IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 7, Issue 2, Ver. I (Mar. - Apr. 2017), PP 26-33
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Performance Analysis of the Sigmoid and Fibonacci Activation Functions in NGA Architecture for a Generalized Independent Component Analysis

James Paul CHIBOLE[1,*], Heywood Ouma ABSALOMS[1, 2], Edward Ng'ang'a NDUNGU[1, 3]

[1]*Department of Telecommunication and Information Engineering, School of Electrical, Electronics and Information Engineering, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.*
[2]*Department of Electrical Engineering, Faculty of Engineering, University of Nairobi, Nairobi, Kenya.*
[3]*Department of Telecommunication and Information Engineering, School of Electrical, Electronics and Information Engineering, Jomo Kenyatta University of Agriculture and Technology, Nairobi, Kenya.*

Abstract: *Activation functions are used to transform the mixed inputs into their corresponding output counterparts. Commonly, activation functions are used as transfer functions in engineering and research. Artificial neural networks (ANN) are the preferred choice for most studies and comparisons of activation functions. The Sigmoid Activation Function is the most common and its popularity arise from the fact that it is easy to derive, its boundedness within the unit interval, and it has mathematical properties that work well with the approximation theory. On the other hand, not so common is the Fibonacci Activation Function with similar and perhaps better features than the Sigmoid. Algorithms have a broad range of applications making it plausible to suspect that different problems call for unique activation functions. The aim of this paper is to have a detailed of the role of the activation functions and then analyse the performance of two of them – the Sigmoid and the Fibonacci – in a non-ANN setup using the most basic artificially generated signals. Results show that the Fibonacci activation function performs better with the set of signals applied in the natural gradient algorithm.*

Keywords: *Fibonacci Activation Function, Sigmoid Activation Function, Independent Component Analysis, Natural Gradient Algorithm.*

I. Introduction

The use of adaptive algorithms to solve various problems is predominant because they can adapt their behaviour to reflect the changing characteristics of the modelled system. Karlik and Olgac observe that there has been enormous research to investigate methods aimed at improving the performance of ANN through optimised training methods, network structure and learn parameters with little work dedicated to activation functions [1]. Adaptive algorithms that do not use the ANN in their implementation have also received little attention. On activation functions, the sigmoid is seen as the preferred choice because it attains the approximation power much better when compared to other established functions studied in the approximation theory of polynomials and spines [2]. In some advanced studies, the use of the hyperbolic tangent sigmoid in the first and second layers and a linear function in the output layers of an ANN network are the most effective when used within the decision algorithm for transmission system fault detection [3]. Despite the likely use of Fibonacci Activation Function, there is no real application of it in literature other than in [4] and only on the experimental basis. Fibonacci activation function has also been picked up in [5] and used in the separation of two speech signals with promising results. A significant observation is that most of the studies on activations functions use neural networks. Therefore, a comparison of the Fibonacci activation function (FAF) to the widely preferred alternative, the Sigmoid activation function (SAF), is necessary. A general understanding of activation functions and why they are critical in the algorithm is not well expounded in literature. What is the performance of FAF when compared to SAF for separating artificially generated signals in terms of quality and stability?

II. Role Of Activation Functions

Input signals are often non-linear in nature. However, during the mixing process, such inputs are combined linearly. In fact, inputs to algorithms are simply a linear transformation.

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} \qquad (1)$$

where $\boldsymbol{x}$ is a vector of mixed signals, $\boldsymbol{A}$ is the mixing matrix and $\boldsymbol{s}$ is the vector of the input signals. To retain back the original signals s, a reverse transformation through a de-mixing matrix $\boldsymbol{W}$ is necessary but often not sufficient.

$$\boldsymbol{y} = \boldsymbol{Wx} \qquad (2)$$

The linear transformation in Equation 2 is necessary but will need to be passed through an activation function even if the inputs are linear. Because most inputs are non-linear, $\boldsymbol{y}$ can effectively resemble $\boldsymbol{s}$ if the transformation passes through a non-linear activation function. Otherwise, it is not possible to retain the non-linearity of the input signals. Non-linearity here means that it is not possible to reproduce the output when the inputs are combined linearly.

While the input takes values in the range $[-\infty, \infty]$, the output controlled by the activation function gives bounded values within the intervals [1, 0] or [-1, 1]. Without an activation function, it is not possible to map the wide inputs to relatively small output parameters.

One essential characteristic of an activation function is that it must be differentiable within its bounded interval. Differentiation helps to give the direction of adjusting the weights. When the derivative value is significant, it means the corresponding weights will equally have large weight adjustments. Calculus rules are that a significant derivative value indicates the minima is still far. For each iteration, the weights of the algorithm are adjusted to correspond to the direction of the steepest descent on the surface of the cost function defined by the total error. Computing the error is by subtracting the expected from the observed values. Then each weight within the matrices of weights is adjusted according to the gradient error calculated. In a more general sense, the derivation is done along the activation function curve as the expected value using optimization techniques such as the natural gradient in finding the minima of the objective function.

The first derivative determines the first point on the curve by ensuring that there is a tangent with a slope of zero on the line tangent. A slope of zero suggests the location of the *minima* and it can be local or global for the function. However, the first derivative also suggests something significant: it informs the algorithm if headed in the correct direction that will take it to the minimum of the function. The derivative value, that is, the slope at that point decreases gradually. What this means is that for a minimized function, its derivative must be calculated and that the value must be decreasing if the algorithm is following the correct direction. It is for this reason that the activation function must be differentiable within its range, revealing it critical role in algorithms. The better the choice of the activation functions the better the algorithm.

2.1. Comparison of Activation Functions

This paper compares the commonly used activation function the Sigmoid to the less explored in the literature, the Fibonacci. [4] showed that the Fibonacci activation function outperforms another (name not known) but having the equation:

$$\varphi(y) = \frac{2}{3}y^3 + \frac{1}{3}y^5 \qquad (3)$$

In fact, the SAF is so common that there is an ANN named after it, the Sigmoid Neurones. Sigmoid neurones are modified version of the perceptions where small changes in the bias and weights reflect as small variations in the output [6]. Equation 4 shows the FAF

$$\varphi(y) = \frac{\sqrt{5}-1}{2}y^3 + \frac{3-\sqrt{5}}{2}y^5 \qquad (4)$$

while the SAF has the equation

$$\varphi(y) = \frac{1}{1+e^{-y}} \qquad (5)$$

and their MATLAB generated graphs are shown in Figure 1, below.

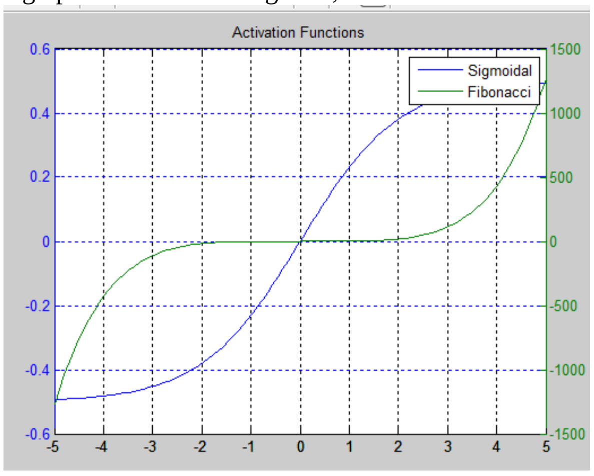

Figure 1: Sigmoid and Fibonacci Activation Functions

An activation function is the cumulative density function (cdf) and the reason for its use is explained below. Assuming there is a random variable s, having probability density $P_s(S)$. The cumulative density function (cdf) is given as:

$$F(s) = P(S \leq s) \quad (6)$$

Suppose, this is the Gaussian density then

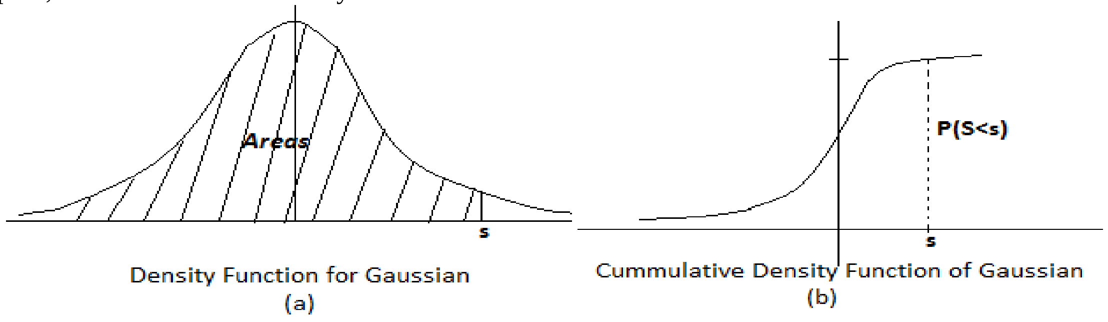

Figure 2: Density function (a) and its cdf (b)

The height of the function s (Figure 2 (b)) is equal to the area of the Gaussian density (Figure 2 (a)), up to the period s. Going further (Figure 2 (b)) the function (Figure 2 (a)) becomes Gaussian. Equation 7 is another form of writing the cdf of Equation 6.

$$F(S) = \int_{-\infty}^{s} P_s\,(t)dt \quad (7)$$

Suppose there is a random variable s and the aim is to model the distribution of this variable. Two option exist: the first option is by specifying the density P_s(s) or the second option is to specify the cdf $F(S)$. Equation 7 is used to relate the two options. Most literature prefers the cdf option because of the easy in the calculation and because it is continuous over the specified space. Continuous here means it is differentiable over its range. In signal processing and other applications, the cdf is truly referred to as the activation function. It is, therefore, easy to recovery the density P_s(s) by taking the cdf and computing its derivative

$$P_s(S) = F'(S) \quad (8)$$

There is always a step in algorithms when the random variable for s is assumed by either specifying the density P_s or the cdf. In the real sense, the assumptions involve the choice of an activation function that can model the output of the signals to be separated. The cdf or the activation function has to be some function increasing from 0 to 1 or some other specified range.

III. Generalized Independent Component Analysis

Let us assume that the data comes from n original sources as speakers

$$s \in \mathbb{R}^n \quad (9)$$

where $s_j^{(i)} = signal\ from\ speaker\ j\ at\ time\ i$

we observe

$$x^{(i)} = As^{(i)} \quad (10)$$

where $X \in \mathbb{R}^n$. This means that there are n microphones and each of them records some linear combination of what the speaker says:

$$x_j^{(i)} = \sum_k A_{jk}\, s_k^{(i)} \quad (11)$$

Equation 11 can be interpreted to mean the *jth* microphone is recording a linear combination of all the speakers are saying at time *i*. The goal is to find the matrix $W = A^{-1}$ so that

$$s^{(i)} = Wx^{(i)} \quad (12)$$

It is possible to recover the original sources as a linear combination of $Wx^{(i)}$.

The use of ICA is because $s_j^{(i)} \backsim Uniform\ [-1, 1]$, each of the speaker outputs uniform white noise

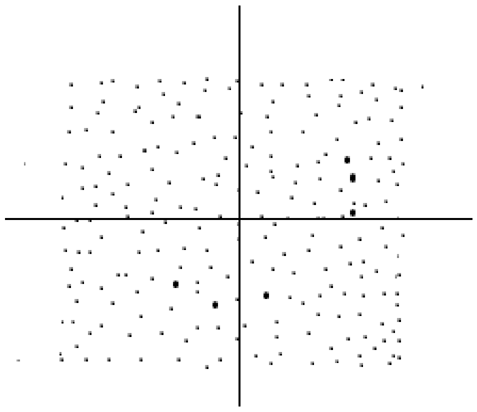

Figure 3: Original source signals $s_1$and s_2

Figure 3 is part of the $s_1$and s_2, the original sources and is what the two speakers will be outputting because each of the speakers is giving out uniform [-1, 1] independent random variables. The microphone, however, records

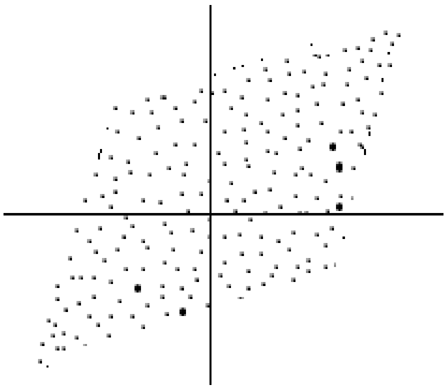

Figure 4: Microphone observed signals

From Figure 4, it is easy to tell the axis of this parallelogram and it is also easy to note which linear transformation is responsible for transforming Figure 4 to Figure 3.

There are ambiguities in ICA. One of them is that it is not possible to recover the original indexing of the order. In particular, the generated data from speaker-1 and speaker-2 when running on ICA there is the possibility of ending up with the reversed order of speakers. That corresponds to taking Figure 4 and flipping it along the 45^0 axis. The axis reflects the picture in either way and it will still be within the box. There is no way an algorithm can tell which was speaker-1 and which was speaker-2. The ordering is ambiguous. The other source of ambiguity is the sign of the sources. It is not possible to tell whether you got back positive $s_1^{(i)}$ or whether you got back negative $s_1^{(i)}$. In Figure 4, what that corresponds to is depicted by taking the figure and reflecting it along the vertical axis, it will reflect along the horizontal axis and you will still get the box. So, it turns out that in this example, there is a guarantee of recovering positive s^i or negative s^i. The two are, therefore, the only ambiguities in this case – permutation of the speakers and the sign of the speakers. It turns out that the reason these are the only source of ambiguity was that $s_j^{(i)}$ is non-Gaussian, an essential characteristic for ICA to work.

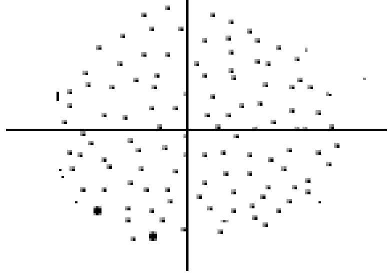

Figure 5: Gaussian Signals

Suppose the original sources were Gaussian (Figure 5), then $s^{(i)} \backsim \mathcal{N}(0, I)$ - Non-Gaussian. Gaussian is a spherically symmetric distribution. Figure 5 shows the continuous of a Gaussian distribution with identity covariance. If the data is Gaussian, then it is impossible to use ICA.

Mathematically, let $s \in \mathbb{R}^n$ and the probability density function of $s = P_s(s)$. Therefore, $X = AS = W^{-1}s$ and where $s = Wx$. Will this information at hand, next is to find the probability density function of W, that is, $P_x(X)$. It is easy to substitute the above equations and get

$P_x(X) = P_s(Wx)$ (13)

Equation 13 is only possible for most functions but not for continuous density function. For continuous density function, the right formula is

$P_x(X) = P_s(Wx)|w|$ (14)

where the additional $|w|$ on Equation 14 is the determinant of the matrix w. (Shortly we will see this determinant in use in the NGA in an applied sense, given as $|detW|$). As an example to illustrate this, suppose

$P_s(s) = I\{0 \leq s \leq 1\}$ (15)

showing that the density of s is uniform distribution over [0, 1]. Figure 6 illustrates this distribution.

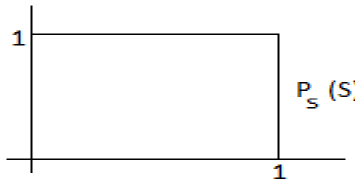

Figure 6: Density distribution of s over [0, 1] interval

If s is uniform over [0, 1], then x will be the uniform distribution over [0, 2], because $x = 2s$, therefore, $A = 2$ and $w = \frac{1}{2}$. For x, the density would be

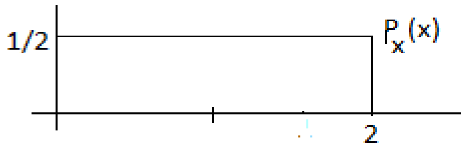

Figure 7: Density distribution of x over [0, 2] interval

In this case $P_x(x) = I\{0 \leq x \leq 2\}.\frac{1}{2}$. where I denote the indicator.

IV. Natural Gradient Model Architecture

The natural gradient algorithm (NGA) is an improvement on the shortcoming of the stochastic gradient algorithm (SGA), which demands the specifying of the initial condition. In both SGA and the modified NGA, the aim is to adjust the coefficient of the demixing matrix. Adjusting $W(k)$ means that the joint Probability Density Function (jPDF) of the separated signal $y(k)$ is made as close as possible to the assumed distribution $\bar{p}_y(y)$ [7] in each iteration of the algorithm. The measure has been successfully done by Cardoso using the divergence measure of Kullback-Leibler [17].

$KL(p_y \parallel \bar{p}_y) = \int p_y(y) \log\left(\frac{p_y(y)}{\bar{p}_y(y)}\right) dy$ (16)

where $p_y(y)$ is the actual distribution and $\bar{p}_y(y)$ is the assumed distribution of the estimated signal vector. In an ideal situation $p_y = \bar{p}_y$ giving a KL of zero. In fact, Equation 16 can correctly be referred to as the objective function [5].

The tricky part is in choosing $\bar{p}_y(y)$, which makes Equation (16) unreliable in many applications. A cost function that is instantaneous and which has the same expected value as that of the $KL(p_y \parallel \bar{p}_y)$ is given by

$$\check{J}(W) = -log\left(\prod_{i=1}^{m} p_s(y_i(k))\, |detW|\right) \qquad (17)$$

where $detW$ indicates the determinant W. The cost function (Equation 17) can be computed further to yield the stochastic gradient descent given as

$W(k+1) = W(k) + \mu(k)[W^{-T}(k) - \varphi(y(k)x^T(k))]$ (18)

here $\varphi(y) = [\varphi(y_1), \dots, \varphi(y_m)]^T$, $\varphi(y) = -\delta log p_s(y/dy)$, is the cdf or algorithmically the activation function. $\mu(k)$ is the step size used by the algorithm to move from one iteration to the next. According to [7], Equation (18) is able to perform Blind Signal Processing (BSS) for any simple choice of $\varphi(y)$. The Stochastic gradient descent of Equation (18) is limited regarding application because of its slow convergence. A modification to the stochastic gradient descent to remove the mixing matrix A's ill-condition is the natural gradient algorithm by Amari [18] or the relative gradient by Gardoso [17]. In NGA, the initial condition is not put into consideration because the algorithm aims at the overall system matrix $C(k)$. The full derivation of the NGA is found in [7] with its final equation given as:

$$C(k+1) = C(k) + \mu(k)[I - \varphi(C(k)s(k))s^T(k)C^T(k)]C(k) \qquad (19)$$

From Equation 19 it is evident that the mixing matrix A does not play any role in the separation process as it only used in the initial condition $C(0) = W(0)A$. Therefore even if the mixing matrix is poorly chosen, it will not affect the quality of separated signal. The NGA works even better if the inputs $s(k)$ adhere to the ICA's independence requirement. Despite the modifications, both SGA and NGA require a good choice of the activation function $\varphi(.)$ as shown in Equation 19.

V. Experiment

Four sub-Gaussian signals were used as inputs to the algorithm

$$\left.\begin{aligned} s_1(t) &= sawthooth(t) \\ s_2(t) &= square(t) \\ s_3(t) &= \sin(t) \\ s_4(t) &= \cos(t) \end{aligned}\right\} \qquad (20)$$

The four inputs have a duration of 100 seconds with a sample time of 0.01. The MATLAB waveforms of these signals is shown below.

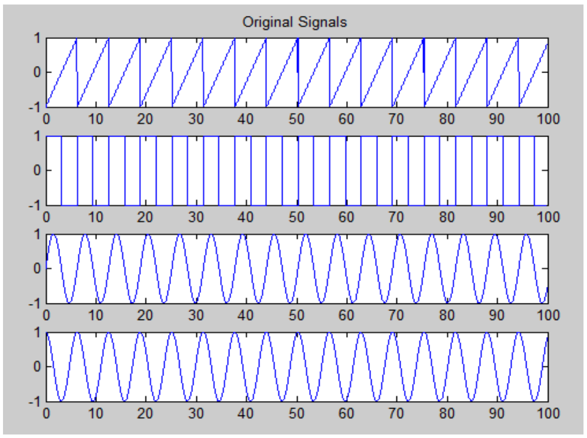

Figure 8: From top to bottom – sawthooth, square, sine and cosine signals

The inputs are then mixed together using a 4-by-4 mixing matrix generated randomly in the range [-1,+1]. The output of this mixture is shown below

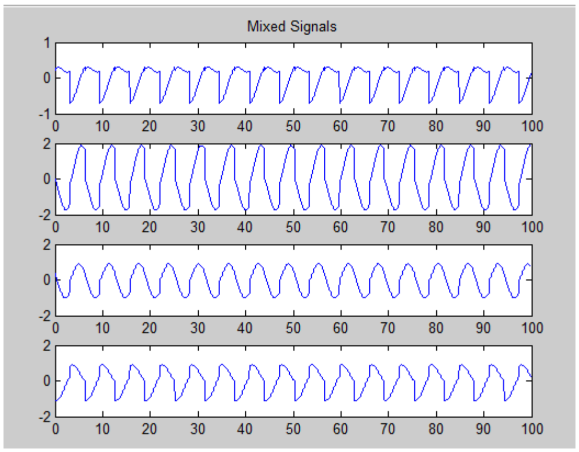

Figure 9: Mixed input signals

The NGA algorithm is now applied to the mixture of Figure 9 with the Fibonacci and later Sigmoid activation function in turns. The learning rate is set at $\boldsymbol{\mu = 0.001}$ while the initial demixing matrix is set at $\boldsymbol{W_o = 0.1I}$.

VI. Performance Comparison

The outputs from the two activation functions are shown below

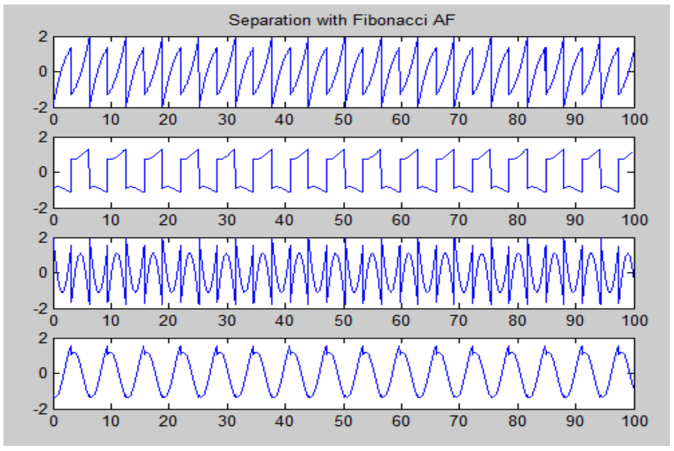

Figure 10: Separated Signals using FAF

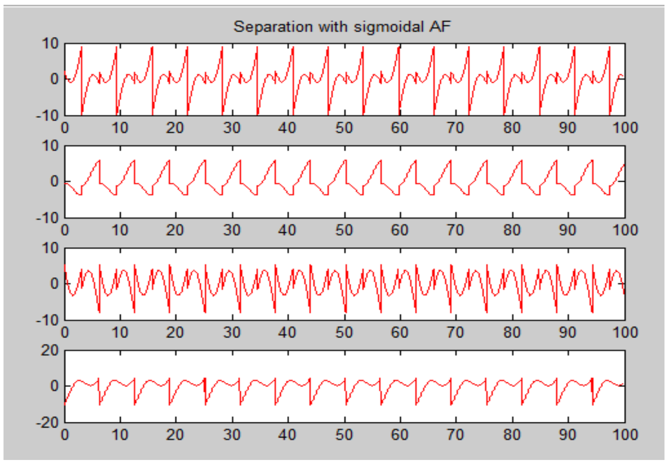

Figure 11: Separated Signals using SAF

It is evident from the two outputs of Figures 10 and 11 that the FAF realizes much better results than the SAF. Further tests on the convergence and stability of the algorithm using FAF and SAF were done with results shown in Figure 12.

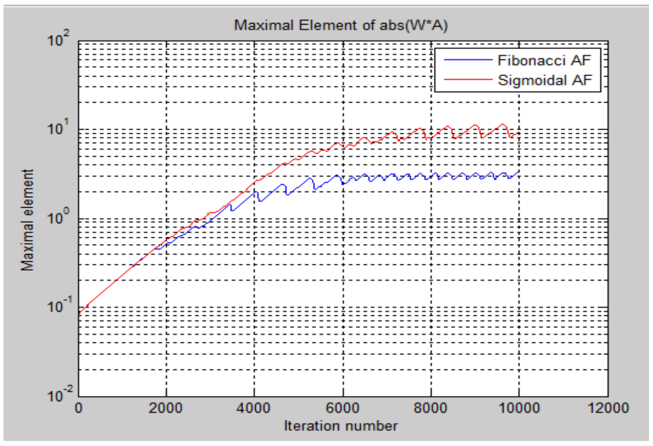

Figure 12: Convergence rate of FAF and SAF

At 10000 iterations it is clear that the algorithm converges much faster when FAF is employed than SAS. It also indicates that the stability of the algorithm is much higher when FAF is used as opposed to SAF and shown by the few elements required to reach convergence.

Finally, a test on dynamic error was also done.

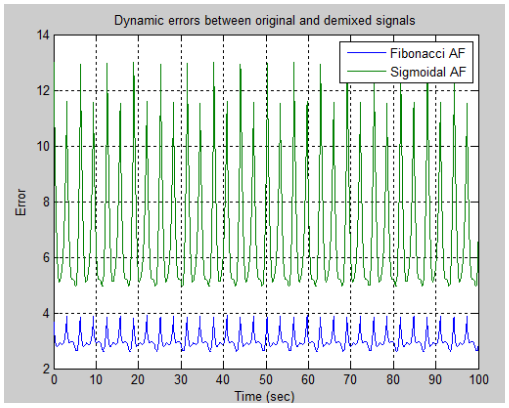

Figure 13: Dynamic error

It is clear that the dynamic error when the input are compared to their corresponding output signals reveal that when FAF is used it gives a smaller error when compared to SAF.

VII. Conclusion

The findings reveal that the performance of algorithms is significantly affected by the choice of the activation function. With a focus on activation function research; there are possibilities of algorithms performing better on problems that have long been identified with specific AF. For example, despite the overwhelming use of the Sigmoid in research and engineering applications, this study has shown that Fibonacci is an equally promising AF. Specifically, the findings show that FAF, when used in NGA, makes the algorithm more stable and converges much faster than when SAF is used. FAF also realizes a better error than SAF.

References

[1]. B. Karlik and V. A. Olgac, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," International Journal of Artificial Intelligence And Expert Systems (IJAE), vol. 1, no. 4, pp. 111-122, 2011.
[2]. B. DasGupta and G. Schnitger, "The Power of Approximating: A Comparison of Activation Functions," Advances in Neural Information Processing Systems, vol. 5, pp. 615-622, 1993.
[3]. N. Suttisinthong, B. Seewirote and A. Ngaopitakkul, "Selection of Proper Activation Functions in Back-propagation Neural Network algorithm for Single-Circuit Transmission Line," in Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS , Hong Kong, 2014.
[4]. L. Lei, W. Yu and W. Xing-Hui, "Natural gradient algorithm based on a class of activation functions and its applications in BSS," in Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, Dalian, 2006.
[5]. J. P. Chibole, "Blind separation of two human speech signals using natural gradient algorithm by employing the assumptions of independent component analysis," in Proceedings of the 2014 International Annual Conference on Sustainable Research and Innovation, Nairobi, 2014.
[6]. M. Nielsen, Neural Networks and Deep learning, New York: Determination Press, 2015.
[7]. D. C. Scott, "7 Blind Signal Seperation and Blind Deconvolution," New York, CRC Press LLC, 2002.
[8]. J. F. Cardoso, "Blind signal separation: statistical principles," in Proceedings of the IEEE, Cambridge, MA, 1998.
[9]. S. Amari, "Natural gradient works efficiently in learning," Neural Computation, vol. 10, pp. 251-276, 1998.