Abstract: In this paper, it is presented how nonlinearly mixed signals can be retrieved uniquely by using a novel approach based on signal restoration methodology rather than the conventional technique of mere signal separation. A new mathematical model of the nonlinear mixing system has been developed culminating in the formulation of a stable unique inverse solution, which has an identical structure to the multilayer neural network. In addition, we show how the optimum framework for the nonlinear demixing system can be obtained directly from the derived mixing model. It is further shown how the proposed schemes using the multilayer Polynomial Neural Network (PNN) can be utilised to acquire the desired solution. Moreover, the corresponding learning algorithm based on the generalised stochastic gradient descent method combined with a modified genetic algorithm (GA) has been developed to yield a novel and more effective approach in updating the parameters of the PNN. Both synthetic and real-time simulations have been conducted to verify the efficacy of each proposed scheme.

Keywords: Nonlinear Independent Component Analysis, Nonlinear Signal Restoration, Nonlinear System Identification, Adaptive Polynomial Neural Network.
1. INTRODUCTION

Signal restoration is primarily concerned about two things: (i) retrieving information about a signal that has been corrupted with unwanted interference and (ii) reconstructing the signal of interest based on the extracted information. To date, a plethora of methodologies can be found in the open literature from various perspectives for solving signal restoration problems. Most recently, a new technique known as the Independent Component Analysis (ICA) has surged to the frontier of many signal restoration research avenues and has become a key ingredient in diverse applications spanning from multi-user cellular radio networks to speech recognition and astronomy to financial forecasting ([1-4] and references therein). Most of the existing ICA algorithms are based on the model that the independent source signals are linearly distorted i.e. they are both linearly and spatially mixed by an invertible matrix. For some applications, linear mixing models can provide sufficient approximations but frequently, channel dynamics are more complex and require nonlinear models such as in underwater acoustics, magnetic recording channels, microwave and satellite communications [5]. In biomedical research, identification of nonlinear dynamics is a subject of interest since many physiological signals undergo nonlinear transformations; for example, the auditory nervous system is modelled as memoryless nonlinearity [6]. Hence, the most appropriate representation of the mixing system in all of these examples would be nonlinear. For nonlinear mixing model [8-14], the linear ICA models are not strictly applicable and the existing schemes fail to extract the sources in non-linear mixtures. Commonly used schemes in linear ICA are variants based on the Kullback-Leibler Divergence (KLD) criteria [15]. However, in nonlinear mixtures there is no guarantee that the solution achieved at the outputs of any nonlinear demixing system will correspond to the desired source signals even when the KLD is minimal. This naturally arises since the KLD functional is invariant under any nonlinear invertible mappings and hence, the restored signals based on any linear model can be related to the original source signals via any unknown nonlinear map. We may therefore state that the solution obtained during the process of signal reconstruction is not unique in the case of nonlinear mixture regardless of the fact that the information of independent signals has been successfully retrieved. In this paper, we show how the Polynomial Neural Network (PNN) combined with a modified genetic algorithm (GA) can be used as the nonlinear demixing system to restore nonlinearly mixed signals uniquely in a blind mode.
2. DEVELOPMENTS OF NONLINEAR MIXING AND DEMIXING MODELS

Linear models play a very crucial role in the development of various signal processing techniques. The obvious advantage of linear models is their inherent simplicity. However, as mentioned in Section 1, several practical situations often arise in which the performances of linear models are unacceptable. Hence, the need to develop nonlinear model and nonlinear signal processing techniques becomes vital in these cases. In particular, the developed model should not only able to demonstrate its flexibility in modelling nonlinear systems but also capable of handling the linear model as a special case. In order to derive the desirable nonlinear model endowed with these characteristics, the following is considered:

Lemma 1

If an equation can be expressed in the following form:

\[ f \left[ G(x, y) \right] = F \left[ f(x), f(y) \right] \]  

where one of the functions \( F \) (or \( G \)) is a continuous group operation for the \( x, \ y \) of an interval, then equation (1) has a strictly monotonic continuous function \( f \) if and only if the other function \( G \) (or \( F \)) is also a continuous group operation. Now, by letting \( G(x, y) = x + y \) and \( F[f(x), f(y)] = f(x) \oplus f(y) \) with application of lemma 1 there exists strictly monotonic continuous function such that

\[ u \oplus v = f(f^{-1}(u) + f^{-1}(v)) \]  

Moreover, we can define from (2) the operator ‘\( \oplus \)’ such that

\[ \alpha \oplus u = u \oplus u \oplus \cdots \oplus u = f \left( f^{-1}(u) + f^{-1}(u) + \cdots + f^{-1}(u) \right) = f \left( \alpha f^{-1}(u) \right) \]  

where \( \alpha \in \mathbb{R} \) i.e. the field of real number.
Theorem 1

If the nonlinear mixing system with \( q \) inputs and \( p \) outputs is defined as:

\[
x = \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix}^T = \begin{bmatrix} g_1(s^T) & g_2(s^T) & \cdots & g_p(s^T) \end{bmatrix}^T
\]

(4)

with

\[
x_i = g_i(s) = m_{i1} \otimes s_1 \oplus m_{i2} \otimes s_2 \oplus \cdots \oplus m_{iq} \otimes s_q
\]

(5)

where \( m_{ij} \in \mathbb{R} \) and \( s_i \) is the \( i^{th} \) source signal. Then, by applying lemma 1 we have

\[
x_i = f(f^{-1}(m_{i1} \otimes s_1) + f^{-1}(m_{i2} \otimes s_2) + \cdots + f^{-1}(m_{iq} \otimes s_q))
\]

\[
= f(m_{i1} f^{-1}(s_1) + m_{i2} f^{-1}(s_2) + \cdots + m_{iq} f^{-1}(s_q))
\]

\[
= f(m_i^T f^{-1}(s))
\]

(6)

Therefore, by substituting (6) into (4), the nonlinear mixing system can be described by the following model:

\[
x = \begin{bmatrix} g_1(s) \\ g_2(s) \\ \vdots \\ g_p(s) \end{bmatrix} = \begin{bmatrix} m_{11} \otimes s_1 \oplus m_{12} \otimes s_2 \oplus \cdots \oplus m_{1q} \otimes s_q \\ m_{21} \otimes s_1 \oplus m_{22} \otimes s_2 \oplus \cdots \oplus m_{2q} \otimes s_q \\ \vdots \\ m_{p1} \otimes s_1 \oplus m_{p2} \otimes s_2 \oplus \cdots \oplus m_{pq} \otimes s_q \end{bmatrix} = \begin{bmatrix} f(m_1^T f^{-1}(s)) \\ f(m_2^T f^{-1}(s)) \\ \vdots \\ f(m_p^T f^{-1}(s)) \end{bmatrix}
\]

(7)

where \( M = \begin{bmatrix} m_1 & m_2 & \cdots & m_p \end{bmatrix}^T \) with dimension \( p \times q \) and \( m_i = [m_{i1} \ m_{i2} \ \cdots \ m_{iq}]^T \). From (7), we recognise that the nonlinear mixture is fundamentally a synthesis of two nonlinear functions, one of which is the inverse of the other and a matrix sandwiched between these two functions. Moreover, the optimal demixer system for the nonlinearly mixed signals in Eqn. (7) is given by
\[
\begin{bmatrix}
  f(\mathbf{m}_1^T f^{-1}(\mathbf{x})) \\
  f(\mathbf{m}_2^T f^{-1}(\mathbf{x})) \\
  \vdots \\
  f(\mathbf{m}_q^T f^{-1}(\mathbf{x}))
\end{bmatrix} = f(\mathbf{M}^+ f^{-1}(\mathbf{x}))
\]

where it is assumed that \( p \geq q \) and \( \mathbf{M}^+ = [\mathbf{m}_1 \mathbf{m}_2 \cdots \mathbf{m}_q]^T \) is the pseudoinverse of \( \mathbf{M} \). Specifically, if the map \( f \) defined in (2) and (3) is a linear function i.e. \( f(u) = u \), then the group operator ‘\( \oplus \)’ reduces to an addition operator while ‘\( \otimes \)’ a multiplication. In this case, the nonlinear mixture simplifies to an instantaneous linear mixture i.e. \( \mathbf{x} = f(\mathbf{M} f^{-1}(\mathbf{s})) = \mathbf{M}s \) [1-4] and the source signals can be recovered simply by applying \( \mathbf{M}^+ \) to the observed signals.

### 3. NONLINEAR DEMIXING SYSTEM

In this section, a general Polynomial Neural Network (PNN) demixer employing a set of polynomial nonlinearities in the hidden layers is firstly described. The structure of the proposed demixer is depicted in Fig. 1 where a 3-layer PNN is used for simplicity. The cost function is then formulated and the learning rules are derived for updating the parameters of the demixer. The obtained learning rules are then modified to incorporate the inverse polynomial nonlinearity at the last layer so as to mimic the desired structure set out in theorem 1 and finally, two new schemes for training the parameters using a modified Genetic Algorithm are developed.

Demixers such as the Self-Organising Map (SOM) [11], Radial Basis Function (RBF) [12] and FMLP with sigmoidal nonlinearity [14] are intrinsically nonlinear due to the user’s prior selection of its hidden neuron function. None of these parameters are explicitly designed to control the amount of nonlinearity of the neuron’s function. Thus, this leads to an oversized network which inevitably subjects to huge computational complexity. Moreover, the utilisation of fixed nonlinearity of the hidden neurons tends to cause the neural network to ‘overfit’ resulting in the performance to poorly generalise in the absence of a priori information. The invariance
of the KLD functional further aggravates the situation since non-unique independent outputs can be produced but subject to unknown nonlinear transformation. As a consequence, arbitrary nonlinear demixer trained using conventional KLD will result in an infinite number of non-unique solutions induced at the outputs of the demixer. Thus, the first step towards obviating generation of non-unique independent outputs is to regulate the inherent capability of the demixer from ‘overfitting’. This is followed by a suitable design of the cost function for training the demixer parameters. The use of polynomial function expansion method is exactly what is required for implementing the first step and this can be directly achieved by using the PNN. The hidden neurons in the PNN have a set of adjustable polynomials, which provides the required flexibility of the nonlinear demixer. They exists in the following form:

$$g(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \cdots + a_Nx^N = \sum_{n=0}^{N} a_n x^n$$  \hspace{1cm} (9)$$

where \(\{a_n\}_{n=1}^{N}\) are the set of coefficients of the polynomial and the integer \(N\) being the order of the expansion. The amount of nonlinearity in the hidden neuron function is explicitly controlled by the set of coefficients \(\{a_n\}_{n=1}^{N}\) and thus prevent the network from suffering the ‘overfitting’ phenomenon. Moreover, the usefulness of polynomials have been demonstrated from the Weierstrass theorem [16] which states that for every function \(f \in C^0([a,b]; \mathbb{R})\) (the space of continuous function from \([a,b]\) to \(\mathbb{R}\)), there exists a sequence of polynomials \((p_0, p_1, p_2, \ldots)\) that converges uniformly to \(f\) on \([a,b]\):

$$\forall \epsilon > 0 : \exists l > L(\epsilon) : \forall x \in [a,b] : \|f(x) - p_l(x)\|_2 < \epsilon$$ \hspace{1cm} (10)$$

The theorem is of crucial importance in the present context of nonlinear signal restoration because it guarantees the existence of useful polynomial approximations. The input-output relationship of the 3-layer perceptron polynomial neural network is described by

$$y^{(l)}_i = \sum_{j=1}^{N_4} \sum_{l=1}^{N_3} w_{ij}^{(l)} g_j^{(l-1)}(y^{(l-1)}_j)$$ \hspace{1cm} for \(i = 1, 2, \ldots, N_4\) $$

with \(g_k^{(0)}(y_k^{(0)}) = x_k\) and the following hidden neurons function
\[ g^{(2)}_k(y^{(2)}_k) = b_{0,k} + b_{1,k}y^{(2)}_k + b_{2,k}(y^{(2)}_k)^2 + \cdots + b_{N,k}(y^{(2)}_k)^N = \sum_{n=0}^{N} b_{n,k}(y^{(2)}_k)^n \]  \hspace{1cm} (12)

\[ g^{(1)}_j(y^{(1)}_j) = a_{0,k} + a_{1,k}y^{(1)}_j + a_{2,k}(y^{(1)}_j)^2 + \cdots + a_{N,k}(y^{(1)}_j)^N = \sum_{n=0}^{N} a_{n,k}(y^{(1)}_j)^n \]  \hspace{1cm} (13)

which is evidently given by the polynomial series expansion of its input and the coefficients \( \{a_{n,k}\}_{n=0}^{N} \) and \( \{b_{n,j}\}_{n=0}^{N} \) are the sets of variable parameters to be optimised.

### 3.1 Regularised Cost Function for Signal Restoration

Specifically, it is desirable that the outputs of the nonlinear demixer are: (i) as independent as possible and (ii) as close as possible to the source signals in the norm-2 sense, when the cost function reaches its minimum point. To facilitate such provision, the set of constraints acting as explicit ‘regulariser’ is added to the original cost function as follows:

\[
\Psi = -h(y) + \sum_i h_i(y_i) + \sum_i \alpha_i f_i(y_i, s_i)
\]

\[
= -h(x) - \log \left| \det \frac{dy}{dx} \right| - \sum_i \log \tilde{q}_i(y_i) + \sum_i \alpha_i f_i(y_i, s_i)
\]  \hspace{1cm} (14)

where the original cost function stems from the KLD [1-3] which is commonly used in source separation problem, \( y_i \) is the \( i^{th} \) output of a demixer, \( h(y) \) and \( h_i(y_i) \) denote the joint and marginal entropy respectively, \( \alpha_i \)'s are the scalar constants chosen to provide the required amount of weights on the constraints and \( f_i(y_i, s_i) \)'s are the constraints constructed from the \textit{a priori} information about the source distributions which may assume the form of cumulants and/or moments that are intended to be matched with the outputs of the nonlinear demixer, i.e.,
where $M$ is the order of the cumulant matching at the outputs of the nonlinear demixer. The expression given in (14) shall be referred to as the constrained KLD ($c$-KLD) which represents our current signal restoration cost function. The KLD term appearing in the first half of Eqn. (14) will extract the independent signal information from the mixtures while the set of constraints together with extracted information assist in reconstructing the original signals.

**Theorem 2**

The effective cost function for the 3-layer PNN demixer assumes the following form:

$$J = -\log \left| \det \frac{\partial f^{(3)}}{\partial y} \right| - \sum_i \log q_i(y^{(3)}_i) + \sum_i \alpha_i f_i(y^{(3)}, s_i)$$

$$= -\log \left| \det W_3 \right| - \log \left| \det W_2 \right| - \log \left| \det W_1 \right| - \sum_n \sum_i \log \left| n b_{i,n} (y^{(2)}_i)^{n-1} \right| - \sum_m \sum_i \log \left| m a_{i,m} (y^{(1)}_i)^{m-1} \right|$$

$$- \sum_i \log q_i(y^{(3)}_i) + \sum_i \alpha_i f_i(y^{(3)}, s_i)$$

**Proof:** See Appendix A.

**Lemma 2**

Let $x = (x_1, x_2, \ldots, x_p)$ be a random variable and let $y = h(\Theta, x)$ be a function of $x$, differentiable with respect to the non-random parameter $\Theta$ and such that $y$ accepts a differentiable pdf $p_Y(y)$. Then, the derivatives of the entropy $H(y)$ [9] with respect to $\Theta$ is given by

$$\frac{dH(y)}{d\Theta} = -E \left[ \psi_Y(h(\Theta, X)) \frac{dh(\Theta, X)}{d\Theta} \right]$$

(17)

where $\psi_Y(y) = \hat{p}_Y(y)/p_Y(y)$ and $\hat{p}_Y(y) = \frac{dp_Y(y)}{dy}$. 

8
Theorem 3

By application of lemma 2, the generalised stochastic gradient descent update rules corresponding to Theorem 2 for the 3-layer PNN demixer with parameters $\Theta = \{W_1, W_2, W_3, [d_{a,k}]_{n=0}^{N}, [d_{b,k}]_{n=0}^{N}, [d_{c,k}]_{n=0}^{N}\}$ are given by

$$W_3(t + 1) = W_3(t) + \eta_3 \left[ I - \left[ \sum_{n=1}^{N} n(n-1) \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-2}] \right] \phi^{(2)} \right] (y^{(3)})^T W_3(t)$$

$$W_2(t + 1) = W_2(t) + \eta_2 \left[ I - \left[ \sum_{n=1}^{N} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-1}] \right] \phi^{(2)} \right] (y^{(2)})^T W_2(t)$$

$$W_1(t + 1) = W_1(t) + \eta_1 \left[ I - \left[ \sum_{m=1}^{N} m \text{diag}[a_m] \text{diag}[(y^{(1)})^{m-1}] \right] \phi^{(1)} \right] (y^{(1)})^T W_1(t)$$

$$\text{diag}[b_n(t + 1)] = \text{diag}[b_n(t)] - \eta_b \left[ \left[ \sum_{n=1}^{N} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-1}] + \phi^{(2)} \right] (y^{(3)})^T W_3(t) \right]$$

$$\text{diag}[a_m(t + 1)] = \text{diag}[a_m(t)] - \eta_a \left[ \left[ \sum_{n=1}^{N} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-1}] + \phi^{(2)} \right] (y^{(3)})^T W_3(t) \right]$$

where $\eta_1, \eta_2, \eta_3, \eta_a$ and $\eta_b$ are the constant fixed step sizes that control the amount of updates.

**Proof:** See Appendix B.

### 3.2 New Demixing Schemes for Parameter Learning Algorithm

The algorithm in (18)-(22) is used primarily for updating the parameters of a general PNN demixer. It has yet to exploit the optimal structure established in theorem 1. However, if the PNN is used for implementing such
required structure, then the learning algorithm will undertake a simpler form since only $W_3$, $W_2$ and 
$\left\{ \left[ \theta_{n,j} \right]_{n=0}^{N} \right\}_{j=1}^{V_1}$ need to be updated while $W_1$ and $g_k^{(1)}(y_k^{(1)})$ can be pre-computed as the inverse of $W_3$ and 
$g_k^{(2)}(\cdot)$, respectively. While the overall computational complexity of the algorithm has been substantially reduced, the new scheme necessitates us to select at least odd order polynomial in order for $g_k^{(2)}(\cdot)$ to be invertible.

The generalised stochastic gradient algorithm exemplified in (18)–(22) searches for the solution in a multidimensional space along the steepest descent direction. Such search can be extremely slow and ineffective if the e-KLD has many plateaus distributed throughout the landscape or when the algorithm is trapped in local minima. In such cases, parallel search such as the Genetic Algorithms (GA) [17,18] may offer a better strategy in ameliorating both problems. There are however a few issues to be aware of, particularly that the GA is not so proficient in fine tuning the optimum solution even after locating an appropriate region in the solution space whereas gradient algorithms exhibit good performance only in local optimisation. Moreover, since the search space is enormously vast, the randomisation that occurs during the mutation process of the GA may result in unproductive attempts in searching along incorrect directions. The shortcomings are further augmented by its huge demand of computational complexity in evaluating the fitness for all chromosomes in a population. Therefore, a hybrid learning algorithm that combines both GA and gradient descent algorithm is devised in order to incorporate the merits of both methods. This approach employs the mutation operator as a searching tool for the gradient method to acquire a quicker trajectory in learning the optimal solution during the adaptation phase. This is advantageous when the gradient descent algorithm is trapped at a local minimum or that the convergence rate is relatively small due to the convergence to plateaus resulting in the rate of change of the cost function dropping within a particular range, which will then activate the genetic search by randomly perturbing the values of the current demixer weights to generate a number of new sets of weights. The algorithm then chooses the set that optimises the fitness function as the survivor among these new sets of weights and original parent weights. Commencing from the new state, the survivor will be adapted by the gradient descent algorithm until it converges to the global solution. These procedures are repeated whenever the gradient method converges to another local minimum or the convergence rate is found to be too small at a regular interval. The new demixer
learning algorithms based on the hybrid of modified GA and gradient algorithm are outlined in the following two schemes:

**• Scheme 1 (for general PNN structure)**

**Step 1:** Initialise \( \Theta(0) = [W_1(0), W_2(0), W_3(0), [a_{n,k}(0)], [b_{n,k}(0)]], \) \( \delta_1 = \) constant, \( \delta_2 = \) constant

**Step 2:** Compute \( y^{(3)}(t) = W_3(t)g^{(2)}(W_2(t)g^{(1)}(W_1(t)x(t))) \) \( (23) \)

\[
F_1(t) = -\log|\det W_1(t)W_2(t)W_3(t)| - \sum_i \log \frac{dg_i^{(1)}(y_i^{(1)}(t))}{dy_i^{(1)}(t)} - \sum_i \log \frac{dg_i^{(2)}(y_i^{(2)}(t))}{dy_i^{(2)}(t)} + \sum_i \alpha_i f_i(s_i(t), y_i^{(3)}(t)) - \sum_i \log \tilde{q}_i(y_i^{(3)}(t))
\]

\[
\Delta F_1 = \frac{F_1(t) - F_1(t - \lambda)}{\lambda} \quad (\lambda \neq 0)
\] \( (25) \)

**Step 3:** If \( F_1(t) > \delta_1 \), update \( \Theta_1(t) \) according to the gradient descent algorithm in (18)–(22).

**Step 4:** If \( F_1(t) > \delta_1 \) and \( |\Delta F_1| < \delta_2 \), activate the GA and generate a new set according to

\[
\Theta_{1,m}(t) = \Theta_1(t) + \mu_m(t)N_m(0,1) \quad m = 1,2,\ldots,M
\] \( (26) \)

Compute the fitness for each \( \Theta_{1,m}(t) \) according to \( F_1(t) \) and together with \( \Theta_1(t) \), select the best survivor that optimises the fitness function.

**• Scheme 2 (for PNN using the structure established in theorem 1)**

**Step 1:** Initialise \( \Theta_2(0) = [W_2(0), W_3(0), [b_{n,k}(0)]], \) \( \delta_1 = \) constant, \( \delta_2 = \) constant

**Step 2:** Compute \( y^{(3)}(t) = W_3(t)g(W_2(t)g^{-1}(W_3^{-1}(t)x(t))) \) where \( g_i(u_i) \) is selected by the user. \( (27) \)

\[
F_2(t) = -\log|\det W_2(t)| - \sum_i \log \frac{dg^{-1}_i(y_i^{(1)}(t))}{dy_i^{(1)}(t)} - \sum_i \log \frac{dg_i(y_i^{(2)}(t))}{dy_i^{(2)}(t)} + \sum_i \alpha_i f_i(s_i(t), y_i^{(3)}(t)) - \sum_i \log \tilde{q}_i(y_i^{(3)}(t))
\]

\[
\Delta F_2 = \frac{F_2(t) - F_2(t - \lambda)}{\lambda} \quad (\lambda \neq 0)
\] \( (29) \)

**Step 3:** If \( F_2(t) > \delta_1 \), update \( \Theta_2(t) \) according to the gradient descent algorithm in (18)–(22).

**Step 4:** If \( F_2(t) > \delta_1 \) and \( |\Delta F_2| < \delta_2 \), activate the GA and generate a new set according to
\( \Theta_{2,m}(t) = \Theta_2(t) + \mu_m(t)N_m(0,1) \quad m = 1,2,\ldots,M \) (30)

Compute the fitness for each \( \Theta_{2,m}(t) \) according to \( F_2(t) \) and together with \( \Theta_2(t) \), select the best survivor that optimises the fitness function.

In both schemes, \( N_m(0,1) \) denotes a random sample drawn from a zero mean, unit variance gaussian distribution while \( \mu_m(t) \) is the step size that controls the perturbation which is also adapted according to 
\[
\mu_m(t) = \mu_m(t-1)\exp(-t N(0,1) - \tau_0 N_m(0,1))
\]
where \( N(0,1) \) and \( \tau_0 \) are some fixed random and deterministic constants, respectively. This is to ensure that good step size evolution will be kept while the bad ones will die out over time.

4. RESULTS

In the first experiment, a simple nonlinear mixture is considered since such study can assist us in gaining insights into the efficacy of each proposed scheme. Let the nonlinear function in (7) assumes the form of a general 3rd order polynomial \( f(u) = u + \gamma u^3 \) where \( \gamma \in \mathbb{R} \) is a variable that controls the amount of nonlinearity in the function. Hence, the nonlinear mixing system for the case of 4 sources and 4 sensors can be derived and expressed analogous to (6) as

\[
x_i = f(m_{1i} f^{-1}(s_1) + m_{12} f^{-1}(s_2) + m_{13} f^{-1}(s_3) + m_{14} f^{-1}(s_4)) \quad \text{for} \quad i = 1,2,3,4
\]

Observation Equation

\[
\text{Observation Equation} = m_{11} \left( \frac{\alpha_1^{1/3}}{\epsilon_{1\gamma}} - \frac{2}{\alpha_1^{1/3}} \right) + m_{12} \left( \frac{\alpha_2^{1/3}}{\epsilon_{2\gamma}} - \frac{2}{\alpha_2^{1/3}} \right) + m_{13} \left( \frac{\alpha_3^{1/3}}{\epsilon_{3\gamma}} - \frac{2}{\alpha_3^{1/3}} \right) + m_{14} \left( \frac{\alpha_4^{1/3}}{\epsilon_{4\gamma}} - \frac{2}{\alpha_4^{1/3}} \right)
\]

\[
+ \gamma \left( m_{11} \left( \frac{\alpha_1^{1/3}}{\epsilon_{1\gamma}} - \frac{2}{\alpha_1^{1/3}} \right) + m_{12} \left( \frac{\alpha_2^{1/3}}{\epsilon_{2\gamma}} - \frac{2}{\alpha_2^{1/3}} \right) + m_{13} \left( \frac{\alpha_3^{1/3}}{\epsilon_{3\gamma}} - \frac{2}{\alpha_3^{1/3}} \right) + m_{14} \left( \frac{\alpha_4^{1/3}}{\epsilon_{4\gamma}} - \frac{2}{\alpha_4^{1/3}} \right) \right)^3
\]

where \( \alpha_i = \left( 108 s_i + 12 \sqrt{\frac{12 + 81 \gamma s_i^2}{\gamma}} \right)^{1/3} \). Four sub-gaussian sources are generated synthetically with 30dB white gaussian noise perturbing each sensor. The parameters \( \{m_{ij}\} \) are randomly selected from a Gaussian distribution. The sources and the outputs of the mixture with \( \gamma = 2 \) are displayed in Fig. 2(a)-(b), respectively. The performance of the proposed algorithms outlined in scheme 1 and scheme 2 will be studied using this
mixture. The truncated 4th order Edgeworth series in (b.4) (Appendix B) is used to estimate the marginal entropy since both sources are sub-gaussian distributed. Scheme 1 employs a 3-layer PNN where the 1st hidden layer assumes the form of a 5th order polynomial while the 2nd hidden layer a 3rd order polynomial. Scheme 2 employs a similar structure but the 2nd hidden layer utilises a 3rd order odd polynomial in the form of 

\[ g_i(y^{(2)}_i) = y^{(2)}_i + b_1(y^{(2)}_i)^3 \]

while the 1st hidden layer which is the inverse of \( g_i(\cdot) \) has been derived and expressed in the following closed form function:

\[
g_i^{-1}(y^{(1)}_i) = -\left( \frac{12}{6} \right)^{\frac{1}{3}} \left\{ -\left( 9y^{(1)}_i + \frac{12 + 8lb_i(y^{(1)}_i)^2}{b_i} \right)^\frac{2}{3} + b_1(12)^{\frac{1}{3}} \right\}
\]  

(32)

Note that in computing the nonlinearity of \( g_i(y^{(2)}_i) \) and \( g_i^{-1}(y^{(1)}_i) \), only a single coefficient is tuned and updated which subsequently leads to a simpler algorithm compared with scheme 1. The results of the restored signals based on linear ICA [1], gradient descent PNN using Eqns. (18)-(22), PNN Scheme 1 and PNN Scheme 2 with \( \lambda = 4 \) are displayed in Fig. 3(a)-(d), respectively where the step sizes used are set to 0.0005. The convergence plots of the fitness function for the last three algorithms are displayed in Fig. 4. From this plot, it is observed that both PNNs with modified GA algorithms successfully relocate to the new points after converging to the local minima (or plateaus) while the gradient descent PNN algorithm is trapped indefinitely in the local minima (or plateaus). Concurrently, we compare the attained performance with three existing nonlinear demixing algorithms: RBF [12], Bayesian Ensemble Averaging (BEA) method [13] and FMLP with logistic nonlinearity [14]. As a comparison, we propose the following performance index:

\[
\Omega = \frac{1}{nT} \sum_{t=1}^{T} \sum_{i=1}^{n} \left[ \frac{s_i(t)}{\sqrt{E(s_i^2)}} - \frac{y^{(L)}_i(t)}{\sqrt{E(y^{(L)}_i^2)}} \right]^2 
\]  

(33)

where \( y^{(L)}_i(t) \) is the \( i^{th} \) final output of the demixer at time \( t \). Fig. 5 shows the performance indices achieved by the six tested demixers for a range of Signal to Noise ratio (SNR). From the plot, it is shown that the proposed
schemes have outperformed the three nonlinear demixing algorithms and that the PNN Scheme 2 achieves the best result. We also observe that the performance attained by the linear ICA falls far from being optimum.

In order to test the efficacy of the proposed schemes under more complex situation, the same set of source signals is passed through the following nonlinear mixing model:

\[
\mathbf{x} = \mathbf{M}_4 \sinh^{-1}(\mathbf{M}_3 \tanh(\mathbf{M}_2 \sinh^{-1}(\mathbf{M}_1 \mathbf{s})))
\]  \hspace{1cm} (34)

where \( \mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3 \) and \( \mathbf{M}_4 \) are \( 4 \times 4 \) randomly chosen invertible matrices. It is difficult to explicitly show how the above nonlinearly distorted signals can be decomposed according to (7) but we show by simulation that such signals can be uniquely retrieved. PNN Scheme 1 employs a 3-layer PNN where the order of polynomial is taken up to the 13\(^{th}\) order at both hidden layers while Scheme 2 uses an odd polynomial up to the 9\(^{th}\) order. The entire update process with \( \lambda = 7 \) is executed using the same settings as in the first experiment. Fig. 6 shows the performance indices achieved by the six tested demixers for a range of SNRs. From the plot, we identify that the proposed schemes have outperformed the rest of the algorithms and that the best result is attained by PNN Scheme 2.

In the third experiment, real-life recording of speech signal mixture [14] is used. The experiment was conducted in an auditorium and acoustic absorbers were used to avoid the echoes. The recordings were taken between the two speakers. In the set up the distance between the signals and the microphones was 2 meter. We allowed the recording amplifier to operate in the saturation region (class-C operation). The recorded signals were sampled at 24K bits per second. The original source recordings are displayed in Fig. 7(a) along with the received signals at the input of the demixer in Fig. 7(b). Scheme 1 employs a 3-layer PNN where the order of polynomial is taken up to the 9\(^{th}\) order at both hidden layers while scheme 2 uses an odd polynomial up to the 5\(^{th}\) order. The entire update process with \( \lambda = 20 \) is executed using the same settings as in the first experiment. The restored speech signals based on linear ICA [1], FMLP with 3 layers [14] (where both hidden layers nonlinearity is given by \( \tanh(\cdot) \)), PNN scheme 1 and PNN scheme 2 are displayed in Fig. 8(a)-(d), respectively. The convergence plots of the fitness function for both schemes and the FMLP are displayed in Fig. 9. Similar to Fig. 4, the trajectories followed by the proposed algorithm show that the PNN demixers are able to leap out from continually dwelling in the local minima (or plateaus) and subsequently produce better results. In addition, we deduce that the
difference in the fitness value between the two steady-state solutions achieved by the PNN schemes arises as a natural outcome of using a finite order polynomial in estimating the inverse of a polynomial having the same order. Moreover, the disparity between the performance of the two schemes becomes more substantial when the mixture is increasingly nonlinear. Finally, the performance indices achieved by the six tested demixers are plotted in Fig. 10 for a range of SNRs. On the other hand, Table 1 shows the results of the speech recognition accuracy after feeding unseen 102 sentences with a total of 1209 nonlinearly distorted spoken words through the demixers. The results are obtained only after all learning algorithms have converged to their steady-state solutions. The outputs of the demixers are evaluated by using the SCLITE software (version 1.5) where the test results are compared with the reference files and the performance benchmark tests results are given in the form of reports using the options in SCLITE. The results acquired from both performance index and speech recognition accuracy clearly show the efficacy of the proposed schemes in restoring real-life recording of nonlinearly mixed signals. Similar to the previous experiments, our analysis has shown that the PNN Scheme 2 is the most efficient with the highest percentage of correct word recognition.

5. SUMMARY

The technique of restoring nonlinearly mixed signals using 3-layer PNN has been presented coupled with the derivation of the stochastic gradient descent algorithm for training the parameters. In addition, a new learning algorithm based on the hybrid of the modified Genetic Algorithm and gradient descent algorithm has been developed to instigate a more efficient basis for locating the desired solution. The overall development constitutes a novel framework for restoring nonlinearly mixed signals in a blind manner. Moreover, the novel framework can be generalised further to include multiple layers PNN (more than 3 layers) where the nonlinearity at every layer can be adapted according to the proposed schemes. This paper also shows that the optimal structure of the demixing system for any nonlinear mixtures modelled in (7) is given by the feedforward multilayer neural network with at least 2 hidden layers as established in theorem 1. Finally, three sets of experiments have been conducted and meticulously studied which successfully demonstrate the efficacy of the proposed algorithm in retrieving and reconstructing original sources that have been mixed nonlinearly.
6. ACKNOWLEDGEMENTS

The authors are grateful to Prof. O.R. Hinton (University of Newcastle) for his supports and the discussions with Dr. S. Sali (University of Newcastle) have greatly contributed to this work. The authors would also like to thank the anonymous reviewers for their helpful suggestions and comments in improving the draft of the manuscript.

7. REFERENCES


7. APPENDIX

A. Derivation of the Effective Cost Function for the 3-layer PNN Nonlinear Demixer

The input-output relationship of the PNN demixer can be expressed as

\[
\begin{align*}
y_i^{(3)} &= \sum_{j=1}^{N_k} w_{ij}^{(3)} g_j^{(2)}(y_j^{(2)}) \quad \text{for } i = 1, \ldots, N_4 \\
y_j^{(2)} &= \sum_{k=1}^{N_k} w_{jk}^{(2)} g_k^{(1)}(y_k^{(1)}) \\
y_k^{(1)} &= \sum_{l=1}^{N_j} w_{kl}^{(1)} x_l 
\end{align*}
\] (a.1)

In vector form, the derivatives of the outputs with respect to the inputs are as follows:

\[
\frac{dy^{(3)}}{dx} = W_3 \sum_{n=0}^{N_i} \text{diag}[b_n] \cdot \text{diag}[(y^{(2)})^{n-1}] W_2 \sum_{m=0}^{N_j} \text{diag}[a_m] \cdot \text{diag}[(y^{(1)})^{m-1}] W_1
\] (a.2)

where diag[b_n] = diag[b_{0,n}, b_{1,n}, \ldots, b_{N_j,n}] and similarly, diag[a_m] = diag[a_{0,m}, a_{1,m}, \ldots, a_{N_j,m}]. Hence, using the c-KLD as effective cost function, we have

\[
J = -\log|\det\frac{dy^{(3)}}{dx}| - \sum_i \log \tilde{q}_i(y_i^{(3)}) + \sum_i \alpha_i f_i(y_i^{(3)}, s_i) \\
= -\log|\det W_3| - \log|\det W_2| - \log|\det W_1| - \sum_n \sum_i \log |n b_{i,n} (y_i^{(2)})^{n-1}| - \sum_m \sum_i \log |m a_{i,m} (y_i^{(1)})^{m-1}| \\
- \sum_i \log \tilde{q}_i(y_i^{(3)}) + \sum_i \alpha_i f_i(y_i^{(3)}, s_i)
\] (a.3)

where \(\alpha_i\)'s are the scalar constants, \(f_i(y_j, s_j)\)'s are the constraints, \(W_1, W_2, W_3\), coefficients \(\{a_{n,k}\}_{n=0}^{N_i} \) and \(\{b_{n,j}\}_{n=0}^{N_j} \) are the sets of variable parameters to be optimised in this demixer network.
B. Derivation of the Generalised Stochastic Gradient Descent Learning Algorithm

The differential of the cost function in (a.3) yields the following expression after tedious derivation:

\[
\begin{align*}
\frac{\partial J}{\partial W} &= -\text{tr}[dW_3W_1^{-1}] - \text{tr}[dW_2W_2^{-1}] - \text{tr}[dW_1W_1^{-1}] + \frac{d}{\sum_{n} \sum_{i} \log |n_{b_{i,n}} (y_i^{(2)})^{n-1}|} \\
&+ \sum_{n} \sum_{i} \log |n_{a_{i,m}} (y_i^{(1)})^{m-1}| - \sum_{i} \log |n_{g_{i}}(y_i^{(3)}) + \alpha_i f_i(y_i^{(3)}, s_i)| \\
&= -\text{tr}[d\xi_1] - \text{tr}[d\xi_2] - \text{tr}[d\xi_3] + \left[\phi^{(2)}\right]^T \\
&= \sum_{n} \sum_{i} m \left[\phi^{(2)}\right]^T \text{diag}(\mathbf{a}_m) \text{diag}(\mathbf{y}_i^{(1)})^{m-1} + \sum_{m} m(n-m) \text{diag}(\mathbf{a}_m) \text{diag}(\mathbf{y}_i^{(1)})^{m-2} \text{dy}^{(1)} \\
&+ \left[\phi + \mathbf{a} \cdot \mathbf{f}^T\right] \text{dy}^{(2)} \\
\end{align*}
\]

where

\[
\phi^{(2)} = \begin{bmatrix}
\frac{1}{\sum_{n} n_{b_{1,n}} (y_i^{(2)})^{n-1}} & \frac{1}{\sum_{n} n_{b_{2,n}} (y_i^{(2)})^{n-1}} & \cdots & \frac{1}{\sum_{n} n_{b_{N,n}} (y_i^{(2)})^{n-1}} \\
\frac{1}{\sum_{m} m_{a_{1,m}} (y_i^{(1)})^{m-1}} & \frac{1}{\sum_{m} m_{a_{2,m}} (y_i^{(1)})^{m-1}} & \cdots & \frac{1}{\sum_{m} m_{a_{N,m}} (y_i^{(1)})^{m-1}} \\
\end{bmatrix}^T
\]

(2.2)

\[
\phi^{(1)} = \begin{bmatrix}
\frac{1}{\sum_{n} n_{b_{1,n}} (y_i^{(2)})^{n-1}} & \frac{1}{\sum_{n} n_{b_{2,n}} (y_i^{(2)})^{n-1}} & \cdots & \frac{1}{\sum_{n} n_{b_{N,n}} (y_i^{(2)})^{n-1}} \\
\frac{1}{\sum_{m} m_{a_{1,m}} (y_i^{(1)})^{m-1}} & \frac{1}{\sum_{m} m_{a_{2,m}} (y_i^{(1)})^{m-1}} & \cdots & \frac{1}{\sum_{m} m_{a_{N,m}} (y_i^{(1)})^{m-1}} \\
\end{bmatrix}^T
\]

(3.2)

and \(d\xi_1 = dW_1W_1^{-1}, d\xi_2 = dW_2W_2^{-1}, d\xi_3 = dW_3W_3^{-1}\) are the non-integrable differentials that define basis of the tangent space of each matrix in the Riemannian sense [1]. The symbol ‘\(\circ\)’ denotes the Hadamard product,

\[
\mathbf{a} = [\alpha_1, \alpha_2, \cdots, \alpha_{N_4}]^T, \quad \mathbf{f} = [\mathbf{j}_1, \mathbf{j}_2, \cdots, \mathbf{j}_{N_4}]^T \quad \text{with} \quad \mathbf{j}_i = \frac{df_i(y_i^{(3)}, s_i)}{dy_i^{(3)}},
\]

\[
\phi^{(j)} = [\phi_1^{(j)}(y_i^{(j)}) \cdots \phi_{N_4}^{(j)}(y_i^{(j)})]^T \quad \text{with} \quad \phi_i^{(j)}(y_i^{(j)}) = \frac{-\tilde{g}_i^{(j)}(y_i^{(j)})}{\tilde{g}_i^{(j)}(y_i^{(j)})} \quad \text{where} \quad \tilde{g}_i^{(j)}(y_i^{(j)})
\]

and \(\tilde{g}_i^{(j)}(y_i^{(j)})\) are the first and second order derivatives of the polynomial nonlinearity with respect to \(y_i^{(j)}\).
\( \varphi^{(3)} = [\varphi_1^{(3)}(y_1^{(3)}) \varphi_2^{(3)}(y_2^{(3)}) \cdots \varphi_N^{(3)}(y_N^{(3)})]^T \) with \( \varphi_i^{(3)}(y_i^{(3)}) = -\frac{d \log q_i(y_i^{(3)})}{d y_i^{(3)}} \) and \( q_i(y_i^{(3)}) \) is the marginal pdf defined at the outputs of the 3\textsuperscript{rd} layer which can be approximated using probability expansion series such as the Gram-Charlier [1] or Edgeworth [2] series. As an example, using the truncated Edgeworth series up to 4\textsuperscript{th} order, \( \varphi_i^{(3)}(y_i^{(3)}) \) assumes the following form:

\[
\varphi_i^{(3)}(y_i^{(3)}) \approx \left( -\frac{1}{2}k_3 + \frac{7}{4}k_4^2 + \frac{3}{2}k_3^2k_4 \right)(y_i^{(3)})^2 + \left( -\frac{1}{6}k_4 + \frac{1}{2}k_3^2 \right)(y_i^{(3)})^3 \tag{b.4}
\]

where \( k_3 \) and \( k_4 \) are the third and fourth order cumulant of \( y_i^{(3)} \), respectively. By considering infinitesimal changes of the cost function with respect of the parameters \( \Theta = \{ W_1, W_2, W_3, \{(d_{n,k})_{k=0}^N \}_{n=1}^{N_2}, \{(\theta_{n,j})_{j=0}^N \}_{n=1}^{N_3} \} \), the learning algorithm can be derived as follows: Starting with the output layer, the derivatives of cost function with respect to the differentials are given by

\[
\frac{dJ}{d\xi_3} = -I + [\varphi + a \circ f][y^{(3)}]^T \tag{b.5}
\]

\[
\frac{dJ}{d\xi_2} = -I + \left[ \sum_{n=2}^{N_2} n(n-1) \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-2}] \right] \left[ \varphi^{(2)} \right][y^{(2)}]^T \tag{b.6}
\]

\[
+ \left[ \sum_{n=1}^{N_3} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-1}] \right] W_3^T \left[ \varphi^{(3)} \right][y^{(1)}]^T
\]

\[
\frac{dJ}{d\xi_1} = -I + \left[ \sum_{m=1}^{N_3} \text{diag}[a_m] \text{diag}[(y^{(1)})^{m-1}] \right] W_2^T \left[ \sum_{n=2}^{N_1} n(n-1) \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-2}] \right] \left[ \varphi^{(2)} \right][y^{(1)}]^T
\]

\[
+ \left[ \sum_{m=1}^{N_2} \text{diag}[a_m] \text{diag}[(y^{(1)})^{m-1}] \right] W_1^T \left[ \sum_{n=1}^{N_2} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-1}] \right] W_3^T \left[ \varphi^{(3)} \right][y^{(1)}]^T \tag{b.7}
\]

\[
+ \left[ \sum_{m=2}^{N_2} m(m-1) \text{diag}[a_m] \text{diag}[(y^{(1)})^{m-1}] \right] \left[ \varphi^{(1)} \right][y^{(1)}]^T
\]
Proceeding in the similar way, the derivatives with respect to the sets \( \{a_{n,k}\}_{k=0}^{N_2} \) and \( \{b_{n,j}\}_{j=0}^{N_1} \) of the neuron hidden polynomial function are given by

\[
\frac{dJ}{d \text{diag}[b_n]} = \text{diag}[m[\phi^{(1)}][(y^{(2)})^{(n-1)}]^T + W_3^T[\phi^{(3)}][(y^{(2)})^n]^T]
\]

(b.8)

\[
\frac{dJ}{d \text{diag}[a_m]} = \text{diag}\left[m[\phi^{(1)}][(y^{(1)})^{(m-1)}]^T + W_2^T\sum_{n=2}^{N_1} n(n-1) \text{diag}[b_n] \text{diag}[(y^{(2)})^{n-2}] \left[\phi^{(2)}][(y^{(1)})^n]^T \right.ight.
\]

(b.9)

\[
+ W_2^T\sum_{n=1}^{N_1} n \text{diag}[b_n] \text{diag}[(y^{(2)})^{n+1}] W_3^T[\phi^{(3)}][(y^{(1)})^n]^T \right]
\]

Using the fact that \( d\xi_i = dW_iW_i^{-1} \) and therefore, \( \frac{dW_i(t)}{dt} = \frac{d\xi_i(t)}{dt} W_i(t) = -\eta \frac{dJ}{d\xi_i(t)} W_i(t) \), we have the weights update algorithm given as \( W_i(t+1) = W_i(t) - \eta_i \frac{dJ}{d\xi_i(t)} W_i(t) \) for \( i = 1,2,3 \). On the other hand, the update equation for the coefficients of the polynomials follows directly from the steepest descent gradient:

\[
\text{diag}[b_n(t+1)] = \text{diag}[b_n(t)] - \eta_b \frac{dJ}{d\text{diag}[b_n(t)]} \quad \text{and} \quad \text{diag}[a_m(t+1)] = \text{diag}[a_m(t)] - \eta_a \frac{dJ}{d\text{diag}[a_m(t)]}.
\]

This completes the derivation of the generalised stochastic gradient descent algorithm for training the network parameters.
Figure 1

Figure 2
Figure 3
Figure 4

Figure 5
Figure 6

Figure 7
Figure 8
Figure 9

Figure 10
<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Correctly recognised (%)</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PNN Scheme 1</td>
<td>80.2</td>
<td>21.9</td>
</tr>
<tr>
<td>PNN Scheme 2</td>
<td>85.1</td>
<td>16.2</td>
</tr>
<tr>
<td>FMLP</td>
<td>75.8</td>
<td>26.4</td>
</tr>
<tr>
<td>BEA</td>
<td>73.7</td>
<td>28.1</td>
</tr>
<tr>
<td>RBF</td>
<td>63.2</td>
<td>37.2</td>
</tr>
<tr>
<td>Linear ICA</td>
<td>41.1</td>
<td>58.9</td>
</tr>
</tbody>
</table>

Table 1
Figure Caption

**Figure 1**: 3-layer Polynomial Neural Network (PNN) as the nonlinear demixer.

**Figure 2**: Simulated signals.
   (a) Original source signals. (b) Outputs of the nonlinear mixture.

**Figure 3**: Restored signals using the proposed algorithms at 30dB SNR.
   (a) Linear ICA. (b) PNN using eqns. (18)-(22). (c) PNN Scheme 1. (d) PNN Scheme 2.

**Figure 4**: Convergence of the fitness function (for experiment 1).

**Figure 5**: Performance index of each tested algorithm at different SNR (for experiment 1).

**Figure 6**: Performance index of each tested algorithm at different SNR (for experiment 2).

**Figure 7**: Real-life recordings.
   (a) Original speech signals. (b) Observed signals.

**Figure 8**: Restored signals using the proposed algorithms.
   (a) Linear ICA. (b) FMLP (3-layer). (c) PNN Scheme 1. (d) PNN Scheme 2.

**Figure 9**: Convergence of the fitness function (for experiment 2).

**Figure 10**: Performance index of each tested algorithm at different SNR (for experiment 3).
Table Caption

Table 1: Speech Recognition Accuracy (for experiment 3).