_3)x_1 \\ & -(x_2 \sin\psi_i - x_3)(x_2 \cos\phi_i - x_3)x_1]^2 = 0\,, \\ & i = 1, 2, 3 \end{cases}$$

where $0 \le x_i \le 1$ and

$$\begin{aligned} E_i = & \; x_2(\cos\psi_i - \cos\psi_0) - x_2 x_3(\sin\psi_i - \sin\psi_0 \\ & -(x_2 \sin\psi_i - x_3)x_1 \\ F_i = & -x_2 \cos\phi_i - x_2 x_3 \sin\phi_i + x_2 \cos\phi_0 \\ & +x_1 x_3 + (x_3 - x_1)x_2 \sin\phi_0. \end{aligned}$$

The constants ϕ_i and ψ_i are given as follows

$$\begin{aligned} \phi_0 = & \; 1.3954170041747090114, \\ \phi_1 = & \; 1.7444828545735749268, \\ \phi_2 = & \; 2.0656234369405315689, \\ \phi_3 = & \; 2.4600678478912500533, \\ \psi_0 = & \; 1.7461756494150842271, \\ \psi_1 = & \; 2.0364691127919609051, \\ \psi_2 = & \; 2.2390977868265978920, \\ \psi_3 = & \; 2.4600678409809344550. \end{aligned}$$

Case study 4 : *Economics modeling application* [23, 45]. This problem arises in economics modeling. It can be extended for general dimensions n as follows

$$\begin{cases} f_i(x) = & (x_i + \sum_{j=1}^{n-i-1} x_j x_{j+i})x_n - c_i = 0, \\ & i = 1, 2, ..., n-1 \\ f_n(x) = & \sum_{j=1}^{n-1} x_j + 1 = 0 \end{cases}$$

where $c_i = 0$ for all i as in [23] and $-10 \le x_i \le 10$.

Case study 5 : *Chemical equilibrium application* [23, 46, 48]. This problem describes a chemical equilibrium system. It concerns the combustion of propane in air to form ten products, which are transformed to ten equations in ten variables. To solve this problem, the system can be reduced to the following systems of five equations in five

variables

$$\begin{cases} f_1(x) = & x_1x_2 + x_1 - 3x_5 = 0 \\ f_2(x) = & 2x_1x_2 + x_1 + x_2x_3^2 + R_5x_2 - R_1x_5 \\ & +2R_7x_2^2 + R_4x_2x_3 + R_6x_2x_4 = 0 \\ f_3(x) = & 2x_2x_3^2 + 2R_2x_3^2 - 8x_5 + R_3x_3 \\ & +R_4x_2x_3 = 0 \\ f_4(x) = & R_6x_2x_4 + 2x_4^2 - 4R_1x_5 = 0 \\ f_5(x) = & x_1(x_2 + 1) + R_7x_2^2 + x_2x_3^2 + R_5x_2 \\ & +R_2x_3^2 + x_4^2 - 1 + R_3x_3 + R_4x_2x_3 \\ & +R_6x_2x_4 = 0 \end{cases}$$

where $-100 \leq x_i \leq 100$ and the constants used in this system are

$$\begin{aligned} R_1 = & \ 10, R_2 = 0.193, \ R_3 = 0.002597/\sqrt{40}, \\ R_4 = & \ 0.003448/\sqrt{40}, \ R_5 = 0.00001799/40, \\ R_6 = & \ 0.0002155/\sqrt{40}, \ R_7 = 0.00003846/40. \end{aligned}$$

Case study 6 : *Combustion application* [23, 45]. This problem is a typical chemical equilibrium problem which represents a combustion problem. The system consists of ten equations in ten unknowns as follows

$$\begin{cases} f_1(x) = & x_2 + 2x_6 + x_9 + 2x_{10} - 10^{-5} = 0 \\ f_2(x) = & x_3 + x_8 - 3 \times 10^{-5} = 0 \\ f_3(x) = & x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} \\ & -5 \times 10^{-5} = 0 \\ f_4(x) = & x_4 + 2x_7 - 10^{-5} = 0 \\ f_5(x) = & 0.5140437 \times 10^{-7}x_5 - x_1^2 = 0 \\ f_6(x) = & 0.1006932 \times 10^{-6}x_6 - 2x_2^2 = 0 \\ f_7(x) = & 0.7816278 \times 10^{-15}x_7 - x_4^2 = 0 \\ f_8(x) = & 0.1496236 \times 10^{-6}x_8 - x_1x_3 = 0 \\ f_9(x) = & 0.6194411 \times 10^{-7}x_9 - x_1x_2 = 0 \\ f_{10}(x) = & 0.2089296 \times 10^{-14}x_{10} - x_1x_2^2 = 0 \end{cases}$$

where $-20 \leq x_i \leq 20$.

Case study 7 : *Rosenbrock function* [49]. This function is well-known for testing the performance of the optimization methods. The two-dimensional function has the global minimum inside a long, narrow, parabolic shaped flat valley. It is unimodal for dimensions 2 and 3, and has 2 minima for higher dimensions. The high-dimensional function is highly nonseparable and is used as one of the difficult test functions. Rosenbrock function can be written in the form of system of nonlinear equations in general dimensions n as follows

$$\begin{cases} f_{2i-1}(x) & = 10(x_{i+1} - x_i^2) = 0, \ i = 1, 2, ..., n-1 \\ f_{2i}(x) & = 1 - x_i = 0, \ i = 1, 2, ..., n-1. \end{cases}$$

In this work, we set $n = 10$ and $-100 \leq x_i \leq 100$. The global solution is $(1, 1, ..., 1)$.

Case study 8 : *SINQUAD function* [50]. This function is multimodal and nonseparable and it is one of the test functions from CUTE: Constrained and Unconstrained Testing Environment. It can be scaled up to arbitrary dimension n and can be written in the form of the following system

$$\begin{cases} f_1(x) = (x_1 - 1)^2 = 0 \\ f_i(x) = \sin(x_i - x_n) - x_1^2 + x_i^2 = 0, \ i = 1, ..., n-1 \\ f_n(x) = x_n^2 - x_1^2 = 0. \end{cases}$$

We set $n = 10$ and $-100 \leq x_i \leq 100$. There are $2^{n-1} + 1$ solutions in the forms $(1, 1, ..., 1, 1)$ and $(1, a_1, a_2, ..., a_8, -1)$ where $a_j \in \{0.2357835607, -1\}$.

Case study 9 : *Proposed function 1.* This function can be written in the form of the system of three nonlinear equations in n variables where $n \geq 3$. The first two equations give the intersection of two n-spheres which is an $(n-1)$-sphere. The last equation chooses the two intersection points where $x_1 = 0.05$. This system is expected to be highly nonseparable since the searching points must lie in the $(n-1)$-sphere. It is shown as follows

$$\begin{cases} f_1(x) = & x_1^2 + x_2^2 + \cdots + x_n^2 - 100 = 0 \\ f_2(x) = & (x_1 - 0.1)^2 + x_2^2 + \cdots + x_n^2 - 100 = 0 \\ f_3(x) = & x_1^2 + (x_2 - x_3)^2 + (x_3 - x_4)^2 + \cdots \\ & +(x_{n-1} - x_n)^2 - 0.0025 = 0. \end{cases}$$

We set $n = 10$ and $-100 \leq x_i \leq 100$. There are two solutions which are $(0.05, a, a, ..., a)$ and $(0.05, -a, -a, ..., -a)$ where $a = \sqrt{(100 - 0.05^2)/(n-1)}$.

Case study 10 : *Proposed function 2.* This function can be written in the form of the system of three nonlinear equations in n variables where n is even. The system has one obvious solution at $(n, n, ..., n)$. It is presented as follows

$$\begin{cases} f_1(x) = & x_1 + x_2 + \cdots + x_n - n^2 = 0 \\ f_2(x) = & x_1^2 + x_2^2 + \cdots + x_n^2 - n^3 = 0 \\ f_3(x) = & x_1^2 - x_2^2 + x_3^2 - x_4^2 + \cdots + x_{n-1}^2 - x_n^2 = 0. \end{cases}$$

We set $n = 10$ and $-100 \leq x_i \leq 100$.

For the first experiment, the proposed DE-R is tested and compared with two basic DE algorithms. The basic DEs follow the settings as recommended in [27] to use NP in the range of $5 \times D$ to $10 \times D$, $F = 0.5$ and $CR = 0.9$. Since the maximum D of all test problems is 10, $NP = 50$ and $NP = 100$ are chosen. So we denote these basic DEs as DE59-50 and DE59-100, respectively.

The DE-R method, denoted by DE59-50-R, uses $NP = 50$, F in $[0.5, 0, 7]$ and the same $CR = 0.9$. The settings and features of DE59-50, DE59-100 and DE59-50-R are summarized in Table 3. Each algorithm is run 30 times for each problem. The maximum number of function evaluations $maxnf$ is set to 1000000 and the VTR (value to reach) for $fbest$ is set to 10^{-20} to guarantee that each $f_i(xbest)$ is less than 10^{-10}. If the $fbest$ value is less than the VTR before reaching the $maxnf$, the successful run and the number of function evaluations used (nf) are recorded. We report the number of successful runs (NS), the mean of function evaluations (Mean nf) and the percentage of standard derivation (%SD).

4.2. Experiment 2: Performance comparison of DE-R with PSO and PSO-NMM methods

The second experiment compares the performance of the DE-R algorithm with those of the PSO and PSO-NMM using the same setting as in [10]. The objective function f is the mean square defined by

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)^2. \tag{4}$$

We choose the following PSO and PSO-NMM variants that obtain good results in [10]: PSO-12, PSO-15, PSO-16, PSO-NMM-12, PSO-NMM-15, and PSO-NMM-16. They are compared on the case studies 1, 4, 5, and 6. Note that for the case study 4, the parameters $c_i = 1$ for $i = 1, 2, ..., n-1$ where $n = 5$. Each algorithm conducts 100 independent runs for each problem. The Min, Mean and SD values of $fbest$ of DE-R at each $maxnf$ are compared with those of PSO and PSO-NMM. The PSO-12 and PSO-15 methods use $maxnf = 50000$ while PSO-16 uses $maxnf = 100000$. The PSO-NMM-12, PSO-NMM-15 and PSO-NMM-16 variants extend the corresponding PSO variants respectively. For each PSO-NMM method, if its PSO phase obtains the solution that reaches the tolerance 10^{-15} or uses up all the $maxnf$ without reaching the tolerance, then it enters the NMM phase by applying the local search using the Nelder-Mead method with the additional function evaluations of 200000. To compare the DE-R with each PSO-NMM method on each test problem, the $maxnf$ for DE-R needs to be adjusted properly. For the former case, Neurophysiology and Economics modeling applications, the DE-R method uses only 200000 as the $maxnf$. For the latter case, Chemical equilibrium and Combustion applications, DE-R uses the sum of the function evaluations of both two phases.

5. Results and discussion

This section shows performance comparison of the DE-R with the basic DE algorithms and the performance comparison of DE-R with PSO and PSO-NMM methods. In each report table, the best values are indicated in bold. The discussion for each experiment is given as follows.

5.1. Performance comparison of the DE-R with the basic DE algorithms

The performance comparison of DE59-50, DE59-100 and DE59-50-R are shown in Table 4, Figure 2 and Figure 3. The table presents the NS, Mean and %SD of each method for all 10 test systems. For the ability and stability of solving each problem, the number of successful runs out of the total 30 runs is considered first. From Table 4, the results show that the DE59-50 algorithm can successfully solve 6 out of 10 problems. It cannot solve the problems 6, 8, 9 and 10, and gives no successful runs out of total 30 runs for these problems. It can solve problem 5 (Chemical Equilibrium Applications) and problem 7 (Rosenbrock problem), but gives high Means and %SDs. However, DE59-50 gives the best results for the first four problems which appear to be relatively easy problems. We can conclude that DE59-50 is good for easy problems but cannot be recommended for solving difficult problems. This is because of its too small population size.

The DE59-100 algorithm can solve all 10 problems. This shows that the increased population size of 100 can increase the solving ability of the DE algorithm with $F = 0.5$ and $CR = 0.9$. However, DE59-100 gives very high Mean values. This shows that its speed of convergence is rather slow.

The proposed DE59-50-R algorithm can successfully solve all 10 problems. It also gives the best results (smallest Means) for the last 6 problems which appear to be difficult problems. For the first four easy problems, DE59-50-R gives the second best results. This shows that the incorporated new mutation strategy and restart technique can prevent the premature convergence and stagnation, and still gives fast convergence speeds for difficult problems as clearly shown in Figure 2 and Figure 3.

Some solutions obtained by DE59-50-R for all application systems and SINQUAD function are reported. We show only 4 different solutions for each problem. Since we set the $VTR = 10^{-20}$, the absolute values of each $f_i(x)$ must be less than 10^{-10}. Table 5 shows some solutions of Neurophysiology application. Our method gives 30

Table 3. The settings and features of DE59-50, DE59-100 and DE59-50-R.

DE algorithm	NP	F	CR	Mutation and other feature
DE59-50	50	0.5	0.9	basic mutation
DE59-100	100	0.5	0.9	basic mutation
DE59-50-R	50	$[0.5, 0.7]$	0.9	basic mutation & xbest-mutation and a restart technique

Table 4. Performance comparison of DE59-50, DE59-100 and DE59-50-R using NS, Mean nf, %SD values at $VTR = 10^{-20}$ averaged over 30 independent runs.

Systems	Methods	NS	Mean nf	%SD
1: Neurophysiology application	DE59-50	30	**27272.70**	16.98
	DE59-100	30	76787.67	19.51
	DE59-50-R	30	40233.67	16.99
2: Robot kinematics application	DE59-50	30	**24942.53**	28.27
	DE59-100	30	57355.60	18.13
	DE59-50-R	30	34721.30	17.70
3: Automative steering application	DE59-50	30	**2303.30**	12.34
	DE59-100	30	4307.83	8.89
	DE59-50-R	30	2682.10	12.03
4: Economics modeling application	DE59-50	30	**12780.60**	5.96
	DE59-100	30	30193.97	4.63
	DE59-50-R	30	21831.93	7.96
5: Chemical equilibrium application	DE59-50	30	190249.03	47.06
	DE59-100	30	124793.33	5.70
	DE59-50-R	30	**30582.23**	3.95
6: Combustion application	DE59-50	0	-	-
	DE59-100	30	95503.70	3.79
	DE59-50-R	30	**59380.20**	4.13
7: Rosenbrock function	DE59-50	30	193992.90	50.51
	DE59-100	30	110388.83	3.62
	DE59-50-R	30	**59565.40**	2.52
8: SINQUAD function	DE59-50	0	-	-
	DE59-100	30	173259.07	12.29
	DE59-50-R	30	**81755.37**	8.90
9: Proposed function 1	DE59-50	0	-	-
	DE59-100	30	109782.20	6.95
	DE59-50-R	30	**65107.80**	5.72
10: Proposed function 2	DE59-50	0	-	-
	DE59-100	30	638354.80	14.20
	DE59-50-R	30	**160827.47**	12.69

different solutions for 30 runs. The values of each variable can be positive or negative. There are two trends of solutions. First, their magnitudes are pairwise equal as $|x_1| = |x_2|$, $|x_3| = |x_4|$ and $|x_5| = |x_6|$. For another trend, all their components are different. We notice that the absolute values of x_5 and x_6 are quite smaller than other components.

Some solutions of Robot kinematics application are presented in Table 6. The authors in [46] claimed that there are 16 solutions. Our results show that the proposed method gives 10 different solutions in 30 runs. All $|x_i|$ are not equal but have roughly the same order. Table 7 shows some solutions of Automative steering application. Our method gives all 30 different solutions for 30 runs. The absolute values of x_1 are bigger than others and the values of x_2 and x_3 have quite the same order. For the Economics modeling application, 30 different solutions are obtained. Some of them are reported in Table 8. Each variable can be positive or negative. The absolute values of x_{10} are much smaller than others which have the same order.

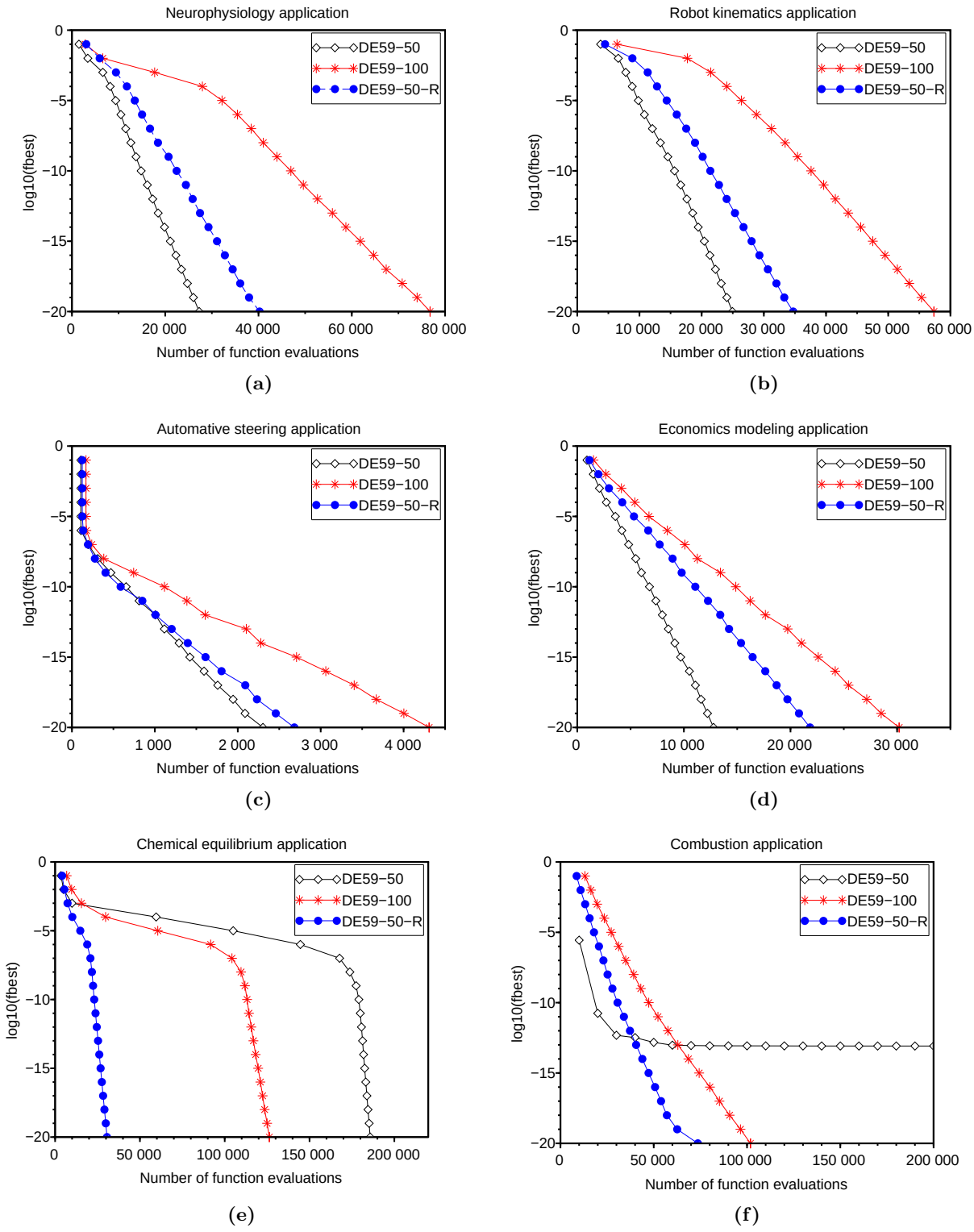

Figure 2. Convergence graphs of DE59-50, DE59-100 and DE59-50-R for systems 1 - 6.

There are four real solutions of the Chemical equilibrium application reported in [48]. Our method gives all four solutions as shown in Table 9. For the Combustion application, the proposed method gives 30 different solutions in 30 runs. Some solutions are presented in Table 10. It shows that the absolute values of x_5, x_6, x_9 and x_{10} are bigger than those of other components which are rather small.

In case of the synthetic test problems, the numbers of solutions for each of Rosenbrock function, proposed functions 1 and 2 are at most 2 and all these solutions are found. Thus we show only the solutions of SINQUAD function which has $2^{n-1}+1$ solutions. Some of them are shown in Table 11. After running the proposed method 30

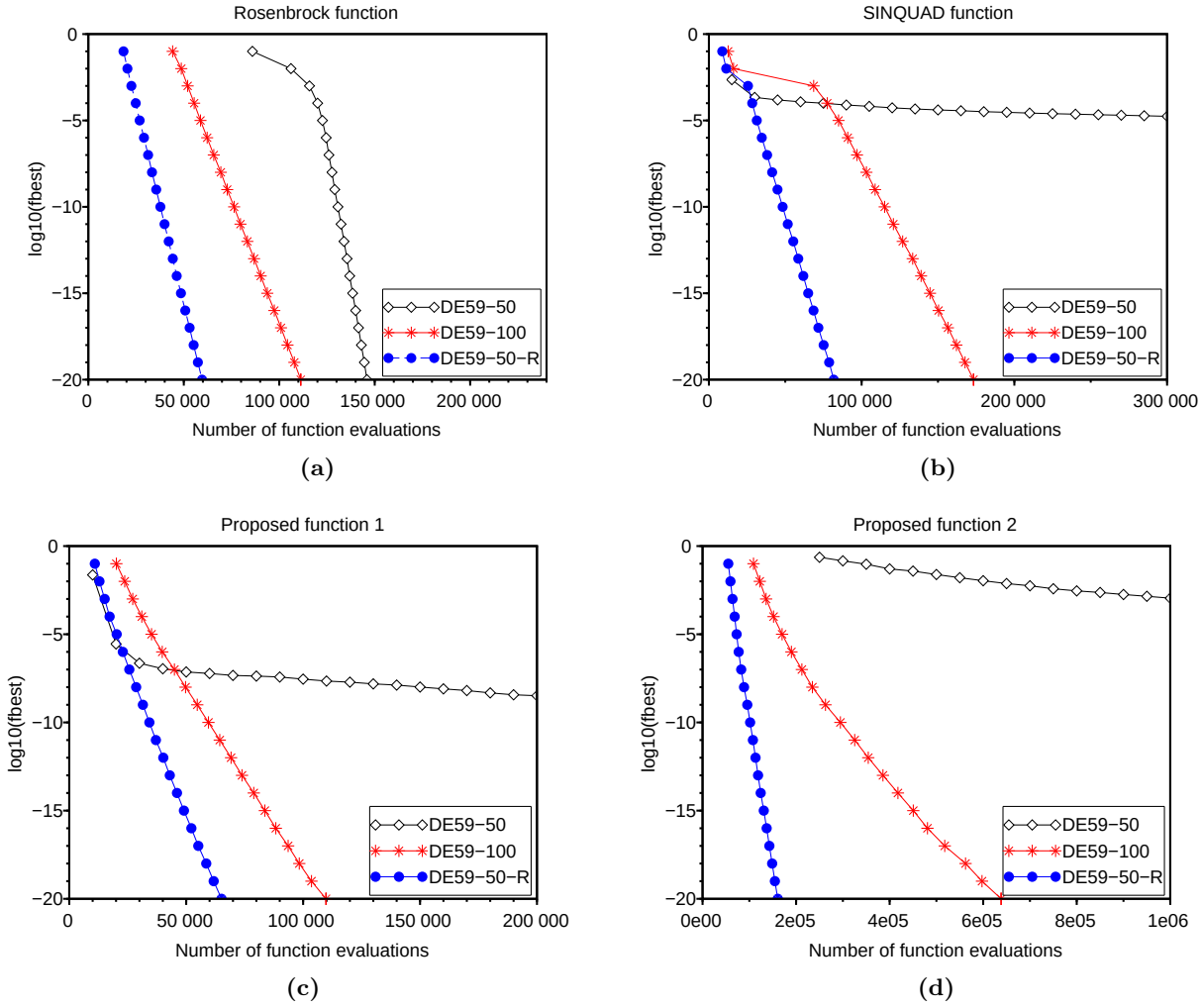

Figure 3. Convergence graphs of DE59-50, DE59-100 and DE59-50-R for systems 7 - 10.

Table 5. Some solutions of Neurophysiology application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	9.7749269097E-01	- 4.2695388140E-02	9.2272247439E-02	- 8.2491292630E-01
x_2	- 9.7749277453E-01	4.2695288767E-02	- 1.1674880868E-01	7.9424101299E-01
x_3	- 2.1096928480E-01	9.9908813614E-01	9.9573381602E-01	5.6525981992E-01
x_4	2.1096889745E-01	- 9.9908814042E-01	- 9.9316147514E-01	6.0760284168E-01
x_5	- 2.9012525772E-05	1.0549841840E-04	1.0199560579E-09	- 4.1050316028E-11
x_6	- 2.9012444215E-05	1.0549834938E-04	1.0098557221E-09	4.2217134911E-11
$f_1(x)$	1.9671153595E-11	- 5.6388893555E-11	1.6900481015E-11	- 1.7983725620E-11
$f_2(x)$	- 4.6815662458E-11	1.6239010137E-11	3.8434810889E-11	- 7.0937811181E-11
$f_3(x)$	2.2663307018E-12	6.7481831402E-11	1.7678400883E-11	2.0558188234E-12
$f_4(x)$	- 6.9223284026E-11	- 6.2704301665E-14	- 8.0570377011E-13	4.4194815939E-11
$f_5(x)$	- 8.0752967592E-12	- 1.3367709799E-11	- 2.2980110987E-11	2.3198700376E-11
$f_6(x)$	2.6177939401E-11	1.0192828755E-12	- 5.0234619236E-12	3.9133806914E-13

times, it can give 27 different solutions.

To show that the proposed method can also give high quality solutions, it is applied to solve two difficult application systems and one difficult synthetic function (Neurophysiology and Combustion systems, and SINQUAD function) by setting 3 different levels of the VTR values: 10^{-20}, 10^{-30} and 10^{-40}. The solutions once reaching each VTR level are recorded in the same run in order to investigate their behaviors and accuracies. The results are shown in tables 12, 13, and 14, respectively. One set of three solutions (each one at each accuracy level) from the same run is reported for each problem.

Table 6. Some solutions of Robot kinematics application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	1.6443166583E-01	1.6443166582E-01	6.7155426177E-01	6.7155426182E-01
x_2	- 9.8638847688E-01	- 9.8638847685E-01	7.4095537891E-01	7.4095537883E-01
x_3	- 9.5472843449E-01	2.3961601716E-01	- 2.3961165919E-01	9.5472976979E-01
x_4	2.9747876626E-01	- 9.7086773778E-01	- 9.7086881339E-01	2.9747448069E-01
x_5	- 9.1115479620E-01	- 9.9763539824E-01	9.5791710189E-01	- 1.2877823744E-01
x_6	4.1206423943E-01	- 6.8728539911E-02	- 2.8704498935E-01	9.9167341678E-01
x_7	9.9132241509E-01	- 6.1550840708E-01	- 5.2790902637E-01	9.6931180772E-01
x_8	- 1.3145291671E-01	7.8813031971E-01	8.4930092421E-01	- 2.4583453656E-01
$f_1(x)$	- 2.4255042419E-12	4.6582404600E-11	2.1869284161E-11	- 1.5662637853E-11
$f_2(x)$	3.1039615322E-11	4.5871084708E-12	- 1.3820167233E-11	5.0422999109E-12
$f_3(x)$	2.1743830694E-12	1.2673250470E-11	1.4527121259E-11	- 5.7471936498E-11
$f_4(x)$	1.5060341862E-11	2.7537028213E-11	5.3715809578E-11	- 2.9148350400E-12
$f_5(x)$	4.0200731632E-11	- 1.2329803845E-11	3.2028157904E-11	- 1.9571233523E-11
$f_6(x)$	- 9.7184482684E-12	- 6.6788574671E-11	3.0616620350E-11	- 2.4448998381E-11
$f_7(x)$	5.4400040028E-11	1.0798473227E-11	- 5.2414739216E-12	- 1.9850676658E-11
$f_8(x)$	- 2.2364221586E-11	2.5351054589E-11	- 1.3006706823E-11	- 3.8787306700E-11

Table 7. Some solutions of Automative steering application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	1.1192696492E-01	1.4669784820E-01	2.1001674043E-02	1.8459769700E-02
x_2	3.8819470790E-05	2.5842964204E-05	1.0358260865E-05	2.5086298124E-05
x_3	1.3969968025E-05	9.5393185049E-06	9.4604945191E-05	4.0484643474E-06
$f_1(x)$	- 2.4953389394E-12	5.1027141958E-13	1.5509436820E-11	1.0749400610E-12
$f_2(x)$	6.2113867145E-12	- 1.8836413824E-12	2.9312085336E-11	1.9709660895E-11
$f_3(x)$	9.1878352205E-11	3.1456991936E-11	4.8961749305E-11	6.6461220923E-11

Table 8. Some solutions of Economics modeling application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	- 6.1626101672E+00	6.0154692495E+00	5.1444125822E+00	3.5697160902E-01
x_2	8.4423418690E+00	2.4451383666E+00	- 8.8035624273E-01	- 3.2209182844E+00
x_3	- 6.0135423035E+00	- 4.3358334024E+00	1.8536153932E+00	6.4891799317E+00
x_4	6.6724322251E+00	9.7533235680E-01	9.9392766203E-01	- 8.4197837365E+00
x_5	1.4648933274E+00	- 6.7099720313E+00	8.8882743353E-01	3.8711746522E+00
x_6	- 9.4952931192E+00	2.9005275369E+00	8.0171844852E-01	5.6239280280E+00
x_7	- 1.8950537683E+00	- 2.0062068336E+00	- 3.9152883109E+00	6.0633634547E-01
x_8	2.5753259373E+00	3.0111923766E+00	- 5.5692586733E-01	1.0345842326E+00
x_9	3.4115059994E+00	- 3.2956476191E+00	- 5.3299310986E+00	- 7.3414727781E+00
x_{10}	- 2.1904782760E-13	- 5.8674381000E-14	- 6.1325687030E-13	4.1496727200E-14
$f_1(x)$	2.8765042169E-11	2.4583005293E-12	- 2.7179403893E-12	- 3.7642815954E-12
$f_2(x)$	8.8232434540E-13	- 2.3163223689E-12	- 1.6660038822E-11	3.1943021197E-13
$f_3(x)$	4.1467842031E-12	3.4719167082E-12	6.0666202644E-13	- 6.1741234660E-13
$f_4(x)$	1.0720305561E-11	- 8.5259062950E-14	4.7145635625E-12	- 2.4212746954E-12
$f_5(x)$	- 1.1227997919E-11	6.1242977433E-13	- 1.3063316353E-12	3.0065629487E-12
$f_6(x)$	- 7.4689413357E-13	- 7.3251520844E-13	1.7618556611E-11	- 1.8728334542E-12
$f_7(x)$	- 2.4172620758E-12	- 4.7228264171E-13	1.2805457393E-12	1.0217298215E-12
$f_8(x)$	4.0410941261E-12	9.8653194558E-13	1.7156652338E-11	- 6.5818503348E-14
$f_9(x)$	- 7.4728297801E-13	1.9337008404E-13	3.2686168644E-12	- 3.0464709312E-13
$f_{10}(x)$	4.9998227780E-11	- 8.8817841970E-16	- 8.0000006619E-11	- 9.9991126490E-12

From tables 12 and 13, the solution for both application problems at $VTR = 10^{-20}$ are quite different from those at $VTR = 10^{-30}$ and 10^{-40} where at these higher accuracies we obtained similar solutions with more accurate solutions at $VTR = 10^{-40}$. For the solutions of SINQUAD function as in Table 14, we can see the same trend

Table 9. All solutions of Chemical equilibrium application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	3.1141022831E-03	2.7571773851E-03	2.4710000144E-03	2.1533077099E-03
x_2	3.4597924347E+01	3.9242289252E+01	4.3879222733E+01	5.0549570315E+01
x_3	6.5041778861E-02	- 6.1387603945E-02	5.7784455215E-02	- 5.4144807517E-02
x_4	8.5937805056E-01	8.5972442500E-01	- 8.6020547295E-01	- 8.6067132299E-01
x_5	3.6951859146E-02	3.6985043297E-02	3.6965520015E-02	3.7000695742E-02
$f_1(x)$	4.4613771011E-11	- 4.0732806017E-11	- 2.5496216249E-11	- 2.4678356580E-11
$f_2(x)$	7.7566347442E-12	- 3.0284536164E-11	5.4004083757E-11	2.1900292344E-11
$f_3(x)$	- 5.7297743156E-11	- 7.1773320794E-12	- 2.5416176759E-11	8.9192338411E-12
$f_4(x)$	1.4782175484E-11	- 2.1605606193E-11	- 7.5870421057E-11	- 2.4946711363E-11
$f_5(x)$	- 4.8531411962E-11	- 6.2299256465E-11	- 2.2982240893E-11	- 3.8321561636E-11

Table 10. Some solutions of Combustion application.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	- 2.1256693800E-07	- 1.2076437010E-06	- 8.2052168644E-08	- 2.9426051511E-07
x_2	- 8.1757590664E-06	4.2644337543E-06	- 1.4477162217E-07	- 4.7401492243E-06
x_3	- 6.7527163990E-04	- 4.2090672635E-05	5.1227964677E-05	- 6.0923109793E-05
x_4	- 4.1833078103E-06	6.4157325619E-06	2.1289152054E-06	3.0404307410E-06
x_5	1.6567014001E-04	4.5312194101E-04	- 2.2149062432E-04	3.2378004814E-04
x_6	1.2934173578E-03	8.0745848233E-04	- 2.3204194677E-04	1.6392254306E-04
x_7	7.0916610888E-06	1.7921487387E-06	3.9355599926E-06	3.4797831273E-06
x_8	7.0527161222E-04	7.2090667874E-05	- 2.1227993570E-05	9.0923084400E-05
x_9	5.3586029742E-04	- 3.0507240444E-04	4.9435402384E-04	- 1.1232728214E-03
x_{10}	- 1.5522596511E-03	- 6.5205449512E-04	- 1.0062660059E-05	4.0508395318E-04
$f_1(x)$	- 4.8273876190E-11	3.7401625020E-12	3.8557848129E-11	2.1883831795E-11
$f_2(x)$	- 2.7678047280E-11	- 4.7604545021E-12	- 2.8892893645E-11	- 2.5393269464E-11
$f_3(x)$	- 5.6092310138E-11	1.8660651569E-12	4.0505311148E-11	2.6580721802E-11
$f_4(x)$	1.4367258554E-11	3.0039278904E-11	3.5190617200E-11	- 3.0043345588E-12
$f_5(x)$	8.4709844719E-12	2.1834044602E-11	- 1.1392318563E-11	1.6557120143E-11
$f_6(x)$	- 3.4477399301E-12	4.4934787964E-11	- 2.3406963800E-11	- 2.8432143924E-11
$f_7(x)$	- 1.7500064230E-11	- 4.1161624304E-11	- 4.5322799486E-12	- 9.2442190882E-12
$f_8(x)$	- 3.8015147210E-11	- 4.0044070424E-11	1.0271567782E-12	- 4.3230264589E-12
$f_9(x)$	3.1455493138E-11	- 1.3747522017E-11	3.0610441206E-11	- 7.0974973960E-11
$f_{10}(x)$	1.0965489675E-17	2.0599143169E-17	- 1.9304158566E-20	7.4580841134E-18

Table 11. Some solutions of SINQUAD function.

	Solution 1	Solution 2	Solution 3	Solution 4
x_1	1.0000013135E+00	1.0000020233E+00	9.9999578687E-01	9.9999635081E-01
x_2	- 1.0000013135E+00	2.3578778570E-01	- 9.9999578691E-01	- 9.9999635080E-01
x_3	2.3578630346E-01	2.3578778573E-01	2.3577476333E-01	- 9.9999635077E-01
x_4	2.3578630350E-01	2.3578778572E-01	- 9.9999578687E-01	- 9.9999635077E-01
x_5	2.3578630340E-01	- 1.0000020233E+00	- 9.9999578688E-01	- 9.9999635083E-01
x_6	- 1.0000013135E+00	- 1.0000020234E+00	2.3577476328E-01	- 9.9999635075E-01
x_7	2.3578630343E-01	2.3578778581E-01	- 9.9999578683E-01	- 9.9999635080E-01
x_8	2.3578630341E-01	2.3578778575E-01	2.3577476333E-01	- 9.9999635085E-01
x_9	- 1.0000013135E+00	2.3578778572E-01	- 9.9999578684E-01	- 9.9999635081E-01
x_{10}	- 1.0000013135E+00	- 1.0000020233E+00	- 9.9999578686E-01	- 9.9999635083E-01
$f_1(x)$	1.7252114825E-12	4.0939003788E-12	1.7750473715E-11	1.3316562876E-11
$f_2(x)$	2.0599966177E-11	- 2.9519581224E-11	2.7519320156E-11	7.3162587100E-12
$f_3(x)$	1.2524037363E-11	- 3.2059285782E-12	3.5025565781E-11	- 2.5635049639E-11
$f_4(x)$	4.9083563602E-11	- 1.0487832824E-11	- 1.0218825786E-11	- 2.1720181209E-11
$f_5(x)$	- 2.9645889910E-11	- 1.5809797915E-11	- 4.2824632729E-12	3.1798563782E-11
$f_6(x)$	5.6868731946E-11	- 7.3832051584E-12	- 1.0347889212E-11	- 4.7247761259E-11
$f_7(x)$	- 6.3057475908E-12	6.1809182772E-11	- 4.7515213986E-11	9.0893959026E-13
$f_8(x)$	- 2.4836653567E-11	1.0494015379E-11	3.5985284197E-11	5.4105275815E-11
$f_9(x)$	- 1.9166890297E-12	- 1.3320324010E-11	- 4.3021475271E-11	1.6147416737E-11
$f_{10}(x)$	1.9149126729E-12	- 5.2741366829E-11	- 2.2285395751E-11	3.4775182733E-11

Table 12. Solutions of Neurophysiology application at $VTR = 10^{-20}$, 10^{-30} and 10^{-40}.

	Solution at $VTR = 10^{-20}$	Solution at $VTR = 10^{-30}$	Solution at $VTR = 10^{-40}$
x_1	9.774926909651037E-01	9.774910633733470E-01	9.774910827638724E-01
x_2	- 9.774927745309928E-01	- 9.774910647465765E-01	- 9.774910827654670E-01
x_3	- 2.109692848010614E-01	- 2.109768258014221E-01	- 2.109767359618408E-01
x_4	2.109688974538331E-01	2.109768194390176E-01	2.109767359544526E-01
x_5	- 2.901252577197261E-05	- 9.810715906270790E-08	2.444699263961000E-10
x_6	- 2.901244421520325E-05	- 9.810715863387080E-08	2.444699263882000E-10
$f_1(x)$	1.967115359491345E-11	2.220446049250313E-16	0
$f_2(x)$	- 4.681566245778868E-11	- 2.220446049250313E-16	0
$f_3(x)$	2.266330701846849E-12	8.737850539388192E-17	- 3.153751293361295E-22
$f_4(x)$	- 6.922328402630745E-11	- 1.434521764580082E-17	6.260981755202151E-21
$f_5(x)$	- 8.075296759227116E-12	- 2.701156197524962E-16	1.071345911685355E-21
$f_6(x)$	2.617793940060294E-11	6.272936219045604E-16	- 3.157529343873903E-21
Mean nf	40233.67	57500.57	77651.60
%SD	16.99	12.58	5.00

Table 13. Solutions of Combustion application at $VTR = 10^{-20}$, 10^{-30} and 10^{-40}.

	Solution at $VTR = 10^{-20}$	Solution at $VTR = 10^{-30}$	Solution at $VTR = 10^{-40}$
x_1	- 2.125669379957560E-07	1.371784251667994E-07	1.379796690717610E-07
x_2	- 8.175759066442083E-06	- 9.619976447815970E-08	- 1.024640702937120E-07
x_3	- 6.752716398985501E-04	1.565137776437351E-05	1.560729129475898E-05
x_4	- 4.183307810306110E-06	7.224452410444500E-09	6.565809411140000E-11
x_5	1.656701400100495E-04	3.763180720674607E-07	3.703652200388360E-07
x_6	1.293417357824663E-03	1.844925109281104E-07	2.085321794461283E-07
x_7	7.091661088782333E-06	4.996387774070343E-06	4.999967170952946E-06
x_8	7.052716122205029E-04	1.434862223559040E-05	1.439270870524103E-05
x_9	5.358602974230222E-04	- 2.040883524405975E-07	- 2.282373401823797E-07
x_{10}	- 1.552259651139891E-03	4.965651547579558E-06	4.956818525791918E-06
$f_1(x)$	- 4.827387619045422E-11	9.657869664593532E-17	0
$f_2(x)$	- 2.767804727950166E-11	- 3.609037981661123E-17	0
$f_3(x)$	- 5.609231013833449E-11	- 5.014435047745458E-18	0
$f_4(x)$	1.436725855439561E-11	5.511287633945886E-16	1.694065894508601E-21
$f_5(x)$	8.470984471899497E-12	5.264730829992297E-16	1.728855061170187E-21
$f_6(x)$	- 3.447739930109454E-12	6.835193007961348E-17	1.049095379878782E-21
$f_7(x)$	- 1.750006423022506E-11	- 5.218880731519356E-17	- 4.028719824372849E-22
$f_8(x)$	- 3.801514721037196E-11	- 1.388394783354915E-16	2.272081369646687E-21
$f_9(x)$	3.145549313765976E-11	5.544608392318181E-16	- 3.954927090396494E-22
$f_{10}(x)$	1.096548967520139E-17	9.105212626901849E-21	8.907628343995374E-21
Mean nf	59380.20	93172.65	129106.13
%SD	4.13	4.11	4.23

of behaviors at all 3 different VTR values.
From these 3 tables, we can conclude that the proposed method DE-R can give more accurate solutions by setting higher precision values of VTR. From $VTR = 10^{-20}$ to 10^{-30} and from 10^{-30} to 10^{-40}, DE-R requires greater numbers of function evaluations. The increased percentages of Means are 35.83% and 35.04% for Neurophysiology application. For Combustion application, they are 59.91% and 60.51%. And for SINQUAD function, they are 41.08% and 58.25%, respectively.

5.2. Performance comparison of DE-R with PSO and PSO-NMM methods

The performance comparison of DE-R, PSO variants, and PSO-NMM variants on 4 nonlinear benchmark problems are presented in Table 15 and Table 16. The DE-R is the same as DE59-50-R in the first experiment except that it uses the setting of $maxnf$ as described in Section 4.2.

Both tables report Min, Mean and SD of $fbest$ based on the corresponding $maxnf$'s. From the results, the first two problems are relatively easy problems whereas the last two problems are more difficult problems with the Chemical equilibrium

Table 14. Solutions of SINQUAD function at $VTR = 10^{-20}$, 10^{-30} and 10^{-40}.

	Solution at $VTR = 10^{-20}$	Solution at $VTR = 10^{-30}$	Solution at $VTR = 10^{-40}$
x_1	1.000001313473061E+00	1.000000000741824E+00	9.999999999921714E-01
x_2	- 1.000001313492704E+00	- 1.000000000741824E+00	- 9.999999999921714E-01
x_3	2.357863034574390E-01	2.357835623069449E-01	2.357835607415836E-01
x_4	2.357863035031185E-01	2.357835623069448E-01	2.357835607415836E-01
x_5	2.357863034047495E-01	2.357835623069443E-01	2.357835607415836E-01
x_6	- 1.000001313528972E+00	- 1.000000000741824E+00	- 9.999999999921714E-01
x_7	2.357863034339120E-01	2.357835623069443E-01	2.357835607415836E-01
x_8	2.357863034107586E-01	2.357835623069438E-01	2.357835607415836E-01
x_9	- 1.000001313470187E+00	- 1.000000000741824E+00	- 9.999999999921714E-01
x_{10}	- 1.000001313474019E+00	- 1.000000000741824E+00	- 9.999999999921714E-01
$f_1(x)$	1.725211482508887E-12	5.503030808840126E-19	6.128739500341215E-23
$f_2(x)$	2.059996617731485E-11	0	0
$f_3(x)$	1.252403736273777E-11	4.232725281383409E-16	0
$f_4(x)$	4.908356360244781E-11	3.677613769070831E-16	0
$f_5(x)$	- 2.964588990961303E-11	- 1.179611963664229E-16	0
$f_6(x)$	5.686873194576947E-11	0	0
$f_7(x)$	- 6.305747590751309E-12	- 1.179611963664229E-16	0
$f_8(x)$	- 2.483665356711739E-11	- 4.371503159461554E-16	0
$f_9(x)$	- 1.916689029712870E-12	0	0
$f_{10}(x)$	1.914912672873470E-12	0	0
Mean nf	81755.37	115341.27	182524.47
%SD	8.90	6.03	18.83

Table 15. Performance comparison of PSO, PSO-NMM and DE-R methods on Neurophysiology and Economics modeling applications averaged over 100 independent runs.

Problems	$maxnf$	Methods	Min	Mean	SD
Neurophysiology application	50000	PSO-12	1.01E-29	1.41E-11	1.11E-10
		PSO-15	0	1.48E-11	1.39E-10
		DE-R	3.75E-36	**1.06E-13**	1.03E-12
	100000	PSO-16	0	2.37E-10	2.34E-09
		DE-R	0	**1.52E-37**	1.49E-36
	200000	PSO-NMM-12	0	6.14E-30	3.71E-29
		PSO-NMM-15	0	3.09E-32	1.27E-31
		PSO-NMM-16	0	2.36E-32	1.16E-31
		DE-R	0	**5.39E-96**	5.39E-95
Economics modeling application	50000	PSO-12	2.77E-30	2.52E-27	4.08E-27
		PSO-15	2.47E-33	**1.03E-32**	4.56E-33
		DE-R	7.40E-33	1.36E-32	4.30E-33
	100000	PSO-16	2.47E-33	**9.69E-33**	4.32E-33
		DE-R	7.40E-33	1.35D-32	4.41E-33
	200000	PSO-NMM-12	4.93E-33	4.91E-32	4.13E-32
		PSO-NMM-15	2.47E-33	1.02E-32	4.46E-33
		PSO-NMM-16	2.47E-33	**9.60E-33**	4.24E-33
		DE-R	7.40E-33	1.37E-32	4.15E-33

application as the most difficult one. The DE-R gives the best Mean values for all cases of Neurophysiology application, Chemical equilibrium application, and Combustion application. It shows much better performances on Chemical equilibrium application and Combustion application, especially on Chemical equilibrium application where it gives the Mean values in the order of 10^{-33} while the PSO and PSO-NMM give the values in the order of 10^{-4}. For the Economics modeling application, all methods produce nearly the same results with PSO-15, PSO-16 and PSO-NMM-16 giving only slightly better results. Thus, we can conclude that the DE-R outperforms all the compared methods.

6. Conclusions

In this paper, we have proposed an efficient improvement of the differential evolution algorithm

Table 16. Performance comparison of PSO, PSO-NMM and DE-R methods on Chemical equilibrium and Combustion applications averaged over 100 independent runs.

Problems	*maxnf*	Methods	Min	Mean	SD
Chemical equilibrium application	50000	PSO-12	9.84E-06	5.79E-04	3.04E-03
		PSO-15	3.55E-06	1.61E-03	5.10E-03
		DE-R	1.54E-34	**1.05E-33**	1.34E-33
	100000	PSO-16	4.16E-07	5.73E-04	3.05E-03
		DE-R	1.54E-34	**1.08E-33**	1.33E-33
	250000	PSO-NMM-12	2.58E-34	5.33E-04	3.04E-03
		PSO-NMM-15	1.22E-34	1.60E-03	5.11E-03
		DE-R	1.54E-34	**9.19E-34**	1.26E-33
	300000	PSO-NMM-16	5.93E-34	5.33E-04	3.04E-03
		DE-R	1.54E-34	**1.19E-33**	1.60E-33
Combustion application	50000	PSO-12	3.08E-11	2.76E-08	4.34E-08
		PSO-15	1.01E-11	4.88E-09	8.68E-09
		DE-R	3.68E-21	**3.56E-18**	9.62E-18
	100000	PSO-16	3.99E-12	2.22E-09	4.21E-09
		DE-R	9.83E-37	**4.41E-22**	1.87E-21
	250000	PSO-NMM-12	1.45E-33	4.69E-17	1.04E-16
		PSO-NMM-15	1.28E-33	6.91E-17	2.37E-16
		DE-R	1.29E-42	**2.43E-23**	1.17E-22
	300000	PSO-NMM-16	4.69E-34	5.05E-17	1.09E-16
		DE-R	2.49E-44	**7.80E-24**	3.34E-23

by using a mixing scheme of two mutation operations and a restart technique for solving the nonlinear systems. The designed algorithm has the advantage of integrating both the global and local search techniques to balance the exploration and exploitation. It can successfully solve all ten selected test problems with varying degrees of difficulty and outperforms the two basic differential evolution algorithms using the recommended setting from the literature. It also outperforms the compared methods recently developed in the literature. This performance results from the proper modification to the basic DE algorithms and shows that the DE algorithm with the restart technique is a promising tool for solving complex systems of nonlinear equations. Future study could investigate on designing and applying the differential evolution algorithms to more complicated nonlinear problems in high dimensions such as those derived from difficult nonlinear ODEs and PDEs, and those from learning models of artificial neural networks.

Acknowledgements

The authors would like to thank the Department of Mathematics, Faculty of Science, Khon Kaen University, Thailand.

References

[1] Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information sciences*, 237, 82–117 .

[2] Siddique, N., & Adeli, H. (2015). Nature inspired computing: An overview and some future directions. *Cogn Comput*, 7, 706–714.

[3] Nanda, S. J., & Panda, G. (2014). A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation*, 16, 1—18.

[4] José-García, A., & Gómez-Flores, W. (2016). Automatic clustering using nature-inspired metaheuristics: A survey. *Applied Soft Computing*, 41, 192—213.

[5] Hamm, L., Brorsen, B. W., & Hagan, M. T. (2007). Comparison of Stochastic Global Optimization Methods to Estimate Neural Network Weights. *Neural Process Lett*, 26, 145—158.

[6] Piotrowski, A. P. (2014). Differential evolution algorithms applied to neural network training suffer from stagnation. *Applied Soft Computing*, 21, 382—406.

[7] Raja, M. A. Z., Umar, M., Sabir, Z., Khan, J. A., & Baleanu, D. (2018). A new stochastic computing paradigm for the dynamics of nonlinear singular heat conduction model of the human head. *Eur. Phys. J. Plus*, 133(364), DOI 10.1140/epjp/i2018-12153-4.

[8] Sabir, Z., Manzar, M. A., Raja, M. A. Z., Sheraz, M., & Wazwaz, A. M. (2018). Neuroheuristics for nonlinear singular Thomas-Fermi systems. *Applied Soft Computing*, 65, 152-169.

[9] Raja, M. A. Z., Shah, Z., Manzar, M. A., Ahmad, I., Awais, M.,& Baleanu, D. (2018). A new stochastic computing paradigm for nonlinear Painlevé II systems in applications of random matrix theory. *Eur. Phys. J. Plus*, 133(254), DOI 10.1140/epjp/i2018-12080-4.

[10] Raja, M. A. Z., Zameer, A., Kiani, A. K., Shehzad, A., & Khan, A. R. (2018). Nature-inspired computational intelligence integration with Nelder-Mead method to solve nonlinear benchmark models. *Neural Comput & Applic*, 29, 1169-1193.

[11] Ahmad, I., Zahid, H., Ahmad, F., Raja, M. A. Z., & Baleanu, D. (2019). Design of computational intelligent procedure for thermal analysis of porous fin model. *Chinese Journal of Physics*, 59, 641-655.

[12] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92), 577–593.

[13] Martínez, J. M. (1994). Algorithms for solving nonlinear systems of equations. *In* : E. Spedicato, ed. *Algorithms for Continuous Optimization-The state of the art*. Kluwer Academic Publishers, London, 81–108.

[14] Kelley, C. T. (1995). *Iterative methods for solving linear and nonlinear equations*. SIAM, Philadelphia.

[15] Dennis, J. E., & Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, Philadelphia.

[16] Ortega, J. M., & Rheinboldt, W. C. (2000). *Iterative solution of nonlinear equations in several variables*. SIAM, Philadelphia.

[17] Kelley, C. T. (2003). *Solving nonlinear equations with Newton's method*. SIAM, Philadelphia.

[18] Karr, C. L., Weck, B., & Freeman, L. M. (1998). Solutions to systems of nonlinear equations via a genetic algorithm. *Eng. Appl. Artif. Intell.*, 11(3), 369–375.

[19] Grosan, C., & Abraham, A. (2008). A new approach for solving nonlinear equations systems. *IEEE Transactions on Systems Man and Cybernetics-Part A: Systems and Humans*, 38(3), 698–714.

[20] Hirsch M. J., Pardalos, P. M., & Resende, M. G. C. (2009). Solving systems of nonlinear equations with continuous GRASP. *Nonlinear Analysis: Real World Applications*, 10, 2000–2006.

[21] Jaberipour, M., Khorram, E., & Karimi, B. (2011). Particle swarm algorithm for solving systems of nonlinear equations. *Computers and Mathematics with Applications*, 62(2), 566–576.

[22] Pourjafari, E., & Mojallali, H. (2012). Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. *Swarm and Evolutionary Computation*, 4, 33–43.

[23] Oliveira, H. A., & Petraglia, A. (2013). Solving nonlinear systems of functional equations with fuzzy adaptive simulated annealing. *Applied Soft Computing*, 13, 4349–4357.

[24] Raja, M. A. Z., Kiani, A. K., Shehzad, A., & Zameer, A. (2016). Memetic computing through bio-inspired heuristics integration with sequential quadratic programming for nonlinear systems arising in different physical model. *SpringerPlus*, 5:2063, DOI 10.1186/s40064-016-3750-8.

[25] Zhang, X., Wan, Q., & Fan, Y. (2019). Applying modified cuckoo search algorithm for solving systems of nonlinear equations. *Neural Comput & Applic*, 31, 553–576.

[26] Storn, R., & Price, K. (1995). *Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces*. Technical Report TR-95-012, ICSI, Berkeley.

[27] Storn, R., & Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *J Glob Optim*, 11(4), 341–359.

[28] Storn, R. (2008). Differential evolution research-Trends and open questions. *In* : U. K. Chakraborty, ed. *Advances in Differential Evolution*. Springer, Berlin, 1–31.

[29] Neri, F., & Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artif Intell Rev*, 33, 61–106.

[30] Das, S., & Suganthan, P. N. (2011). Differential evolution: a survey of the state-of-the-art. *IEEE Trans Evol Comput*, 15(1), 4–31.

[31] Lampinen, J., & Zelinka, I. (2000). On stagnation of the differential evolution algorithm. *Proceedings of the 6th international Mendel conference on soft computing*, 76–83.

[32] Gämperle, R., Müller, S. D., & Koumoutsakos, P. (2002). A parameter study for differential evolution. *Proceedings of the conference in neural networks and applications (NNA), fuzzy sets and fuzzy systems (FSFS) and evolutionary computation (EC), WSEAS*, 293–298.

[33] Fan, H. Y., & Lampinen, J. (2003). A trigonometric mutation operation to differential evolution. *J Glob Optim*, 27(1), 105–129.

[34] Das, S., Konar, A., & Chakraborty, U. K. (2005). Two improved differential evolution schemes for faster global search. *ACM-SIGEVO Proceedings of genetic and evolutionary computation conference*, 991–998.

[35] Kaelo, P., & Ali, M. M. (2007). Differential evolution algorithms using hybrid mutation. *Comput Optim Appl*, 37, 231–246.

[36] Das, S., Abraham, A., Chakraborty, U. K., & Konar, A. (2009). Differential evolution with a neighborhood-based mutation operator. *IEEE Trans Evol Comput*, 13(3), 526–553.

[37] Neri, F., & Tirronen, V. (2009). Scale factor local search in differential evolution. *Memet Comput J*, 1(2), 153–171.

[38] Qin, A. K., & Suganthan, P. N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. *Proceedings of the IEEE congress on evolutionary computation*, 1785–1791.

[39] Salman, A., Engelbrecht, A. P., & Omran, M. G. (2007). Empirical analysis of self-adaptive differential evolution. *Eur J Oper Res*, 183(2), 785–804.

[40] Soliman, O. S., & Bui, L. T. (2008). A self-adaptive strategy for controlling parameters in differential evolution. *Proceedings of the IEEE congress on evolutionary computation*, 2837–2842.

[41] Yang, Z., Tang, K., & Yao, X. (2008). Self-adaptive differential evolution with neighborhood search. *Proceedings of the world congress on computational intelligence*, 1110–1116.

[42] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans Evol Comput*, 13(2), 398–417.

[43] Zaharie, D. (2002). Critical values for control parameters of differential evolution algorithm. *Proceedings of the 8th international Mendel conference on soft computing*, 62–67.

[44] Zaharie, D. (2003). Control of population diversity and adaptation in differential evolution algorithms. *Proceedings of the 9th international Mendel conference on soft computing*, 41–46.

[45] Verschelde, J., Verlinden, P., & Cools, R. (1994). Homotopies exploiting newton polytopes for solving sparse polynomial systems. *SIAM J. Numer. Anal.*, 31, 915–930.

[46] Morgan, A., & Shapiro, V. (1987). Box-bisection for solving second-degree systems and the problem of clustering. *ACM Transaction on Mathematical Software*, 13, 152–167.

[47] Pramanik, S. (2002). Kinematic synthesis of a six-member mechanism for automotive steering. *ASME Journal of Mechanical Design*, 124, 642–645.

[48] Meintjes, K., & Morgan, A. (1990). Chemical equilibrium systems as numerical test problems. *ACM Transaction on Mathematical Software*, 16, 143–151.

[49] More, J., Garbow, B., & Hillstrom, K. (1981). Testing unconstrained optimization software. *ACM Transaction on Mathematical Software*, 7, 17–41.

[50] Bongartz, I., Conn, A., Gould, N., & Toint, Ph. (1995). CUTE: constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21, 123–160.

Jeerayut Wetweerapong *completed M.Sc. degree in Mathematics from West Virginia University, US in 1995 and Ph.D. degree in Mathematics from Khon Kaen University, Thailand in 2012. He has been teaching and doing research in field of scientific computing and optimization at Department of Mathematics, Khon Kaen University.*

http://orcid.org/0000-0001-5053-3989

Pikul Puphasuk *completed M.Sc. degree in Mathematics from Khon Kaen University, Thailand in 2002 and Ph.D. degree in Applied Mathematics from Suranaree University of Technology, Thailand in 2009. She is an assistant professor at Department of Mathematics, Khon Kaen University. Her research areas include computational sciences, numerical analysis and optimization.*

http://orcid.org/0000-0001-9069-1703