JAIDM
*Journal of **AI** and **Data Mining***
Vol. 1, No.2, 2013, 111-118.

Estimation of LPC coefficients using Evolutionary Algorithms

H. Marvi[1], Z. Esmaileyan[2], A. Harimi[3*]

1.Electrical engineering department, Shahrood university of technology, Shahrood, Iran
2.Electrical engineering department science and research branch, Islamic Azad Univercity, Shahrood, Iran
3.Electrical engineering department, Shahrood branch, Islamic Azad Univercity, Shahrood, Iran

Received 31 January 2012; accepted 19 February 2013
**Corresponding author: a.harimi@gmail.com (A. Harimi)*

Abstract
The vast use of Linear Prediction Coefficients (LPC) in speech processing systems has intensified the importance of their accurate computation. This paper is concerned with computing LPC coefficients using evolutionary algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Differential Evolution (DE) and Particle Swarm Optimization with Differentially perturbed Velocity (PSO-DV). In this method, evolutionary algorithms try to find the LPC coefficients which can predict the original signal with minimum prediction error. To this end, the fitness function is defined as the maximum prediction error in all evolutionary algorithms. The coefficients computed by these algorithms are compared to coefficients obtained by traditional autocorrelation method in terms of the prediction accuracy. Our results showed that coefficients obtained by evolutionary algorithms predict the original signal with less prediction error than autocorrelation methods. The maximum prediction error is achieved by autocorrelation method: GA, PSO, DE and PSO-DV are 0.35, 0.06, 0.02, 0.07 and 0.001, respectively. This finding shows that the hybrid algorithm, PSO-DV, is superior to other algorithms in computing linear prediction coefficients.

Keywords: *Linear prediction coefficients, evolutionary, algorithms, PSO, DE, PSO-DV*

1. Introduction
Linear predictive coding (LPC) is a very powerful method for speech analysis [1, 2]. This method is widely used because it is fast and simple and yet an effective way of estimating the main parameters of speech signals. Linear predictive coding gets its name from the fact that it predicts the current sample of speech signal, x[n], as a linear combination of its past *p* samples, as follow:

$$x[n] = \sum_{k=1}^{p} a_k x[n-k] + e[n] \quad (1)$$

where a_k and en[n] represent the LPC coefficients and the estimation error, respectively. The linear prediction model described by equation (1) can be schematically shown as Figure 1 wherein z^{-1} indicates the transfer function of a delay system.

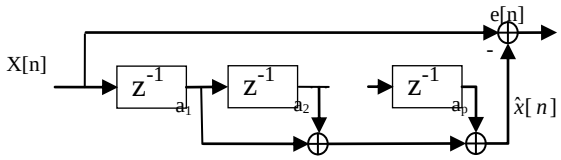

Figure 1. Signal estimation using LPC coefficients.

The prediction error can be written as follow:

$$e[n] = x[n] - \tilde{x}[n] = x[n] - \sum_{k=1}^{p} a_k x[n-k] \quad (2)$$

where $\hat{x}[n]$ is the estimated signal. The basic approach is to find a set of a_k coefficients that minimize the mean square prediction error over a short time of speech signal. There are several traditional methods, such as the covariance method, the autocorrelation method, and the lattice method to determine the LPC coefficients [1-3]. The vast use of Linear Prediction Coefficients (LPC) in speech processing systems

has intensified the importance of their accurate computation.

Our contribution in this work is to determine LPC coefficients using four evolutionary algorithms: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), DE (Differential Evolution) and PSO-DV. These algorithms try to find the best LPC coefficients which predict the speech signal with less prediction error. The LPC coefficients obtained from traditional autocorrelation method are used here as a benchmark to verify whether the proposed evolutionary methods can find better coefficients.

The paper is organized as follows: Next section focuses on the evolutionary algorithms include GA, PSO, DE and PSO-DV are detailed by the proposed method in section 3. The experimental results for applying evolutionary algorithms to determine optimal LPC coefficients are presented and discussed in section 4, and finally the paper is concluded in section 5.

2.Evolutionary algorithms

The use of *evolutionary* strategies (ESs) is to solve non-linear optimization problems has attracted much attention recently. The common underlying idea behind all the evolutionary algorithms is the same: A population of individuals tries to survive under the environmental pressure. The fittest individuals have the more chance to survive and this yields a rise in the fitness of the population. Given a fitness function to be maximized (or minimized), a set of candidate solutions can randomly be created. Based on the fitness function, some of the better candidates are chosen to generate the next population using various generation operators, such as recombination and mutation. Recombination is an operator applied to two or more selected candidates (the so-called parents) and generate one or more new candidates (the children). Mutation is applied to one candidate and generates one new candidate. Various evolutionary algorithms take different strategies to upgrade a population in consecutive generations. In this section, four evolutionary algorithms, GA, PSO, DE and PSO-DV are described.

2.1.Genetic Algorithm (GA)

A basic element of the biological genetics is the chromosomes. Chromosomes cross over each other, and mutate themselves, and a new set of chromosomes is generated. Based on the requirement, some of the chromosomes survive. This is the cycle of one generation in biological genetics. The above process is repeated for many generations and finally the best set of chromosomes based on the requirement are available. The Mathematical algorithm equivalent to the above behavior used as the optimization technique is called as an Artificial Genetic Algorithm [4, 5]. GA flowchart is shown in Figure 2.

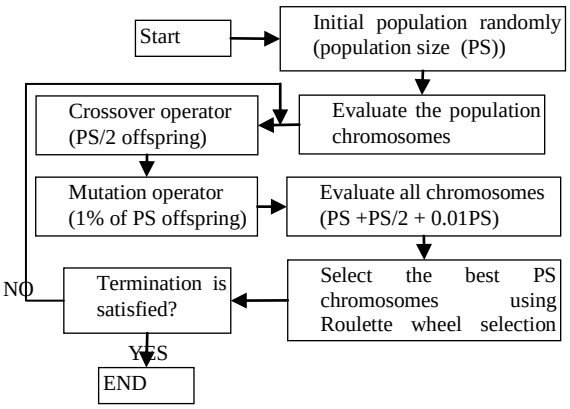

Figure 2. GA flowchart.

As can be seen from figure 2, at the step 1 the chromosomes are initialized randomly, and then GA iteratively examines various sets of coefficients generated during the genetic operators crossover and mutation [6]. In each iteration, GA chooses the qualified chromosomes, which result in the minimum estimation error using Roulette wheel selection method. This algorithm is detailed in [7-9]. The classic crossover and mutation operators are schematically shown in Figure 3.

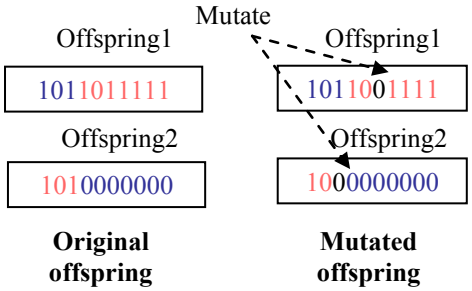

Figure 3. Genetic operators: crossover and mutation.

Table 1 shows the parameter setup for the GA employed here.

Table 1. Genetic Algorithm (GA) parameters setup.

Modeling description	setting
Population size	100
Selection technique	Roulette wheel
Crossover type	One point crossover
Crossover rate	0.9
Mutation rate	0.001
Iteration number	100

2.2.Particle Swarm Optimization (PSO)

In PSO, there is a society, wherein each of the members named as a particle is considered as a possible solution for the available problem. The number of particles which is usually chosen below 100 is named as swarm size. In step 1 the particles are initialized randomly at the position $x_i(0)$. Then, particles are evaluated using a fitness function. The best position each particle has been so far, and The position of the best particle in the society are named pbest (personal best) and gbest (global best). Each particle has its own velocity, which is updated in each iteration. The velocity and position of *i-th* particle at current iteration could be written as [10, 11]:

$$v_i(t+1) = \zeta[\ \omega.v_i(t) + c_1.\Phi_1.(\ pbest_i(t) - x_i(t)) + c_2.\Phi_2.(\ gbest(t) - x_i(t)) \quad (3)$$

and

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (4)$$

where $v_i(t)$ and $x_i(t)$ are vectors which denote the previous velocity and position of the *i-th* particle in sequence. ζ is called the constriction factor. Venter and Sobeiski termed c_1 as 'self confidence' and c_2 as 'swarm confidence' [4,12].

Since in this problem each particle represents an one possible solution for linear prediction coefficients, dimensionality of problem (each vector dimension) depends on the number of LPC coefficients or estimation degree. Φ_1 and Φ_2 stand for a uniformly distributed random number in the interval (0, 1). There are many suggestions from researchers about ζ , c_1 and c_2. In this study, these parameters are determined as [4, 13]:

$$\zeta = 2/|4 - \phi - \sqrt{\phi^2 - 4\phi}| \quad (5)$$

where

$$c_1 = (c_{1f} - c_{1i})(t / t_{Max}) + c_{1i} \quad (6)$$

and

$$c_2 = (c_{2f} - c_{2i})(t / t_{Max}) + c_{2i} \quad (7)$$

where $c_{1i}=c_{2f}=2.5$, $c_{2i}=c_{1f}=0.5$ and t_{Max} is the number of maximum allowable iterations. In this study the swarm size is set to 100.

2.3.Differential Evolution (DE)

In 1995, Price and Storn proposed a new floating point encoded evolutionary algorithm for global optimization and named it DE owing to a special kind of differential operator invoked to create new offspring from parent chromosomes instead of classical crossover or mutation [4, 14]. Easy methods of implementation and negligible parameter tuning made the algorithm quite popular very soon.

Like any other evolutionary algorithm, DE also starts with a population of *PS* D-dimensional search variable vectors. We will represent subsequent generations in DE by discrete time steps like t=0, 1, 2 ...t, t+1 etc. Since the vectors are likely to be changed over different generations we may adopt the following notation for representing the *i-th* vector of the population at the current generation (i.e. at time t=t) as:

$$\vec{X}_i(t) = [x_{i,1}(t), x_{i,2}(t), ..., x_{i,D}(t)] \quad (8)$$

These vectors are referred in literature as 'genomes' or 'chromosomes'. DE is a very simple evolutionary algorithm and works through a simple cycle, presented in Figure 3.

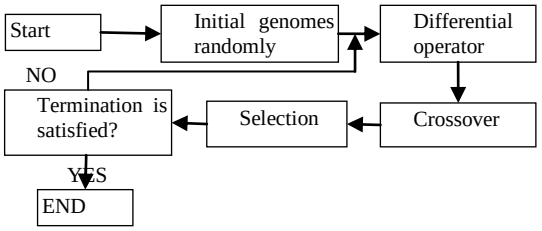

Figure 4. DE flowchart.

In step 1 the chromosomes are initialized randomly at position $x_i(0)$. Now in each generation (or one iteration of the algorithm) to change each population member $\vec{x}_i(t)$, a donor vector $\vec{v}_i(t)$ is created. To create $\vec{v}_i(t)$ for *i-th* member, three other parameter vectors (e.g., the r1, r2, and r3-th vectors) are chosen in a random fashion from the current population. Next, a scalar number F scales the difference of any two of the three vectors and the scaled difference is added to the third one whence we obtain the donor vector $\vec{v}_i(t)$. The process for the, *j-th* component of each vector can be expressed as [15,16]:

$$V_{i,j}(t) = X_{r1,j}(t) + F(X_{r2,j}(t) - X_{r3,j}(t)) \quad (9)$$

Next, to increase the potential diversity of the population, a crossover scheme comes into play. The crossover is performed on each of the *D* variables whenever a randomly picked number between *0* and *1* is lower the Crossover Rate (*CR)*. Then the trial vector forms as:

$$\vec{U}_i(t) = [U_{i,1}(t), U_{i,2}(t)..U_{i,D}(t)] \quad (10)$$

wherein:

$$U_{i,j}(t)=\begin{cases}V_{i,j}(t), rand(0,1)<CR\\ X_{i,j}(t), otherwise\end{cases} \quad (11)$$

CR and F are two control parameters for DE. For each trial vector, $\vec{x}_i(t)$, an offspring vector, $\vec{U}_i(t)$, is created. To keep the population size constant over subsequent generations, the next step of the algorithm calls for 'selection' to determine which one of the target vectors and trial vectors will survive in the next generation (i.e. at time t = t+1). DE actually involves the Darwinian principle of 'Survival of the fittest' in its selection process which may be outlined as:

$$\vec{X}_i(t+1)=\begin{cases}\vec{U}_i(t), f(\vec{U}_i(t))\le f(\vec{X}_i(t))\\ \vec{X}_i(t), f(\vec{X}_i(t))< f(\vec{U}_i(t))\end{cases} \quad (12)$$

where f( ) is the function to be minimized. So, if the new trial vector yields a better value of the fitness function, it replaces its target in the next generation; otherwise, the target vector is retained in the population. Hence the population either gets better (the fitness function) or remains constant but the population never deteriorates. Table 2 shows the parameter setup for the DE employed here.

Table 2. Differential Evolution (DE) parameters setup.

Modeling description	setting
Number of genomes	100
Crossover rate (CR)	0.9
Scaling factor (F)	0.01
Iteration number	100

2.4.PSO-DV

PSO-DV is a hybrid evolutionary algorithm introduces a differential operator (borrowed from DE) in the velocity-update scheme of PSO [4, 11]. The operator is invoked on the position vectors of two randomly chosen particles, not on their individual best positions. Further, unlike the PSO scheme, a particle is actually shifted to a new location only if the new location yields a better fitness value, that is,, a selection strategy has been incorporated into the swarm dynamics. In the proposed algorithm, for each particle *i* in the swarm two other distinct particles, say j and k ($i \neq j \neq k$) are selected randomly. The difference between their positional coordinates is taken as a difference vector:

$$\vec{\delta}=\vec{X}_k-\vec{X}_j, k\neq j\neq i \quad (13)$$

Then the d-th velocity component (1 < d < n) of the target particle i is updated as:

$$V_{id}(t+1)=\begin{cases}\omega V_{id}(t)+\beta\delta_d+c_2\varphi_2.(p_{gd}\\ -X_{id}(t)), rand(0,1)\le CR\\ V_{id}(t), otherwise\end{cases} \quad (14)$$

where *CR* is the crossover rate, *d* is the *d-th* component of the difference vector defined earlier, and β is a scale factor in [*0, 1*]. Now, a new trial location Tr_i is created for the particle by adding the updated velocity to the previous position X_i :

$$\vec{T}r_i=\vec{X}_i(t)+\vec{V}_i(t+1) \quad (15)$$

The particle is placed at this new location only if the coordinates of the location yield a better fitness value. Thus, if we are seeking the minimum of an n-dimensional function $\vec{f}(x)$, then the target particle is relocated as follows:

$$\vec{X}_i(t+1)=\begin{cases}\vec{T}r_i, f(\vec{T}r_i)\le f(\vec{X}_i(t))\\ \vec{X}_i(t), otherwise\end{cases} \quad (16)$$

Therefore, every time its velocity changes, the particle either moves to a better position in the search space or sticks to its previous location. The current location of the particle is thus the best location it has ever found [4]. Table 3 shows the parameter setup for the DE employed here.

Table 3. Differential Evolution (DE) parameters setup.

Modeling description	setting
Swarm size	100
Crossover rate (CR)	0.5
Scaling factor (β)	0. 1
Iteration number	100

3.Proposed method

Designing the individuals and the fitness function are two common parts of coding a non-linear optimization problem to be solved by ESs. Each individual is known as chromosome in GA and a particle in PSO should be able to represent a possible solution for the available problem. The fitness function must be capable to evaluate individuals as possible solutions for the optimization problem.

In this study, the problem is to find LPC coefficients which can estimate the speech signal with less prediction error. Therefore, each set of LPC coefficients can be an individual in the ES and all individual construct the population. Since the main problem is to minimize the prediction error, the fitness functions is defined as the maximum prediction error as:

$$f(i)=\max(|\hat{s}[n]-s[n]|), 1\le i\le N \quad (17)$$

Where $\hat{s}[n]$ and $s[n]$ indicate the predicted signal and original signal, respectively. N is the number of individuals presented in the population. Minimizing the fitness function during the algorithm improves the prediction accuracy. Figure 5 represents the diagram of the proposed evolutionary algorithm to find the optimal LPC coefficients.

In Figure 5, Ci , $1 \le i \le 8$ represents the ith LPC coefficient. In this study, we use 8 LPC coefficients to predict the speech signal (order 8 LPC).

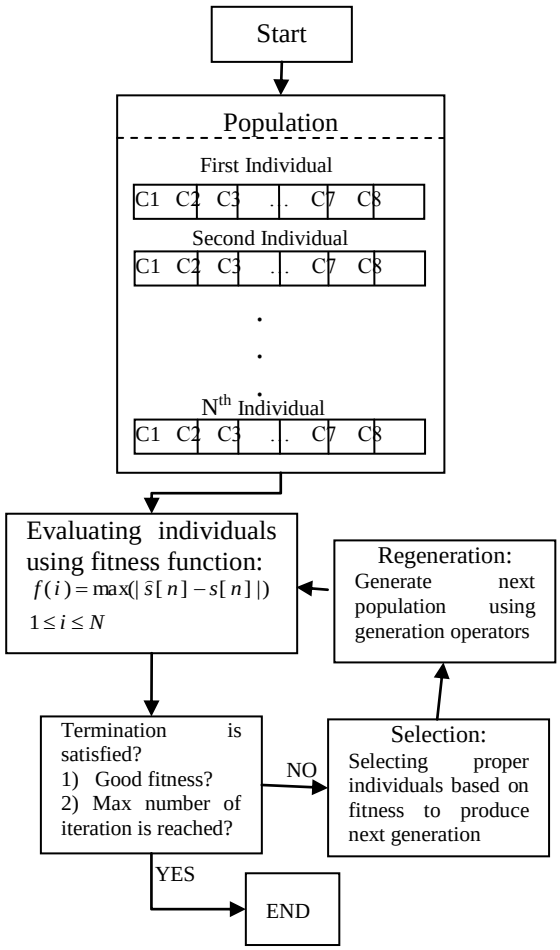

Figure 5. Diagram of the proposed method.

As can be seen from Figure 5, each individual include 8 LPC coefficients, which can be a possible solution for the problem. In step 1, individuals are initialized randomly. After that, the population is upgraded in an evaluation-selection-regeneration loop. In this loop individuals are evaluated using fitness function described by equation (17). The best individuals are chosen in the selection stage to reconstruct the next generation. Finally, next population is produced using generation operators. Various evolutionary algorithms are usually different in the employed selection and generation strategies.

In this study, we employ four different evolutionary algorithms (GA, PSO, DE and PSO-DV) to find optimal LPC coefficients which can estimate the speech signal with less prediction error. The results of our experiments are presented and discussed in the next section.

4.Experimental results

In this section, we compare the proposed evolutionary algorithms and traditional autocorrelation method by means of the prediction error. The first 8 LPC coefficients estimated from autocorrelation, GA, PSO, DE and PSO-DV algorithms are used to estimate two 20ms frames of speech.

Figures 6 to 15 show the original signals, estimated signals and the prediction error curves obtained for the two frames using autocorrelation method, GA, PSO, DE and PSO-DV algorithms.

As can be seen from these figures, all of the evolutionary algorithms employed to estimate signal using LPC coefficients outpoint the traditional autocorrelation method in term of prediction error. The error reduction rate of these algorithms is shown in Figure 16 and 17, respectively for the two speech frames.

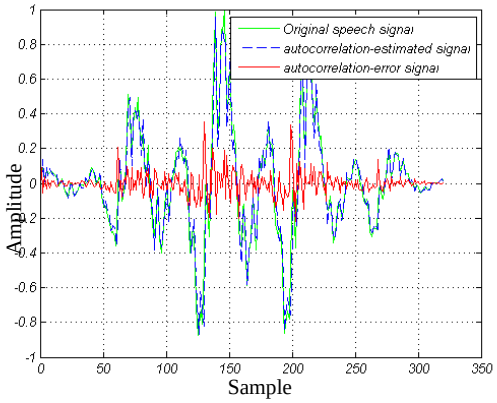

Figure 6. Autocorrelation method (frame 1).

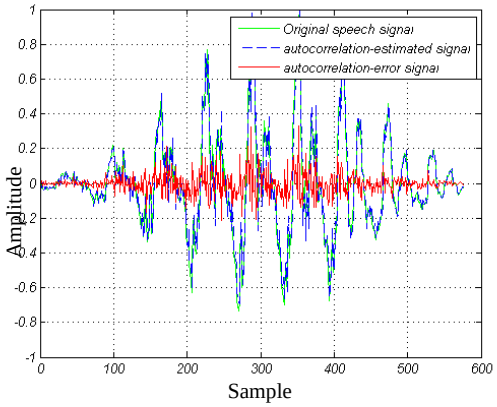

Figure 7. Autocorrelation method (frame 2).

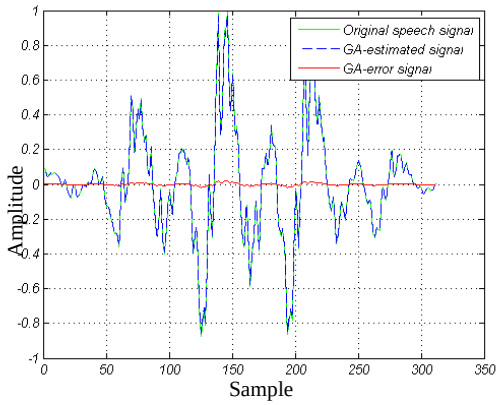

Figure 8. GA (frame 1).

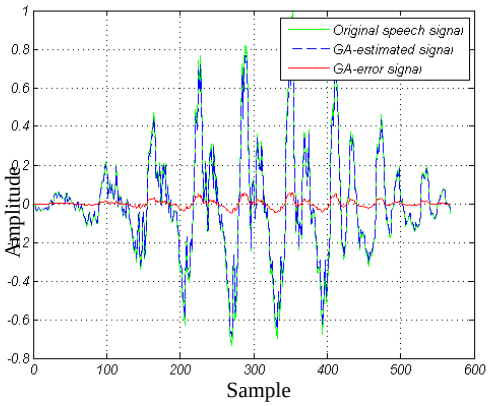

Figure 9. GA (frame 2).

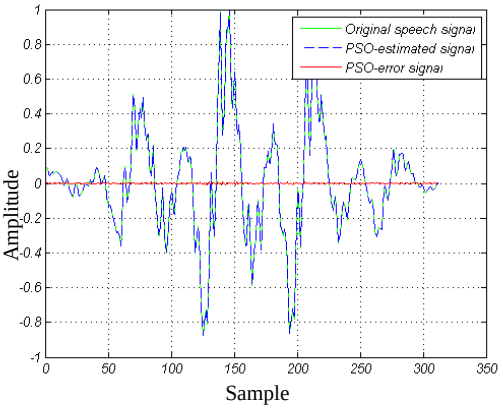

Figure 10. PSO (frame 1).

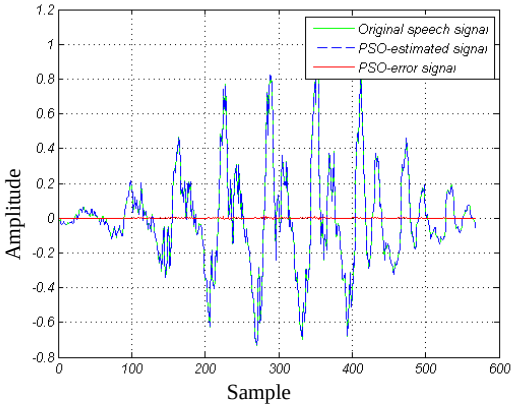

Figure 11. PSO (frame 2).

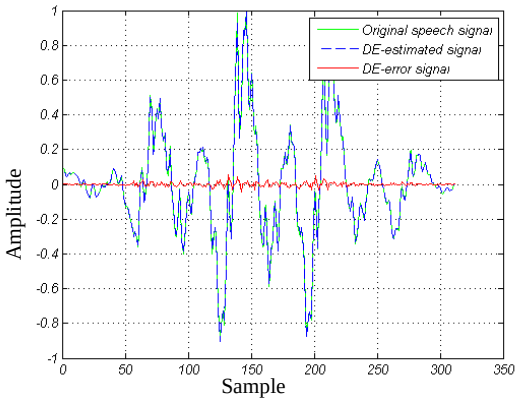

Figure 12. DE (frame 1).

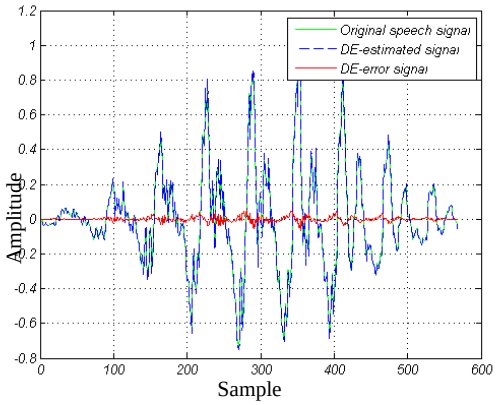

Figure 13. DE (frame 2).

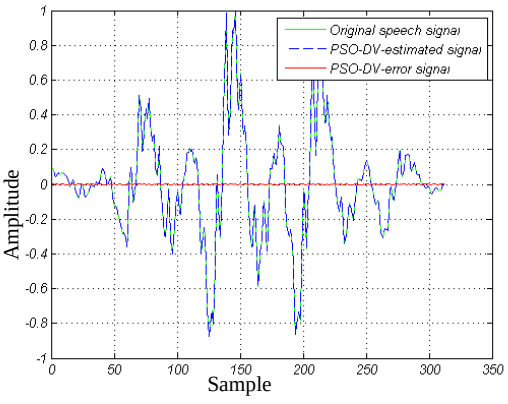

Figure.14. PSO-DV (frame 1).

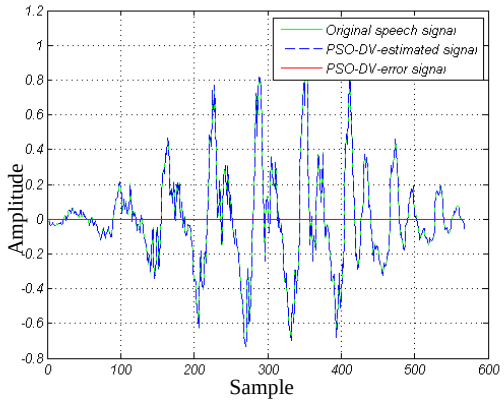

Figure 15. PSO-DV (frame 2).

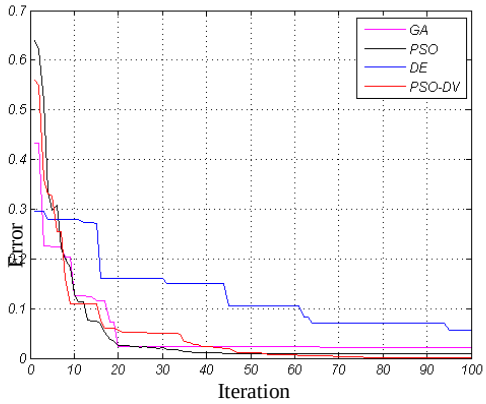

Figure 16. Error rate for GA, PSO, DE and PSO-DV (frame 1).

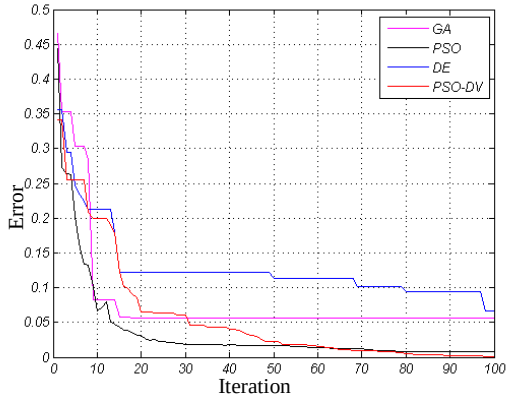

Figure.17. Error rate for GA, PSO, DE and PSO-DV (frame 2).

From Figures16 and 17, it can be seen that after 50 iterations the error rates of the GA, PSO and PSO-DV are converged, while DE is not converged yet. Also, it could be seen that PSO-DV outperforms other techniques in term of convergence speed and predicted error. It is interesting to see that all the evolutionary algorithms reached to the error rate lower that autocorrelation scheme after just 10 iterations. According to Figures 16 and 17, PSO seems to be the most efficient method for iteration number below 50, while PSO-DV is superior since then.

Figure 18 shows the minimum prediction error obtained using each method.

As can be seen from Figure 18, the proposed PSO-DV is superior to autocorrelation and also other evolutionary algorithms. Also, the success of evolutionary algorithms in computing efficient LPC coefficients which result in minimum prediction error encourages the idea of employing these algorithms for this problem.

The LPC coefficients obtained by each method are presented in Tables 4 and 5 for first and second speech frames, respectively. As can be seen from these tables, the computed coefficients are by no means similar, but all of them estimate the original speech signal properly.

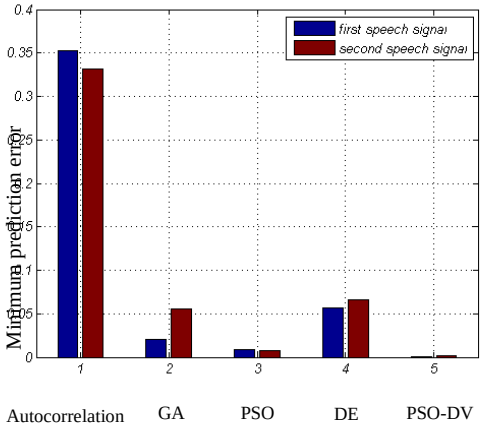

Figure 18. minimum prediction error achieved each various methods (frame 1 and 2).

5.Conclusion

The purpose of the current study was to determine the LPC coefficients for speech signal using evolutionary algorithms: GA, PSO, DE and PSO-DV. The findings in this study suggest that the evolutionary algorithms are superior to traditional methods, such as Autocorrelation method in terms of prediction accuracy. The following conclusions can be drawn from the present study.

First, evolutionary algorithms, such as GA, PSO, DE and PSO-DV can predict speech signal more accurate than traditional autocorrelation method.

Second, our experiments show that the hybrid algorithm, PSO-DV, which resulted from the contribution of PSO and DE, is the fastest algorithm and DE is the slowest one. According to our experiments, GA, PSO and PSO-DV converged after 50 iterations while DE did not converge after 100 iterations.

Moreover, PSO-DV achieved the best results after 100 iterations. After the PSO-DV, PSO and GA were the most powerful methods to estimate the speech signal with less prediction error.

Another flexibility of evolutionary algorithms is to find solutions with any desirable criterion. In other words, we can limit our results with more constrictions, such as stability of the constructed LPC filter. This could be realized by designing a proper fitness function for the problem. Algorithms try to find the fittest solution to the function. In this work the fitness function is defined as maximum prediction error, therefore, algorithms tried to find LPC coefficients with less prediction error.

It is clear that for the hardware implementation of a signal processing system, the LPC coefficients should be quantized and its effect on the efficiency of the system would not remain optimal after the quantization procedure the coefficients. Evolutionary algorithms can be employed to find the optimal quantized coefficients. Moreover, applying LPC coefficients obtained by evolutionary algorithms for speech processing applications may improve their efficiency.

Table 4. LPC coefficients obtained by each method for frame 1.

Method	C1	C2	C3	C4	C5	C6	C7	C8
Autocorrelation	1.4849	-0.8596	0.0345	0.7340	-0.7169	0.3087	0.3699	-0.4936
GA	0.0389	-0.1377	0.0643	-0.0688	0.1961	-0.1710	0.1506	0.8969
PSO	-0.0212	0.0475	-0.0489	0.0162	0.0369	-0.0602	0.0489	0.9794
DE	-0.0336	0.0425	-0.1215	0.1630	-0.1065	0.0468	0.0562	0.9379
PSO-DV	-0.0008	0.0008	0.0007	-0.0015	0.0015	-0.0011	0.0003	1

Table 5. LPC coefficients obtained by each method for frame 2.

Method	C1	C2	C3	C4	C5	C6	C7	C8
Autocorrelation	1.6647	-1.1798	0.5273	0.3132	-0.9179	1.1388	-0.8051	0.1637
GA	-0.1694	0.3606	-0.3501	0.0077	0.3515	-0.5047	0.5034	0.7723
PSO	0.0020	-0.0066	0.0210	-0.0349	0.0116	0.0176	-0.0253	1.0136
DE	-0.0520	0.0916	-0.1205	0.1200	-0.1426	0.0844	-0.0608	1.0789
PSO-DV	-0.0019	0.0042	-0.0039	-0.0006	0.0042	-0.0049	0.0051	0.9978

References

[1] Huang, X., Acero, A. and Hon, H. W. (2001). Spoken Language Processing. Upper saddle River. NJn And Prentice Hall.

[2] Rabiner, L. R. and Schafer, R. W. (1978). Digital processing of speech signals. Englewood cliffs. NJ, Prentice Hall.

[3] Gopal, E. S. (2007). Algorithm collections for digital signal processing applications using matlab. Natural institute of Technology, Tiruchi, India.

[4] Das, S., Abraham, A. and Konar, A. (2007). Particle Swarm Optimization and differential Evolution Algorithms: Technical analysis, Applications and Hybridization perspectives. Dept of Electronics and Telecomunications Engineering, Jadavpur University, Kolkata, 700032.

[5] Manoj, V. J. and Elias, E. (2009). Design of multiplier-less non uniform filter bank trans multiplexer using genetic algorithm. Signal Processing. Volume 89, Issue 11.

[6] Lim, Y. H., Tana, J. and Abramsonb, D. (2012). Solving Optimization Problems in Nimrod/OK using a Genetic Algorithm. Procedia Computer Science. 9, 1647 – 1656.

[7] Xiang , L., Gang, D. and BSTBGA. (2013). A hybrid genetic algorithm for constrained multi-objective optimization problems. Computers & Operations Research. 40, 282–302.

[8] Gen, M., Cheng, R. (2000). Genetic Algorithms and Engineering Optimization. vol. 68, Wiley Interscience Publication.

[9] Goldberg, D.E. (1989). Genetic Algorithm in search, optimization and machine learning. Addison-Wesley, Reading, MA.

[10] Wong, T. C. and Ngan, S. C. (2012). A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop. Applied Soft Computing.

[11] Epitropakis, M. G., Plagianakos, V. P. and Vrahatis, M. N. (2012). Evolving cognitive and social experience in Particle Swarm Optimization through Differential Evolution: A hybrid approach. Information Sciences. 216, 50–92.

[12] Upendar, J., Gupta, C. P. and Singh, G.K. (2010). Design of two-channel quadrature mirror filter bank using particle swarm optimization. Digital Signal Processing. Volume 20, Issue 2.

[13] Khare, A. and Rangnekar, S. (2012). Particle swarm optimization: A review, Applied Soft Computing .

[14] Chang, W. D. (2009). Two-dimensional fractional-order digital differentiator design by using differential evolution algorithm. Digital Signal Processing. Volume 19, Issue 4.

[15] Mohamed, A. W. and Sabry, H. Z. (2012). Constrained optimization based on modified differential evolution algorithm. Information Sciences. 194, 171–208.

[16] Mohamed, A. W., Sabry, H. Z. and Khorshid, M. (2012). An alternative differential evolution algorithm for global optimization. Journal of Advanced Research. 3, 149–165.