Automatic Parameter Calibration for Frequency Modulated Sound Synthesis with a faster Differential Evolution

SOHAN GHORAI , SUMIT KUMAR PAL
Department of Electronics & Communication Engineering
Narula Institute of Technology,
Kolkata-700109, INDIA
e-mail: mr.sohan.ghorai@ieee.org, pal.sumit99@gmail.com

Abstract: — Frequency Modulated sound synthesis is a technique widely used for replicating sound of a natural music instruments to use it in a computer sound synthesizer. Many sound synthesis technique have been successful in reproducing the sounds of musical instruments. Sound synthesis techniques require parameter calibration. However, this task can be difficult and time consuming because of the non-linear parameter space. Two difficult challenges have to handle for synthesizing sound, that is-proper parameter extraction from a difficult search space and search time, which matters for a real time application. This article presents an application of stagnation adaptive DE scheme for faster parameter identification of a FM synthesized sound to match an unknown target sound. The simulation result shows that DE can perfectly reproduce the sound signal over other algorithm and also have a very faster convergence rate than classic DE.

Key Words: — differential evolution, sound synthesis, frequency modulation, optimization, parameter identification

1 Introduction

Replicating the sounds of musical instruments using parametric synthesis technique is a problem frequently addressed in the field of computer music. The success of any particular synthesis algorithm is dependent on the selection of suitable controls and synthesis parameters. It is very difficult and time consuming for a particular synthesis algorithm to find a best set of parameter manually. Thus, diverse optimization methods have been used for automatic calibration, such as HMM [3], neural network [4], cellular automata [5], particle swarm [1], genetic algorithm [7] etc. Another simple method to create complex audio spectra was proposed by John Chowing in 1973, which uses a six dimension frequency modulated signal equation [1]. In this technique, frequency of a carrier oscillator is altered in accordance with the amplitude of a modulating signal [1]. As shown in chowning's original paper, and in much of the literature that followed, FM synthesis made it possible to create complex spectra with only a limited number of oscillators. FM sound synthesizer is a six-dimension problem [12], which aims to optimize a set of parameters to match the target sounds [16]. However, it is an expensive task to achieve desired target sound through manual or automatic adjustment of parameter values because of non-linear nature of target sound spectrum [9]. Several evolutionary algorithms taken this problem as an unconstrained optimization problem and used to extract parameters. Real coded GA [16] have been extensively used for estimating the parameters of FM synthesizer. Simulation studies of different variant of GA like one of the proposed gradual distributed real coded GA by Herreta and Lozano [15] [16] found better result in finding parameters. An extensive study on evolutionary computation applied to FM audio synthesis parameter [16] extraction done by Mitchell [9].

Particle Swarm Optimization (PSO), a population based stochastic optimization technique [13] proposed by Eberhart and Kennedy was applied on this problem. Like GA, it does not have any biological operator such as crossover and mutation. The potential solution called particles, fly through the problem space by tracking the coordinates in the problem space which are associated with the best fitness value.

Differential Evolution (DE), developed by Storn and Price [10], is a robust yet simple global stochastic optimization algorithm. Several experimental studies shows that DE has power to dominate over many algorithms in terms of robustness and convergence speed [10] [11]. Some recent study shows that DE can

perfectly reconstruct a signal without any error [16] than any other tested so far.

The rest of the paper introduces a study to improve the speed of convergence and error variance of classic DE algorithm to meet faster computational efficiency.

2 Problem Definition of FM Sound Synthesis

Frequency Modulated (FM) sound synthesis provides an efficient solution to create complex spectra of sound wave with less computational burdens. The original six-dimension problem was originated from a simple frequency modulation signal expression as follows. Let us consider that f_c and f_m are the carrier and modulator frequencies, respectively. M is the modulation index and A represent the output amplitude of the signal. The function for simple FM can be expressed as follows [9]:

$$y(t) = Asin(w_c + M\sin(w_m t)) \quad (1)$$

$$w_m = 2\pi f_m$$
$$w_c = 2\pi f_c$$
$$M = \frac{d}{f_m}$$

where $y(t)$ is the modulated carrier amplitude with respected to time. The modulating signal varies the carrier signal frequency of FM in the range expressed by peak frequency deviation d , which is the product of modulation index M and modulating frequency expressed by f_m [9][16]. If modulation index $M = 0$, there will be no modulation of carrier signal frequency, and therefore the generated signal will be the sine wave having frequency same as carrier frequency f_c .

The FM sound synthesizer equation is formed by repetitive inserting of another carrier modulator within the modulating sine wave of equation (1). Thus a six-dimensional problem [8] can be expressed as follows.

$$y(t) = a_1\sin(\omega_1 t\theta + a_2\sin(\omega_2 t\theta + a_3\sin(\omega_3 t\theta))) \quad (2)$$

The FM sound synthesis problem aims to tune the parameters of a FM synthesizer consisting of six variables [$a_1, \omega_1, a_2, \omega_2, a_3, \omega_3$] to match with a target sound. In our experiment, our aim is to optimize the parameters to achieve the following target sound.

$$y_0(t) = 1.0\sin(0.5\, t\theta + 1.5\sin(4.8\, t\theta + 2.0\sin(4.9\, t\theta))) \quad (3)$$

For equation (2) and (3), $\theta = 2\pi/100$ and the parameters for equation (2) are initialized [12] in the range [-6.4, 6.35].

Now, to convert this problem as a minimization problem to fit it with DE, the fitness function is expressed as summation of square errors [12] between the estimated wave (2) and the target wave (3) as follows:

$$f(\vec{X}) = \sum_{t=0}^{100}(y(t) - y_0(t))^2 \quad (4)$$

For perfect reconstruction of the sound wave, fitness function value should be zero.

3 Differential Evolution Algorithm

Differential Evolution algorithm is a population based direct search tool, which supports real number encoding of parameters [17]. It uses tournament selection method and adopts differential strategy, which changes population by differential vector of individuals in mutation scheme. Mutation plays an important role in differential evolution algorithm, that can effectively utilize the characteristics of population distribution by introducing the concept of difference vector and avoid inefficient variation of search performance in GA, while enhancing the search speed [17]. Like other evolutionary algorithms, population vector is randomly generated within search space without the prior knowledge about the nature of search space [9][17]. There are three basic steps, named as mutation, crossover and selection. DE creates NP number of chromosome (fitness vector) having real valued parameters as a population of each generation.

$$\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ..., x_{D,i,G}].$$

where i=[1,2,3, ….. , NP].

$\overline{X}_{i,G}$ is a D-dimensional parameter vector, can be called as an individual. G denotes one generation. At each generation, algorithm uses evolutionary operators like mutation and crossover operation to generate a trial vector $\overline{U}_{i,G}$ for each individual target vector $\overline{X}_{i,G}$, in the current population.

3.1 Mutation:

After initialization of population, DE creates a mutation vector $\overline{V}_{i,G}$ corresponding to each target vector $\overline{X}_{r,G}$ through differential strategy. A scaled difference between two randomly generated parameter vector from current population is added to base vector to produce mutant vector $\overline{V}_{i,G}$. The process can be expressed as

$$\vec{V}_{i,G} = \vec{X}_{r_1^i,G} + F \cdot (\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}).$$

There are several other strategies for generating mutant vector described by Storn and Price can be summarized as follows [10]:

"DE/rand/1": $\overline{V}_{i,G} = \overline{X}_{r_1,G} + F.(\overline{X}_{r_2,G} - \overline{X}_{r_3,G})$

"DE/best/1": $\overline{V}_{i,G} = \overline{X}_{best,G} + F.(\overline{X}_{r_1,G} - \overline{X}_{r_2,G})$

"DE/current to best/1":

$$\overline{V}_{i,G} = \overline{X}_{i,G} + F.(\overline{X}_{best,G} - \overline{X}_{i,G}) + .(\overline{X}_{r_1,G} - \overline{X}_{r_2,G})$$

"DE/Best/2":

$$\overline{V}_{i,G} = \overline{X}_{best,G} + F.(\overline{X}_{r_1,G} - \overline{X}_{r_2,G}) + F.(\overline{X}_{r_3,G} - \overline{X}_{r_4,G})$$

"DE/rand/2":

$$\overline{V}_{i,G} = \overline{X}_{r_1,G} + F.(\overline{X}_{r_2,G} - \overline{X}_{r_3,G}) + F.(\overline{X}_{r_4,G} - \overline{X}_{r_5,G})$$

Where indices named r_1, r_2, r_3, r_4, r_5 are random and mutually exclusive integers taken from the range [1,NP]. F is a factor initialized in the range [0, 2] for scaling differential vectors and $\overline{X}_{best,G}$ is the individual vector having best fitness value in the population at current generation G. In conventional real coded GAs, mutation takes the form of a random perturbation of a fixed type, whose purpose is to prevent premature convergence, but it can sometime generate destructive population. DE avoids this problem by mutating base vectors with population-derived difference vectors. As generations pass, these differences tend to adapt to the natural scaling of the problem. For example, if the population becomes compact in one variable but remains widely dispersed in another, the difference vectors sampled from it will be small in the first variable, yet large in the other. This automatic adaptation significantly improves the convergence of the algorithm.

3.2 Crossover:

To maintain the diversity of the population, a crossover operation comes into play after generating the donor vector through mutation. The donor vector exchanges some fraction of its components with the target vector $\vec{X}_{i,G}$ and forms the *trial* or *mutant* vector

$$\vec{U}_{i,G} = [u_{1,i,G}, u_{2,i,G}, u_{3,i,G}, ..., u_{D,i,G}].$$

Different DE algorithms suggested by Storn and Price generally use two kinds of crossover methods - *exponential* and *binomial* [1].

Most commonly used crossover is binomial crossover.

Binomial crossover is performed on the *D* variables of each vector. The process exchanges gene when a crossover probability (*Cr* value) is less than or equal to a randomly generated number between 0 and 1. The number of parameters that will be inherited from the donor has a binomial distribution due to the selection of rand(0,1) function. The scheme may be expressed as:

$$u_{j,i,G} = \begin{cases} V_{j,i,G} & if\ (rand_{i,j}(0,1) \le Cr\ or\ j = j_{rand} \\ X_{j,i,G} & otherwise \end{cases}$$

Where, as before, $rand_{i,j}[0,1]$ is a uniformly distributed random number, which is called anew for each *j*-th component of the *i*-th parameter vector. $j_{rand} \in [1,2,....,D]$ is a randomly chosen index, which ensures that $\vec{U}_{i,G}$ gets at least one component from $\vec{V}_{i,G}$. It is instantiated once for each vector per generation [1].

3.3 Selection:

To maintain constant population size over subsequent generations, the next step of DE calls for a greedy *selection* process, in which the target or trial vector survives to the next generation (at $G = G+1$) based on the better fitness value among them. The selection process can be expressed as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G} & if\ (U_{i,G}) \le f(X_{i,G}) \\ X_{i,G} & if\ (U_{i,G}) > f(X_{i,G}) \end{cases}$$

Where $f(\overline{X}_{i,G})$ is the objective function that is to be minimize. Thus, it selects the trial vector best suited for that given objective function.

4 Stagnation Adaptive DE (DESSAS) for Faster and Better Optimization

In our algorithm, three factors dominantly affect the performance of DE algorithm.

- Q-number best vector selection for Q_{best} Mutation.

- Stagnation sense window to trigger mutation strategy.
- Stagnation threshold to adapt *F* and *Cr* values.

For each generation, ***Q*** number of best individual is selected from the entire population. Initially ***Q***=20 is taken by default but after 100 generation, if standard deviation of fitness values of Q best vectors is low enough then ***Q*** is changed to 5. Low standard deviation indicates best population loses diversity due to either unimodality of function or local premature convergence.

A stagnation sense window (namely ***t*** window) is used to switch mutation strategy dynamically. After passing 50 generation, ***t*** takes the average value of last 50 generation's minimum fitness value of population.

IF the value of t is nearly equal to the current generation's minimum fitness value, then it sense the situation of stagnation and changes the mutation strategy to DE/Q_{opp}/1 where the base vector is randomly selected from pool of (NP-Q) vectors, which is the set of all present generation population vector excluding ***Q*** best vector. The strategy helpful for finding some of the least explored places in a premature convergence condition.

ELSE randomly selects base vector from ***Q*** and take mutation strategy DE/Q_{best}/1 to speed up the convergence process.

If, after changing the mutation strategy due to stagnation, found no improvement of fitness values, then algorithm starts adapting *F* to increase the difference vector distance for exploiting more space.

For each generation, algorithm checks stagnation by comparing fitness value of current generation with the previous one. If it is reduced, then it stores the successful *F* and *Cr* value and gives reward by increasing stagnation threshold window ***t*** to reduce computational complexity.

In the selection process, we have used binomial crossover scheme.

Following section describes our DE algorithm in steps as follows:

Step 1: Set the generation number G=0 and randomly initialise a population of NP individuals $P_G = \{\vec{X}_{1,G}, ..., \vec{X}_{NP,G}\}$ with each $\vec{X}_{i,G} = [x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, ..., x_{D,i,G}]$ individuals having dimension D and uniformly distributed in the range $[\vec{X}_{\min}, \vec{X}_{\max}]$ .where $\vec{X}_{\min} = \{x_{1,\min}, x_{2,\min}, ..., x_{D,\min}\}$ and $\vec{X}_{\max} = \{x_{1,\max}, x_{2,\max}, ..., x_{D,\max}\}$ with i=[1,2,..,NP]. Then evaluate fitness of each individual.

Step 2: FOR generation g=1:GEN

For each generation, ***Q*** number of best individual is selected from the entire population. Initially ***Q***=20 is taken by default but after 100 generation, if standard deviation of Q best vectors is low enough then ***Q*** is changed to 5. Low standard deviation indicates best population loses diversity due to either unimodality of function or local premature convergence.

A stagnation window (namely ***del***) is initialized with initial value adapted by the algorithm. The adaption is based on the objective function to be minimized. The stagnation window, ***del*** plays critical role in mutation strategy selection and F and Cr adaption. As generation passes, ***del*** gradually reduced to facilitate exploration first and exploitation later by using formula del=[(abs(best fitness value)/g)+***bias***]. ***bias*** parameter is the reward and punishment factor that further enhances batter parameter retention or failed parameter replacement.

WHILE the stopping criteria is not satisfied

DO

FOR j=1 to NP

Step 3 (Mutation Step): For each generation choose two random individual from population mutually different from each other and j. then randomly select one base vector from ***Q*** best selection.

One of the stagnation sense window (namely ***t*** window) is used to switch mutation strategy dynamically. After passing 50 generation, ***t*** takes the average value of last 50 generation's minimum fitness value of population.

t= sum[A(g-50):A(g-1)]/10

IF (***t*** - minimum(fitness) <***del*** && rand()>0.5) then change the mutation strategy to DE/Q_{opp}/1 where the base vector is randomly selected from pool of (NP-Q) vectors, which is the set of all present generation population vectors excluding ***Q*** best vector.

$$\vec{V}_{i,G} = Q_{opp} + (\overline{X}_{r_1,G} - \overline{X}_{r_2,G})$$

ELSE randomly select base vector from ***Q*** and take mutation strategy DE/Q_{best}/1

$$\vec{V}_{i,G} = Q_{best} + (\overline{X}_{r_1,G} - \overline{X}_{r_2,G})$$

k window triggers the F (scale factor) value adaptation, where adopted F value is F= mode(F_array)+0.1*randn(). Where F_array is the array of successful F values in the past generations. The function creates new F value around the neighbourhood of most successful F value archive. ***k*** window is the average value of last 40 generation's minimum fitness value. If ***k*** value is not reduced by some certain threshold amount (0.25***del***) then it triggers the F adaptation, otherwise it retains the previous F value.

Step 4 (Crossover Step): Generate a trial vector for the i-th target vector through uniform crossover.

Then verify whether the generated trial vectors are within boundary constraints.

The crossover constant Cr is taken 0.9 initially, which is adapted along with the search process. The adaptation is triggered by the stagnation sense window ***z***. ***z*** is the average value of last 20 generation's minimum fitness value. ***z***= sum[A(g-20):A(g-1)]/20. If ***z*** value is not reduced by some certain threshold amount (0.5***del***), then it triggers the Cr adaptation, otherwise it retains the previous Cr value.

Crossover rate Cr is adapted by using the formula

Cr= mode(Cr_array)+***pt*********smu***.

Where ***smu***= randn() and Cr_array is the array of successful Cr values in the past generation.

IF ***smu***<0 then ***pt***=0.2

ELSE take ***pt***=0.1

Step 5 (Selection Step): calculate the fitness of trial individual

$$\vec{U}_{i,G} = \{u_{1,i,G}, ..., u_{D,i,G}\}$$

IF $f(\vec{U}_{i,G}) \le f(\vec{X}_{i,G})$, THEN

$$\vec{X}_{i,G+1} = \vec{U}_{i,G}$$

IF $f(\vec{U}_{i,G}) < f(\vec{X}_{best,G})$, THEN

$$\vec{X}_{best,G} = \vec{U}_{i,G}$$

END IF

ELSE

$$\vec{X}_{i,G+1} = \vec{X}_{i,G},$$

$$f(\vec{X}_{i,G+1}) = f(\vec{X}_{i,G})$$

END IF

END FOR

- Increase the generation count. g= g+1
- Store minimum fitness value over generation to A array.
- Check stagnation by comparing fitness value of current generation with the previous one. If it is reduced, then store the successful F and Cr value and give reward by increasing stagnation threshold window. If it is not reduce by some threshold amount, then discard current F and Cr value and give punishment by decreasing stagnation threshold window.
- Generate stagnation parameters ***t***, ***k***, ***z***, ***u***
- Store ***Q*** best vectors in a ***P*** matrix
- Measure the diversity of best vectors by taking the standard deviation of ***P*** matrix.
- Store the successful F values through generation in an array.
- Store the successful Cr values through generation in an array.
- If fitness value is reduced, then current F and Cr value is taken for the use in future generation till further stagnation replaces it. If fitness value is not reduced then last successful F and Cr value is applied in every even generation and newly generated value is tried in every odd generation.

END FOR

5 Performance Evaluation of Stagnation Adaptive DE (DESSAS)

5.1 Numerical Benchmark:

DESSAS algorithm is tested using a set of 25 well-known boundary constrained standard benchmark function from the special session and competition on real parameter optimization held under the IEEE CEC 2005. These functions contain a diverse set of problem features including multimodality, ruggedness, noise in fitness, ill-conditioning, nonseparability and interdependence (rotation) for single objective optimization and are based on classical benchmark function such as Rosenbrock's, Rastrigin's, Swefel's, Griewank's and Ackley's function.

5.2 Algorithms Compared:

The performance of the DESSAS algorithm is compared with the following algorithms that include five state-of-the art DE variants:

1) JADE
2) jDE
3) SADE
4) EPSDE

Unless otherwise stated, for all the contestant algorithms, we employ the best-suited parametric setup chosen with guidelines from their respective literatures. The population size NP for the DE variants has been kept equal to 150 irrespective of the problem dimension D. also we have compared our main DE variant with above mentioned contestant algorithms at D=50.

5.3 Results on Numerical Benchmark:

System: Windows 7 Home (SP1)
CPU: Pentium ® 4 (3.00 GHz)
RAM: 2GB
Language: Matlab 7.10
Algorithm: DESSAS

Experiment is conducted on 30-D and 50-D problem. We choose population size to be 150 for 30-D problem and 250 for 50-D problem. We have taken 25 run for each function. Function error value is taken after number of function evaluations [FES] =3 x 10^5 for 30-D problem and FES=5 x 10^5 for 50-D problem. We have compared our algorithm with other state-of-the-art algorithm in 30-D and results are listed in table-1. We have also listed our performance results for 50-D problem in Table-2 and Table-3. We have not listed our whole comparison result due to space problem. In order to evaluate how the performance of our algorithm changes with the scaling of the search spaces, we also compared our results with other algorithms in functions f_1 to f_{14} in 100 dimensions. Table-1 shows the mean and standard deviation of the best-of-run errors for 25 independent runs of each of 5 algorithms on 25 numerical benchmarks for D=30. Table-2 and Table-3 reports the same values for functions f_1 to f_{14} tested in D=50. It is to be noted that best-of-the-run error corresponds to the absolute difference between the actual output value $f(\overline{X}_{best})$ and the actual optimum f^* of a particular objective function. Convergence graph of some functions given below:

Fig.1 convergence graph of shifted rotated high conditioned Elliptic Function

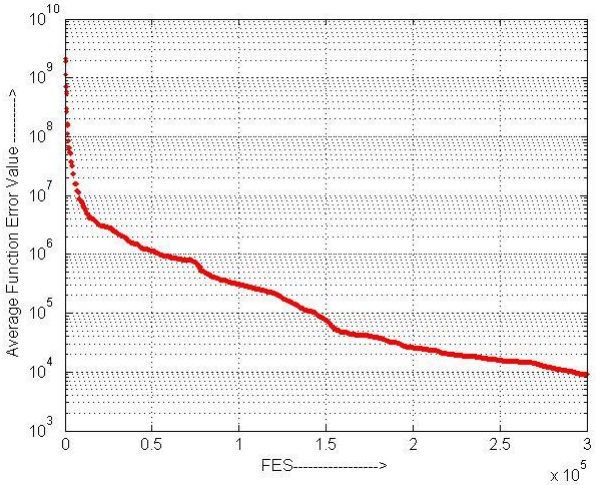


Fig.2 convergence graph of Shifted Rastrigin's Function

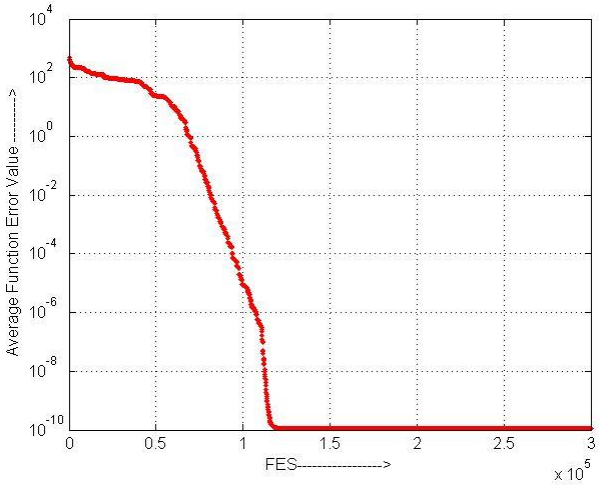


Fig.3 convergence graph of Schwefel's Problem 2.13

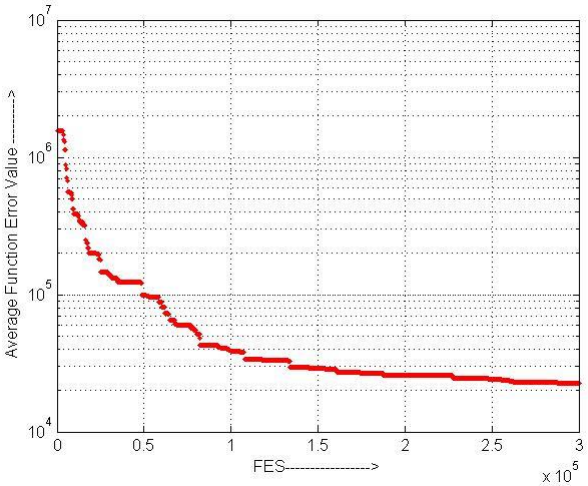
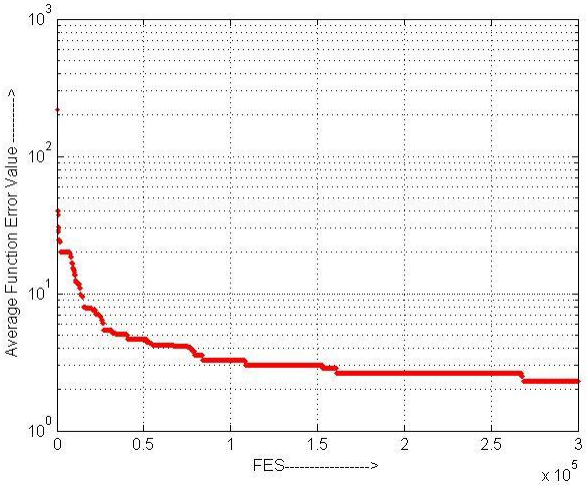


Fig.4 convergence graph of expanded extended Griewank's plus Rosenbrock's function



6 Experimental Study with FM Sound Synthesis

In our experiment, we have compared the performance of our stagnation adaptive DE with classic DE, Particle Swarm Optimization (PSO) and GA.

For GA algorithm, non-uniform mutation operator is applied to overcome premature convergence. The probability of updating a chromosome by mutation is taken 0.125 and the crossover probability is taken 0.6. For PSO, the parameters ω, c_1, c_2 are set to 10, 0.72984, 1.49618 respectively. For classic DE, control parameters, *F* and *Cr* are fixed to 0.5 and 0.9 respectively but for our DE algorithm *F* and *Cr* are initialized to 0.5 and 0.9 respectively. We used same settings for the common parameters of PSO and DE. The population size (N_p) is taken 100 and the maximum number of function evaluations is set to 1,00,000. We have taken 30 independent runs of each algorithms and the mean best function values and standard deviation are recorded for study.

The computational result of PSO, GA, classic DE and stagnation adaptive DE shown in Table 4. "mean error" represents best-of-run errors (the error between estimated sound and the target sound) for 30 independent runs. "std_dev" denotes the standard deviation of errors for 30 runs. "AFE" represents average number of function evaluation requires to perfectly reach the zero error. Since we have set our algorithm termination criteria MAX_FES=1,00,000 function evaluation, so result is taken within this run length.

Table 4. Performance Table for different tested Algorithm

Algorithms	AFE	mean error	std_dev
PSO	100000	8.79	5.32
GA	100000	6.3e-15	4.83
classic DE	68680	0	0
stagnation adaptive DE	28700	0	0

From the results, it can be observed that stagnation adaptive DE can perfectly extract parameters with less number of function evaluation. It is also tested that standard deviation of function evaluation for 30 runs for classic DE is 8.14e+03 whereas for stagnation adaptive DE is 6.24e+03. Figure 5 and Figure 6 presents convergence graph of classic DE and stagnation adaptive DE respectively. It is observed that our DE algorithm converges faster than classic DE.

Fig. 5 convergence graph of classic DE algorithm

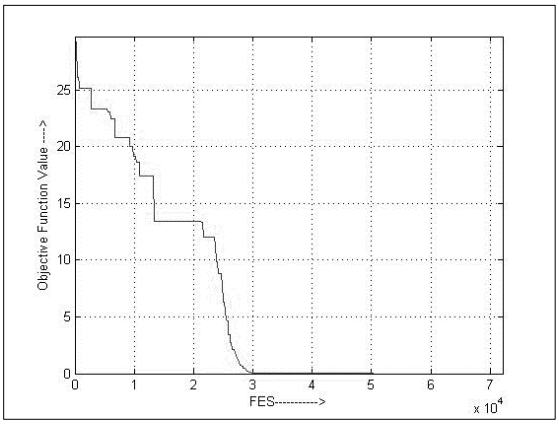


Fig. 6 convergence graph of stagnation adaptive DE

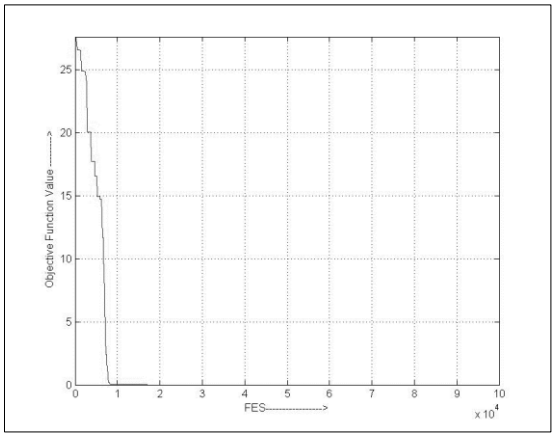
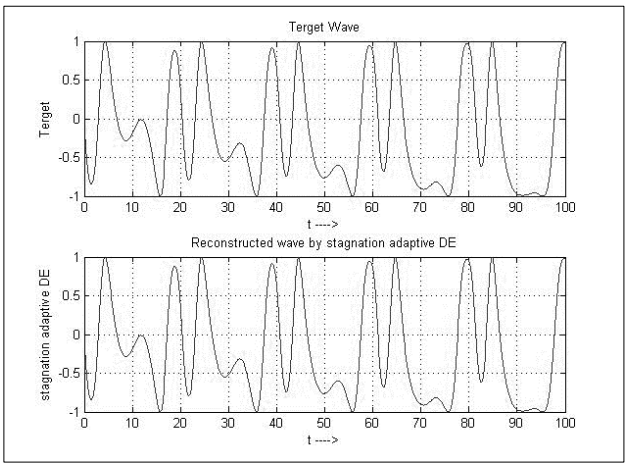


Figure 7 shows the target wave and reconstructed wave by our DE algorithm. It can be observed that our DE algorithm can perfectly mimics the target sound wave.

Fig. 7 Target wave and estimated wave constructed using stagnation adaptive DE



7 Conclusion

In this paper, we have taken benchmark problem described in [12] to conduct experimental studies on it. We have refined classic DE algorithm to work faster and more robust on a different benchmark problem. One of the problems is FM sound synthesis, which has complex multimodal landscape. Our DE can successfully extract parameters with no variance and almost 50% less iteration require than classic DE. On the other hand, PSO and GA fail to optimize parameters perfectly for same initial condition. We want to investigate our algorithm's performance on modified FM synthesizer and high dimensional problem in future.

Table 1
Experimental Results of JADE, jDE, SaDE, EPSDE and DESSAS over 25 independent runs on 25 test functions of 30 variables with 300000 FES

Function		JADE	jDE	SaDE	EPSDE	DESSAS
		Mean error ± Std Dev	**Mean error ± Std Dev**	**Mean error ± Std Dev**	**Mean error ± Std Dev**	**Mean error ± Std Dev**
Unimodal Functions	***F1***	0.00E+00±0.00E+00=	0.00E+00±0.00E+00=	0.00E+00±0.00E+00=	0.00E+00±0.00E+00=	0.0000E+00 ± 0.0000E+00
	F2	1.02E-28±1.23E-28>	1.07E-06±1.54E-06<	7.26E-06±1.52E-05<	4.41E-26±5.17E-26<	0.0000E+00 ± 6.3553E-14
	F3	8.37E+03±7.23E+03<	1.87E+05±1.12E+05>	4.20E+05±2.88E+05>	8.63E+05±3.91E+06>	1.0835E+04 ± 6.6986E+03
	F4	1.62E-16±5.14E-16<	4.93E-02±1.36E-01>	1.22E+02±2.54E+02>	3.11E+02±2.01E+03>	4.5365E-08 ± 4.4086E-08
	F5	8.71E-08±5.12E-07<	5.89E+02±4.43E+02<	3.04E+03±5.61E+02>	1.28E+03±6.12E+02>	8.2731E+02 ± 4.1861E+02
Basic Multimodal Functions	***F6***	2.17E+01±2.76E+01>	2.25E+01±2.33E+01>	5.01E+01±3.32E+01>	5.94E-01±1.42E+00>	1.2101E-10 ± 2.7002-10
	F7	8.13E-03±7.41E-03<	1.17E-02±6.87E-03<	1.52E-02±1.21E-02>	1.63E-02±1.61E-02>	1.3297E-02 ± 9.9340E-03
	F8	2.09E+01±1.57E-01=	2.09E+01±4.92E-02=	2.09E+01±4.57E-02=	2.09E+01±5.29E-02=	2.0962E+01 ± 4.7658E-02
	F9	0.00E+00±0.00E+00=	0.00E+00±0.00E+00=	2.74E-01±3.91E-01>	3.14E-02±2.69E-01>	0.0000E-00 ± 4.3769E-01
	F10	3.09E+01±4.76E+00<	5.59E+01±8.42E+00<	4.83E+01±1.18E+01<	5.32E+01±3.00E+01<	8.7487E+01 ± 2.2847E+01
	F11	2.92E+01±1.73E+00<	3.49E+01±1.42E+00=	1.37E+01±2.40E+00<	3.63E+01±2.98E+00>	3.5407E+01 ± 5.5710E+00
	F12	5.98E+03±4.69E+03<	8.34E+03±8.43E+03<	3.21E+03±2.32E+03<	4.28E+04±6.55E+03<	1.8589E+05 ± 2.8421E+03
Expanded Multimodal Functions	***F13***	1.41E+00±1.09E-01<	1.62E+00±1.32E-01<	3.96E+00±2.80E-01>	1.92E+00±1.38E-01<	3.0830E+00 ± 9.1590E-01
	F14	1.27E+01±3.10E-01<	1.33E+01±2.09E-01=	1.23E+01±2.75E-01<	1.35E+01±2.39E-01=	1.3484E+01 ± 3.0905E-01
Hybrid Composition Functions	***F15***	3.41E+02±1.73E+02>	3.68E+02±8.12E+01>	3.81E+02±7.82E+01>	2.10E+02±1.79E+01<	2.2531E+02 ± 3.5227+01
	F16	1.26E+02±1.32E+02>	7.96E+01±2.91E+01<	8.62E+01±6.97E+01=	1.19E+02±8.91E+01>	8.6708E+01 ± 1.6799E+01
	F17	1.51E+02±1.29E+02>	1.35E+02±3.84E+01<	7.54E+01±3.86E+01<	1.71E+02±1.07E+02>	1.4604E+02 ± 4.9106E+01
	F18	9.04E+02±1.13E+00>	9.03E+02±1.00E+01>	8.66E+02±6.17E+01>	8.21E+02±3.41E+00>	6.3333E+02 ± 2.8867E+02
	F19	9.02E+02±7.98E-01>	9.04E+02±1.21E+00>	8.63E+02±6.09E+01>	8.23E+02±3.30E+00>	7.2000E+02 ± 2.3874E+02
	F20	9.04E+02±8.45E-01>	9.04E+02±1.16E+00>	8.72E+02±6.13E+01>	8.27E+02±4.17E+00=	8.2000E+02 ± 4.4721E+01
	F21	5.01E+02±4.61E-13<	5.00E+02±4.72E-13<	5.49E+02±1.80E+02<	8.53E+02±1.00E+02=	8.5902E+02 ± 7.8336E-01
	F22	8.54E+02±1.84E+01>	8.76E+02±1.93E+01>	9.18E+02±1.90E+01>	5.10E+02±7.12E+00>	5.0044E+02 ± 5.3786E-01
	F23	5.52E+02±7.15E+01<	5.30E+02±2.68E-04<	5.43E+02±4.57E-03<	8.61E+02±6.89E+01=	8.6710E+02 ± 1.1847E+00
	F24	2.01E+02±2.27E-14<	2.00E+02±2.47E-14<	2.01E+02±6.11E-13<	2.14E+02±1.54E+00>	2.1075E+02 ± 2.2870E-14
	F25	2.11E+02±7.94E-01<	2.10E+02±7.22E-01<	2.14E+02±2.10E+00>	2.14E+02±2.61E+00>	2.1230E+02 ± 2.5052E+00
> (better)		**11**	**11**	**15**	**16**	
< (worse)		**11**	**9**	**7**	**3**	
= (equals)		**3**	**5**	**3**	**6**	

Mean Error" and "Std Dev" indicate the average and standard deviation of the function error values obtained in 25 runs respectively. ">" sign denotes performance of our algorithm is better than respective one. "<" sign denotes performance of our algorithm is worse than respective one. "=" sign denotes performance of our algorithm is more or less same than respective one.

Table 2
Error Value and Standard Deviation achieved at 1e+3 FES for the DESSAS algorithm at D=50

FES		F1	F2	F3	F4	F5	F6	F7
5e+5	1st (best)	0.0000E+00	5.6843E-14	3.9652E+03	3.1031E-09	1.6872E+02	5.6843E-14	2.8422E-14
	7th	0.0000E+00	5.6843E-14	4.5715E+04	3.1691E-08	3.5555E+02	5.6843E-14	2.8422E-14
	13th (median)	0.0000E+00	5.6843E-14	9.9839E+04	1.4579E-07	7.2474E+02	3.4106E-13	9.8573E-03
	19th	0.0000E+00	1.1369E-13	4.0837E+05	3.3913E-06	1.2259E+03	3.9866E+00	1.9678E-02
	25th (worse)	5.6843E-14	1.3636E+04	2.4348E+06	2.3367E+04	1.7162E+03	3.9866E+00	4.6729E-02
	Mean	0.0000E+00	5.4545E+02	4.2142E+05	9.3467E+02	7.8359E+02	1.4352E+00	1.3876E-02
	Std Dev	1.1603E-14	2.7272E+03	6.9365E+05	4.6734E+03	4.4413E+02	1.9530E+00	1.3664E-02

Table 3
Error Value and Standard Deviation achieved at 1e+3 FES for the DESSAS algorithm at D=50

FES		F8	F9	F10	F11	F12	F13	F14
5e+5	1st (best)	2.0839E+01	0.0000E+00	2.1889E+01	2.7489E+01	2.0833E+04	2.1356E+00	1.2716E+01
	7th	2.0918E+01	0.0000E+00	5.2548E+01	3.2556E+01	2.8593E+04	2.6089E+00	1.3069E+01
	13th (median)	2.0946E+01	0.0000E+00	7.7821E+01	3.6557E+01	4.4120E+04	2.9172E+00	1.3596E+01
	19th	2.0956E+01	3.4106E-13	1.1293E+02	3.9644E+01	6.1502E+04	3.4263E+00	1.3660E+01
	25th (worse)	2.1016E+01	2.1772E+01	1.8354E+02	4.0732E+01	3.4769E+05	9.1752E+00	1.3802E+01
	Mean	2.0937E+01	1.8983E+00	9.0971E+01	3.5763E+01	5.7858E+04	3.4589E+00	1.3397E+01
	Std Dev	4.3654E-02	5.8642E+00	4.6841E+01	4.3433E+00	6.3555E+04	1.6671E+00	3.4753E-01

References:

[1] J. Chowning, "The synthesis of complex audio spectra by means of frequency modulation", *Journal of the Audio Engineering Society*, vol. 21, no. 7, 1973, pp. 526–534.

[2] M. H. S. Heise and J. Loviscach, "Automatic cloning of recorded sounds by software synthesizers," in *Audio Engineering Society Convention 127*, 2009.

[3] T. M. T. K. T. Yoshimura, K. Tokuda and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," in *Sixth European Conference on Speech Communication and Technology*, 1999, pp. 1315–1318.

[4] M. Roth and M. Yee-King, "A comparison of parametric optimization techniques for musical instrument tone matching," in *Audio Engineering Society Convention 130*, 5 2011, pp. 972–980.

[5] J. Serquera and E. Miranda, "Evolutionary sound synthesis: rendering spectrograms from cellular automata histograms," *Applications of Evolutionary Computation*, 2010, pp. 381–390.

[6] A. Horner, "Evolution in digital audio technology," *Evolutionary Computer Music*, vol. 52, 2007.

[7] I. Fujinaga and J. Vantomme, "Genetic algorithms as a method for granular synthesis regulation," in *Proceedings of the International Computer Music Conference*, 1994, pp. 138–138.

[8] J. Justice, "Analytic signal processing in music computation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 6, 1979, pp. 670–684.

[9] T. J. Mitchell, An exploration of evolutionary computation applied to frequency modulation audio synthesis parameter optimization, *Ph.D. Thesis*, University of the West of England, 2010.

[10] R. Storn, K. Price, "Differential evolution--A simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, vol. 11, 1997, pp. 341-359.

[11] J. Vesterstrom, R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004, pp. 1980-1987.

[12] S. Das, P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems", *Technical Report*, 2010.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", *IEEE International Conference on Neural Networks*, Perth, Australia., 1995, pp.1942-1948.

[14] B. Hutchins, "The frequency modulation spectrum of an exponential voltage-controlled oscillator," *Journal of the Audio Engineering Society*, vol. 23, no. 3, 1975, pp.200–206.

[15] A. Horner, J. Beauchamp, L. Haken, "Methods for multiple wavetable synthesis of musical instrument tones," *Journal of the Audio Engineering Society*, vol. 4, no. 5, 1993, pp. 336-356.

[16] K. Xiong, H. Wang, "Differential Evolution for Frequency Modulation Sound Synthesis Parameter Identification," *IEEE international Conference on Wireless Communication*, Networking and Mobile Computing, 2011, pp. 1-4.

[17] S. Das and P.N. Suganthan, "Differential Evolution-A survey of the state-of-the-art", *IEEE Transaction on Evolutionary Computing*, vol. 15, no. 1, Feb 2011, pp. 4-31.

[18] Neri F. (2005). "Empirical investigation of word-of-mouth phenomena in markets: a software agent approach". *WSEAS Transactions on Computers*, WSEAS Press (Wisconsin, USA), issue 8, vol. 4, pp. 987-994.

[19] Sohan Ghorai, Abhijit Dasgupta, "Differential Evolution with Stagnation Sense Adaptive Strategy (DESSAS)", *6th International Conference on Advance Computing and Communication Technologies (ICACCT 2012)*, November 3-4, 2012, pp 60-65., ISBN: 978-93-82062-70-7.

[20] Sumit.K.Pal, Sohan Ghorai, "Moving Object Tracking System in Video with Kalman Filter", *International Journal of Engineering Research & Technology (IJERT)*, June 2013, Vol. 2 Issue 6, ISSN: 2278-0181.

[21] Maria Angelova, Pedro Melo-Pinto, Tania Pencheva, "Modified Simple Genetic Algorithms Improving Convergence Time for the Purposes of Fermentation Process Parameter Identification", *WSEAS Transactions on Systems*, WSEAS Press (Wisconsin, USA), July 2012, issue 7, vol. 11, pp. 256-267.

[22] Libor Pekař, "On a Controller Parameterization for Infinite-Dimensional Feedback Systems Based on the Desired Overshoot", *WSEAS Transactions on Systems*, WSEAS Press (Wisconsin, USA), June 2013, issue 6, vol. 12, pp. 325-335.

[23] Pallabi Talukdar, Mousmita Sarma, Kandarpa Kumar Sarma, "Recognition of Assamese SpokenWords using a Hybrid Neural Framework and Clustering Aided Apriori Knowledge", *WSEAS Transactions on Systems*, WSEAS Press (Wisconsin, USA), July 2013, issue 7, vol. 12, pp. 360-370.