Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.7, July 2016 – www.caeaccess.org

Stochastic Solution on Convergence Property of Quadratic Functions

Adekunle Y. A.
Babcock University
Ilishan-Remo
Ogun State, Nigeria

Ebiesuwa Seun
Babcock University
Ilishan-Remo
Ogun State, Nigeria

Yoro R. E.
Babcock University
Ilishan-Remo
Ogun State, Nigeria

ABSTRACT
Constraints satisfaction problems (CSPs) aim at solution via algorithms that search a domain space for goal states. The solution of which must satisfy all constraints and guarantees explicit reasoning structure that conveys data about the problem to the algorithm. Thus, it assigns to an output, a set of variables that satisfies a set of constraints in its bid to prune off huge portion of the search space. This study presents solutions to quadratic functions via David Fletcher Powell method and stochastic method of optimization. To aim this purpose, hybrid neural networks are trained using DFP as a pre-processor to yield approximate solutions to the quadratic function. A trial solution of the quadratic equation is written as sum of two parts: (a) first part satisfies the initial condition for unconstrained optimization using DFP and hybrids as separate methods to solve a quadratic function; while (b) second part uses DFP as a pre-processor with adjustable parameters of for the ANN-TLRN hybrid. Results show that presented method introduces a closer form to the analytic solution. These present method is easily extended to solve a wide range of problems.

Keywords
Stochastic, elitist, network, function, optimization, search space, solution

1. INTRODUCTION
Search methods aim to maximize or minimize constraint satisfaction problem objective functions and yield a feasible, optimal (closest to best) solution. CSPs are dynamic, nonlinear and complex – making the search for its solution cumbersome and inexplicable to resolve. Thus, search methods must be tuned to resolve CSP optimization as it relates inputs with uncontrollable parameter in the modeled system to satisfy all possible constraints to yield an output that is both flexible, optimal, easily adapted and robust (Ojugo, 2013a).

Nonlinear systems modeled via mathematical equations, appear in diverse spheres of life as dynamic feats and phenomena in control systems, medicine, communication etc. Such systems are modeled as quadratic functions and various methods are used to solve quadratic/differential equations. Finding the optimal solution of nonlinear, quadratic equations is still a challenge task (Fletcher and Reeves, 1964; Forsythe, 1986 and Friedlander et al, 1999). Many studies have shown that parallel processor computers in order to solve the first order differential equation using Hopfield neural network models as a factor of speed; while some, used feedforward ANN to solve linear/nonlinear ordinary differential equations (Ghalambaz et al, 2011); while Lagaris et al (1998) went further to represent a new method to solve first-order linear ordinary and partial equations using ANN, as was further buttressed by Malek and Shekari (2006). Khan et al (2009) provided a hybrid ANNPSO intelligence model to solve the well-known Wessinger equation (though the model could not satisfy initial and boundary conditions). It was improved by Ghalambaz et al (2011); while Khan et al (2009) furthered to satisfy the solution for initial and boundary condition problems. The study presents comparative stochastic model for solving quadratic equation using DFP as preprocessor to seek optimality and convergence property. The paper is has: Section 1 as detailed description of optimization, Section 2 as problem formulation, Section 3 is brief review of adopted stochastic frameworks, Section 4 discusses results. And finally, conclusions and directions of future research.

1.1. Mathematical Optimization
Mathematical optimization is a selection of the best element from a set of available alternatives. It thus, aims either at a maximization or minimization task of real function that symmetrically choose inputs from an allowed set of variables, to compute the function's output by finding the best available values from a set of alternatives via the objective function in a given domain (Ojugo, 2012b). Armijo (1966) and Ojugo (2012a) notes that:

Definition 1: A continuous optimization defined as a pair (S, f). S is set of possible solutions: $S = R^N$ and N is number of controllable parameters and R is the real number line. Thus, *f* is a multi-objective function (f: S → R) to be optimized – where a solution vector X is as limited between lower and upper bounds ($X_{lb} \leq X \leq X_{ub}$). For maximization task (search for a solution greater than or equal to all other solutions), and minimization task (search for solution that is smaller than or equal to the all other solutions).The set of maximal and minimal solution $S_{max} \subseteq S$ of a function f: S → R defined as:

$$X_{max} \in S_{max} \Leftrightarrow \forall X \in S: f(X_{max}) \leq f(X) \quad (1)$$

$$X_{min} \in S_{min} \Leftrightarrow \forall X \in S: f(X_{min}) \leq f(X) \quad (2)$$

2. DFP: THE PROBLEM FORMULATION
We aim to minimize

$$\boldsymbol{f(x_1x_2) = (x_2 - x_1)^2 + (1 - x_1)^2 \; with \; x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}} \text{ and}$$

$$\boldsymbol{f(x_1x_2) = x_1^2 + 25x_2^2, x_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \; as \; start \; points}$$

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.7, July 2016 – www.caeaccess.org

Dai (2001) notes that a point x* ε A is a global minimum of $f_o(x)$ if: $\boldsymbol{f(x^0) \leq f(x) for\ all\ x\ \varepsilon\ A}$ (3). An ideal optimization allows its objective function to have a unique minimize. Thus, the function in Eq. 3 is a global minimizer. Conversely, a point x* ε A is a local minimize if for any ε > 0, f(x*) ≤ f(x) and for any x ε A, ||x – x*|| ≤ ε

2.1 Line Search Method / Algorithm

Dai et al (2002) and Polyak (1969) notes that numeric method algorithm for unconstrained optimization (as grouped into line search and trust region) aims to minimize the nonlinear function f(x) as in Eq. 4. Thus, an iterative, continuous f(x) with initial point x_1, its k^{th} iteration (new point x_{k+1}) is computed as thus (Barzilar, 1988 and Raydan, 2003):

$$Y = \mathbf{Min}_{x\ \varepsilon\ R^N}\ f(x) \quad (4)$$

The goal of convergence analysis is to study feats of sequence $\{x_k\}$ generated and compare the differences between the convergence performances of the various models (Batruni, 1991; Powell, 1971 and 1976). The sequence x_k generated is said to converge to a point x* if:

$$Y = \lim_{k\to\infty} ||x_k - x^*|| = 0 \quad (5)$$

With the solution for x* as in Eq. 5 not available, a possible replacement is Eq. 6, which unfortunately, also does not guarantee the convergence of x_k will then yield Eq. 6 as given by Daniel (1967) and Gottlieb and Orszag (1977) as thus:

$$Y = \lim_{k\to\infty} ||x_k - x_{k-1}|| = 0 \quad (6)$$

However, the global convergence for unconstrained optimization aims to prove that Eq. 7 ensures x_k is close to the set of stationery points where ∇(x) = 0, or Eq. 8, which ensures that a least subsequence of x_k is close to the set of the stationery points (Dai and Yuan 1999; 2003).

$$\lim_{k\to\infty} ||g_k|| = 0 \quad (7)$$

$$\lim_{k\to\infty} InF||g_k|| = 0 \quad (8)$$

But $g_k = g(x_k) = \nabla f(x_k)$. Local convergence aims at the convergence speed of the sequence generated by the algorithm. In studying this, we assume a sequence x_k converges to a local minimize x*, so that the second order sufficient condition and the Newton method, can converge very slowly (Polak, 1969 and Polyak, 1969).

2.2 Quasi-Newton Method

Dennis and Moore (1974) note Newton's method as the basis for Quasi-Newton, and most widely used for solving nonlinear equations and unconstrained optimization. If for a smooth function f(x) and a positive hessian, then Second order Taylor's expansion yields thus:

$$f(x+p) \cong f(x) + p^T \nabla_f(x) + \frac{1}{2} p^T \nabla_f(x)$$

$$Differentiating\ yields\ \frac{\partial f(x+p)}{\partial p} \cong \nabla_f(x) + \nabla^2 f(x) = 0$$

$$Thus, \nabla^2 f(x) = -\nabla_f(x) \quad (9)$$

As long as $\nabla^2_f(x)$ is a positive definite, $\nabla^2_f(x)p$ is Newton direction, and the next approximation is:

$$x^{k+1} = x^k - \frac{\nabla_f(x)}{\nabla^2 f(x)} \quad (10)$$

Newton direction has proven to be more expensive than Steepest Descent direction. We must compute the hessian matrix and invert it (not applicable with Quasi Newton). Its merits are: (a) convergent rate for Newton method is quadratic, and thus, there is a lot to gain in finding its direction, (b) they form a good starting point if *f*'(x) is positive definite, and (c) they are simple and easy to implement. However, its demerits includes: (a) they are not globally convergent for many problems, (b) may diverge if the starting point approximation is far from the solution, (c) it fails if hessian matrix is not inverted, and (d) requires analytic second order derivatives of *f*. Also, the method is seen as an approximation of the Newton Raphson method (Dennis and Moore, 1977; Liu and Storey, 1991).

$$x^{k+1} = x^k - \frac{f(x)}{f'(x)} \quad (11)$$

Consider behaviour of Quasi Newton method from Broyden's class of unconstrained optimization task given by: *min*{f(x): x ε R^N}. Class consists of iterations of the form:

$$x_{k+1} \leftarrow x_k + \alpha_k \rho_k\ where\ \rho_k = -\beta_k^{-1} g_k \quad (12)$$

g_k is gradient of f at x_k, α_k is stepsize and hessian approximation β_k is updated by Broyden (1967) and Cauchy (1987):

$$\beta_{k+1} = \beta_k - \frac{\beta_k s_k s_k^T \beta_k}{s_k^T \beta_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \emptyset(s_k^T \beta_k s_k) v_k v_k^T \quad (13)$$

This yields two update formulae: For $\emptyset = 0$, yields Broyden, Fletcher, Goldfarb and Shanno (BFGS) method; while $\emptyset = 1$, yields Davidson, Fletcher and Powell (DFP) method (Dennis and Moore, 1977). Dixon (1972) notes that with exact line search, all class members yield same iterates and their performance varies markedly. We assume step-size α_k is chosen by an inexact line search satisfying two conditions as thus:

$$f(x_k + \alpha_k \rho_k) \leq f(x_k) + \sigma \alpha_k g_k^T \rho_k \quad (14)$$

$$g(x_k + \alpha_k \rho_k)^T \rho_k \geq \beta g_k^T \rho_k \quad (15)$$

Dennis and Moore (1977) note several important results about this class of methods is that they do not require an exact line search. If all assumptions hold that: x* is the minimize of *f*, the hessian matrix H is positive definite, $\alpha_k = 1$ in DFP/BGFS, then Eq. 16 is true and x_k converges to x* Q-superlinearly, and its sum will be finite, if $||x_1 – x^*||$ and $||\beta_1 –$

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.7, July 2016 – www.caeaccess.org

H|| are sufficiently small and stepsize α_k=1 will satisfy the line search conditions of Eq. 14 and 15 (Starchuski, 1981; Ritter, 1979 and Griewank and Toini, 1982).

$$\sum_{k=0}^{\infty} \left| |x_{k+1} - x^*| \right| < \infty \quad (16)$$

2.3 DFP as Preprocessor

DFP optimizes (with/without an exact line search). Here, DFP computes solution of a given quadratic functions (as problem domain) via stochastic hybrid models, define convergence properties and state its influence on such convergence properties. The algorithm is (Raydan, 2003; Fletcher, 1987):

At iteration *i*, DO:

1. Step1 – choose $A_i \cong H^{-1}$ (inverse of hessian matrix at i).
2. Step2 – if x^i is optimal, stop; Else obtain the search direction pi by solving: $\rho^i = -Ai\nabla fi(x)$
3. Step3 – Minimize f(x) in the direction of pi via

$$\min_{\alpha} f\left(x^i + \alpha\rho^i\right) \; to \; find \; \alpha$$

4. Step4 – define $\rho^i = x^{i+1} - x^i = \Delta x^i \; (change \; in \; x)$

   $y^i = \nabla f_{i+1}(x) - \nabla f_i(x) = \Delta g_i(change \; in \; gradient)$

5. Step5 – Compute: $\beta_i = \frac{(\rho^i)^T(\rho^i)}{(\rho^i)^T(y^i)}$

$$C_i = \frac{-A_i y^i (A_i y^i)^T}{A_i y^i (y^i)^T}$$

6. Step6 – Update the hessian matrix A as thus

   A_{i+1} = A*i* + B*i* + C*i*

   Set i = *i* + 1 and return to step 2

3. INTELLIGENT OR SOFT COMPUTING

Optimization methods tuned via Artificial Intelligence models have yielded biological, evolutionary and stochastic heuristic whose implementation span across medicine, electronics etc – to mention a few, and examples include genetic algorithm, simulated annealing, ant colony and particle swarm optimization, gravitational search algorithm, fuzzy, bacteria foraging etc., today referred to as soft-computing (Ojugo, 2012a).

3.1 Artificial Neural Network (ANN)

ANN as a data processing model is inspired by biological neurons of the human brain – and consists of interconnected neurons, whose major feat is in their ability to ***learn*** by example via simulation, making them universal estimators. It learns as neurons share electromechanical signals via dendrites and synapse axon that converts the signals. This helps it process data. Learning occurs by adjusting the synapses weight(s), and inputs are summed by adder. Based on the task at hand, its activation function limits its output. A simple mathematical model as in Eq 18 with its synapse weight connections helps modulate the associated inputs and nonlinear feats exhibited in neurons via an activation function as thus (Pham and Liu, 1995):

$$\emptyset = f(net) = f\sum_{i=1}^{m} Xi * Wij \quad (18)$$

Ojugo et al (2013a, b) note encoded, ANN has three basic layers: input, hidden and output, and two configurations: (a) feedforward network (allows data to flow from input-to-output with no feedback as the network extends over multiple layers), and (b) recurrent network (which has a dynamic feedback to help the network undergo relaxation and evolve to a stable state if there is no further change in its activation values and output. Output change is significant and dynamic behavior constitutes its output). Lee and Kang (1990) the configuration of choice is dependent on the application area, feats and system requirement. Various methods are used to set its connection strengths so that learning takes place. These includes: (a) explicit connection via apriori knowledge, and (b) implicit connection ***post-priori*** in which the network is trained to learn patterns that changes its weight in a learning rule. Learning is grouped into:

a. Supervised – here, input vector with set of desired responses, one for each node, is relayed to the output. A forward pass is done and errors between desired and actual response for each node in output is found, and used to determine weight changes based on the learning algorithm Thus, desired output signals is yielded by an external teacher. Examples are Perceptron, delta and back-propagation (Ojugo et al, 2012b).

b. Unsupervised – Here, output is trained to respond to clusters of patterns that help network to discover statistical, salient feats in the input, with no prior knowledge of how patterns are grouped. Thus, model develops its own representation of the input (Conway et al, 1998).

c. Reinforcement – output learns what to do, map states to actions and must discover actions that yield the most reward by trying them. Such actions may affect not only the immediate data, but also the rest states. Its trial and error search and delayed reward are its two distinguishing feats (Plumb et al, 2005).

Study adopts the TLRN (MLP with short memory) architecture with ***unsupervised*** learning and RBF a control model to compare results obtained by training network to yield results and provide a fail-safe to eradicate noise in realtime data-stream. Thus, the network learns from experiences, generalized from previous datasets to new ones with abstract feats, at its inputs containing irrelevant data (Denton and Hung, 1996). Trial-error is used in selecting number of hidden layers and nodes in each hidden layer. Previous results have shown that ANN with a hidden layer outperforms those with two/more – as this only increases the number of parameter that only complicates training. The optimal hidden layer size is found by systematically increasing the number of hidden node until network's performance shows no further improvement or it longer improves significantly. The network is complex enough to accurately simulate dynamic, nonlinear feats. Standard tasks use 15, 30, 45, 60 and 100 hidden nodes (on each layer) to examine model's performance and our study however,

Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 5 – No.7, July 2016 – www.caeaccess.org

adopts ***single*** hidden layer with 18-hidden nodes (Pham and Liu, 1995; Pham and Karaboga,1999 and Pham et al, 2006).

3.2 Gravitation Search Algorithm (GSA)

GSA is based on Newton's laws of gravity and motion with its main idea, being to consider isolated system of masses, where every mass represents a solution to a certain problem. Law of gravity states that every particle attracts another and the gravitational force between particles are directly proportional to the product of their masses and inversely proportional to distance between them (Rashedi et al, 2009a). Thus, an agent's performance depends on its mass. They attract each other via gravitational pull towards those with heavier mass. Agents are initialized at start point, are randomly located in space so that gravitational force is defined as thus:

$$\boldsymbol{Fij = G(t) = \frac{Mi(t) * Mj(t)}{Rij(t) + \varepsilon} \{Xj(t) - Xi(t)\}} \quad \mathbf{(19)}$$

R_{ij} is the Euclidean distance between masses for the objects (i and j) masses, G(t) is gravitation force at time t with small constant ε – which decreases in time to control the population and search's accuracy. Thus, the total force acting on an agent is:

$$\boldsymbol{F_i^d = \sum_{j \in kbest, j \neq 1} rand(i) * Fij} \quad \mathbf{(20)}$$

rand randomizes agents' initial state at intervals [0,1]. Acceleration of i at time t, in *d* dimension is directly proportional to force acting on agent I, and inversely proportional to agent's mass:

$$Aid(t) = \frac{Fid(t)}{Mij(t)} \qquad (21)$$

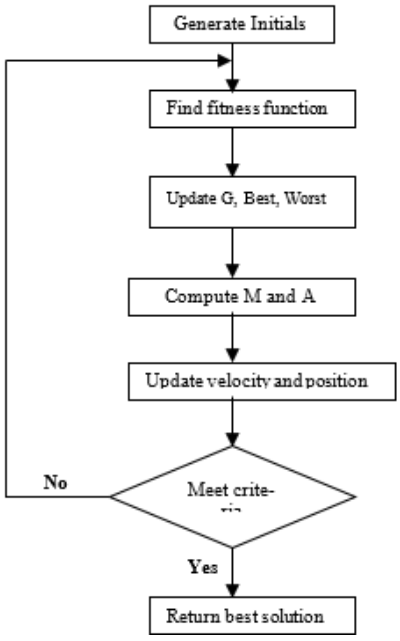

Fig 1: Steps for Gravitational Search Algorithm

The next velocity of an agent is a function of its current velocity plus current acceleration, which updates next position given by X as thus:

$$Vid(t+1) = rand(i) * Vid(t) + Aid(t) \quad (22)$$

$$Xid(t+1) = Xid * Vid(t+1) \quad (23)$$

$V_i^d(t)$ is agent velocity in *d* at time t, rand is a random number between [0,1]. Mass is updated as fitness value of agent i at time t given as:

$$Mi(t) = \frac{Fit(i) - worst(t)}{best(t) - worst(t)} \quad (24)$$

Best(t)/Worst(t) are strongest/weakest agents from their fitness route. For a maximization task as the one at hand, they are defined:

$$worst(t) = \max_{j \in \{1,2 \ldots N\}} Fit(t) \quad (25)$$

$$best(t) = \min_{j \in \{1,2 \ldots N\}} Fit(t) \quad (26)$$

At start, agents are located as solution points trained in ANN, and then passed over to GSA so that with each cycle, agent velocity and position is updated via Eq. 22 and 23; while G and M are computed via Eq. 19 and 24. The model stops if an

optima is found or stops using its stop criterion (this is computational expensive). GSA uses exploration ability to navigate and guarantee its choice value for random agents, and exploitation ability to allow agents of heavier masses move slower in order to attract those of lesser mass as well as locate optima, around a good solution in the shortest time possible (Rashedi et al, 2009b and Ojugo, 2012a).

3.3 ANN-Cultural Genetic Algorithm

Perez and Marwala (2011) GA as inspired by Darwinian evolution and genetics (survival of fittest), consists of a population (data) chosen for natural selection with potential solutions to a specific task. Individuals with genes close to its optimal, is said to be fit. Fitness function determines how close an individual is to the optimal solution. Each solution is an individual whose optimal is found via 4-operators namely:

a. Initialize – From the population, individual data are encoded into format suitable for selection. Each encodings has its merit/demerit. Binary encoding is computationally more expensive to achieve. Decimal encoding has greater diversity in chromosome and greater variance of pools generated; float-point encoding or its combination is more efficient than binary. Thus, it encode as fixed length vectors for one or more pools of different types. The *fitness* function evaluates how close a solution is to its optimal – after which they are chosen for reproduction. If solution is found, function is *good*; else, is *bad* and not selected for crossover. The fitness function is the only part with knowledge of task. If more solutions are found, the higher its fitness value.

b. Selection – Good fit individuals close to optimal are chosen to mate. The larger the number of selected, the better the chances of yielding fitter individuals. This continues until one is chosen, from the last two/three

remaining solutions, to become selected parents to new offspring. Selection ensures the fittest individuals are chosen for mating but also allows for less fit individuals from the pool and the fittest to be selected. A selection that only mates the fittest is *elitist* and often leads to converging at a local optima.

c. Crossover – ensures genes of fitter individuals are exchanged to yield a new, fitter pool. There are two crossover types (depends on encoding type used) as: (a) simple crossover for binary encoded pool via particular- or multi- point; and all genes are from one parent, and (b) arithmetic crossover allows new pool to be created by adding an individual's percentage to another.

d. Mutation alters chromosomes by changing its genes or its sequence, to ensure that a new pool converges to global minima (instead of local optima). Algorithm stops if optimal is found or after number of runs (though computationally expensive) if a number of new pools are created or once no better solution is found. Genes may change based on probability of mutation rate. Mutation improves the much needed diversity in reproduction and its algorithm is as thus:

Mutation Algorithm {}

1. Input: A chromosome rule
2. Output: Mutated solution, a fns of mutation rate
3. Set mutation threshold (between 0 and 1)
4. For each network attribute in chromosome
5. Generate a random number between 0 and 1
6. If random number > mutation threshold then
7. Generate Random value
8. Set solution attribute value with
9. Generated attribute value
10. End if: End For Each

Reynolds (1994) Cultural GA as a variant of GA consists of belief spaces namely: (a) Normative (has specific range values which an individual is bound), (b) Domain (data about task domain), (c) Temporal (data about events' space is available), and (d) Spatial (has topographical data). In addition, an influence function mediates between belief space and the pool – alter individuals to ensure the pool conforms to the belief space. CGA is chosen to yield a pool that does not violate its belief space and reduces number of possible individuals generated as optimum is found (Heppner and Grenander, 1990).

Once initialized, ANN computes individual fitness and 30-individual are selected as the new sub pool via tournament, to determine mating individuals. Thus, after training, selected data are moved to the CGA model – with only crossover and mutation applied, to help the network model learn dynamic and non-linear feats in the historic, obtained data.

With GA, only crossover (single point) and mutation is carried out – and data between 1 and 30 is randomly generated via Gaussian distribution, corresponding to crossover points (since prior now, all genes are from a parent). Now, other parents contribute the rest to yield new individuals, whose genetic makeup is combination of both parents. They are then allowed to undergo mutation from which 3-random genes are selected for another mutation and are allocated new random values that still conforms to the belief space. The number of mutation applied depends on how far CGA is progressed (how fit is the fittest individual in the pool). Thus, number of mutations equals fitness of the fittest individual divided by 2. New individuals replace old ones in pool, with low fitness values (creating a new pool). This continues until individual with a fitness value of 0 is found – indicating that the solution has been reached (Ursem et al, 2002).

Note that initialization and selection via ANN ensures the first 3-beliefs are met; while mutation ensures the fourth is met. Also, an influence function helps influence how many mutations takes place. Knowledge of solution (how close task is to solution) has direct impact on how algorithm is processed. Algorithm stops when best individual has a fitness of 0.

3.4 Model Performance Evaluation

Model performance is evaluated via computed values of mean square error (MSE), mean absolute error (MAE) and mean relative error (MRE), coefficient of efficiency (COE) and coefficient of determination (r^2) (Nash and Sutcliffe, 1970):

$$MSE = {}^{1}/_{n} \sum_{i=1}^{m} \{(Ypi - Yio)2\}^{1/2} \qquad (27)$$

$$MAE = {}^{1}/_{n} \sum_{i=1}^{m} |Ypi - Yio| \qquad (28)$$

$$MRE = {}^{1}/_{n} \sum_{i=1}^{m} \frac{|Ypi - Yio|}{Yio} \qquad (29)$$

$$COE = 1 - \frac{(Q_{obs} - Q_{sim})^2}{(Q_{sim} - Q_{obs})^2} \qquad (30)$$

$$r^2 = \frac{[(Q_{obs} - (1 - Q_{obs}))(Q_{obs} - (1 - Q_{obs}))]^2}{(Q_{obs} - (1 - Q_{obs}))^2 (Q_{sim} - (1 - Q_{sim}))^2} \qquad (31)$$

4. RESULT PRESENTATION / DISCUSSION

Performance analysis is as presented below:

Table 1. Model Performance Evaluation

Model	*MSE*	*MRE*	*MAE*	*COE (R)*	*COD (r^2)*
DFP	0.67	0.91	0.78	0.672	0.650
ANN	0.87	0.79	0.75	0.781	0.966
ANNGSA	0.76	0.81	0.62	0.753	0.921
ANNCGA	0.76	0.77	0.76	0.688	0.812

The result of the models simulation is as thus:

Table 2. Optmization using stachastic methods and DFP

Optimization Method	High	Ave.	Low
DFP	**0.72**	**0.70**	**0.72**
ANN (TLRN)	**0.67**	**0.60**	**0.50**
ANNGSA	**0.56**	**0.49**	**0.45**
ANNCGA	0.91	0.89	0.72

Table 3. Optimization with DFP as pre-processor

Optimization Method	High	Ave.	Low
ANN (TLRN)	**0.87**	**0.81**	**0.50**
ANNGSA	**0.61**	**0.54**	**0.54**
ANNCGA	0.91	0.89	0.72

The results are as thus:

a. ANNCGA took 21 seconds to find the solution after 98 iterations (at best). ANNCGA was run 15 times (to eradicate non-biasness), it found optima each time – and the time varied significantly between 21 seconds and 4 minutes – as its convergence time depends on how close the initial population is to the solution as well as on mutation applied to the individuals in the pool.

b. ANNGSA (at best) 18seconds after 321 iterations. GSA used a gravitational pull and mass update of 282 iterations before finding a solution. It was run 25 times and solution was found each time on a range between 4seconds and 3minutes. Its convergence time depends on initialization, gravitational cum mass updates.

c. ANN arrived at solution 32seconds at best after 401iterations. On 25 runs, solution time is between 42seconds and 8minutes – and its convergence depends on it weight and bias choices.

With a common solution space, fitness function and selection criteria, weights set between 0.2/0.35, and biases = 0.75 yields a better and faster convergence, before the solutions are crossed over to CGA, GSA – as supported by Perez and Marwala (2011), Mandic and Chambers (2001), Meade and Fernandez (1994) and Ojugo et al (2013a, b). Other values led to slower and/or non-convergence. The number of nodes in hidden layer greatly influences a network's performance. If it is small, network may not achieve its accuracy. If it is too many, may result in overtraining.

5. STUDY RATIONALE

Models are educational tools to help compile existing knowledge about a task. They serve as language to communicate hypotheses, help us investigate input parameters crucial in estimation and help us gain better insight of a task. Thus, model development, sensitivity and failure analysis helps reflect on theories and functioning of nature systems. The rationale for the implementation of hybrids are:

a. Artificial Neural Networks: ANN's major feat is its ability to learn to approximate a function – making it a flexible approximator; while SA is an intelligent search heuristics for global optimum. The adaptive property of ANN is a huge merit in modeling dynamic, changing states.

b. Genetic algorithms: GA was not originally developed for optimization; But, it offers statistical guarantee of global convergence to an optimal point. It is best suited for some problems. It start with an initial random population, *and allocate increasing trials to regions of the search space found to have high fitness*. This is a disadvantage if the maximum is in a small region, surrounded on all sides by regions of low fitness. This kind of function is difficult to optimize by any method, and here the simplicity of the iterated search usually wins.

c. DFP Optimize: Many well-behaved continuous functions have been developed which rely on using data about the gradient of the function to guide the direction of search. If the derivative of the function cannot be computed, because it is discontinuous, for example, these methods often fail. Such methods are generally referred to as *hill-climbing*. They can perform well on functions with only one peak (*unimodal* functions). But on functions with many peaks, (multimodal functions), they suffer from the problem that the first peak found will be climbed, and this may not be the highest peak. Having reached the top of a local maximum, no further progress can be made.

6. CONCLUSION/ RECOMMENDATIONS

Model's application as an intellectual or educational tool requires less accurate numerical agreement between predictions and observations. Rather, it requires a feedback mechanism as more important. Models that are understandable and manageable can be fully explored to help us better comprehend the process at hand. We thus, sought a balance between complexity and simplicity (as crucial in studying the relevant processes of a models and better yet, to understand the workings of the model). A detailed model may not be operationally applicable in larger scale, but its study can help to develop an applicable ones. Very simple models may not give enough new data whereas complex ones are not easily understandable. Nonetheless, models support experts in making estimates about a task.

The study recommends that though the techniques and models are time-consuming, DFP/BGFS should be adopted and adapted to act as preprocessor that quickly provide a good initial solution for models aiming to resolve quadratic equations as it will in turn, reduce considerably the time taken as well as improve the quality of the outputted result.

7. REFERENCES

[1] Armijo, S., (1966).*Minimization function having Lipschitz continuous partial derivatives*, Pacific Journal of Mathematics, 16, 1-3.

[2] Barzilar, J and Borwein, J.W.,(1988).*Two point step size gradient methods*, IMA Journal of Numerical Analysis, 141-148.

[3] Batruni, R.,(1991). *Multilayer network with piecewise-linear structure and BP-learning*, IEEE Transaction on Neural Networks, 2, 395–403.

[4] Broyden, C.G.,(1967). *Quasi newton method and their application to function approximation*, Mathematical

Computation, 21, 368-381.

[5] Cauchy, A.,(1987). *Methode generale pour la resolution de systems d'equations simultanees*, Computer Rendition of Science Paris, 25, 46-89.

[6] Caudill M.,(1987). *Neural Networks Primer, Part I*, AI Expert December, 46-52.

[7] Conway, A. J., Macpherson, K and Brown, J. C., (1998). *Delayed time series predictions with neural networks*, Journal of Neurocomputing, 18, 81–89.

[8] Dai, Y.H.,(2001). *Alternate step gradient method*, Report AMSS-2001-041, Academy of Mathematics and Systems Science, 32-56.

[9] Dai, Y.H and Yuan, Y.,(2003). *Alternate minimization gradient method*, IMA J. of Numerical Analysis, 23, 377

[10] Dai, Y.H and Yuan, Y.,(1999). *A nonlinear conjugate gradient method with strong global convergence property*, SIAM Journal of Optimization, 10, 177-182.

[11] Dai, Y.H., Yuan, J.Y and Yuan, Y.,(2002). *Modified two-point stepsize gradient methods for unconstrained optimization*, Computational Optimization and Application, 22, 103-109.

[12] Daniel, J.W.(1967). *Conjugate gradient method for linear/nonlinear operator equations*, SIAM J. Numeric Analysis, 4, 10 – 26.

[13] Dennis, J.E and Moore, J.J.,(1974). *A characterization of superlinear convergence and its application to Quasi Newton methods*, Mathematical Computation, 28, 549.

[14] Dennis, J.E and Moore, J.J.,(1977). *Quasi Newton method: motivation and theory*, SIAM Rev, 19, 46-89.

[15] Denton, J.W and Hung, M.S.,(1996). *A comparison of nonlinear optimization methods for supervised learning in multilayer feedforward networks*, European J. of Operation Research, 93, 358-368.

[16] Dixon, L.C.W.,(1972). *Variable metric algorithms necessary and sufficient conditions for identical behaviour on non-quadratic functions*, J. of Optimization Theory and Application, 10, 34-40.

[17] Fletcher, R.,(1987) *Practical methods of optimization*, John Wiley and Sons, Chichester.

[18] Fletcher, R and Reeves, C.,(1964). *Function minimization by conjugate gradients*, Computational Journal, 7, 149

[19] Forsythe, G.E.,(1986). *On asymptotic directions of s-dimension optimum gradient method*, Numerische Mathematik, 11, 57-76.

[20] Friedlander, A., Martinez, J.M., Molina, B and Raydan, M.,(1999). *Gradient method with retards and generalization*, SIAM J. of Numerical analysis, 36, 275-289.

[21] Ghalambaz, M., Noghrehabadi, A.R., Behrang, M.A., Assareh, E., Ghanbarzadeh, A and Hedayat, N.,(2011). *A hybrid gravitational search neural network method to solve well known Wessinger's equation*, World Academy of Science, Engineering and Technology, 49, 803.

[22] Gottlieb, D and Orszag, S.A.,(1977). *Numerical analysis of spectral methods: theory and applications*, CBMS-NSF Regional Conference Series in Applied Mathematics, 26.

[23] Griewank, A and Toini, P.H.,(1982). *Local convergence analysis of partitioned Quasi Newton* updates, Numerical Mathematics, 39, 429-448.

[24] Hager, W and Zhang, H.,(2003). *A new conjugate gradient method wit guaranteed descent and an efficient line search*, SIAM J. of Optimization, 305-333.

[25] Heppner, H and Grenander, U.,(1990). *A stochastic non-linear model for coordinated bird flocks"*, In Krasner, S (Ed.), *The ubiquity of chaos* (233–238). Washington: AAAS.

[26] Jang, J.S.,(1993). *Adaptive fuzzy inference systems*. IEEE Transactions on Systems, Man and Cybernetics, 23, 665–685.

[27] Khan, J., Zahoor, R and Qureshi, I.R,(2009). *Swarm intelligence for problem of non-linear ordinary differential equations and its application to well known Wessinger's equation*. European J. Sci. Research, 34(4), 514-525.

[28] Lagris, I.E., Likas, A and Fotiadis, D.I.,(1998). *Artificial neural networks for solving ordinary and partial differential equation*, IEEE Trans. Neural Network, 9(5), 987

[29] Lee, H and Kang, I.S.,(1990). *Neural algorithms for solving differential equations*, Journal of Computational Physics, 91, 110–131.

[30] Liu, Y and Storey, C.,(1991). *Efficient generalized conjugate gradient algorithms*, J. of Optimization Theory Application, 69, 129-137.

[31] Malek, A and Beidokhti, R.S.,(2006). *Numerical solution for high order differential equations using hybrid neural network - Optimization method*, Applied Mathematics and Computation, 183, 260-271.

[32] Mandic, D. and Chambers, J.,(2001). *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*, John Wiley andSons: New York.

[33] Meade, A.J and Fernandez, A.A.,(1994). *The numerical solution of linear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modeling, 19(12), 1–25.

[34] Nash, J and Sutcliffe, J., (1970). *River flow forecasting with conceptual models*, J. of Hydro. Sci., 10, 282–290.

[35] Ojugo, A.A, (2012a).*Artificial neural networks gravitational search model for rainfall runoff simulation and modeling*, unpublished PhD, Computer Sci. Department, Ebonyi State University Abakiliki, Nigeria.

[36] Ojugo, A.A., Eboka, A.O., Okonta, E.O., Yoro, R.E and Aghware, F.O.,(2012b). *Genetic algorithm rule-based intrusion detection system*, Journal of Emerging Trends in Computing and Information Systems, 3(8), 1182 - 1194.

[37] Ojugo, A., and Yoro, R.E., (2013a). *Computational intelligence in stochastic solution for Toroidal Queen*, Progress in Intel Comp. App., 2(1), 46–56.

[38] Ojugo, A.A., Emudianughe, J., Yoro, R.E., Okonta, E.O and Eboka, A..,(2013b). *Hybrid artificial neural network gravitational search algorithm for rainfall runoff modeling in Hydrology*, Progress Intel. Comp. App., 2(1), 22

[39] Perez, M and Marwala, T.,(2011). *Stochastic optimization approaches for solving Sudoku*, IEEE Transaction on Evolutionary Computation, 256–279.

[40] Pham, D and Karaboga, D.,(1999). *Training Elman and Jordan networks for system identification using GA*, Artificial Intelligence in Engineering, 13, 107–117.

[41] Pham, D., Koc, E., Ghanbarzadeh, A and Otri, S.,(2006). *Optimization of weights of multi-layered perceptrons using bee algorithm*. Proc. of Intelligent Manuf. Systems. Sakarya University, Dept of Industrial Engineering, 38.

[42] Pham, D.T and Liu, X.,(1995). *Artificial Neural Networks for Identification, Prediction and Control*, Springer Verlag, London.

[43] Plumb, A.P., Rowe, R.C., York, P and Brown, M.,(2005). *Optimization of the predictive ability of artificial neural network models*, European J. of Pharmaceutical Sciences, 25, 395-405.

[44] Polak, E and Ribiere, G.,(1969). *Note sur la convergence de directions conjugees*, Rev. Francaise Informat Researche Operationelle, 3(16), 35-43.

[45] Polyak, B.T.,(1969). *The conjugate gradient method in extreme problems*, USSR Computation Mathematics: Mathematic Physics, 9, 94-112.

[46] Powell, M.J.,(1971). *On the convergence of the variable algorithm D*, Winston Mathematics Application, 21.

[47] Powell, M.J.,(1976). *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, In Cottle, R.W and Lemke, C.E (eds.), Nonlinear programming, SIAM Proceedings of AMS, 9, 53-72.

[48] Rashedi, E., Nezamabadi-pour, H and Saryazdi, S.,(2009). *GSA: A Gravitational Search Algorithm*, Information Sciences, 179, 2232–2248.

[49] Rashedi, E., Nezamabadi-pour, H and Saryazdi, S.,(2009). *Filter modeling using gravitational search algorithm*, Energy policy; doi:10.1016/j.engappai.2010.05.007.

[50] Raydan, M.,(1993). *On Barzilai and Borwein choice of stepsize for the gradient method*, IMA Journal Numeric Analysis, 13, 321-326.

[51] Reynolds, R.,(1994). *An introduction to cultural algorithms*, IEEE Transaction on Evolutionary Programming, 131-139.

[52] Ritter, K.,(1979). *Local and superlinear convergence of a class of variable method*, Computing, 23, 287-297

[53] Stachurski, A.,(1981). *Superlinear convergence of a class of variable method*, Mathematical Programming, 14, 178-205.

[54] Ursem, R., Krink, T., Jensen, M.and Michalewicz, Z.,(2002). *Analysis and modeling of controls in dynamic systems*. IEEE Transaction on Evolutionary Computing, 6(4), 378-389