NON LINEAR BLIND SOURCE SEPARATION USING COMPUTATIONALLY INTELLIGENT TECHNIQUES

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS & COMMUNICATION ENGINEERING
By

DEBASHISH SADANGI
Roll No: 10509010

ILA MISHRA
Roll No: 10507011

Under the Guidance of
D.P.Acharya

Department of Electronics & Communication Engineering
National Institute of Technology, Rourkela
Rourkela, Orissa

National Institute of Technology
Rourkela

CERTIFICATE

This is to certify that the thesis entitled **"NON LINEAR BLIND SOURCE SEPARATION USING COMPUTATIONALLY INTELLIGENT TECHNIQUES"** submitted DEBASHISH SADANGI, Final year student of Electronics and communication Engineering, Roll No.:10509010 and ILA MISHRA, Final year student of Electronics and Instrumentation Engineering, Roll No.:10507011 in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Electronics and communication Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/ Institute for the award of any Degree or Diploma.

D.P.Acharya
Project Guide
Department of E.C.E
National Institute of Technology
Rourkela – 769008

ACKNOWLEDGEMENT

We take this opportunity as a privilege to thank all individuals without whose support and guidance we could not have completed our project in this stipulated period of time. First and foremost we would like to express our deepest gratitude to our Project Supervisor Mr.D.P.Acharya, Department of Electronics and Communication Engineering, for his invaluable support, guidance, motivation and encouragement through out the period this work was carried out. His readiness for consultation at all times, his educative comments and inputs, his concern and assistance even with practical things have been extremely helpful.

We would also like to thank all professors and lecturers for their generous help in various ways for the completion of this thesis .We also extend our thanks to our fellow students for their friendly co-operation.

DEBASHISH SADANGI
Roll. No. 10509010
Department of ECE
NIT Rourkela

ILA MISHRA
Roll. No. 10507011
Department of ECE
NIT Rourkela

ABSTRACT

The Independent Component Analysis technique has been used in Blind Source separation of non linear mixtures. The project involves the blind source separation of a non linear mixture of signals based on their mutual independence as the evaluation criteria. The linear mixer is modeled by the Fast ICA algorithm while the Non linear mixer is modeled by an odd polynomial function whose parameters are updated by four separate optimization techniques which are Particle Swarm Optimization, Real coded Genetic Algorithm, Binary Genetic Algorithm and Bacterial Foraging Optimization. The separated mixture outputs of each case was studied and the mean square error in each case was compared giving an idea of the effectiveness of each optimization technique.

Key words: *Nonlinear Blind Source Separation, Particle Swarm Optimization, Genetic Algorithm, Real coded GA, Binary GA, Bacterial Foraging*

CONTENTS

TITLE **PAGE NO.**

CHAPTER 1

INTRODUCTION TO ICA

1.1 Blind source separation 9
1. 2 Independent component analysis 12
1.3 FAST ICA 18
1.4 Flow chart of FASTICA 20
1.5 Results 22

CHAPTER 2

PARTILCLE SWARM OPTIMIZATION BASED ICA

2.1 Introduction to PSO 24
2.2 A basic canonical PSO algorithm 26
2.3 Flow chart of PSO algorithm 28
2.4 blind source separation using PSO and ICA 31
2.5 Nonlinear blind separation based on PSO 33
2.6 Block diagram of nonlinear ICA and PSO 37
2.7Result of blind source separation using ICA and PSO 40

CHAPTER 3

GENETIC ALGORITHM BASED ICA

3.1Introduction to genetic algorithm 42

3.2Simple generational genetic algorithm pseudo code 46

3.3GA element 46

3.4Observations 49

3.5 Nonlinear blind separation based on GA 50

3.6Block diagram of nonlinear ICA using real coded GA 53

3.7 Block diagram of nonlinear ICA using binary GA 56

3.8 Result of BSS using ICA and real coded GA(sine wave & random noise) 59

3.9 result of BSS using ICA and real coded GA(square wave & random noise) 60

3.10 Result of BSS using ICA and BINARY GA(sine wave & random noise) 61

3.11 Result of BSS using ICA and binary GA(square wave & random noise) 62

CHAPTER 4

BACTERIA FORAGING OPTIMIZATION BASED ICA

4.1 Introduction to bacterial foraging algorithm 64

4.2 The optimization function for the bacterial foraging (BF) algorithm 66

4.3 A simple Bacterial Foraging Algorithm 67

4.4 Block diagram of nonlinear ICA using BFO 69

4.5 result of BSS using ICA and BFO(sin wave & random noise) 72

4.6 result of BSS using ICA and BFO(square wave & random noise) 73

4.7 Comparison between various optimization techniques 74

4.8 Discussion 75

CHAPTER 5

CONCLUSION

5.1 Conclusion 77

Reference 78

Chapter 1

INTRODUCTION TO ICA

1.1 Blind source separation

1.2 Independent component analysis

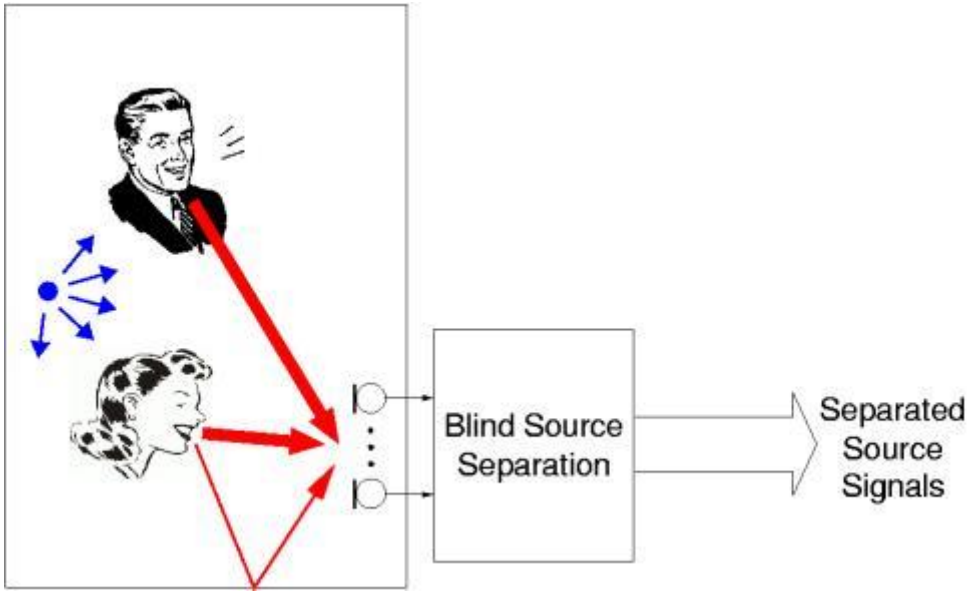
1.3 FAST ICA

1.4 Flow chart of FASTICA

1.5 Results

1.1 BLIND SOURCE SEPARATION

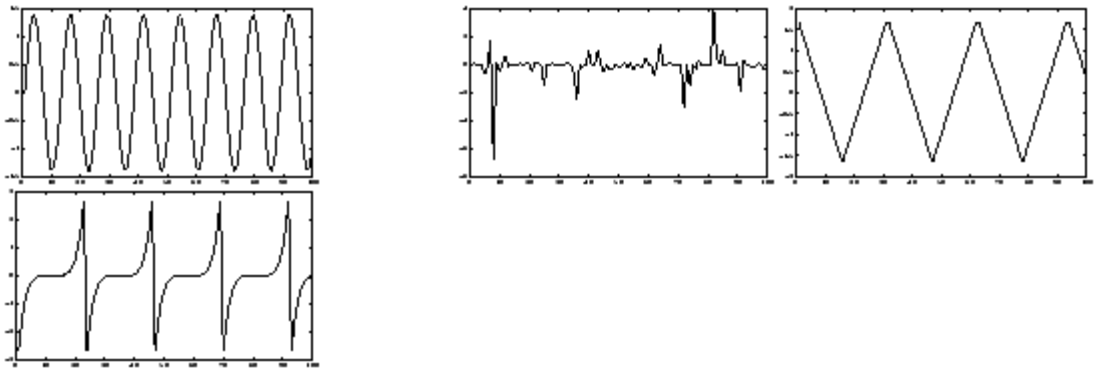
Blind source separation (BSS) refers to the problem of recovering signals from several observed linear mixtures. The strength of the BSS model is that only mutual statistical independence between the source signals is assumed and not a priori information about ,e.g., the characteristics of the source signals, the mixing matrix or the arrangement of the sensors is needed. Therefore BSS can be applied to a variety of situations such as, e.g., the separation of simultaneous speakers, analysis of biomedical signals obtained by EEG or in wireless telecommunications to separate several received signals.



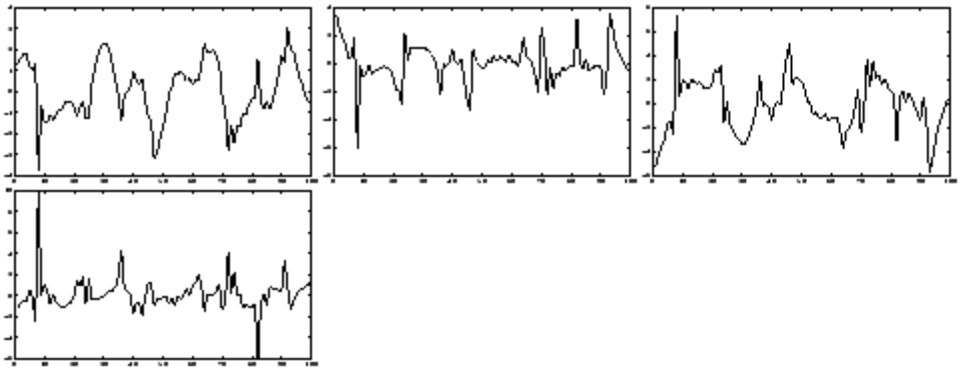
With respect to audio signals, a linear mixture of sources is commonly referred to as "cocktail party problem". How can humans select the voice of a particular speaker from an ensemble of different voices corrupted by music and noise in the background? One approach to solve this problem is to record the mixed audio signals with microphone arrays and subsequently apply blind source separation methods. Several simultaneously active signal sources at different spatial locations can then be separated by exploiting mutual independence of the sources. In the field of audio processing BSS is applicable, e.g., to the realization of noise robust speech recognition, high-quality hands-free telecommunication systems or speech enhancement in hearing aids.

Because temporal redundancies (statistical regularities in the time domain) are "clumped" in this way into the resulting signals, the resulting signals can be more effectively deconvolved than the original signals.

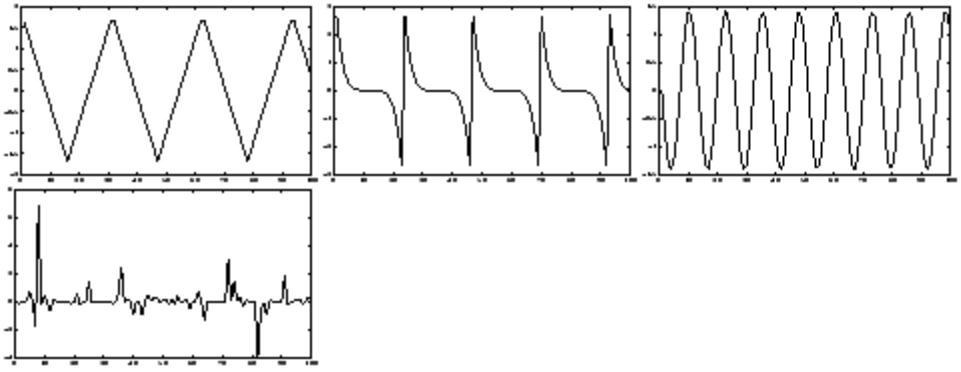
1.1.1 An illustration of blind source separation. This figure shows four source signals, or independent components.



Due to some external circumstances, only linear mixtures of the source signals in Fig. above as depicted here, can be observed.



Using only the linear mixtures in Fig. 6, the source signals in Fig. 5 can be estimated, up to some multiplying factors. This figure shows the estimates of the source signals.



1.2 INDEPENDENT COMPONENT ANALYSIS

Independent component analysis (**ICA**) belongs to a class of blind source separation method for separating data into underlying components ,where such data can take the form of images, sounds, telecommunication channels or stock market prices. It is a computational method for separating a multivariate signal into additive subcomponents supposing the mutual statistical independence of the non-Gaussian source signals. It is a special case of blind source separation.

ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples. In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed nongaussian and mutually independent, and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA.

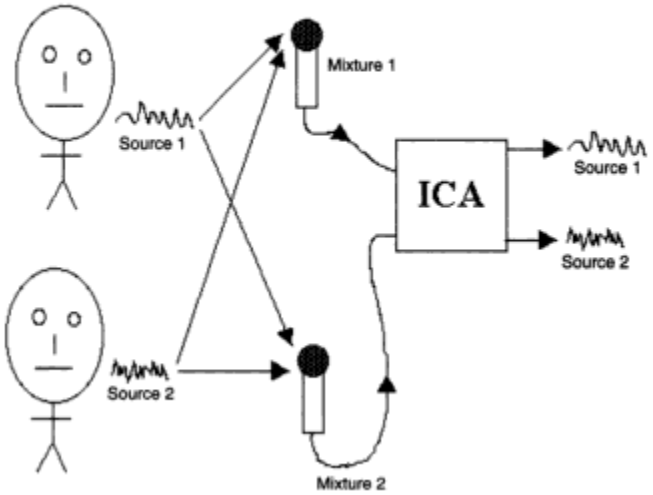
ICA is superficially related to principal component analysis and factor analysis. ICA is a much more powerful technique, however, capable of finding the underlying factors or sources when these classic methods fail completely.

The data analyzed by ICA could originate from many different kinds of application fields, including digital images, document databases, economic indicators and psychometric measurements. In many cases, the measurements are given as a set of parallel signals or time series; the term blind source separation is used to characterize this problem. Typical examples are mixtures of simultaneous speech signals that have been picked up by several microphones, brain waves recorded by multiple sensors, interfering radio signals arriving at a mobile phone, or parallel time series obtained from some industrial process.

1.2.1 DEFINITION

When the independence assumption is correct, blind ICA separation of a mixed signal gives very good results. It is also used for signals that are not supposed to be generated by a mixing for analysis purposes. A simple application of ICA is the “cocktail party problem”, where the underlying speech signals are separated from a sample data consisting of people talking simultaneously in a room. Usually the problem is simplified by assuming no time delays and echoes. An important note to consider is that if *N* sources are present, at least *N* observations

(e.g. microphones) are needed to get the original signals. This constitutes the square ($J = D$, where D is the input dimension of the data and J is the dimension of the model). Other cases of underdetermined ($J < D$) and overdetermined ($J > D$) have been investigated.



. Non-Gaussianity, motivated by the **central limit theorem**, is one method for measuring the independence of the components. Non-Gaussianity can be measured, for instance, by kurtosis or approximations of entropy. Mutual information is another popular criterion for measuring statistical independence of signals.

x(t)= As(t) …….(1)

y(t)= Wx(t) ……...(2)

ICA goal is finding a linear transform given by matrix W so that

the Output y(t) is the copy or estimate of source signal s(t):

In which s(t)=[s1, s2, ...sn] is a 1xn vector

composed by n source signals, x(t)=[x1, x2,.xn]T

is a (nx1) vector composed of n measuring signals and the (nxn) matrix A is called as mixture matrix. That is to be more divulsive,

The components x_i of the observed random vector $x = (x_1, \ldots, x_m)^T$ are generated as a sum of the independent components s_k, $k = 1, \ldots, n$:

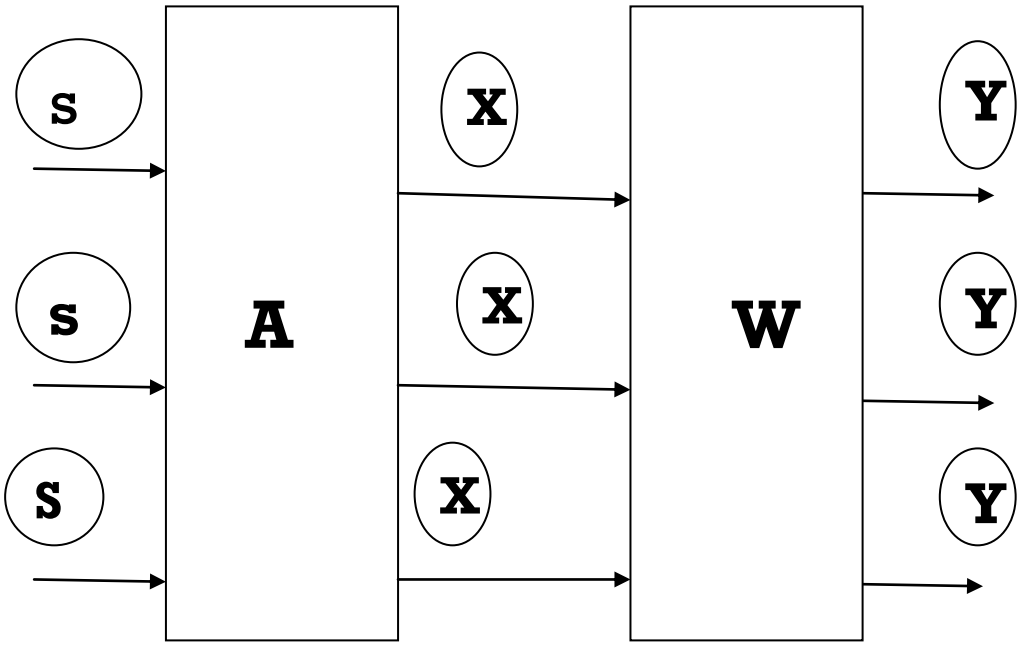
$$x_i = a_{i,1}s_1 + \ldots + a_{i,k}s_k + \cdots + a_{i,n}s_n$$

weighted by the mixing weights $a_{i,k}$.The same generative model can be written in vectorial form as $x = \sum_{k=1}^{n} s_k a_k$, where the observed random vector x is represented by the basis vectors $a_k = (a_{1,k}, \ldots, a_{m,k})^T$. The basis vectors a_k form the columns of the mixing matrix $A = (a_1, \ldots, a_n)$ and the generative formula can be written as $\boldsymbol{x = As}$, where $s = (s_1, \ldots, s_n)^T$.

Given the model and realizations (samples) $x_1, \ldots, x_N$ of the random vector x, the task is to estimate both the mixing matrix ***A*** and the sources ***s*** . This is done by adaptively calculating the ***w*** vectors and setting up a cost function which either maximizes the nongaussianity of the calculated $s_k = (w^T * x)$ or minimizes the mutual information. In some cases, a priori knowledge of the probability distributions of the sources can be used in the cost function.

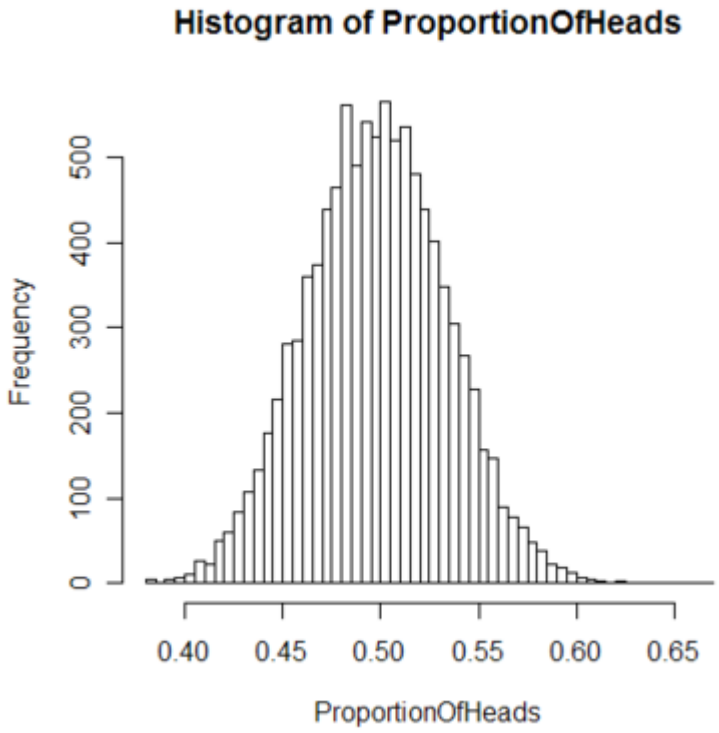
The original sources s can be recovered by multiplying the observed signals x with the inverse of the mixing matrix $W = A^{-1}$, also known as the "immixing matrix".

1.2.2 LINEAR MIXTURE AND SEPARATION MODEL



1.2.3 CENTRAL LIMIT THEOREM (CLT)

The central limit theorem (CLT) states that the re-averaged sum of a sufficiently large number of identically distributed independent random each with finite mean and variance will be approximately normally distributed (Rice 1995). Formally, a **central limit theorem** is any of a set of weak-convergence results improbability theory. They all express the fact that any sum of many independent identically distributed random variables will tend to be distributed according to a particular "attractor distribution".



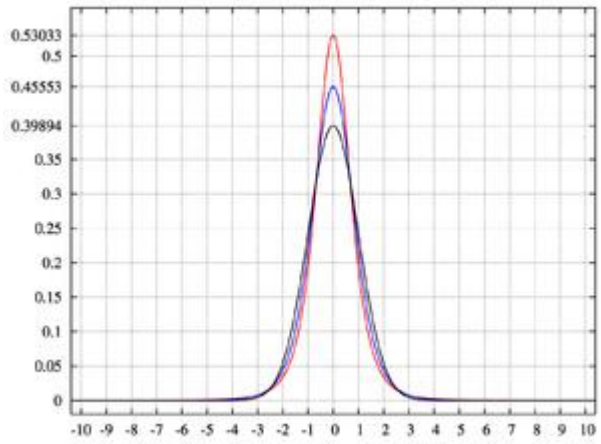
In probability theory and statistics, **kurtosis** (from the Greek word κυρτός, *kyrtos* or *kurtos*, meaning bulging) is a measure of the "peakedness" of the probability distribution of a real-valued random variable. Higher kurtosis means more of the variance is due to infrequent extreme deviations, as opposed to frequent modestly-sized deviations.

the fourth standardized moment is defined as

$$\frac{\mu_4}{\sigma^4},$$

where μ_4 is the fourth moment about the mean and σ is the standard deviation. This is sometimes used as the definition of kurtosis in older works, but is not the definition used here.

Graph of kurtosis



Typical algorithms for ICA use centering, whitening (usually with the Eigen value decomposition), and dimensionality reduction as preprocessing steps in order to simplify and reduce the complexity of the problem for the actual iterative algorithm. Whitening and dimension reduction can be achieved with principal component analysis or singular value decomposition. Whitening ensures that all dimensions are treated equally *a priori* before the algorithm is run. Algorithms for ICA include infomax , FastICA , and JADE, but there are many others also.

In general, ICA cannot identify the actual number of source signals, a uniquely correct ordering of the source signals, nor the proper scaling (including sign) of the source signals.

ICA is important to blind signal separation and has many practical applications. It is closely related to (or even a special case of) the search for a factorial code of the data, i.e., a new

vector-valued representation of each data vector such that it gets uniquely encoded by the resulting code vector (loss-free coding), but the code components are statistically independent

1.3 FASTICA

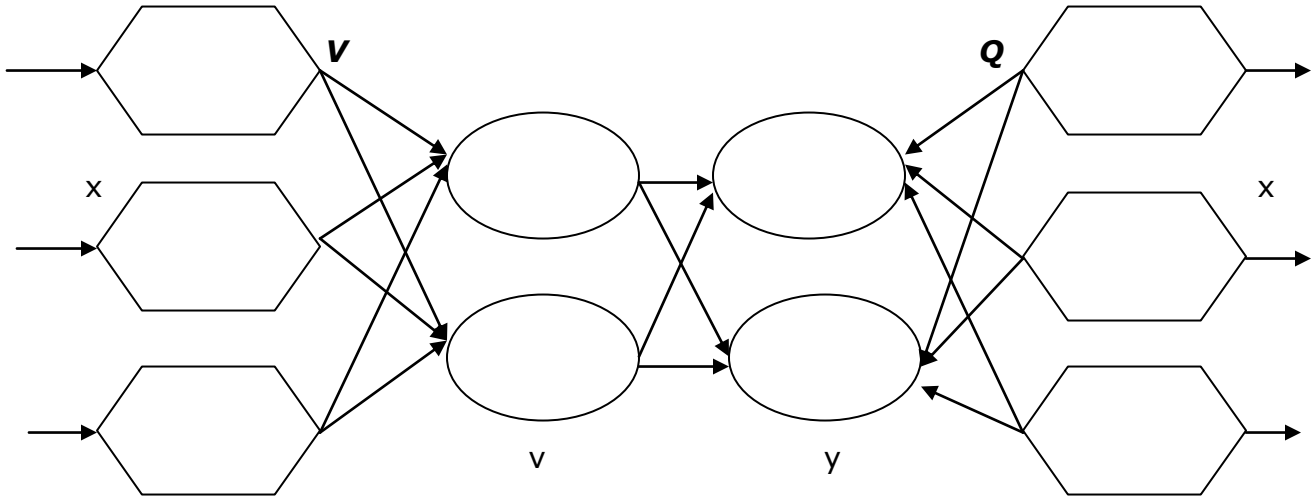
FastICA is an efficient and popular algorithm for independent component analysis invented by Aapo Hyvärinen at Helsinki University of Technology. The algorithm is based on a fixed-point iteration scheme maximizing non-Gaussianity as a measure of statistical independence. It can be also derived as an approximate Newton iteration. The fast fixed point algorithm for ICA converges rapidly to the most accurate solution allowed by data structure.

The basic linear relationship for ICA problem is taken to be

$$x=As$$

s::statistically independent signal A transformation V can be found using standard PCA methods, such that the observed data are linearly transformed to a vector

$$v=V x \ldots\ldots\ldots\ldots (3)$$



Whitening||Separation||Estimation of ICA basis vectors

Prewhitening serves two purposes::sphering the data and determining the number of independent component.

$$v=VAs=Bs$$

where $B=VA$ is an orthogonal matrix.

1.3.1 FIXED POINT ICA ALGORITHM FOR ICA

Step 1.. Prewhiten the observed data x to obtain vector v.

Step 2.. Randomly set the values of initial weight vector w(0) and normalize to the unit length, that is,

$$w(0) \longleftarrow w(0)/||w(0)||_2$$

and set j=1.

Step 3.. Let

$$W(j)=E[v(w^T(j-1)v]-3w(j-1)$$

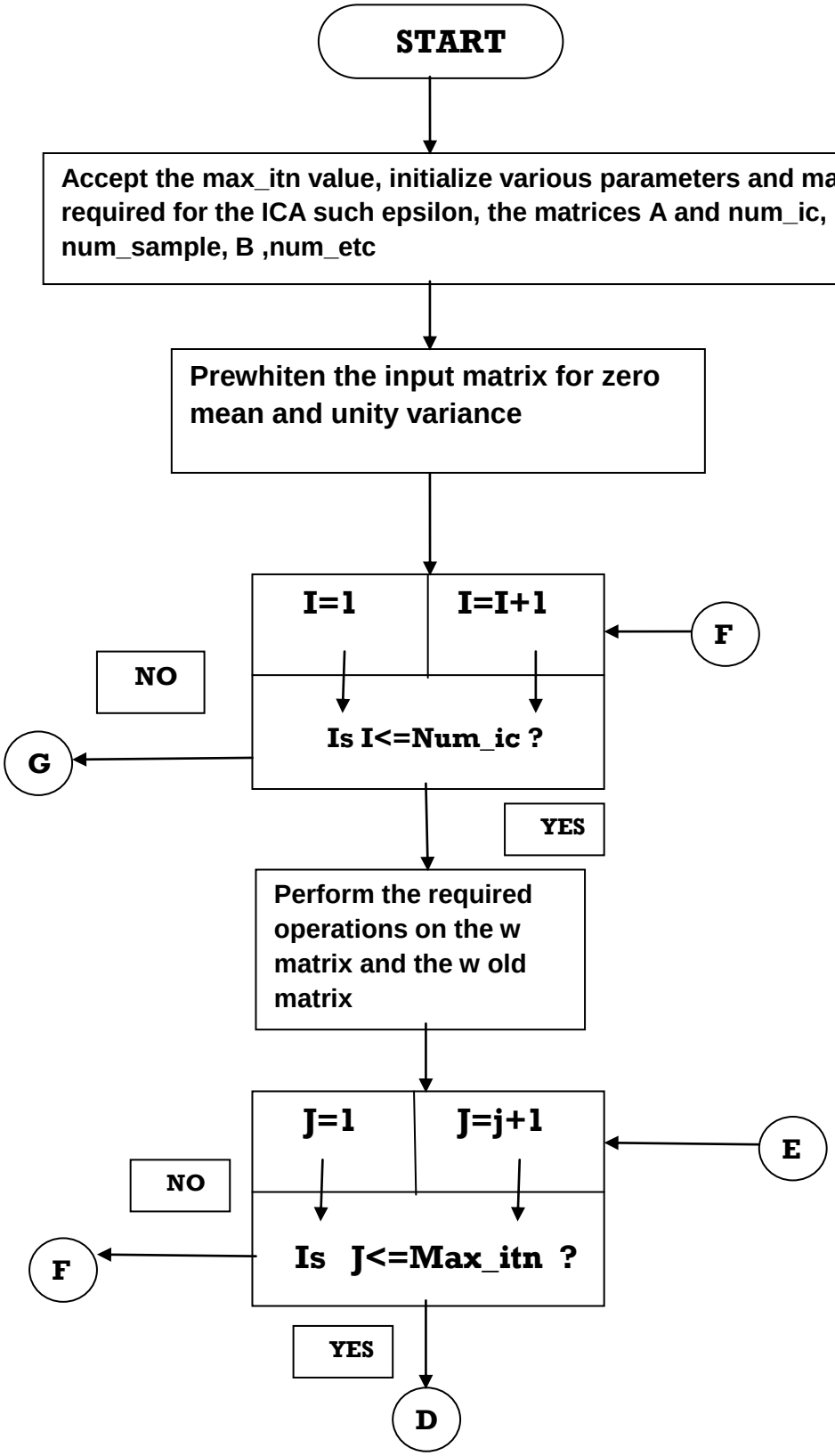
The expectation operator can be estimated using a relatively large number of v vector

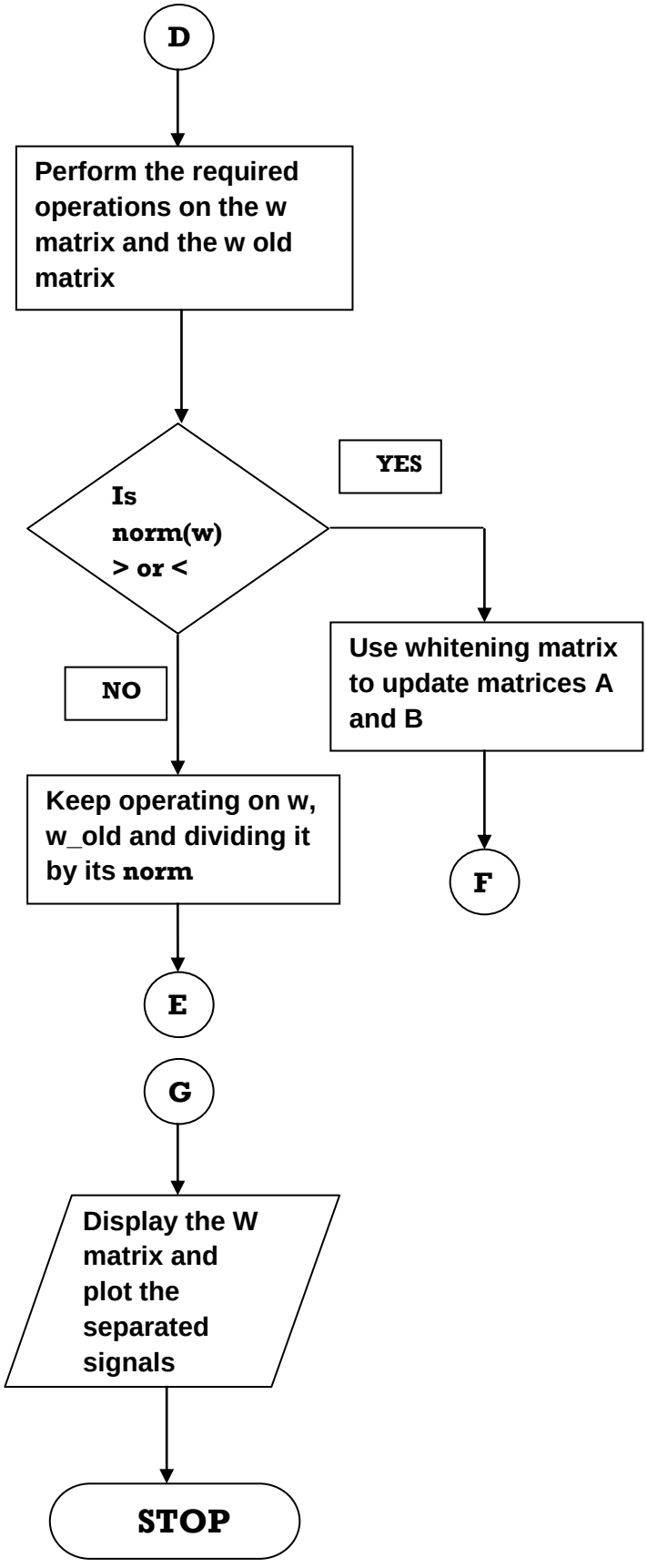
Step 4.. Normalise *w(j)* to the unit length

$$w(j) \longleftarrow w(j)/||w(j)||_2$$

Step 5..*If* $|w^T(j)*w(j-1)|$ is not close to 1,let j$\longleftarrow$j+1 and go to the step 3.Otherwise, output vector w(j).

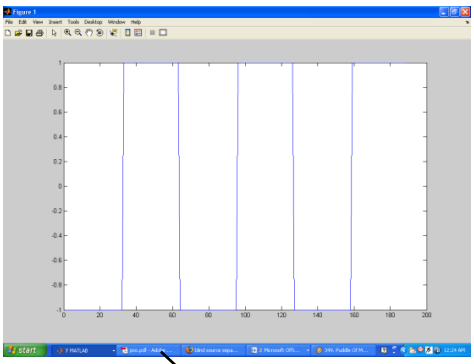
1.4 FLOW CHART OF FAST INDEPENDENT COMPONENT ANALYSIS ALGORITHM



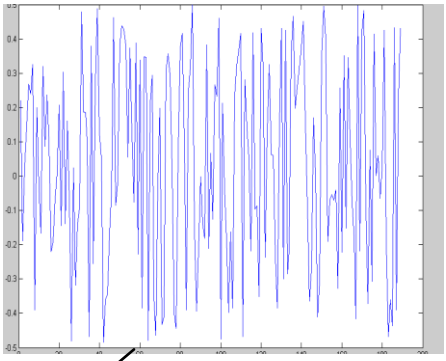

D
Perform the required operations on the w matrix and the w old matrix
Is norm(w) > or <
YES
Use whitening matrix to update matrices A and B
F
NO
Keep operating on w, w_old and dividing it by its norm
E
G
Display the W matrix and plot the separated signals
STOP

1.5 RESULT OF BLIND SOURCE SEPARATION USING FASTICA

INPUT SIGNAL 1

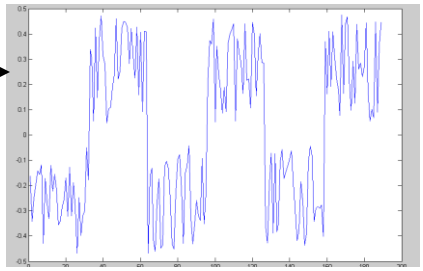


INPUT SIGNAL 2 (RANDOM NOISE)



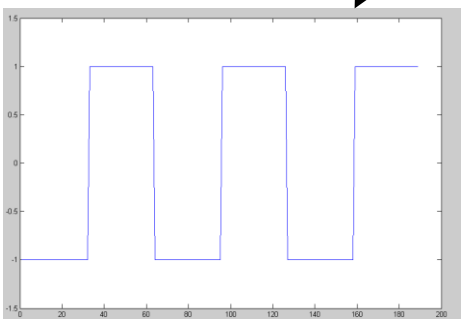
MIXED SIGNAL

LINEAR MIXING

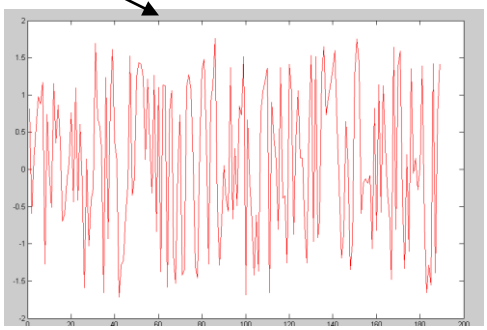


FAST ICA

SIGNAL 1



SIGNAL 2



Chapter 2

PARTILCLE SWARM OPTIMIZATION BASED LEARNING

2.1 Introduction to PSO

2.2 A basic canonical PSO algorithm

2.3 Flow chart of PSO algorithm

2.4 blind source separation using PSO and ICA

2.5 Nonlinear blind separation based on PSO

2.6 Block diagram of nonlinear ICA and PSO

2.7Result of blind source separation using ICA and PSO

2.1 INRODUCTION TO PARTILCLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

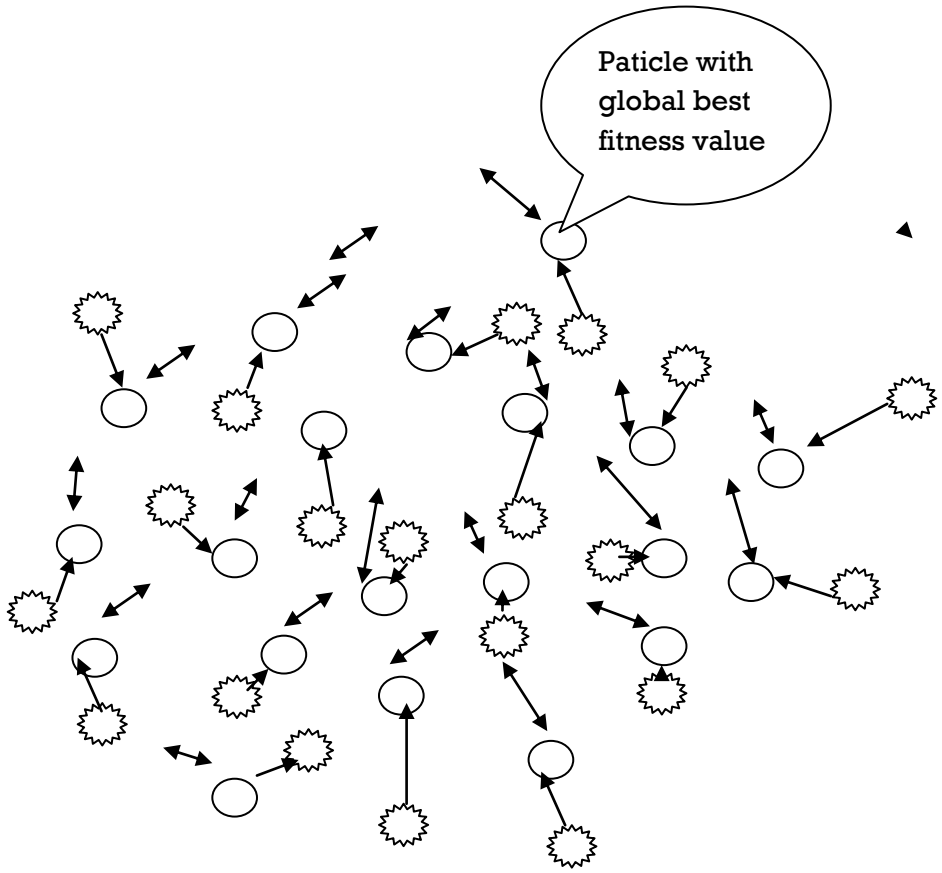
PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called*lbest*. when a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.



Particle's current position

Particle's previous best position

Future direction of motion of the particle

Previous direction of motion of the particle

2.1.1 REPRESENTATION OF A PARTICLE SWARM

- Particle's next direction and velocity of motion decided from its position's difference from that of its previous best position and the global best position

2.2 A BASIC CANONICAL PSO ALGORITHM

The algorithm presented below uses the global best and local bests but no neighborhood bests. Neighborhood bests allow parallel exploration of the search space and reduce the susceptibility of PSO to falling into local minima, but slow down convergence speed. Note that neighborhoods merely slow down the proliferation of new bests, rather than creating isolated subswarms because of the overlapping of neighborhoods: to make neighborhoods of size 3, say, particle 1 would only communicate with particles 2 through 5, particle 2 with 3 through 6, and so on. But then a new best position discovered by particle 2's neighborhood would be communicated to particle 1's neighborhood at the next iteration of the PSO algorithm presented below. Smaller neighborhoods lead to slower convergence, while larger neighborhoods to faster convergence, with a global best representing a neighborhood consisting of the entire swarm. The tendency is now to use partly random neighborhoods (see Standard PSO on the Particle Swarm Central).

A single particle by itself is unable to accomplish anything. The power is in interactive collaboration.

Let $f : \mathbb{R}^m \to \mathbb{R}$ be the fitness function that takes a particle's solution with several components in higher dimensional space and maps it to a single dimension metric. Let there be *n* particles, each with associated positions $\mathbf{x}_i \in \mathbb{R}^m$ and velocities $\mathbf{v}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$. Let $\hat{\mathbf{x}}_i$ be the current best position of each particle and let $\hat{\mathbf{g}}$ be the global best.

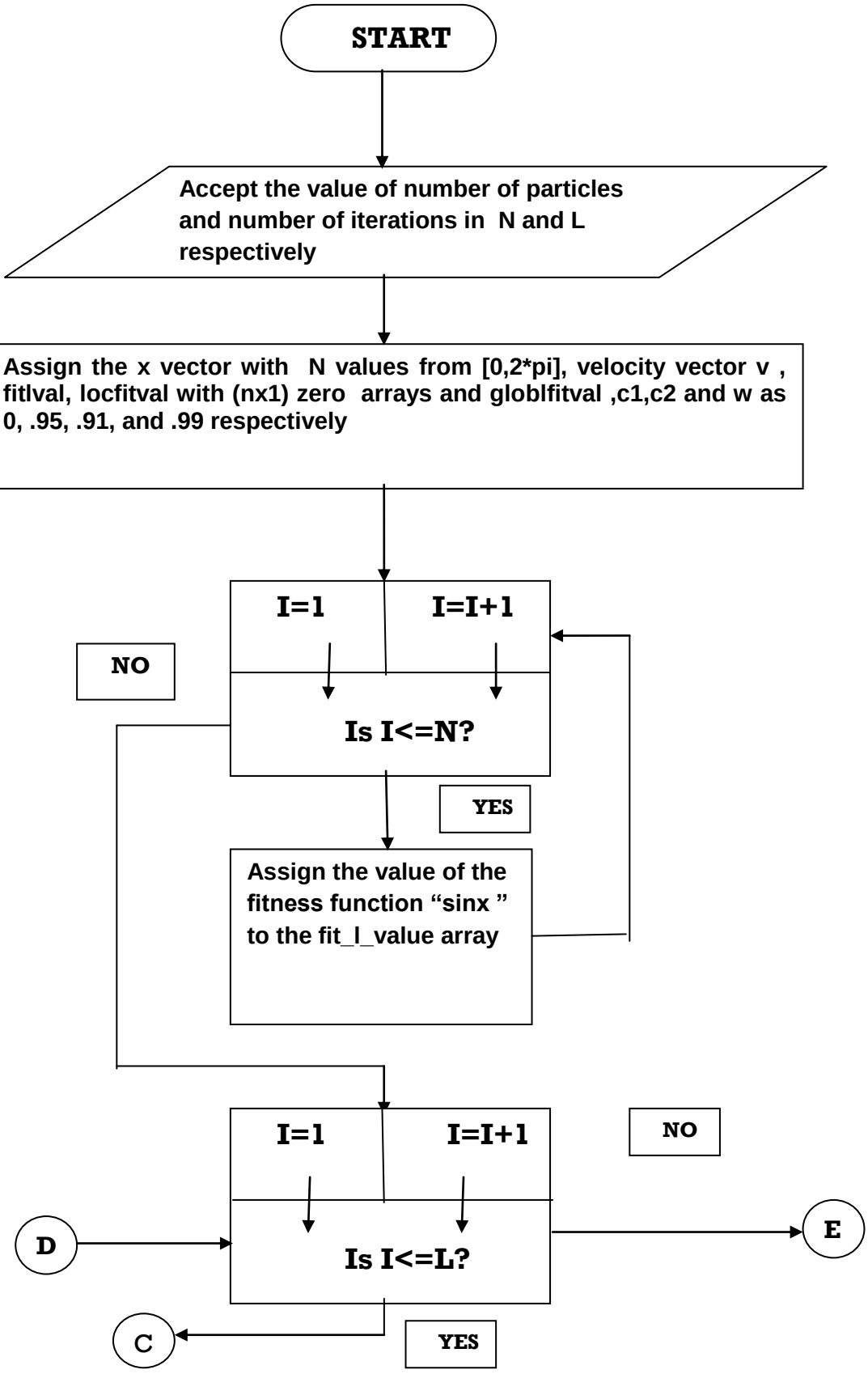
- Initialize $\mathbf{x}_i$ and $\mathbf{v}_i$ for all *i*. One common choice is to take $\mathbf{x}_{ij} \in U[a_j, b_j]$ and $\mathbf{v}_i = \mathbf{0}$ for all *i* and $j = 1, \ldots, m$, where a_j,b_j are the limits of the search domain in each dimension, and *U* represents the Uniform distribution (continuous).
- $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i$ and $\hat{\mathbf{g}} \leftarrow \arg\min_{\mathbf{x}_i} f(\mathbf{x}_i), i = 1, \ldots, n$.
- While not converged:
  - For each particle $1 \leq i \leq n$:
    - Create random vectors $\mathbf{r}_1$, $\mathbf{r}_2$: $\mathbf{r}_{1j}$ and $\mathbf{r}_{2j}$ for all *j*, by taking $\mathbf{r}_{1j}, \mathbf{r}_{2j} \in U[0, 1]$ for $j = 1, \ldots, m$
    - Update the particle positions: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$.

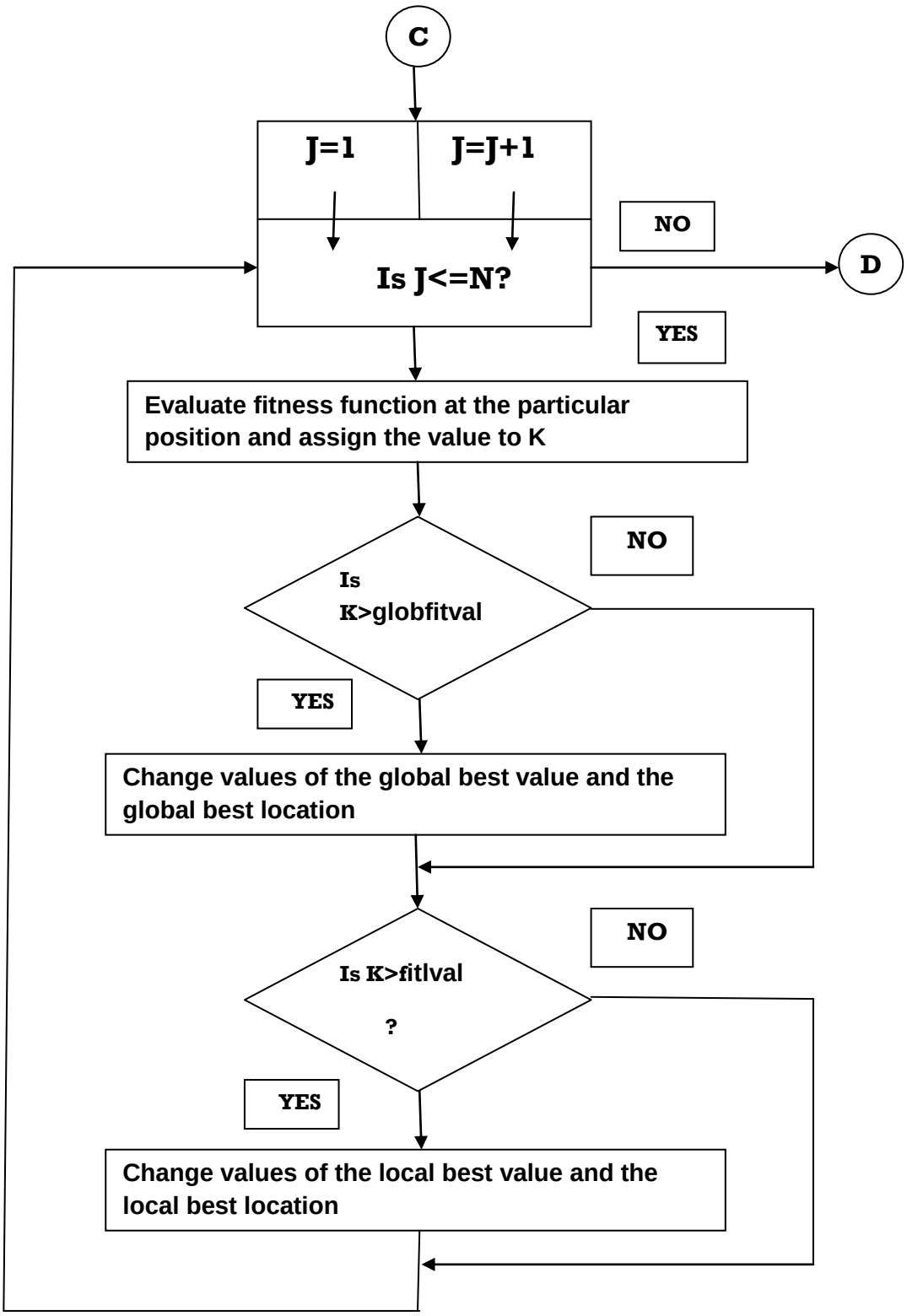
    - Update the particle velocities:
      $\mathbf{v}_i \leftarrow \omega \mathbf{v}_i + c_1 \mathbf{r}_1 \circ (\hat{\mathbf{x}}_i - \mathbf{x}_i) + c_2 \mathbf{r}_2 \circ (\hat{\mathbf{g}} - \mathbf{x}_i)$.
    - Update the local bests: If $f(\mathbf{x}_i) < f(\hat{\mathbf{x}}_i)$, $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i$.
    - Update the global best If $f(\mathbf{x}_i) < f(\hat{\mathbf{g}})$, $\hat{\mathbf{g}} \leftarrow \mathbf{x}_i$.
- $\hat{\mathbf{g}}$is the optimal solution with fitness $f(\hat{\mathbf{g}})$.

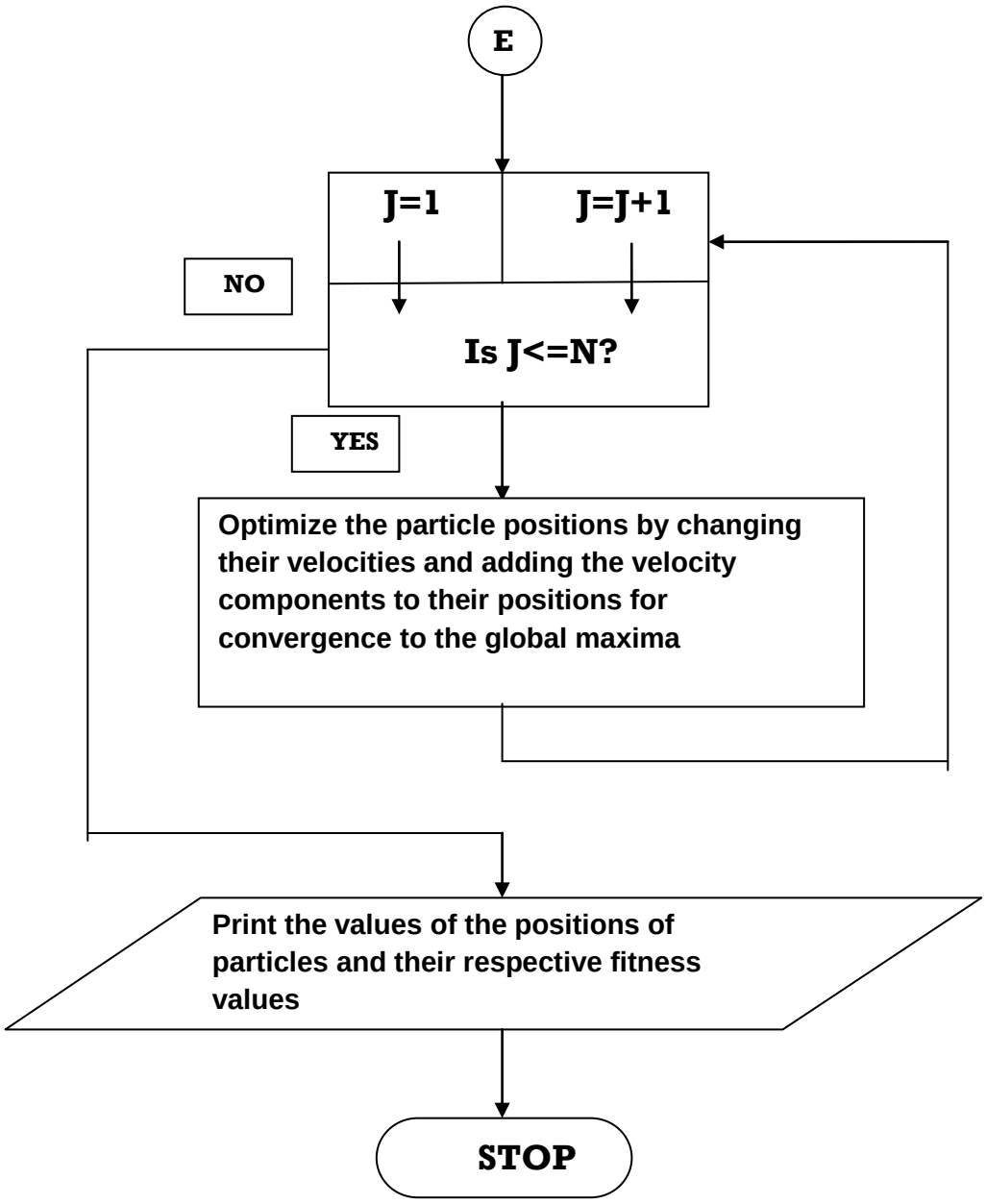
Note the following about the above algorithm:

- ω is an inertial constant. Good values are usually slightly less than 1.
- c_1 and c_2 are constants that say how much the particle is directed towards good positions. They represent a "cognitive" and a "social" component, respectively, in that they affect how much the particle's personal best and the global best (respectively) influence its movement. Usually we take $c_1, c_2 \approx 2$.
- $\mathbf{r}_1, \mathbf{r}_2$are two random vectors with each component generally a uniform random number between 0 and 1.
- ∘operator indicates element-by-element multiplication i.e. the Hadamard matrix multiplication operator.

2.3 FLOW CHART OF PSO ALGORITHM




C
J=1
J=J+1
Is J<=N?
NO
D
YES
Evaluate fitness function at the particular position and assign the value to K
Is K>globfitval
NO
YES
Change values of the global best value and the global best location
Is K>fitlval ?
NO
YES
Change values of the local best value and the local best location


E
J=1
J=J+1
NO
Is J<=N?
YES
Optimize the particle positions by changing their velocities and adding the velocity components to their positions for convergence to the global maxima
Print the values of the positions of particles and their respective fitness values
STOP

2.4 BLIND SOURCE SEPARATION USING PSO AND ICA

2.4.1 INTRODUCTION

In practical circumstances, mixture signal may be mixed by nonlinear system, which leads to a more complicated result than the linear mixture signal. Linear blind separation algorithm isn't suitable no longer.

Nowadays, two kinds of research methods are developed for nonlinear blind separation: the first one extracts the nonlinearity by adopting SOFM, and the second one uses the nonlinear mixture model to
match the practical mixture nonlinear system on the base of linear blind separation. The former makes the network complexity increase exponentially when the number of source signal is large, and comes into being interpolation error when continuous source signal is separated, the latter adopts Newton iteration method, gradient method and natural gradient method to solve problems. And yet the NP problem applied in nonlinear source blind separation is still hard to achieve global optimal solution. Recently many researchers use GA for nonlinear blind separation, which has a good effect but slow convergence rate. The combination of PSO and natural gradient method is proposed for the nonlinear mixture signal blind separation in this paper. The nonlinear signal blind separation algorithm based on particle swarm is established in which high-order odd polynomial is used to fit the nonlinear mixed function and create nonlinear signal blind separation model, PSO is adopted to work out the parameters of polynomial as natural gradient method to iterative linear non-mixed matrix.

2.4.2 NONLINEAR SIGNAL BLIND SEPARATION MODEL

The blind separation of sources problem can be approached by wider point of view by using Independent Component Analysis(ICA). ICA goal is finding a linear transform given by matrix W so that the Output y(t) is the copy or estimate of source signal s(t):

x(t)=As(t) ……..(1)

y(t)=wx(t)……..(2)

In which s(t)=[si, s2,…. ..sn] is a nx l vector composed by n source signals, x(t)=[xi, x2,.xn]T is a n x 1 vector composed of n measuring signals and nxn matrix A is called as mixture matrix.

Nevertheless, the linear mixing model may not be appropriated for some real environment. Even though the nonlinear mixing model is more realistic and practical, most existing algorithms for the BSS problem were developed for the linear model. However,for nonlinear mixing models, many difficulties occur and both the linear ICA and the existing linear demixing methodologies are no longer

applicable because the complexity of nonlinear parameters. Bure[l] resolves the two problems by using the nonlinear model with two-layer sensor construction. The relationship between measuring signal and source signal in nonlinear model is defined as follows:

x(t)=F[As(t)] ……(3)

where F=[fi, f2, .fn]T is a reversible nonlinear transform matrix. If F is linear, then formula (3) degenerates to formula (1).

2.4.3 NON LINEAR MIXTURE AND SEPARATION MODEL

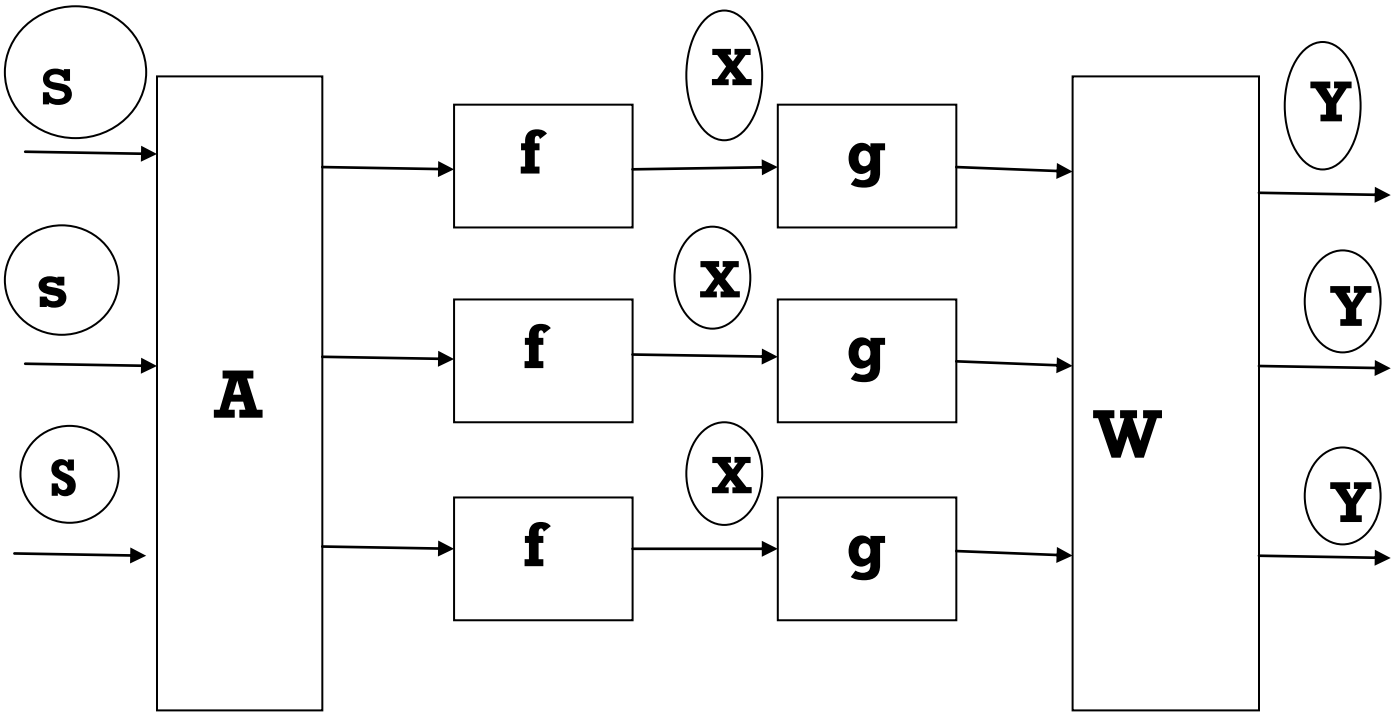


Figure above describes the nonlinear model. There are two parts in mixture system: the linear mixture based on mixture matrix A and each channel's independent nonlinear transform function fi , while separation system is opposite: each channel's independent nonlinear inverse transform g1 and linear separation of matrix W. The output yi(t) can be defined as:

$$yi(t)=\sum_{j-1}^{n} w_{ij} g_j(x_j(t)) \qquad \text{.....(4)}$$

The linear inverse function is usually hard to determine for there is no prior knowledge in mixture and separation system, but we can also fit them by odd polynomial as most of them are origin symmetry.

$$g_j(x_j) - \sum_{k-1}^{p} g_{jk} x_j^{2k-1}$$

$$\text{Then:} \quad yi(t) = \sum_{j-1}^{n} w_{ij} \sum_{k-1}^{p} g_{jk} x_j^{2k-1}$$

$g^j=[\, g_{j1}, g_{j2}, \ldots\ldots g_{jp}]^T$ is the parameter of nonlinear transform function in j channel. A big difficulty in nonlinear model is the parameter computation, as it presents a problem with numerous local minima. Thus we require an algorithm that is capable of avoiding entrapment in such a minimum. As a solution to this first unmixing stage, we propose the Particle Swarm Optimization(PSO) which has been applied successfully in all kinds of multidimensional continuous space optimization problems.

2.5 NONLINEAR BLIND SEPARATION BASED ON PSO

PSO algorithm resembles a school of flying bird developed by Kennedy and Eberharr6] in 1995. It can reach population optimization through collectivity cooperation. Each individual is named as a "particle" which, in fact, represents a potential solution to a problem. Each particle adjusts its flying according to its own flying experience and its companions' flying experience. Each particle is treated as a point in a D-dimensional space. The i^{th} particle is represented as

$X_i=(\, x_{i1}, x_{i2}, \ldots, x_{iD}\,)$. The best previous position (the position giving the best fitness value) of any particle is recorded and represented as

$P_i=(p_{i1},p_{i2},\ldots,p_{iD})$ - The index of the best particle among all the particle in the population is represented by the symbol p_{gd}. The rate of the position change (velocity) for particle i^{th} is represented as

$V_i=(v_{i1}, v_{i2},\ldots, v_{iD})$.

The particles are manipulated according to the following equation:

$$v_{id}(t+1) = w\times v_{id}(t)+c_1\times rand()\times[p_{id}(t)-x_{id}(t)] + c_2\times rand()\times[p_{gd}(t)-x_{id}(t)] \qquad (7)$$

$$x_{id}(t+1) = x_{id}(t)+v_{id}(t+1)$$

$$1\le i\le n, 1\le d\le D \qquad (8)$$

Where ci and c2 are two positive constants called acceleration constants, rand() is the random function in the range[0,l], w is the inertia factor when big for global exploration and small for local exploitation. The position range is [-XMAXd , XMAXd] and velocity range is [-VMAXd , VMAXd] in d^{th} $1\le d\le D$ dimension. The boundary value will be chosen if position or speed goes beyond the range when iterating. The particle's position and velocity are initialized at random, then iterated according to equations (7) and (8) until a satisfactory solution is searched out.Analyzing equation (7), we can find out that if the best previous position Pi doesn't change in a long time, the velocity update will be mainly determined by $w*v_{id}(t)$ and the velocity will be slower and slower when the particle is near Pi. So the particle swarm reflects a strong coherence, which means the particle swarm converges quickly.

2.5.1 Evaluation function::

To perform the PSO, first is very important to define the fitness function. this fitness function is constructed having in mind that the output sources must be independent from their nonlinear mixtures. For this purpose, we mutual information is adopted as the evaluation function as the measure of independence:

I(y)==log|W|-∑ E [∑(2k-1)*xi^(2k-2)*gik] + ∑H(y(i))

When I(y)=0 means that yi is independent of each other, and the calculation of $H(y_i)$ needs to estimate the distribution density function of y_i firstly. In fact, the distribution density function of y_i is unknown, which can be approached by Edgeworth expansion. But in this paper, Gram-charlier expansion is chosen for the divergence problem in training existing in Edge-worth expansion algorithm. The entropy of each component is only related to the three-order and four-order commutations:

$$H(yi)=\frac{\log(2\pi e)}{2}-\frac{1}{2\cdot 3!}(k_3^i)^2-\frac{1}{2\cdot 4!}(k_4^i)^2 + \frac{3}{8}(k_3^i)^2 k_4^i+\frac{1}{16}(k_4^i)^3 \qquad (13)$$

In which , $k_3^i = m_3^i, k_4^i = m_4^i - 3$, centralizing and prewhitening on the signal should be done before calculation assuring the expectation value equals '0' and variance equals 1. In order to enhance the independence of all random variables, the reciprocal value of equation (9) is chosen as the evaluation function. When the reciprocal value reaches maximum, each signal is independent of each other:

$$\text{Eval-function}(y)=\frac{1}{I(y)} \qquad \text{.....(15)}$$

2.5.2 IMPLEMENTATION OF ALGORITHM::

The parameter space of nonlinear mixed andseparate system can be divided into two parts: linear parameter space and nonlinear parameter space. Through the analysis above it's easy to know that the two spaces are independent, and so is the nonlinear parameter space of each channel. Let

$$G=[G_1 \otimes G_2 \otimes \ldots \otimes G_n] \otimes G_j=[g_{j1} \otimes g_{j2} \otimes \ldots \otimes g_{jp}]$$

for nonlinear parameter space .Independent component analysis is used for the estimation of linear parameter W.

Step 1: Fetch source signal; centralize and prewhiten the signal.

Step 2: Initialization

2.1 initialize the parameters of nonlinearnon-mixed function. Generate initial particles

$G=[G_1 \mathbb{X} G_2 \mathbb{X} \ldots\ldots \mathbb{X} G_n] \mathbb{X} G_j=[g_{j1} \mathbb{X} g_{j2} \mathbb{X} \ldots\ldots \mathbb{X} g_{jp}]$

randomly.

2.2 initialize the parameters of linear non-mixed matrix. Generate W=rand() randomly and standardize

$W=W/||W||.$

Step 3: Centralize and prewhiten yi(t) and calculate the evaluation function according to equation (14).

3.1 if a certain particle's current evaluation value is better than the best previous evaluation value, then set the best previous evaluation value equal to the current evaluation value and the best previous position to the particle's current position.

3.2 seek the optimum solution for all local and global populations and if better than the best previous solution then update P_l and P_g.

3.3 run orient-migration and local deeply search when the best previous position Pi doesn't change for a long time or changes little continuously.

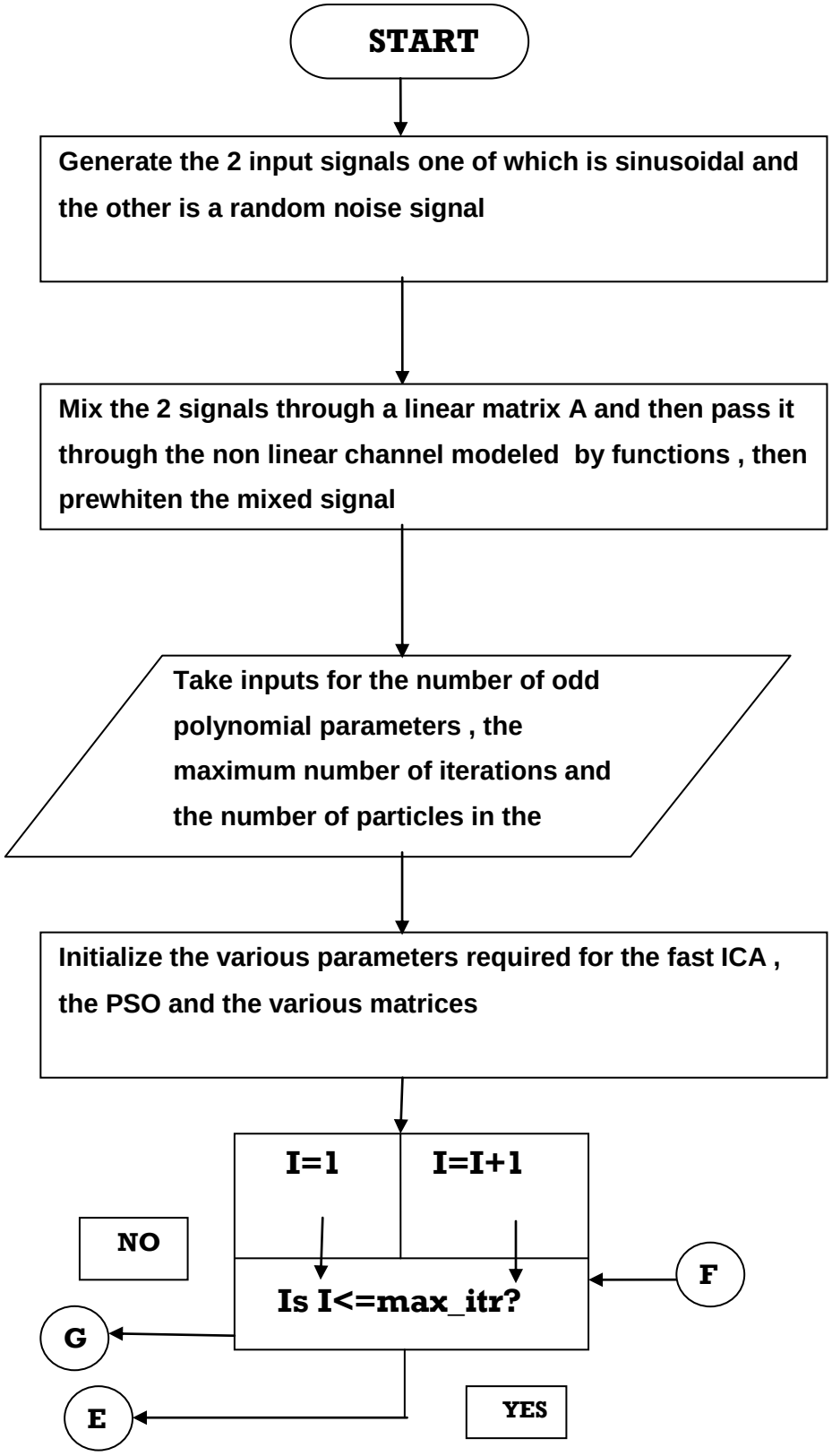
Step 4: Parameter update

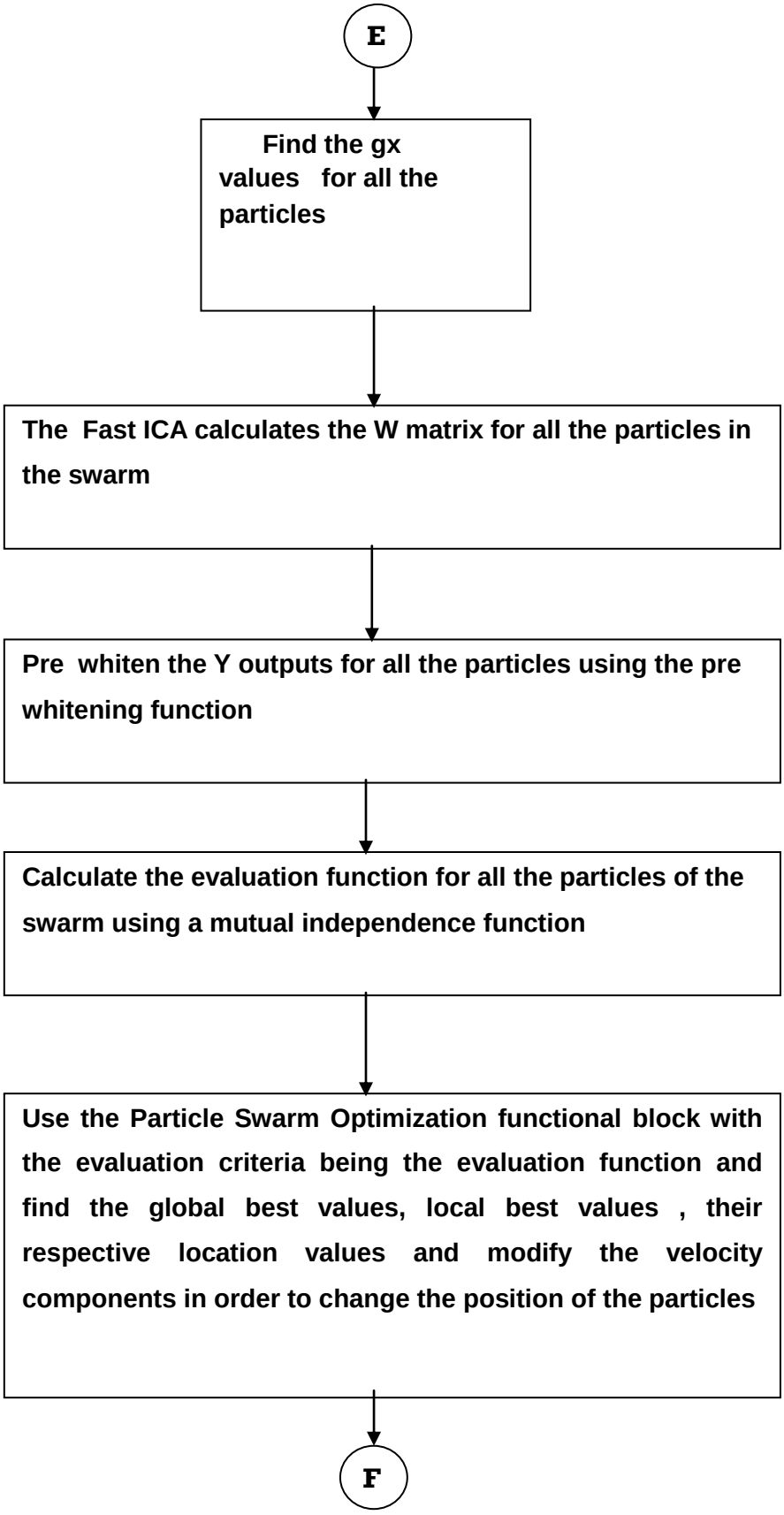
4.1 for nonlinear parameter, every particle calculate $\mathbb{X}_v$ and $\mathbb{X}_r$ according to equation (7), X_v and X_r equation (8).

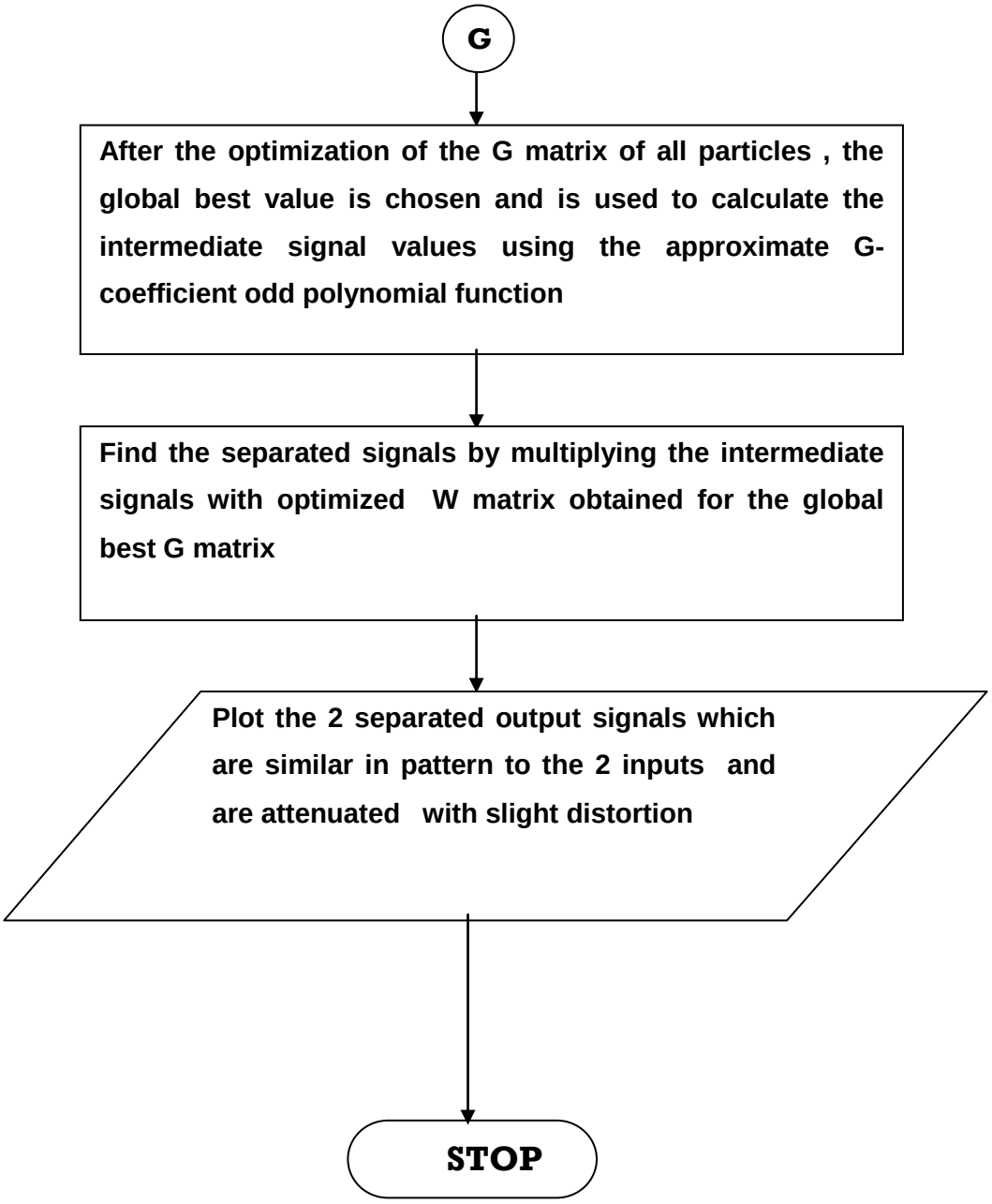
4.2 for linear parameter, update according to equation (15) by natural gradient algorithm and re-standardize W.

Step 5: Loop to step (3) until a criterion is met or a maximum number of iterations.

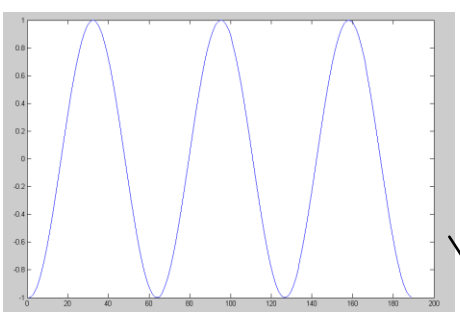
2.6 Block diagram of nonlinear ICA and PSO is given below::



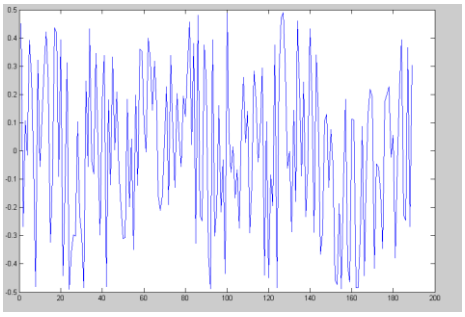

E
Find the gx values for all the particles
The Fast ICA calculates the W matrix for all the particles in the swarm
Pre whiten the Y outputs for all the particles using the pre whitening function
Calculate the evaluation function for all the particles of the swarm using a mutual independence function
Use the Particle Swarm Optimization functional block with the evaluation criteria being the evaluation function and find the global best values, local best values , their respective location values and modify the velocity components in order to change the position of the particles
F


G
After the optimization of the G matrix of all particles , the global best value is chosen and is used to calculate the intermediate signal values using the approximate G-coefficient odd polynomial function
Find the separated signals by multiplying the intermediate signals with optimized W matrix obtained for the global best G matrix
Plot the 2 separated output signals which are similar in pattern to the 2 inputs and are attenuated with slight distortion
STOP

2.7 RESULT OF BLIND SOURCE SEPARATION USING ICA AND PSO

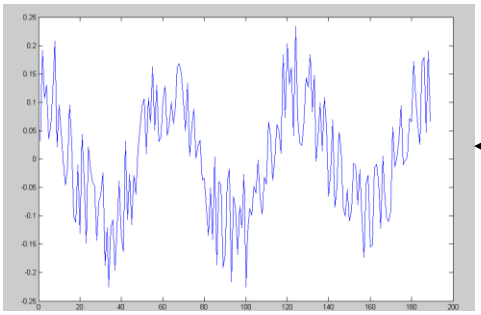


INPUT 1

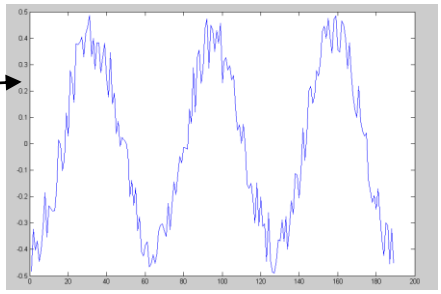


INPUT 2

Linear Mixer and Non Linear Channel

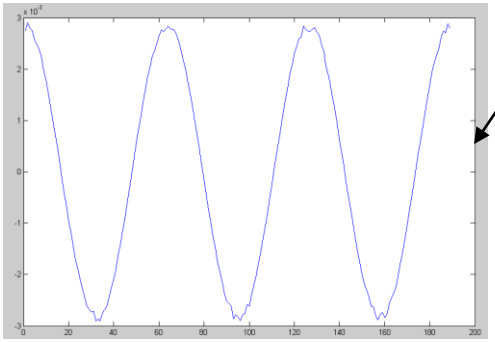


MIXED SIGNAL1

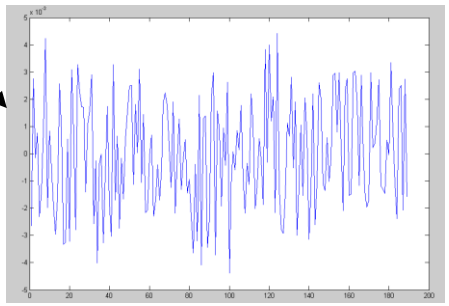


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and PSO for approximation of the non linear functions



OUTPUT SIGNAL 1



OUTPUT SIGNAL 2

Chapter 3

GENETIC ALGORITHM BASED LEARNING

3.1Introduction to genetic algorithm

3.2Simple generational genetic algorithm pseudo code

3.3GA element

3.4Observations

3.5 Nonlinear blind separation based on GA

3.6Block diagram of nonlinear ICA using real coded GA

3.7 Block diagram of nonlinear ICA using binary GA

3.8 Result of BSS using ICA and real coded GA(sine wave & random noise)

3.9 result of BSS using ICA and real coded GA(square wave & random noise)

3.10 Result of BSS using ICA and BINARY GA(sine wave & random noise)

3.11 Result of BSS using ICA and binary GA(square wave & random noise)

3.1INTRODUCTION TO GENETIC ALGORITHM

3.1.1 INTRODUCTION

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (also known as evolutionary computation) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination)

3.1.2 METHODOLOGY

Genetic algorithms are implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Genetic algorithms find application in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

A typical genetic algorithm requires:

- a genetic representation of the solution domain,
- a fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used,

but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise

Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly, then improve it through repetitive application of mutation, crossover, inversion and selection operators.

- Individual - Any possible solution
- Population - Group of all individuals
- Search Space - All possible solutions to the problem
- Chromosome - Blueprint for an individual
- Trait - Possible aspect of an individual
- Allele - Possible settings for a trait
- Locus - The position of a gene on the chromosome
- Genome - Collection of all chromosomes for an individual

- ***Initialization***

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

- ***Selection***

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

Selection is a genetic operator that chooses a chromosome from the current generation's population for inclusion in the next generation's population. Before making it into the next generation's population, selected chromosomes may undergo crossover and / or mutation (depending upon the probability of crossover and mutation) in which case the offspring chromosome(s) are actually the ones that make it into the next generation's population.

Genetic Server and Genetic Library include the following types of selection:

1. **Roulette** - A selection operator in which the chance of a chromosome getting selected is proportional to its fitness (or rank). This is where the concept of survival of the fittest comes into play.
2. **Tournament** - A selection operator which uses roulette selection N times to produce a tournament subset of chromosomes. The best chromosome in this subset is then chosen as the selected chromosome. This method of selection applies addition selective pressure over plain roulette selection.
3. **Top Percent** - A selection operator which randomly selects a chromosome from the top N percent of the population as specified by the user.
4. **Ranked-** A selection operator which selects the best chromosome (as determined by fitness). there are two or more chromosomes with the same best fitness, one of them is chosen randomly.
5. **Random** - A selection operator which randomly selects a chromosome from the population.

- ***Reproduction***

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation.

For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", recent researches (Islam Abou El Ata 2006) suggested more than two "parents" are better to be used to reproduce a good quality chromosome.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

- ***Termination***

This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

3.2 SIMPLE GENERATIONAL GENETIC ALGORITHM PSEUDOCODE

- Choose initial population
- Evaluate the fitness of each individual in the population
- Repeat until termination: (time limit or sufficient fitness achieved)
- Select best-ranking individuals to reproduce
- Breed new generation through crossover and/or mutation (genetic operations) and give birth to offspring
- Evaluate the individual fitnesses of the offspring
- Replace worst ranked part of population with offspring

3.3 OBSERVATIONS

There are several general observations about the generation of solutions via a genetic algorithm:

- Repeated fitness function evaluation for complex problems is often the most prohibitive and limiting segment of artificial evolutionary algorithms. Finding optimal solution to complex high dimensional, multimodal problems often requires very expensive fitness function evaluations. In real world problems such as structural optimization problems, one single function evaluation may require several hours to several days of complete simulation. Typical optimization method can not deal with such a type of problem. In this case, it may be necessary to forgo an exact evaluation and use an approximated fitness that is computationally efficient. It is apparent that amalgamation of approximate models may be one of the most promising approaches to convincingly use EA to solve complex real life problems.
- The "better" is only in comparison to other solution. As a result, the stop criterion is not clear.
- In many problems, GAs may have a tendency to converge towards local optima or even arbitrary points rather than the global optimum of the problem. This means that it does not "know how" to sacrifice short-term fitness to gain longer-term fitness. The likelihood of this occurring depends on the shape of the fitness landscape: certain problems may provide an easy ascent towards a global optimum, others may make it easier for the function to find the local optima. This problem may be alleviated by using a different fitness function, increasing the rate of mutation, or by using

selection techniques that maintain a diverse population of solutions, although the No Free Lunch theorem proves that there is no general solution to this problem.

- Elitism is to copy the best chromosomes (solutions)to new population before applying crossover and mutation .When creating a new population by crossover or mutation the best chromosome might be lost. It forces GA to retain some number of the best individuals at each generation. Its has been found that elitism significantly improves performance.
- GAs cannot effectively solve problems in which the only fitness measure is a single right/wrong measure, as there is no way to converge on the solution (no hill to climb). In these cases, a random search may find a solution as quickly as a GA. However, if the situation allows the success/failure trial to be repeated giving (possibly) different results, then the ratio of successes to failures provides a suitable fitness measure.

- Selection is clearly an important genetic operator, but opinion is divided over the importance of crossover versus mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation, and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic). There are many references in Fogel (2006) that support the importance of mutation-based search, but across all problems the No Free Lunch theorem holds, so these opinions are without merit unless the discussion is restricted to a particular problem.
- Often, GAs can rapidly locate good solutions, even for difficult search spaces. The same is of course also true for evolution strategies and evolutionary programming.

- For specific optimization problems and problem instances, other optimization algorithms may find better solutions than genetic algorithms (given the same amount of computation time). Alternative and complementary algorithms include evolution strategies, evolutionary programming, simulated annealing, Gaussian adaptation, hill climbing, and swarm intelligence (e.g.: ant colony optimization, particle swarm optimization) and methods based on integer linear programming. The question of

which, if any, problems are suited to genetic algorithms (in the sense that such algorithms are better than others) is open and controversial

- The implementation and evaluation of the fitness function is an important factor in the speed and efficiency of the algorithm.

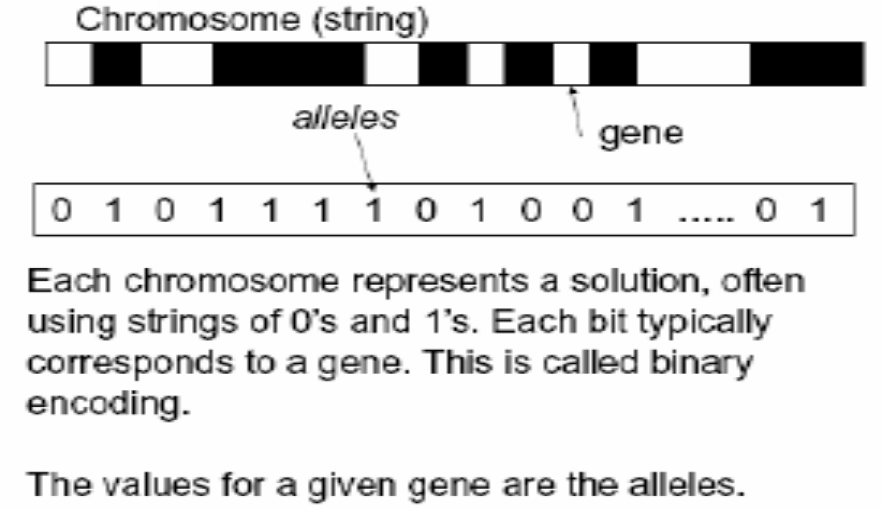
This project presents new encoding methods

- binary genetic algorithm (BGA)
- new converting methods for the real-coded genetic algorithm (RCGA).

These methods are developed for the specific case in which some parameters have to be searched in wide ranges since their actual values are not known. The oversampling effect which occurs at large values in the wide range search are reduced by adjustment of resolutions in mantissa and exponent of real numbers mapped by BGA. Owing to an intrinsic similarity in chromosomal operations, the proposed encoding methods are also applied to RCGA with remapping (converting as named above) from real numbers generated in RCGA. A simple probabilistic analysis and benchmark with two ill-scaled test functions are carried out. System identification of a simple electrical circuit is also undertaken to testify effectiveness of the proposed methods to real world problems. All the optimization results show that the proposed encoding/converting methods are more suitable for problems with ill-scaled parameters or wide parameter ranges for searching.

BINARY CODED GA:

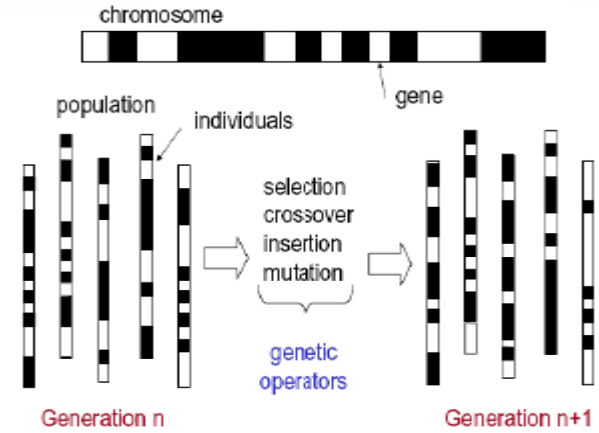
Here ,a population of random strings of 1's and 0's is initialized and rates each string according to the quality of its result. Depending on the problem, the measure of fitness could be business profitability, game payoff, error rate or any number of other criteria. High-quality strings mate; low-quality ones perish. As generations pass, strings associated with improved solutions will predominate. Furthermore, the mating process continually combines these strings in new ways, generating ever more sophisticated solutions. The kinds of problems that have yielded to the technique range from developing novel strategies in game theory to designing complex mechanical systems



REAL CODED GA:

GA tailored for the optimization in real-valued search spaces. In contrast to the canonical GA, the genome consists of (real-valued) object parameters, i.e., evolution operates on the ``natural'' representation. The real-coded GA employs special recombination operators, which are hybrid constructs of intermediate recombination (usually) and mutation. Real-coded GAs can exhibit self-adaptive behavior.

3.4 GA ELEMENT



3.5 NONLINEAR BLIND SEPARATION BASED ON GA::

If we are solving some problem, we are usually looking for some solution, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution is) is called ***search space*** (also state space). Each point in the search space represent one feasible solution. Each feasible solution can be "marked" by its value or fitness for the problem.

3.5.1 Real coded genetic algorithm

Initialization

Initially many individual solutions are randomly generated to form an initial population, covering the entire range of possible solutions (the search space)Each point in the search space represents one possible solution marked by its value(fitness)

Selection

A proportion of the existing population is selected to bread a new bread of generation.

Reproduction

Generate a second generation population of solutions from those selected through genetic operators: crossover and mutation.

In real coded genetic algorithm ,two constant are taken, alpha and beta. These two constants are use in the reproduction process.

child_1=p2+(rand*alpha*(p2-p1));

child_2=p1+(rand*alpha*(p2-p1));

where child_1,child-2 are two child population.

p1 and p2 are parent population

Constant beta is used during mutation

mc=cpop (mutpos (1,1),1)+beta*((2*rand(num_ic,p))-1);

Where, mc is the mutated child

mutpos is the position of mutation

cpop is the total child population

Termination

A solution is found that satisfies minimum criteria

- Fixed number of generations found.
- Allocated budget (computation, time/money) reached
- The highest ranking solution's fitness is reaching or has reached.

3.5.2 Binary coded genetic algorithm

Initialization

Initially many individual solutions all as the string of 1's and 0's are randomly generated to form an initial population, covering the entire range of possible solutions (the search space)Each point in the search space represents one possible solution marked by its value(fitness).after first iteration each time the previous real value has to be changed to binary for further processing

Selection

A proportion of the existing population is selected to bread a new bread of generation. In our project we have taken tournament selection

Reproduction

Generate a second generation population of solutions from those selected through genetic operators: crossover and mutation.

In binary coded genetic algorithm ,

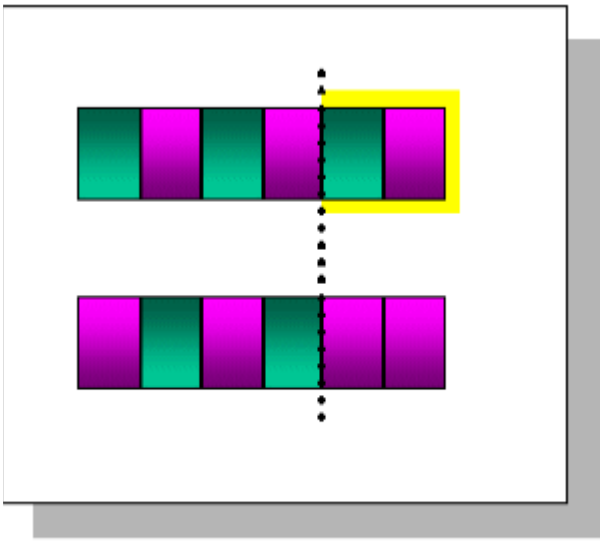
xoverchild(j,1+n:ran_cross(i)+n)=Fit_population(j+1,1+n:ran_cross(i)+n);

here, xoverchild is the child produced due to cross over

j is denotes the parameter that is to be optimized

Fit_population is total parent population

Here in crossover we are taking single point crossover. A random point is chosen on the individual chromosomes (strings) and the genetic material is exchanged from beginning to this point.



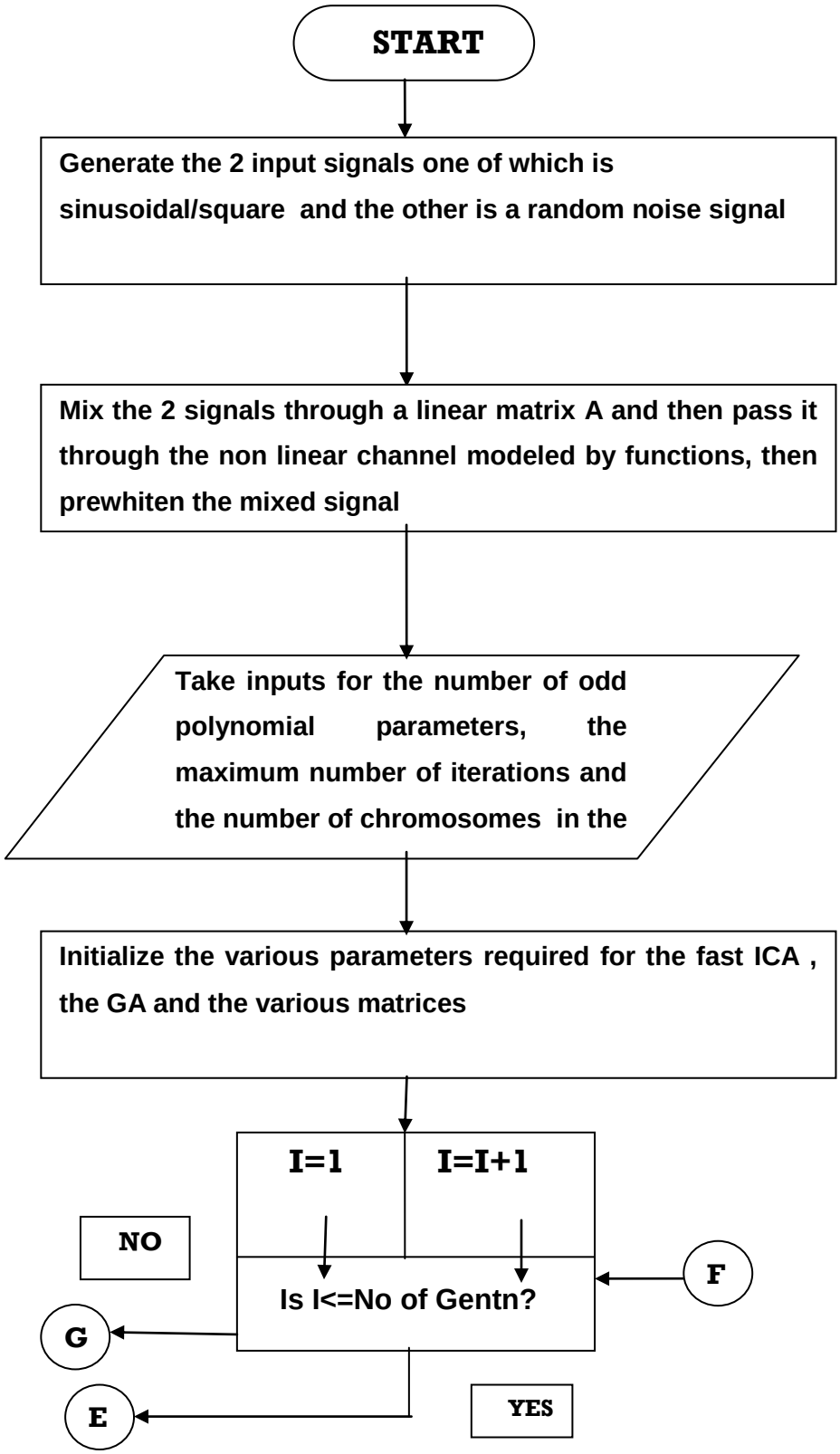
One-Point Crossover

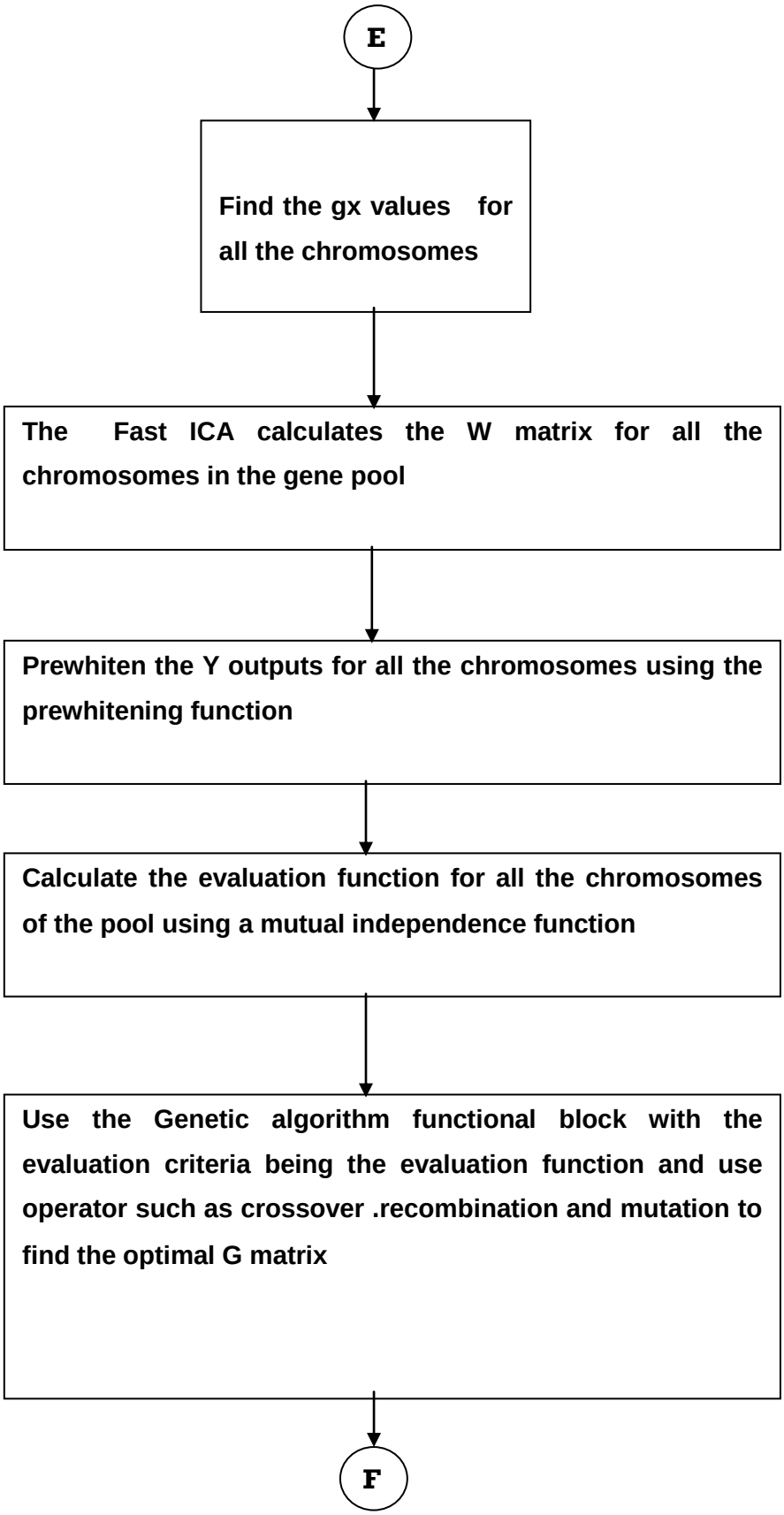
Termination

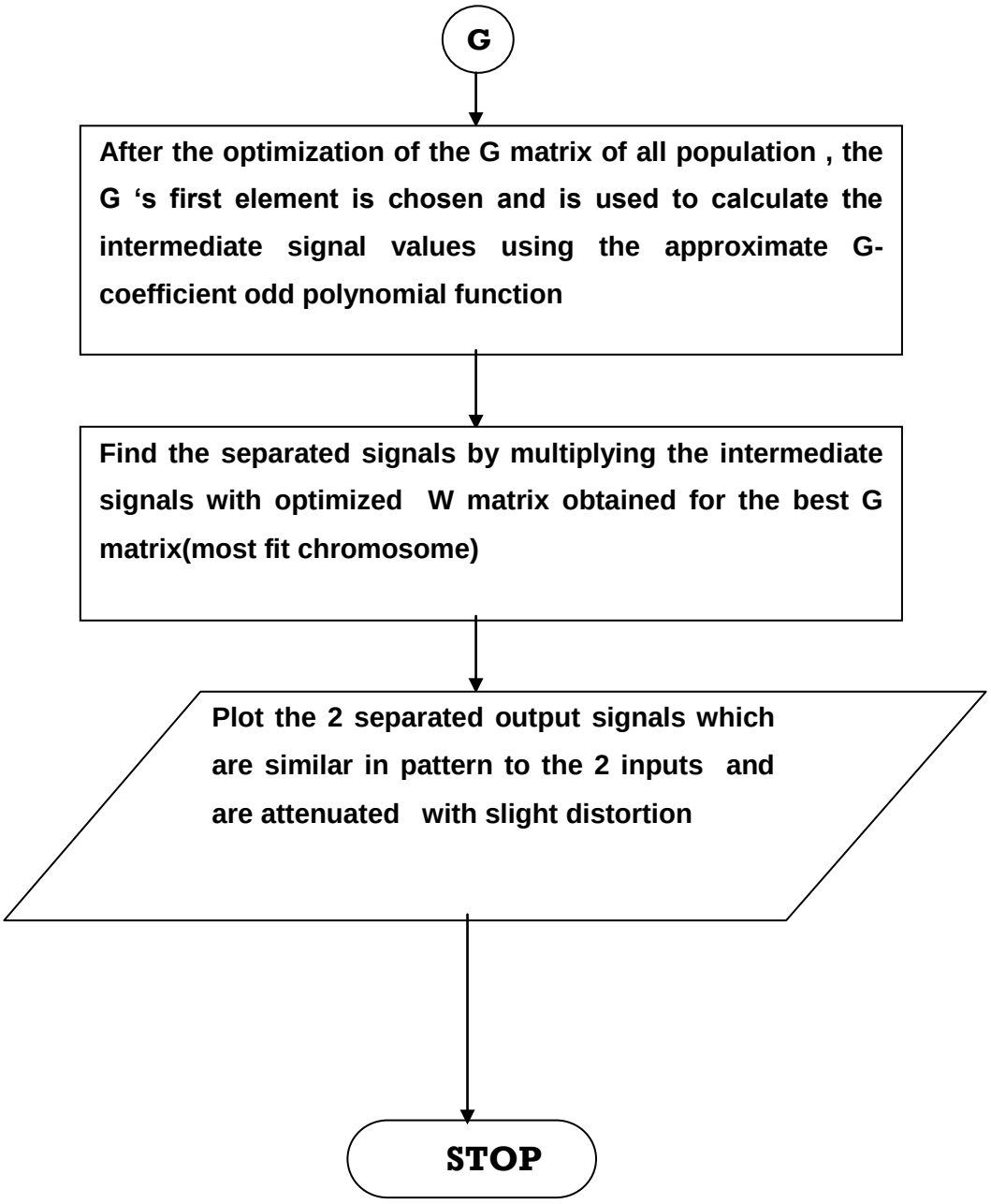
A solution is found that satisfies minimum criteria

- Fixed number of generations found.
- Allocated budget (computation, time/money) reached
- The highest ranking solution's fitness is reaching or has reached.

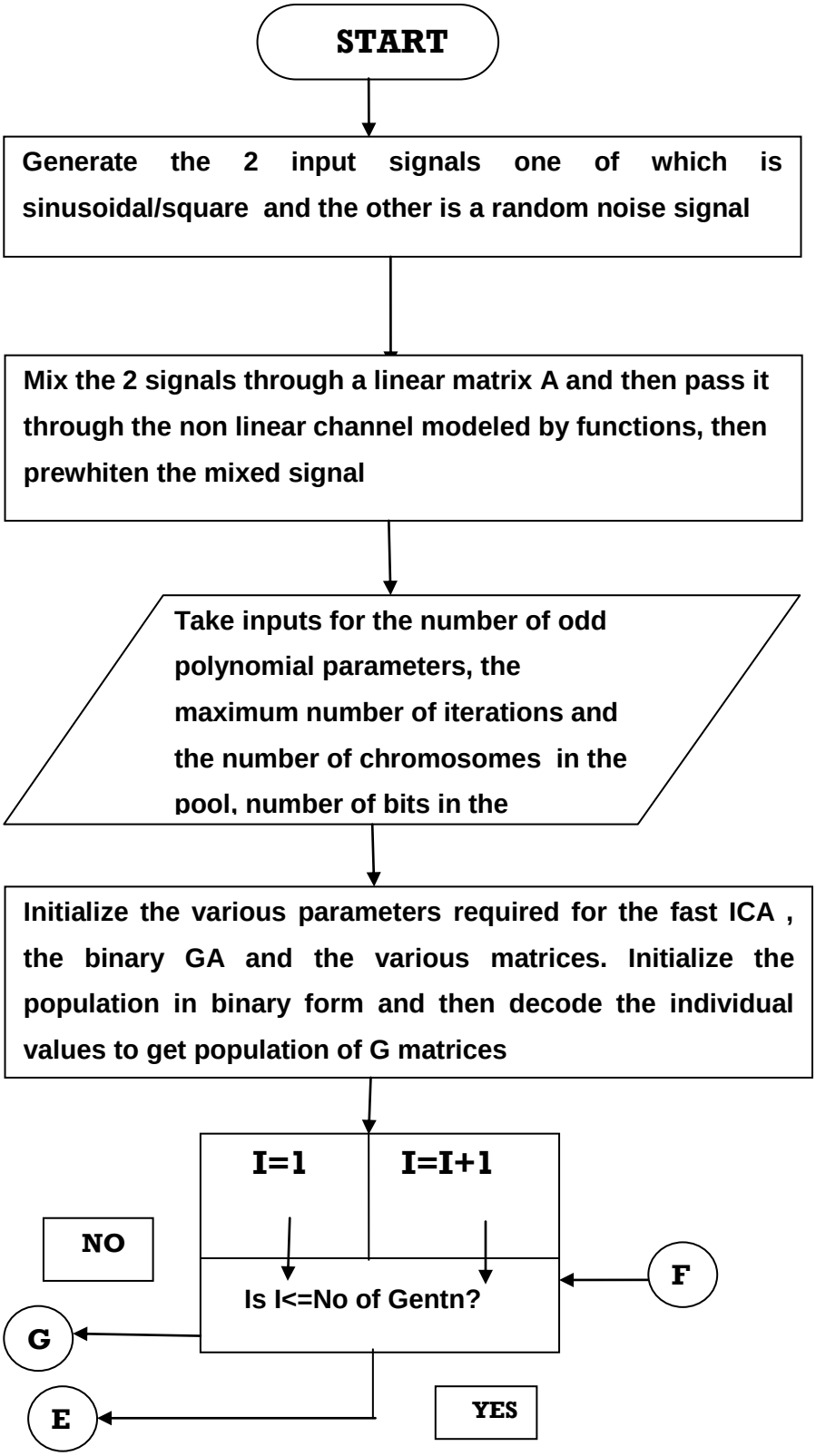
3.6 Block diagram of nonlinear ICA using real coded GA

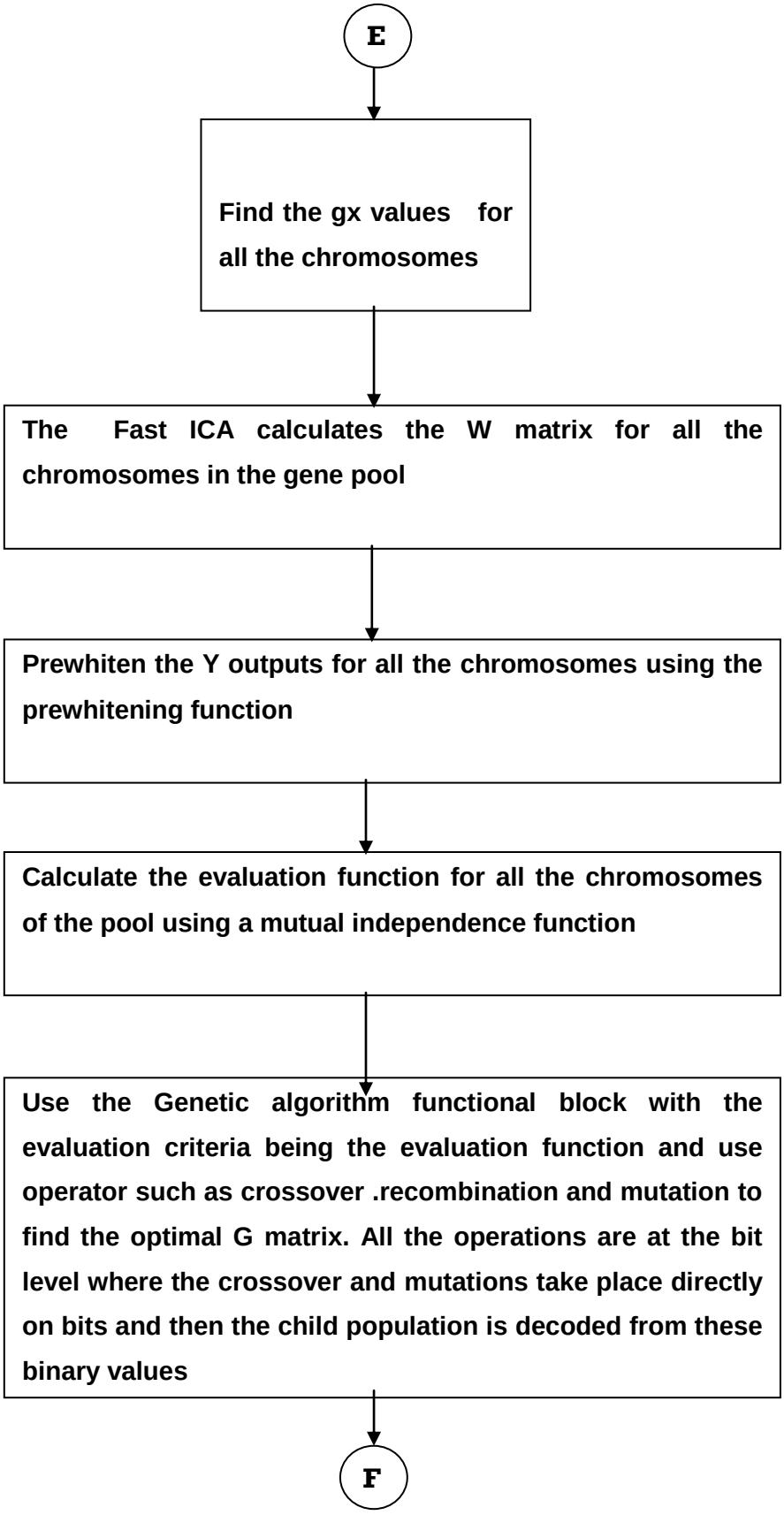


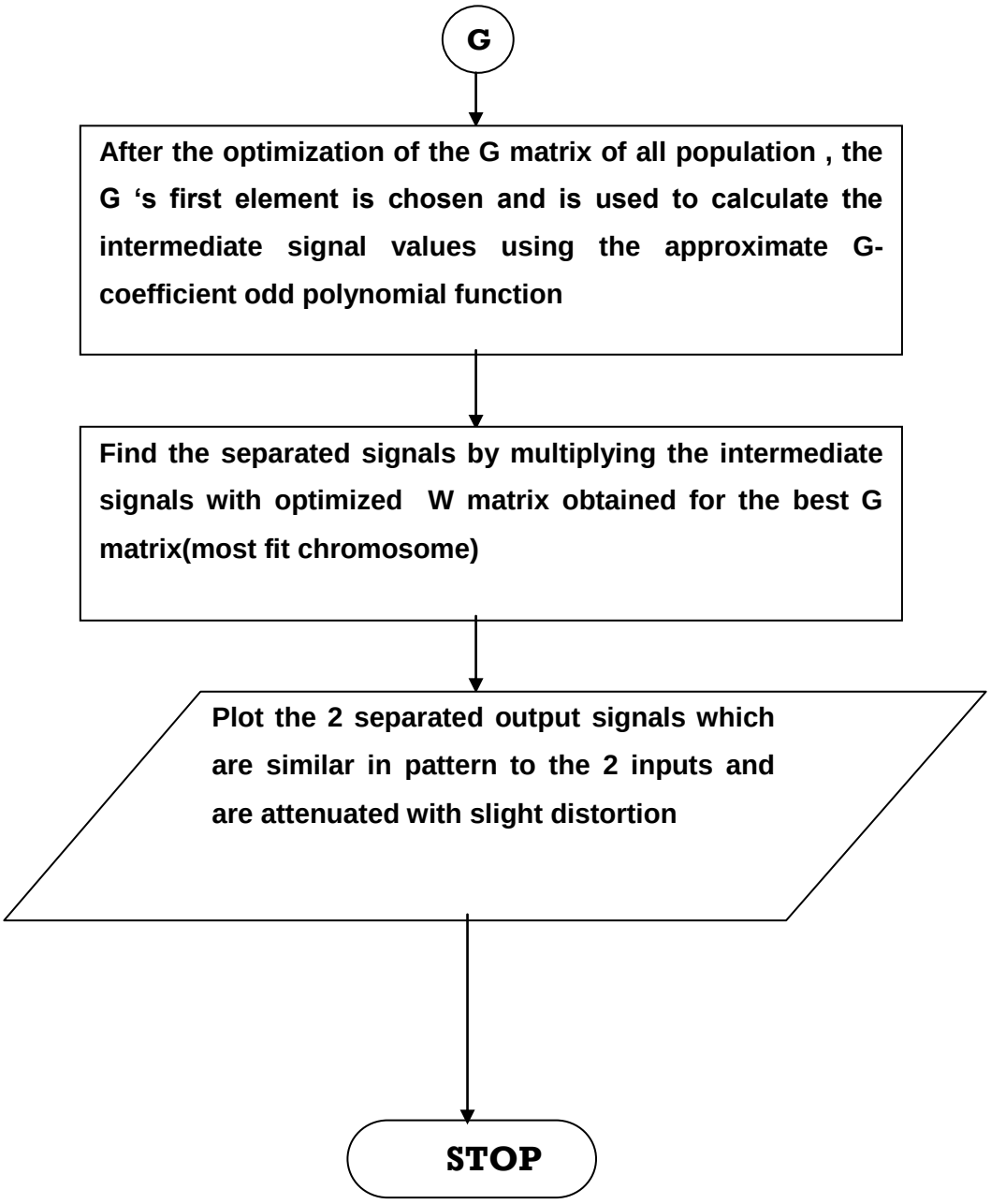

E
Find the gx values for all the chromosomes
The Fast ICA calculates the W matrix for all the chromosomes in the gene pool
Prewhiten the Y outputs for all the chromosomes using the prewhitening function
Calculate the evaluation function for all the chromosomes of the pool using a mutual independence function
Use the Genetic algorithm functional block with the evaluation criteria being the evaluation function and use operator such as crossover .recombination and mutation to find the optimal G matrix
F


G
After the optimization of the G matrix of all population , the G ‘s first element is chosen and is used to calculate the intermediate signal values using the approximate G-coefficient odd polynomial function
Find the separated signals by multiplying the intermediate signals with optimized W matrix obtained for the best G matrix(most fit chromosome)
Plot the 2 separated output signals which are similar in pattern to the 2 inputs and are attenuated with slight distortion
STOP

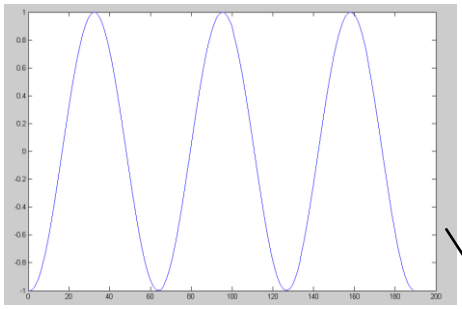
3.7 Block diagram of nonlinear ICA using binary GA



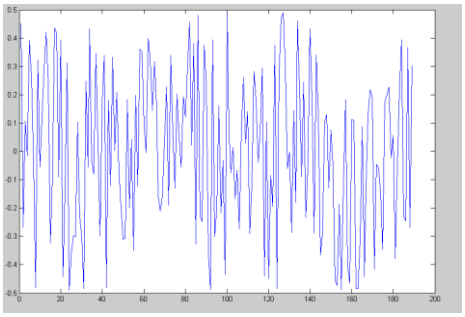

E
Find the gx values for all the chromosomes
The Fast ICA calculates the W matrix for all the chromosomes in the gene pool
Prewhiten the Y outputs for all the chromosomes using the prewhitening function
Calculate the evaluation function for all the chromosomes of the pool using a mutual independence function
Use the Genetic algorithm functional block with the evaluation criteria being the evaluation function and use operator such as crossover .recombination and mutation to find the optimal G matrix. All the operations are at the bit level where the crossover and mutations take place directly on bits and then the child population is decoded from these binary values
F


G
After the optimization of the G matrix of all population , the G ‘s first element is chosen and is used to calculate the intermediate signal values using the approximate G-coefficient odd polynomial function
Find the separated signals by multiplying the intermediate signals with optimized W matrix obtained for the best G matrix(most fit chromosome)
Plot the 2 separated output signals which are similar in pattern to the 2 inputs and are attenuated with slight distortion
STOP

3.8 RESULT OF BSS USING ICA AND REAL CODED GA(SINE WAVE & RANDOM NOISE)

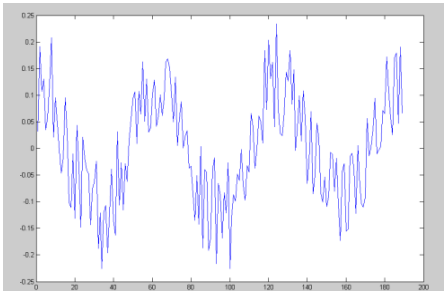


INPUT 1

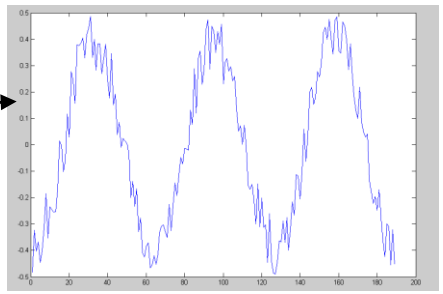


INPUT 2

Linear and Non Linear Mixer

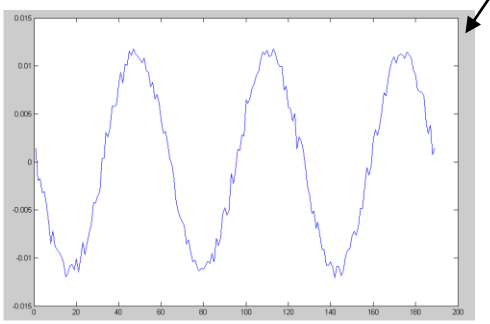


MIXED SIGNAL1

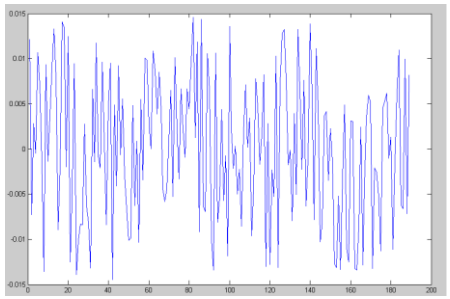


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Real Coded GA for approximation of the non linear functions

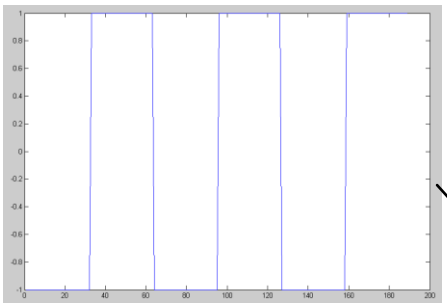


OUTPUT SIGNAL 1

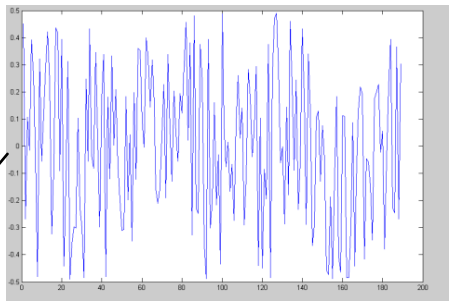


OUTPUT SIGNAL 2

3.9 RESULT OF BSS USING ICA AND REAL CODED GA(SQUARE WAVE & RANDOM NOISE)

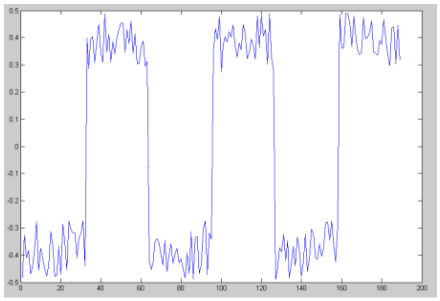


INPUT 1

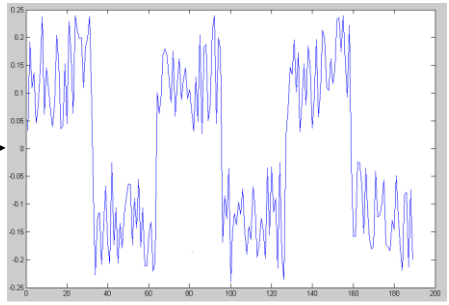


INPUT 2

Linear Mixer and Non Linear Channel

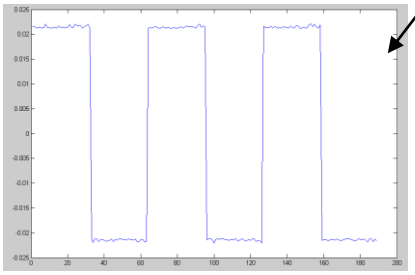


MIXED SIGNAL1

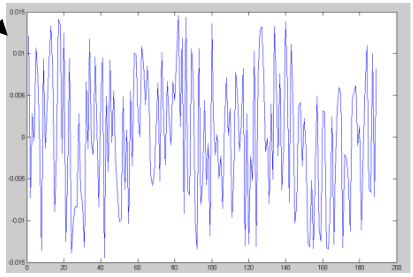


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Real Coded GA for approximation of the non linear functions

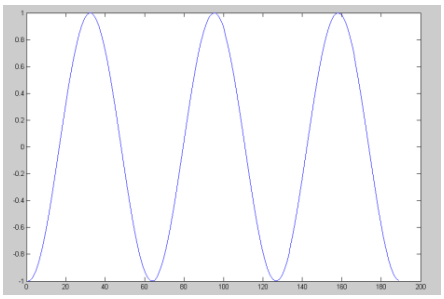


OUTPUT SIGNAL 1

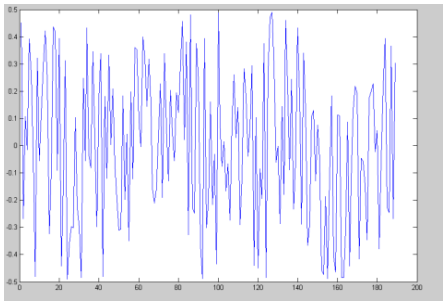


OUTPUT SIGNAL 2

3.10 RESULT OF BSS USING ICA AND BINARY GA(SINE WAVE & RANDOM NOISE)

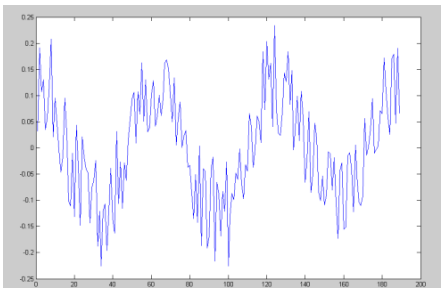


INPUT 1

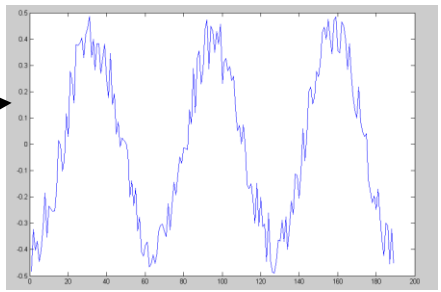


INPUT 2

Linear Mixer and Non Linear Channel

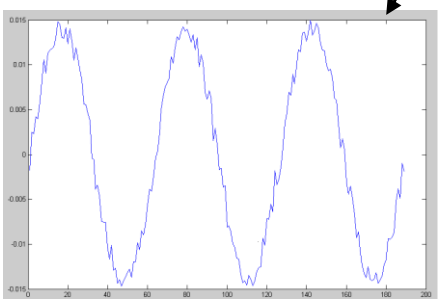


MIXED SIGNAL1

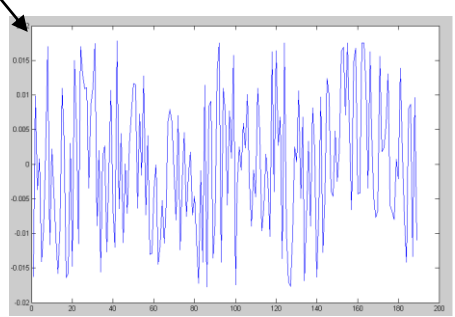


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Real Coded GA for approximation of the non linear functions

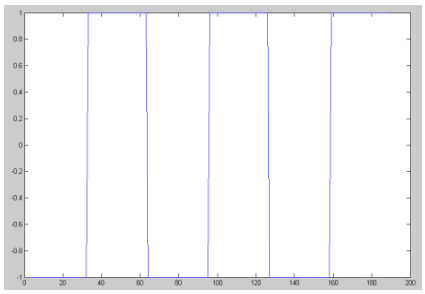


OUTPUT SIGNAL 1

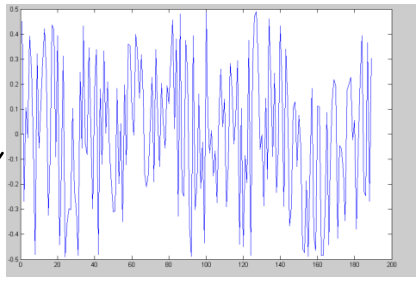


OUTPUT SIGNAL 2

3.11 RESULT OF BSS USING ICA AND BINARY GA(SQUARE WAVE & RANDOM NOISE)

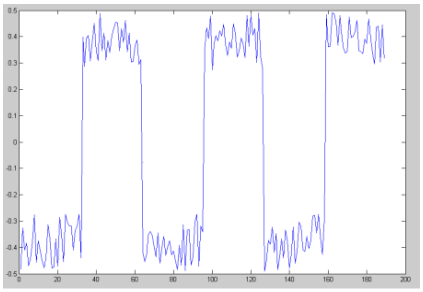


INPUT 1

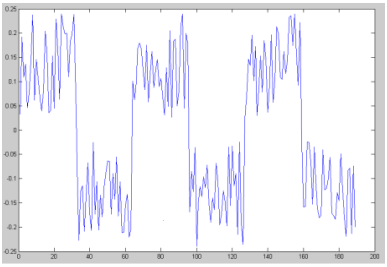


INPUT 2

Linear Mixer and Non Linear Channel

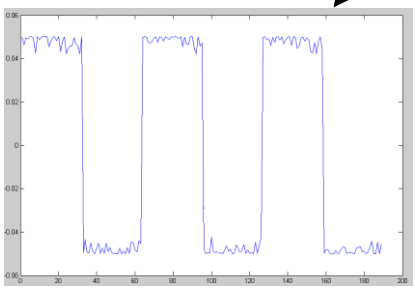


MIXED SIGNAL1

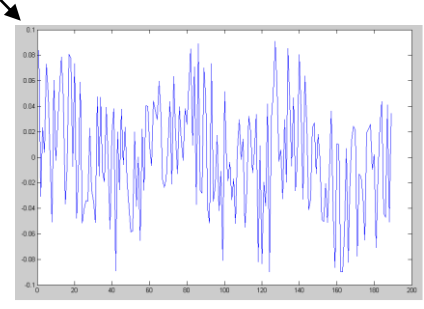


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Binary GA for approximation of the non linear functions



OUTPUT SIGNAL 1



OUTPUT SIGNAL 2

Chapter 4

BACTERIA FORAGING OPTIMIZATION BASED LEARNING

4.1 Introduction to bacterial foraging algorithm
4.2 The optimization function for the bacterial foraging (BF) algorithm
4.3 A simple Bacterial Foraging Algorithm
4.4 Block diagram of nonlinear ICA using BFO
4.5 result of BSS using ICA and BFO(sin wave & random noise)
4.6 result of BSS using ICA and BFO(square wave & random noise)
4.7 Comparison between various optimization techniques
4.8 Discussion

4.1 INTRODUCTION TO BACTERIAL FORAGING ALGORITHM

Natural selection tends to eliminate animals with poor foraging strategies and favor the propagation of genes of those animals that have successful foraging strategies. After many generations poor foraging strategies are either eliminated or shaped into good ones. Animals search for and obtain nutrients in a way that maximizes *E/T* where *E* is energy obtained per time *T.* Recently, search and optimal foraging of bacteria have been used for solving optimization problems .To perform social foraging, an animal needs communication capabilities and over a period of time it gains advantages that can exploit the sensing capabilities of the group. This helps the group to predate on a larger prey, or alternatively, individuals could obtain better protection from predators while in a group.

4.1.1 Overview of chemotactic behavior of Escherichia coli

In our research, we considered the foraging behavior of E. coli, which is a common type of bacteria . Its behavior and movement comes from a set of six rigid spinning (100–200 r.p.s) flagella, each driven as a biological motor. An E. coli bacterium alternates through running and tumbling. Running speed is 10–20 lm/s, but they cannot swim straight.

4.1.2 Decision making in foraging

The chemotactic actions of the bacteria are modeled as follows:

• In a neutral medium, if the bacterium alternatively tumbles and runs, its action could be similar to search.

• If swimming up a nutrient gradient (or out of noxious substances) or if the bacterium swims longer (climb

up nutrient gradient or down noxious gradient), its behavior seeks increasingly favorable environments.

• If swimming down a nutrient gradient (or up noxious substance gradient), then search action is like avoiding unfavorable environments.

Therefore, it follows that the bacterium can climb up nutrient hills and at the same time avoids noxious substances. The sensors it needs for optimal resolution are receptor proteins which are very sensitive and possess high gain. That is, a small change in the concentration of nutrients can cause a significant change in behavior. This is probably the best-understood sensory and decision-making system in biology

4.1.3 Reproduction

Mutations in E. coli affect the reproductive efficiency at different temperatures, and occur at a rate of about 10 per gene per generation. E. coli occasionally engages in a conjugation that affects the characteristics of the population.

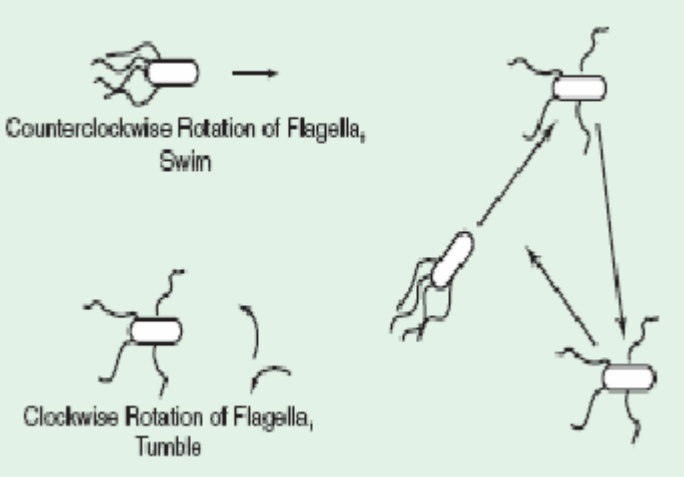
4.1.4 Types of taxis

There are many types of taxis that are used in bacteria such as, aero taxis (attracted to oxygen), Photo taxis (light), thermo taxis (temperature), magneto taxis (magnetic lines of flux) and some bacteria can change their shape and number of flagella (based on the medium) to reconfigure in order to ensure efficient foraging in a variety of media.

4.1.5 Swarming

Bacteria could form intricate stable spatio-temporal patterns in certain semisolid nutrient substances and they can survive through a medium if placed together initially at its center. Moreover, under certainconditions, they will secrete cell-to-cell attractant signals so that they will group and protect each other.

4.1.6 Representation of swim/run and tumble



4.2 The optimization function for the bacterial foraging (BF) algorithm

Optimization consists of four different stages-

- Chemotaxis
- Swarming
- Reproduction
- Elimination & Dispersion

4.2.1 Chemo taxis

It is defined to be a tumble followed by a tumble or tumble followed by a run. To represent a tumble, a unit length random direction Ø(j) is generated, this will be used to define the direction of movement after a tumble.

Mathematically,

$$\theta^{l}(j+1,k,l)= \theta^{l}(j,k,l)+C(i)* Ø(j)$$

Where C(i) =size of the step taken in the random direction. = position of l^{th} bacterium at j^{th} chemo tactic loop, k^{th} reproduction loop and l^{th} elimination-dispersal loop. If at $\theta^{l}(j+1,k,l)$, the cost function is better than at $\theta^{l}(j,k,l)$, another step of size C(i) is taken in the same direction. This swim is continued as long as it continues to better the cost function, but only upto a max. number of steps N_s.

4.2.2 Reproduction

After N_c chemo tactic steps, a reproduction step is taken. The population is sorted in order of ascending order of cost function .Then, S_r = S/2 least healthy bacteria die and the other healthiest bacteria each split into two bacteria, which are placed at the same location.

4.2.3 Elimination and Dispersal

This is done to prevent the bacteria from being trapped in local minima. A bacterium is chosen according to a preset probability P_{ed}, to be dispersed and moved to another position within the environment. They have the effect of possibly destroying the chemo tactic progress. But, they also have the effect of assisting in chemo taxis, since dispersal may place the bacteria near good food sources.

4.3 A simple Bacterial Foraging Algorithm

The algorithm to search optimal values of parameters is described as follows:

[Step 1] Initialize parameters n; N; Nc; Ns; Nre; Ned; Ped, C(i)(i=1,.. N); Ø(j)

where,

n: Dimension of the search space,

N: The number of bacteria in the population,

NC: Chemotactic steps,

Nre: The number of reproduction steps,

Ned: The number of elimination–dispersal events,

Ped: Elimination–dispersal with probability,

C(i): The size of the step taken in the random direction specified by the tumble.

[Step 2] Elimination–dispersal loop: l = l+1.

[Step 3] Reproduction loop: k = k+1.

[Step 4] Chemo taxis loop: j = j+1.

[sub step a] For i = 1,2,. . . ,N, take a chemo tactic step for bacterium i as follows.

[sub step b] Compute fitness function, ITSE (I,j,k,l)

[sub step c] Let ITSElast = ITSE(i, j,k, l) to save this value since we may find a better cost via a

run.

[sub step d] Tumble: generate a random vector Λ(i) , with each element Λ m(i),

m = 1,2,. . . ,p, a Random number on [-1, 1].

[sub step e] Move: Let

$\theta^{l}(i+1,j,k)= \theta^{l}(i,j,k)+C(i)* \Lambda(i)/\sqrt{\Lambda^{T}(i)\,\Lambda(i)}$

This results in a step of size C(i) in the direction of the tumble for bacterium i.

[sub step f] Compute ITSE(i, j + 1,k, l).

[sub step g] Swim.

(i) Let m = 0 (counter for swim length).

(ii) While m < Ns (if have not climbed down too long).

• Let m = m + 1.

• If ITSE(i, j , k,l) >ITSElast(if doing better), let ITSElast = ITSE (i,j+1,k,l)and let

$\theta^{l}(i+1,j,k)= \theta^{l}(i,j,k)+C(i)* \Lambda(i)/\sqrt{\Lambda^{T}(i)\,\Lambda(i)}$

and use this $\theta^{l}(i+1,j,k)$ to compute the new ITSE(I,j+1,k,l) as we did in [sub step f].

• Else, let m = Ns. This is the end of the while statement.

[sub step h] Go to next bacterium (i,1) if i ≠N (i.e., go to [sub step b] to process the next bacterium).

[Step 5] If j < NC, go to step 3. In this case, continue chemo taxis,

[Step 6] Reproduction:

[sub step a] For the given k and l, and for each i = 1,2,. . . ,N, let

$ITSE^{l}_{health} = \sum ITSE(i,j,k,l)$

be the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances).

Sort bacteria and chemo tactic parameters C(i) in order of ascending cost ITSEhealth (higher cost means lower health).

[sub step b] The Sr bacteria with the highest ITSEhealth values die and the remaining Sr bacteria with the best values split (this process is performed by the copies that are made are placed at

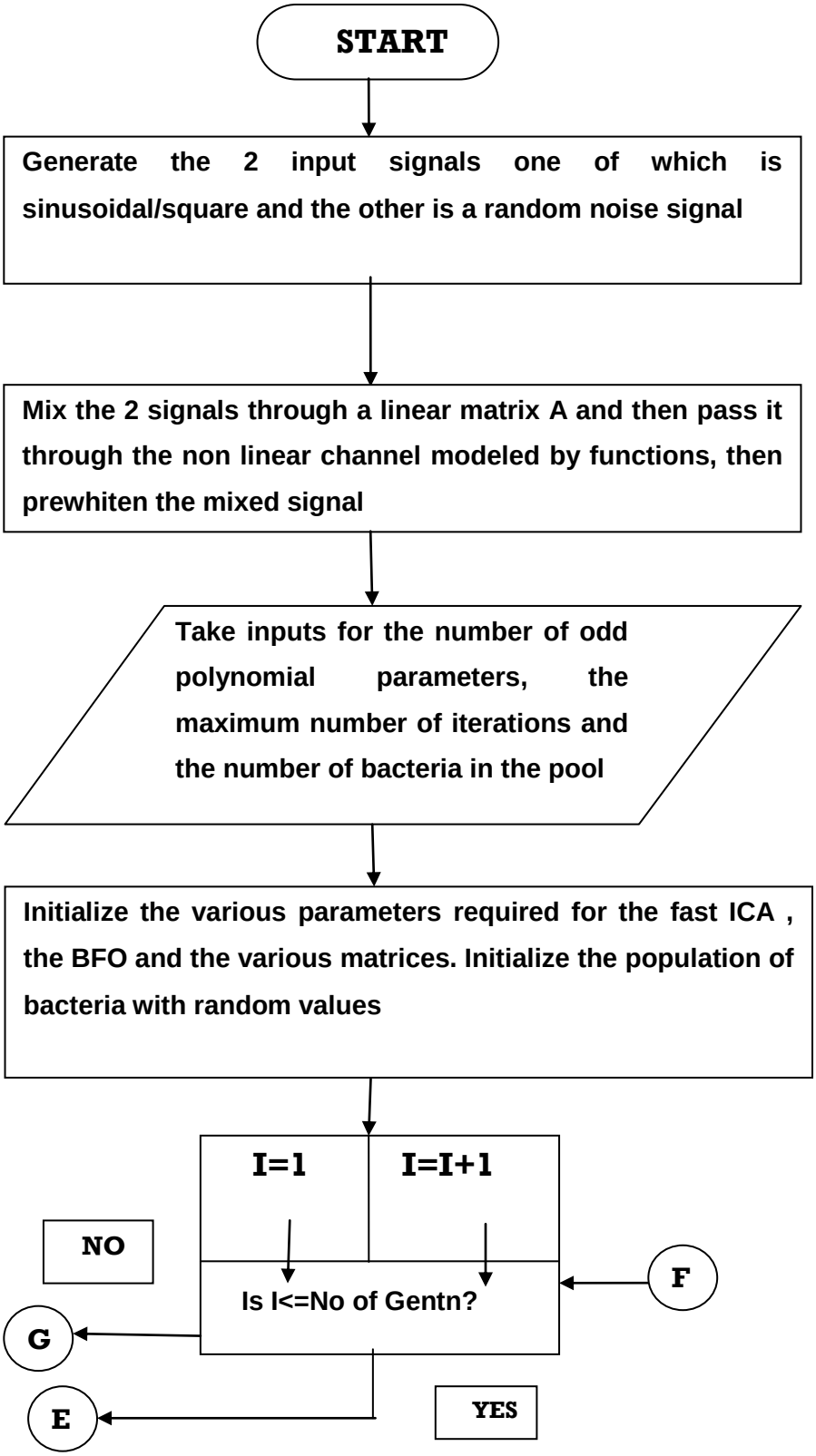
the same location as their parent).

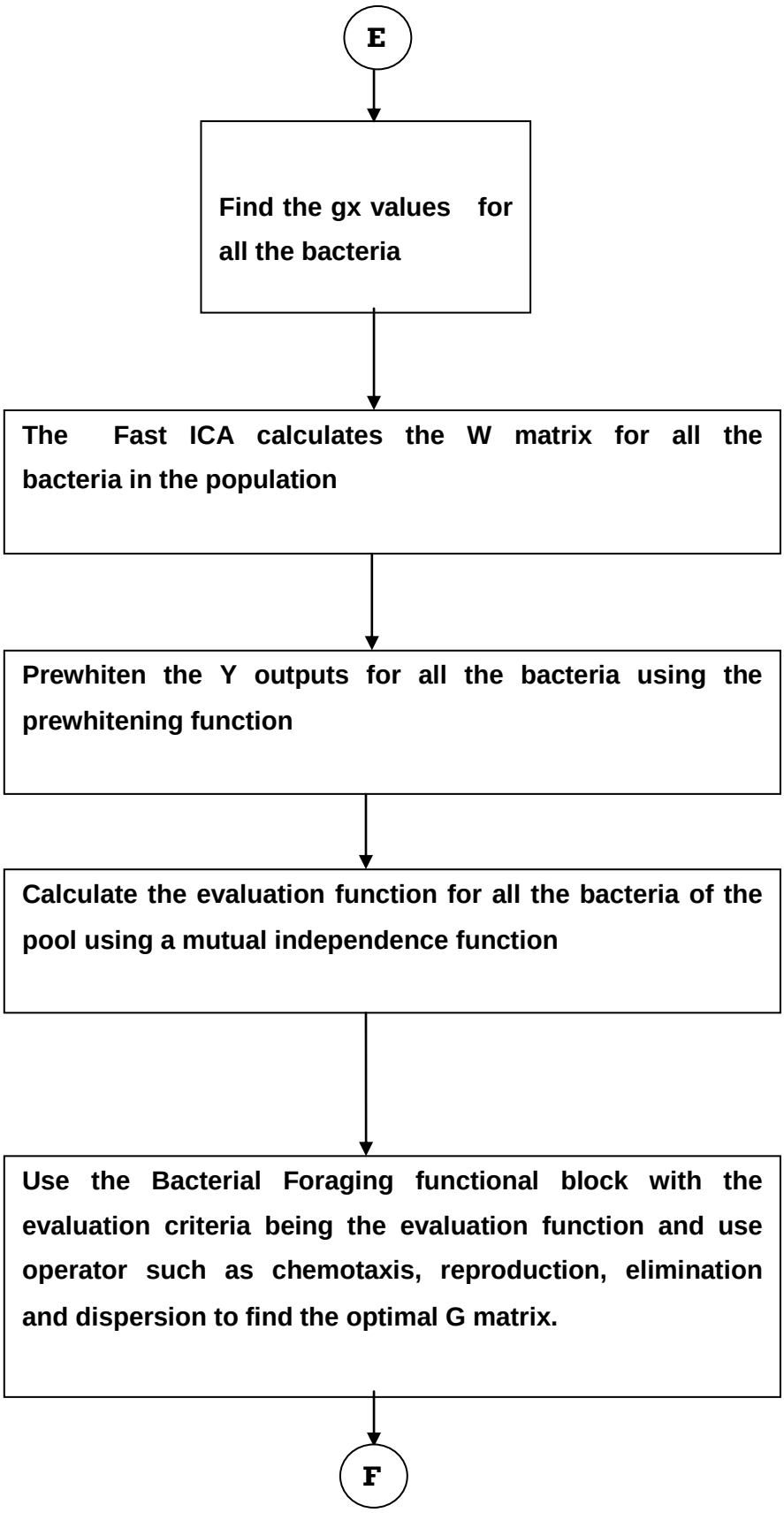
[Step 7] If k < Nre, go to [step 3]. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemo tactic loop.

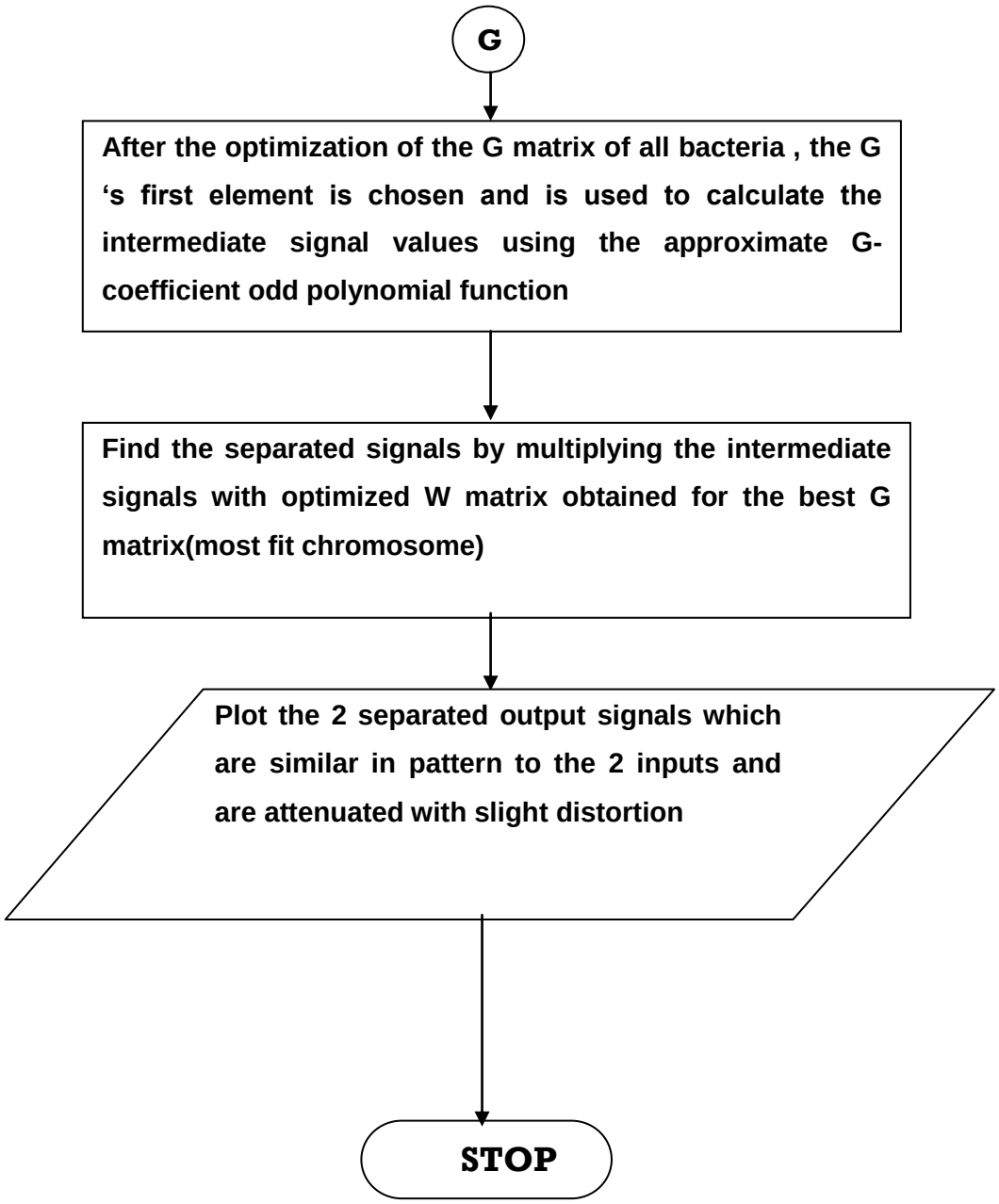
[Step 8] Elimination–dispersal: For i = 1,2,. . . ,N, with probability Ped, eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population constant

. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If l < Ned, then go to [step 2]; otherwise end.

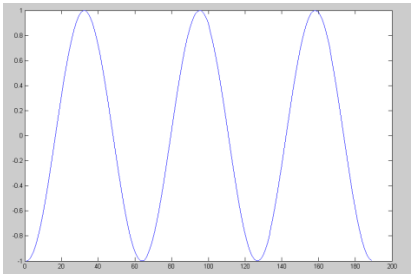
4.4 Block diagram of nonlinear ICA using BFO ::



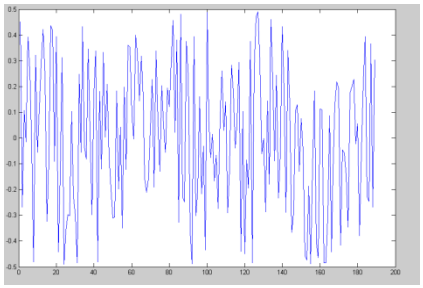

E
Find the gx values for all the bacteria
The Fast ICA calculates the W matrix for all the bacteria in the population
Prewhiten the Y outputs for all the bacteria using the prewhitening function
Calculate the evaluation function for all the bacteria of the pool using a mutual independence function
Use the Bacterial Foraging functional block with the evaluation criteria being the evaluation function and use operator such as chemotaxis, reproduction, elimination and dispersion to find the optimal G matrix.
F


G
After the optimization of the G matrix of all bacteria , the G ‘s first element is chosen and is used to calculate the intermediate signal values using the approximate G-coefficient odd polynomial function
Find the separated signals by multiplying the intermediate signals with optimized W matrix obtained for the best G matrix(most fit chromosome)
Plot the 2 separated output signals which are similar in pattern to the 2 inputs and are attenuated with slight distortion
STOP

4.5 RESULT OF BSS USING ICA AND BFO(SIN WAVE & RANDOM NOISE)

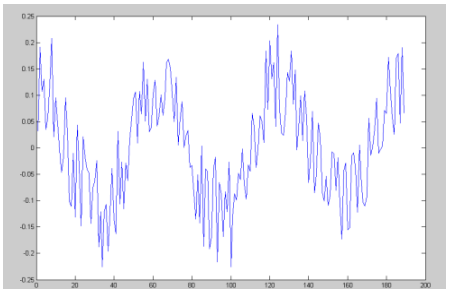


INPUT 1

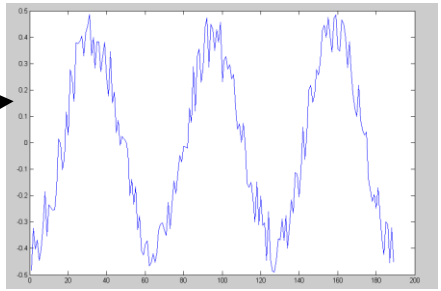


INPUT 2

Linear Mixer and Non Linear Channel

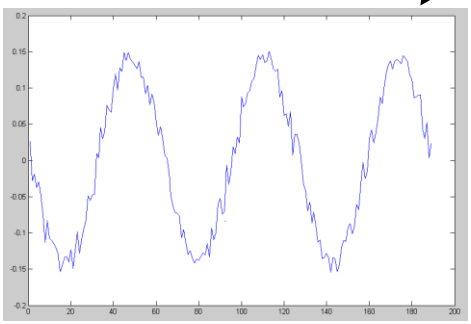


MIXED SIGNAL1

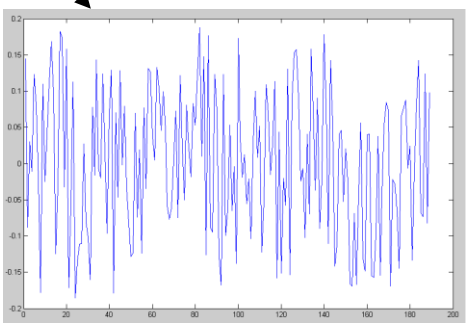


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Bacterial Foraging for approximation of the non

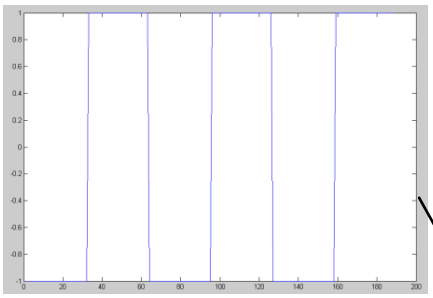


OUTPUT SIGNAL 1

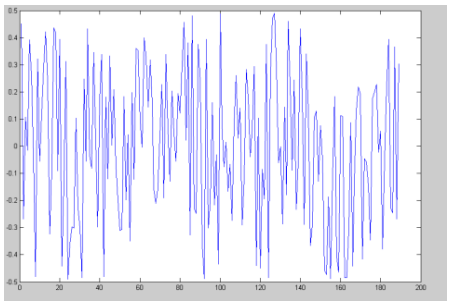


OUTPUT SIGNAL 2

4.6 RESULT OF BSS USING ICA AND BFO(SQUARE WAVE & RANDOM NOISE)

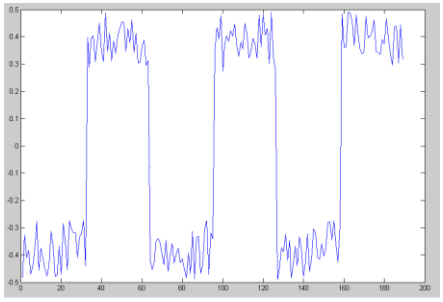


INPUT 1

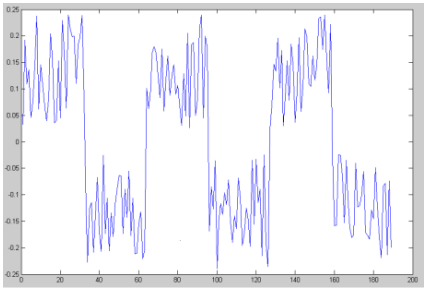


INPUT 2

Linear Mixer and Non Linear Channel

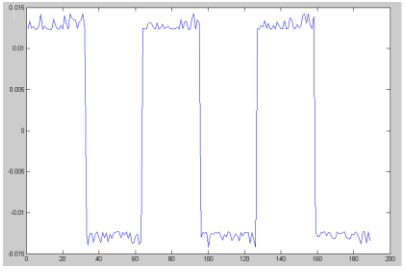


MIXED SIGNAL1

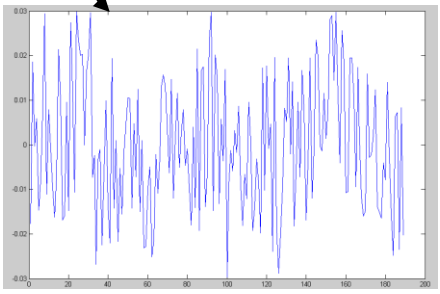


MIXED SIGNAL 2

Non Linear ICA using Fast ICA for optimization of mixer and Real Coded GA for approximation of the non linear functions



OUTPUT SIGNAL 1



OUTPUT SIGNAL

4.7 Comparison between various optimization techniques

Sl. No	Type of input along with the random noise input	Particle Swarm Optimization	Real coded Genetic Algorithm	Binary Genetic Algorithm	Bacterial Foraging
1.	Sinusoidal Input	No. of iterations= 50 No. of particles= 40 No. of parameters=6 MSE for signal= 1.5769e-004 MSE for random Noise= 0.0514	No. of iterations= 200 No. of chromosomes= 20 No. of parameters=5 MSE for signal=.0050 MSE for random Noise=.0719	No. of iterations=100 No. of chromosomes= 10 No. of parameters=3 MSE for signal= 0.0054 MSE for random Noise= 0.0815	No. of iterations=25 No. of bacteria= 12 No. of parameters=5 MSE for signal=.0111 MSE for random Noise=.0655
2.	Square Input	No. of iterations= 200 No. of particles= 40 No. of parameters=6 MSE for signal= .0041 MSE for random Noise=.0807	No. of iterations=50 No. of chromosomes= 20 No. of parameters=6 MSE for signal=9.6164e-4 MSE for random Noise=.0756	No. of iterations=45 No. of chromosomes =15 No. of parameters=4 MSE for signal=.0034 MSE for random Noise=.0555	No. of iterations=25 No. of bacteria =12 No. of parameters=5 MSE for signal=3.6190 MSE for random Noise=0.5730

4.8 DISCUSSION

We observe that the Particle Swarm Optimization technique is easier to implement as it follows a very simple update methodology .The Genetic Algorithms (both real coded and binary) ensure a convergence but may take higher number of iterations. The Bacterial Foraging technique is complex in nature but ensures a rapid convergence. All the above 3 mentioned techniques have their own merits and demerits but are competent enough to perform non linear blind source separation.

Chapter 5

Conclusion

CONCLUSION

In this project, we have used PSO ,GA ,BFO and ICA to carry out non linear mixed signal blind source separation. High order odd polynomial is applied to fit non linear mixed function and establish non linear signal blind separation model. Estimating the parameter of polynomial by PSO algorithm and then iterating the linear non mixed matrix by ICA has obtained good effect. Similarly we have applied genetic algorithm(both binary and real coded) and bacteria foraging technique to estimate the parameter of the odd polynomial and thereafter we have compare the result of the all three algorithms.

REFERENCE

1) WEI YU, LIU ZHENXING AND LI CHANGHAI “IMPROVED Particle Swarm To Non Linear Blind Source Separation”

2) A TALEB C. JUTTEN “Source Separation In Post Non Linear Mixtures : An Entropy Based Algorithm “

3)T.-W. LEE, M. GIROLAMI, A. BELL, AND T. SEJNOWSKI, “An unifying information-theoretic framework for independent component analysis”, International Journal on Mathematical and Computer Modeling, 1998.

3) AAPO HYVARINEN, JUHA KARHUNEN, ERKKI OJA”A Comprehensive Introuction To Independent Component Analysis For Students And Practitioners”.

4) L. B. ALMEIDA, “Linear and nonlinear ICA based on mutual information”, in Proc. Symp. 2000 on Adapt. Sys. for Sig. Proc., Commun. and Control, Lake Louise, Alberta, Canada, 2000.

5) DONG HWA KIM, AJITH ABRAHAM, JAE HOON CHO “A Bacterial Foraging Approach For Global Optimization”

6) ACHARYA D.P, PANDA G., LAKSHMI Y.V.S “A Constrained Genetic Algorithm Based Independent Component Analysis “

7) WWW.WIKIPEDIA.ORG.