Modified Differential Evolution Algorithm for Parameter Estimation in Mathematical Models

Musrrat Ali and Millie Pant
Department of Paper Technology
Indian Institute of Technology Roorkee,
Saharanpur campus, Saharanpur, India.
{musrrat.iitr, millidma}@gmail.com

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, Washington, USA.
ajith.abraham@ieee.org

Vaclav Snasel
VSB Technical University of Ostrava, Czech Republic
vaclav.snasel@vsb.cz

***Abstract*— the parameter estimation or identification problem, which frequently arises, while developing the mathematical models, may be formulated as a nonlinear global optimization problem. Here the objective is to find the set of parameters to minimize the function quantifying the goodness of the fit subject to the system dynamics. The mathematical model of the problem is often multimodal in nature and requires a suitable global optimization method for its solution. In the present study we show the application of a Modified Differential Evolution (MDE) for solving parameter estimation problem. We have considered two test cases. A comparison of numerical results with other algorithms shows the competence of MDE over basic DE and other methods.**

Keywords: Differential Evolution, parameter estimation, mathematical models, opposition-based learning.

I. INTRODUCTION

MATHEMATICAL models used in applied research (biology, physics, economics, etc.) are often defined by a system of ordinary differential equations. It should be noted that in general the solution of such a system need not be an elementary function. Based on experimental data obtained, the parameters of the mathematical models have to be determined. This problem is known in literature as the parameter identification problem. The classical methods for dealing with parameter identification problem include the quasi-linearization method [1, 2] and smoothing the data method [2–4] etc. In the recent years, evolutionary algorithms (EA) like genetic algorithms (GA) [5, 6] have also been used for solving such problems. One of the main advantages to using these techniques is that they require no knowledge or gradient information about the response surface. In the present study, we have used Differential Evolution [7] and its modified version called Modified Differential Evolution (MDE) for solving parameter identification problem.

DE has emerged as a popular choice for solving global optimization problems [8]. Using a few parameters, DE exhibits an overall excellent performance for a wide range of benchmark as well as real-world application problems [9] and has shown a better performance in comparison to other EA. Nevertheless, like most of the EA in their basic forms DE also suffers from certain drawbacks like slow and/ or premature convergence. While slow convergence implies higher computational time, premature convergence leads to suboptimal solution. Therefore efforts are needed to improve its performance.

MDE algorithm developed by the authors [10] is a simple and modified version of basic DE algorithm. It gave a good performance for benchmark problems in terms of solution quality as well as convergence rate. Motivated by its success, in the present study, we have used it for solving parameter identification problem. Two test cases are considered and results obtained by MDE are compared with basic DE and some other algorithms given in literature.

The remainder of the paper is structured as follows. Introduction of parameter identification problem is given briefly in section II. Section III describes the algorithms used in this study (basic DE and MDE). Procedure to solve parameter identification problem is given in Section IV. Experimental settings are given in Section V. Numerical examples are listed in Section VI. Section VII provides comparisons of results. Finally the conclusions based on the present study are drawn in section VIII.

II. PARAMETER IDENTIFICATION PROBLEM

Let us assume that the mathematical model is defined either by a differential equation of the first order

$$\frac{dy}{dx} = f\left(t, y(t), p\right) \tag{1}$$

or a differential equation of the second order

$$\frac{d^2y}{dx^2} = f\left(t, y(t), y'(t), p\right) \tag{2}$$

Where $p = (p_1,...,p_n)^T$ is the vector of n unknown real parameters. Also given is the experimental data (t_i, y_i), $i= 1, \ldots, m$ where t_i represents the values of the independent variable and, y_i, the measured values of the corresponding dependent variable. Usually we have n<<m. With the given data one has to estimate the optimal parameter vector, p^*, and the optimal initial condition for the differential equation (1) or (2) such that

978-1-4244-6588-0/10/$25.00 ©2010 IEEE

$$F(p^*) = \min_{p \in R^n} F(p),\ F(p) = \sum_{i=1}^{m} \left[y(t_i, p) - y_i \right]^2 \quad (3)$$

where $y(t_i ; p)$ is the solution of Eq. (1) or (2).

In this paper we shall consider the problem of determining the optimal parameters of a mathematical model defined by a differential equation of the second order. The initial condition of the model is generally not known. It is not recommended to take the first data for that purpose because the error it contains is not known [11]. The problem of finding the optimal initial condition in the mathematical model could be posed in the following way [1, 2]:

Find the minimum of the functional

$$\phi(\mu, \nu) = \sum_{i=1}^{m} \left[y_{\mu,\nu}(t_i; p^*) - y_i \right]^2 \quad (4)$$

Where the function $y_{\mu,\nu}$ is the solution of the Cauchy problem

$$y'' = f\left(t, y(t), y'(t), p^*\right),\ y'(t_1) = \mu,\ y(t_1) = \nu \quad (5)$$

III. ALGORITHMS

In this section we will describe briefly the working of basic differential evolution (DE) and modified differential evolution (MDE).

A. Differential Evolution (DE)

Throughout the present study we shall follow *DE/rand/1/bin* version of DE and shall refer to it as basic version. This particular scheme is briefly described as:

DE starts with a population of NP candidate solutions: $X_{i,G}$, $i = 1, \ldots, NP$, where the index i denotes the i^{th} individual of the population and G denotes the generation to which the population belongs. The three main operators of DE; mutation, crossover and selection are described as follows:

Mutation: The mutation operation of DE applies the vector differentials between the existing population members for determining both the degree and direction of perturbation applied to the individual subject of the mutation operation. The mutation process at each generation begins by randomly selecting three individuals $X_{r1,G}$, $X_{r2,G}$ and $X_{r3,G}$,in the population set of (say) *NP* elements. The i^{th} perturbed individual, $V_{i,G+1}$, is generated based on the three chosen individuals as follows:

$$V_{i,G+1} = X_{r3,G} + F * (X_{r1,G} - X_{r2,G}) \quad (6)$$

Where, $i = 1 \ldots NP$, $r_1, r_2, r_3 \in \{1 \ldots NP\}$ are randomly selected such that $r_1 \neq r_2 \neq r_3 \neq i$, $X_{r3,G}$ is known as the base vector and F is the control parameter such that $F \in [0, 1]$.

Crossover: once the mutant vector is generated, the perturbed individual, $V_{i,G+1} = (v_{1,i,G+1}, \ldots, v_{n,i,G+1})$, and the current population member, $X_{i,G} = (x_{1,i,G}, \ldots, x_{n,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or "trial" vectors,$U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{n,i,G+1})$, as follows:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} \text{ if } rand_j \leq C_r \vee j = k \\ x_{j,i,G} \quad \text{otherwise} \end{cases} \quad (7)$$

Where, $j = 1 \ldots n$, $k \in \{1, \ldots, n\}$ is a random parameter's index, chosen once for each i. The crossover rate, $Cr \in [0, 1]$, is set by the user.

Selection: The selection scheme of DE also differs from that of other EAs. The population for the next generation is selected from the individual in current population and its corresponding trial vector according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} \text{ if } f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} \quad \text{otherwise} \end{cases} \quad (8)$$

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. In DE trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation.

B. Modified Differential Evolution (MDE)

The basic operators of MDE are same as basic DE but it differs from it according to the following three points:

1. *Initialization phase:* MDE utilizes opposition based learning (OBL) method while DE uses uniform random numbers for initialization of population. In MDE, we randomly construct a population P of NP individuals, dimension of each vector being n, using the following rule:
   $X_{i,j} = X_{min,j} + rand(0, 1)(X_{max,j} - X_{min,j})$,
   Where $X_{min,j}$ and $X_{max,j}$ are lower and upper bound for j^{th} component respectively and rand(0,1) is a uniform random number between 0 and 1.
   We construct another population OP of NP individuals using the following rule:
   $$y_{i,j} = X_{min,j} + X_{max,j} - P_{i,j}$$
   Where $P_{i,j}$ are the points of population P.
   Now, construct initial population S taking NP best individuals from union of these two populations.
2. *Mutation Phase*: In mutation step MDE uses best individual of three points as base vector in contrast to basic DE where anyone of the individual is taken as base vector. This mutation step of MDE is performed as follows:
   Select randomly three distinct individuals X_{r1}, X_{r2} and X_{r3} from population S and select the one having the best fitness (say $X_{r1,G}$) and denote it as X_{tb}. Now perform mutation as follows:
   $$V_i = \boldsymbol{X_{tb}} + F \times (X_{r2} - X_{r3}) \quad (9)$$
3. *Population Set*: Finally, MDE differs from DE in maintaining the population. While DE works on two

populations; current population and advanced population (for next generation), in MDE only one population is considered in which all the operations takes place.

IV. SOLVING PARAMETER IDENTIFICATION PROBLEM USING DE/MDE

In the parameter identification problem the parameters and initial conditions which are to be determined and optimized are randomly generated within the given range. After each iteration, the new values y (t_i ; p) for each possible solution of p are determined using the Runge–Kutta method and fitness of each individual is determined using Equation (3). The DE/MDE then consists of the following steps:

1. Initialize a population of individual (possible solutions of p).
2. Find the values y (t_i ; p) for each individual, p, in the population using the Runga–Kutta method.
3. Evaluate the fitness of each chromosome, p, in the population using Eq. (3).
4. Create new offspring by using DE/MDE operators.
5. Find the new values y (t_i ; p) for each new offspring, p, in the population using the Runga–Kutta method.
6. Evaluate the fitness of each new offspring, p, using Eq. (3) and insert them into the population.
7. If the stopping criterion is satisfied, then stop and return the best individual, otherwise, go to step 4.

V. EXPERIMENTAL SETUP

In order to make a fair comparison we have used C++ rand ( ) function to generate initial population for both DE and MDE algorithms with same seed. The number of individuals in the population is taken as *10*n*, where n is the dimension of the problem. Values of scaling factor F and crossover probability C_r are taken as 0.5 each. Both the algorithms are executed on a PIV PC, using DEV C++, thirty times for each problem. We have compared the algorithms taking two criteria (i) fitness and (ii) number of function evaluations. For fitness evaluation, the termination criterion is taken as maximum number of generations (100, 300, and 500). For function evaluation termination criterion is $|f_{\max} - f_{\min}| \le \varepsilon = 10^{-4}$.

Where $f_{\max}$ and $f_{\min}$ are objective function values at worst and best point of the population.

VI. NUMERICAL EXAMPLES

Two test cases considered in the present study are (1) Enzyme effusion problem and (2) No load loss problem.

A brief description of the problems is given here:

1. Enzyme effusion problem [2]

$$y_1' = p_1(27.8 - y_1) + \frac{p_4}{2.6}(y_2 - y_1) + \frac{4991}{t\sqrt{2\pi}}\exp\left(-0.5\left(\frac{\ln(t) - p_2}{p_3}\right)^2\right)$$

$$y_2' = \frac{p_4}{2.7}(y_1 - y_2)$$

According to the Table 1, one has to estimate the parameter values of p_1, p_2, p_3, p_4 in addition to the initial condition of y_1, y_2.

2. The analytical solution of mathematical model containing a second order ordinary differential equation (ODE) can be stated by

$$y(t, p) = p_1 \exp(p_3 t) + p_2 \exp(p_4 t)$$

According to the given data (t_i ,y_i), i = 1,. . . ,m (see Table 2), one has to estimate the parameter values p_1, p_2, p_3, p_4 of the function.

TABLE I. DATA FOR ENZYME EFFUSION PROBLEM.

t	y_1	t	y_1	t	y_1	t	y_1
0.1	27.8	21.3	331.9	42.4	62.3	81.1	23.5
2.5	20.0	22.9	243.5	44.4	58.7	91.1	24.8
3.8	23.5	24.9	212.0	47.9	41.9	101.9	26.1
7.0	63.6	26.8	164.1	53.1	40.2	115.4	33.3
10.9	267.5	30.1	112.7	59.0	31.3	138.7	17.8
15.0	427.8	34.1	88.1	65.1	30.0	163.2	16.8
18.2	339.7	37.8	76.2	73.1	30.6	186.7	16.8

TABLE II. DATA FOR PROBLEM 2.

t	-1	-2/3	-1/3	0	1/3	2/3	1
y	64.0	66.0	69.5	74.0	80.8	91.0	103.5

VII. NUMERICAL RESULTS AND COMPARISONS

A. Problem 1: Enzyme effusion

a. MDE Vs DE

Table III gives the results for problem 1 taken by fixing the maximum number of generation 100, 300, 500. From last column of this Table which gives the sum of square error (SSE), we see that both algorithms give almost similar results. But if we run both the algorithm to achieve an accuracy 10^{-04} then MDE converge faster than DE. It finds out the result up to desired accuracy in lesser number of function evaluations and also in lesser time (Table IV). A performance curve is given in Figure 1. Plot of experimental data and data obtained by MDE after 500 generation is given in Figure 2.

b. MDE Vs other algorithms given in literature

For this comparison we ran MDE for 100, 300 and 500 generations and the corresponding results are stored in Table V. From this Table it is clear that MDE outperforms all other algorithms.

TABLE IV. RESULTS OBTAINED USING DE/MDE FOR ENZYME EFFUSION PROBLEM TO ACHIEVE THE ACCURACY 10^{-04}

Algorithm	*Time(sec)*	*NFE*
DE	3753	18480
MDE	2407	11760

TABLE III. RESULTS OBTAINED USING DE/MDE FOR ENZYME EFFUSION PROBLEM.

Algorithm	p_1	p_2	p_3	p_4	y_1	y_2	*Gen*	*SSE*
DE	23.4543	37.3712	0.272592	2.65286	0.364299	0.206415	100	4048.69
DE	23.2648	37.0019	0.272790	2.65350	0.364828	0.209529	300	4044.50
DE	23.2548	37.0000	0.272819	2.65363	0.364895	0.209288	500	4044.48
MDE	23.2437	37.0013	0.272873	2.65371	0.364971	0.209080	100	4044.49
MDE	23.2543	37.0000	0.272819	2.65363	0.364895	0.209285	300	4044.48
MDE	23.2543	37.0000	0.272819	2.65363	0.364895	0.209285	500	4044.48

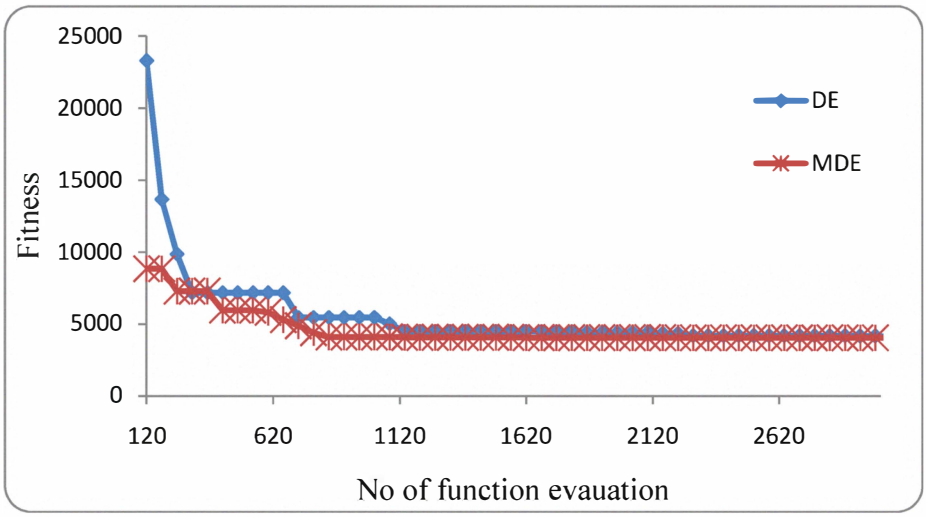

Fig 1: performance curves of Enzyme effusion problem.

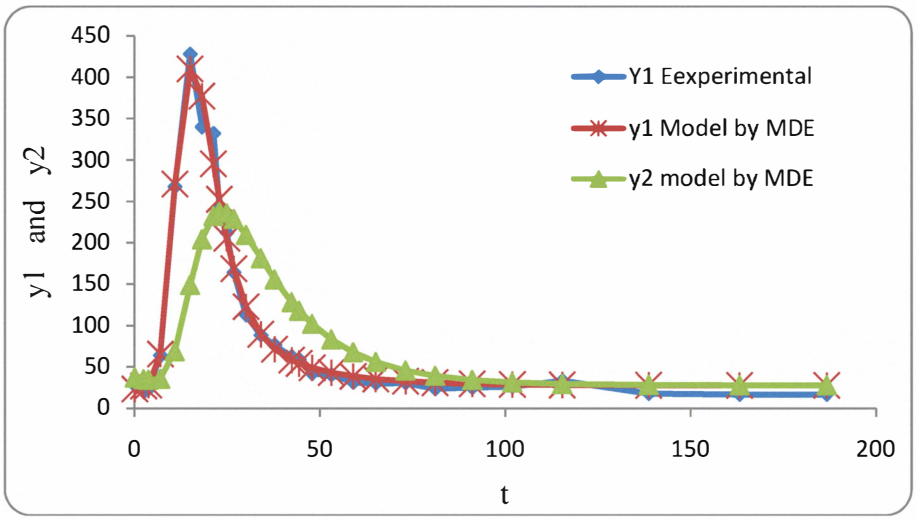

Fig 2: Enzyme effusion model at 500 generation.

TABLE V. COMPARISON OF ENZYME EFFUSION RESULTS BY MDE WITH OTHER ALGORITHMS.

Algorithm	p_1	p_2	p_3	p_4	y_1	y_2	*Gen*	*SSE*
MDE	23.2437	37.0013	0.272873	2.65371	0.364971	0.209080	100	4044.49
MDE	23.2543	37.0000	0.272819	2.65363	0.364895	0.209285	300	4044.48
MDE	23.2543	37.0000	0.272819	2.65363	0.364895	0.209285	500	4044.48
Ref. [2]	0.31900	2.70100	0.41900	0.10310	22.00000	39.00000	--	5076.60
Ref. [5]	0.31938	2.70104	0.38920	0.07819	21.00000	38.75000	100	5229.73
Ref. [5]	0.30501	2.69865	0.40052	0.11663	22.02000	39.44000	200	4547.34
Ref. [5]	0.28452	2.67169	0.39268	0.16144	23.99000	40.14000	500	4068.38
Ref. [6]	0.24540	2.60920	0.33260	0.32170	22.00500	38.60800	100	4431.45
Ref. [6]	0.25610	2.62690	0.34490	0.26960	22.04300	38.40300	200	4193.92
Ref. [6]	0.26190	2.63360	0.35240	0.25750	21.98600	38.70400	300	4136.73

B. Problem 2

a. MDE Vs DE

Table VI gives the results for problem 2 taken by fixing the maximum number of generation 100, 300, 500. From last column of this Table which gives the sum of square error (SSE) that both algorithm gives almost similar results. But if we run both the algorithm to achieve an accuracy 10^{-04} then it can be seen that MDE converge faster than DE. It finds out the result up to desired accuracy in lesser number of function evaluations and computational time (Table VII). A performance curve is given in Figure 3. Plot of experimental data and data obtained by MDE after 500 generation is given in Figure 4.

b. MDE Vs other algorithms given in literature

We compared the performance of MDE with the results available in literature where it has been solved using Genetic Algorithm (GA). For this comparison we executed MDE for 100, 300 and 500 generation and stored the results in Table VIII. From this table it can be clearly observed that MDE outperforms the other algorithms.

TABLE VII. RESULTS OBTAINED USING DE/MDE FOR PROBLEM 2 TO ACHIEVE THE ACCURACY 10^{-04}

Algorithm	*Time(Sec)*	*NFE*
DE	0.30	13564
MDE	0.10	8472

TABLE VI. RESULTS OBTAINED USING DE/MDE FOR PROBLEM 2.

Algorithm	p_1	p_2	p_3	p_4	*Gen*	*SSE*
DE	36.0250	38.1356	0.693334	-0.189512	100	0.325970
DE	34.1450	39.9841	0.715547	-0.170192	300	0.296257
DE	34.1266	40.0000	0.715740	-0.169867	500	0.296017
MDE	35.4350	38.6848	0.698735	-0.182301	100	0.314189
MDE	34.1266	40.0000	0.715740	-0.169867	300	0.296017
MDE	34.1266	40.0000	0.715740	-0.169867	500	0.296017

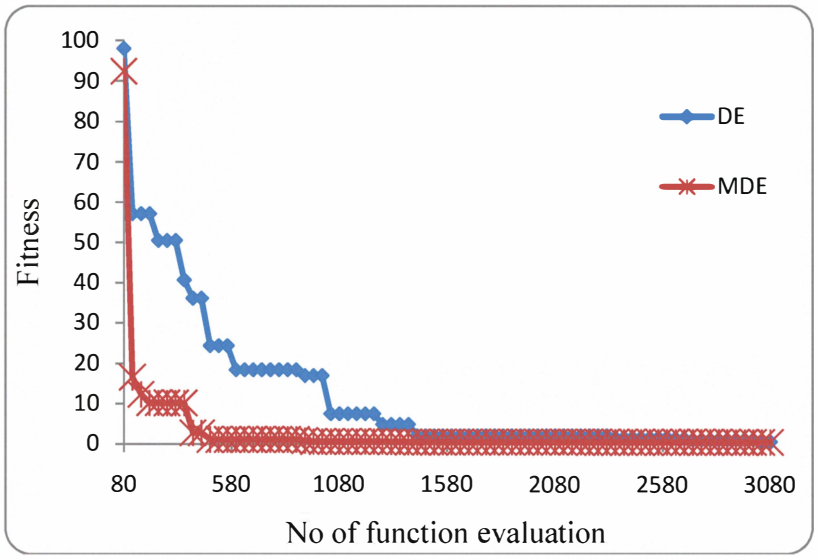

Fig 3: performance curves of problem 2.

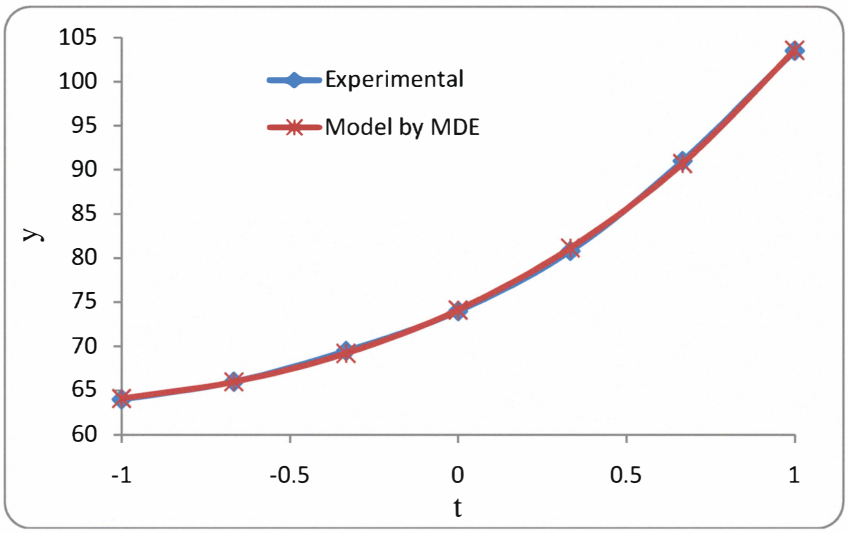

Fig 4: y Vs t of problem 2 at 500 generation.

TABLE VIII. COMPARISON OF PROBLEM 2 RESULTS BY MDE WITH OTHER ALGORITHMS.

Algorithm	p_1	p_2	p_3	p_4	*Gen*	*SSE*
MDE	35.4350	38.6848	0.698735	-0.182301	100	0.314189
MDE	34.1266	40.0000	0.715740	-0.169867	300	0.286017
MDE	34.1266	40.0000	0.715740	-0.169867	500	0.276017
Ref. [2]	43.9805	30.1752	0.60735	-0.28653	--	0.352
Ref. [5]	43.2233	30.8774	0.6170	-0.2812	100	0.4396
Ref. [5]	40.8112	33.3234	0.6400	-0.2464	200	0.3859
Ref. [5]	37.7414	36.3533	0.6753	-0.2123	500	0.3347
Ref. [6]	42.5032	31.5469	0.6265	-0.2749	100	0.3204
Ref. [6]	41.4371	32.6713	0.6346	-0.2563	200	0.2827

VIII. DISCUSSION AND CONCLUSIONS

In this paper we have investigated the performance of DE and MDE, a modified version of DE for solving parameter identification problem. The simulation of results showed that MDE is quite competent for solving such problems in lesser number of function evaluations and time without compromising with the quality of solution.

References

[1] R. Scitovski, D. Juki_c, I. Urbiha, "Solving the parameter identification problem by using TLp spline", Math. Commun. (Suppl.) 1, 2001, pp. 81–91.

[2] R. Scitovski, D. Juki_c, "A method for solving the parameter identification problem for ordinary differential equations of the second order", Appl. Math. Comput. 74 , 1996, pp. 273–291.

[3] R. Gali_c, R. Scitovski, T. Maro_sevic, "Application of the moving least square method in solving the parameter identification problem of a mathematical model (in Croatian)", in: T. Hunjak, Lj. Marti_c, L. Nerali_c (Eds.), Proceedings of the 4th Conference on Operational Research, Zagreb, 1994, pp. 181–191.

[4] R. Gali_c, R. Scitovski, T. Maro_sevic, D. Juki_c, "Optimal initial condition in mathematical model (in Croatian)", in: T. Hunjak, Lj. Marti_c, L. Nerali_c, (Eds.), Proceedings of the 5th Conference on Operational Research, Zagreb, 1995, pp. 62–71.

[5] E.K. Nyarko, R. Scitovski, "Solving the parameter identification problem of mathematical models using genetic algorithm", Applied mathematics and Computation 153 (3), 2004, pp. 651–658.

[6] M.A.Kahlik, M.sherif, S.Saraya and F Areed, "Parameter identification problem: Real coded GA approach", Applied mathematics and Computation 187, 2007, pp. 1495–1501.

[7] R. Storn and K. Price, "Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces", Technical Report TR-95-012, Berkeley, CA, 1995.

[8] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," J. Global Opt., vol. 11(4), 1997, pp. 341–359.

[9] K. V. Price, R. M. Storn, and J. A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Berlin, Germany: Springer-Verlag, 2005.

[10] M.Ali, M.Pant and A.Abraham, "A Modified Differential Evolution Algorithm and Its Application to Engineering Problems", 2009 International Conference of Soft Computing and Pattern Recognition, Malaysia, 2009, pp. 196-201.

[11] J.M. Varah, "A spline least squares method for numerical parameter estimation in differential equations", SIAM J. Sci. Stat. Comput. 3, 1982, pp. 28–46.