Applied Mathematics, 2010, 1, 288-292
doi:10.4236/am.2010.14037 Published Online October 2010 (http://www.SciRP.org/journal/am)

Artificial Neural Networks Approach for Solving Stokes Problem

Modjtaba Baymani, Asghar Kerayechian, Sohrab Effati
Department of Mathematics, Ferdowsi University of Mashhad, Mashhad, Iran
E-mail: *mhbaymani@yahoo.com, krachian@gmail.com, s-effati@um.ac.ir*
Received July 10, 2010; *revised August* 11, 2010; *accepted August* 14, 2010

Abstract

In this paper a new method based on neural network has been developed for obtaining the solution of the Stokes problem. We transform the mixed Stokes problem into three independent Poisson problems which by solving them the solution of the Stokes problem is obtained. The results obtained by this method, has been compared with the existing numerical method and with the exact solution of the problem. It can be observed that the current new approximation has higher accuracy. The number of model parameters required is less than conventional methods. The proposed new method is illustrated by an example.

Keywords: Artificial Neural Networks, Stokes Problem, Poisson Equation, Partial Differential Equations

1. Introduction

CFD stands for Computational Fluid Dynamics, a subgenre of fluid mechanics that uses computers (numerical methods and algorithms) to represent, or model, problems that engage fluid flows. CFD software is usually used to solve equations in a discretized way. The domain is transferred into a grid or mesh – a regular/irregular and 2D/3D surface of cells. After discretization, an equation solver runs to solve the equations of fluid motion (Euler equations, Navier-Stokes equations, etc.). Algorithms from numerical linear algebra, like: Gauss-Seidel, successive over relaxation, Krylov subspace method or algorithms from Multigrid family are typically used. These methods involve millions of calculations, so, as it can be easily observed, computing is time consuming. Also in many problems, even with parallel programming and supercomputers, only approximate solutions can be reached.

There are various optimization methods of computer science which can be used for CFD. Simulated Annealing, Genetic Algorithms, Evolution Strategy, Feed-Forward Neural Networks are popular and we reimplemented in many projects.

In [1] a framework is created for evolutionary optimization which is then tested on aerodynamic design example. The framework was based on covariance matrix adaptation, with the feed-forward neural network as an approximate fitness function. An aerodynamic design procedure which combines neural networks with polynomial fits is introduced in [2] and [3] discussed an artificial neural network which is an approximate model that is used for optimization of the blade geometry by simulated annealing method. Parallel stochastic search algorithm is introduced in [4] and tested on defining a shape of two airfoils. In this work, a performance neural network for solving Stokes equations is presented.

Lagaris, *et al.* [5] used artificial neural networks (ANN) for solving ordinary differential equations and partial differential equations for both boundary value and initial value problems. Canh and Cong [6] presented a new technique for numerical calculation of viscoelastic flow based on the combination of neural networks and Brownian dynamics simulation or stochastic simulation technique (SST). Hayati and Karami [7] used a modified neural network to solve the Berger's equation in one-dimensional quasilinear partial differential equation.

The Stokes equations describe the motion of a fluid in R^n $(n = 2 \text{ or } 3)$. These equations are to be solved for an unknown velocity vector $u(x, y) = (u_i(x, y))_{1\le i\le n} \in R^n$ and pressure $p(x, y) \in R$.

We restrict our attention here to incompressible fluids filling all of R^n $(n = 2)$ as follow:

$$\begin{cases} -\Delta u_1 + \dfrac{\partial p}{\partial x} = f_1 & in\ \Omega \subset R^2 \\ -\Delta u_2 + \dfrac{\partial p}{\partial y} = f_2 & in\ \Omega \\ \dfrac{\partial u_1}{\partial x} + \dfrac{\partial u_2}{\partial y} = 0 & in\ \Omega \end{cases} \quad (1)$$

with boundary conditions:

$$u = (u_1, u_2) = (u^0{}_1, u^0{}_2) = u^0, \text{ on } \partial\Omega .$$

Here, u^0 is given, C^∞ divergence-free vector field on Ω, f_1, f_2 are the components of a given, externally applied force (e.g. gravity). The first and second equations of (1) are just Newton's law $f = ma$ for fluid element subject to the external force $f = (f_1, f_2)$ and to the forces arising from pressure and friction. The third equation of (1) says that the fluid is incompressible. For physically reasonable solutions, we want to make sure $u = (u_1, u_2)$ does not grow as $|(x, y)| \to \infty$. Hence we will restrict our attention to the force f and initial condition u^0 that satisfy

$\left|\partial_x^\alpha u^0(x)\right| \le C_{\alpha K}\left(1+|x|\right)^{-K}$, on R^n, for any α and some K,

$\left|\partial_x^\alpha f(x)\right| \le C_{\alpha K}\left(1+|x|\right)^{-K}$, on R^n, for any α and some K.

We accept a solution of (1) as physically reasonable only if it satisfies $p,\ u \in C^\infty(R^n)$ and

$\int_\Omega |u(x)|\,dx < C$ (bounded energy).

2. Description of the Method

The usual proposed approach for problem (1) will be illustrated in terms of the following general partial differential equation:

$$\begin{aligned} &G_1\left(x, \psi_1(x), \nabla^2\psi_1(x), \frac{\partial\phi}{\partial x}(x)\right) = 0, \quad x \in D \\ &G_2\left(x, \psi_2(x), \nabla^2\psi_2(x), \frac{\partial\phi}{\partial x}(x)\right) = 0, \quad x \in D \\ &G_3\left(x, \frac{\partial\psi_1}{\partial x}(x), \frac{\partial\psi_2}{\partial y}(x)\right) = 0, \quad x \in D \end{aligned} \tag{2}$$

subject to certain boundary conditions (BC's) (for example Dirichlet and/or Neumann), where $x = (x_1, ..., x_n) \in R^n,\ D \subset R^n$ denotes the definition domain and $\psi_1(x), \psi_2(x), \phi(x)$ are the solutions to be computed.

If $\psi_{1t}(x, P_1),\ \psi_{2t}(x, P_2),\ \phi_t(x, P_3)$ denote trial solutions with adjustable parameters P_1, P_2, P_3, the problem (2) is transformed to

$$\min_{P_1, P_2, P_3} \sum_{x_i \in D}\left[\left(G_1(x_i)\right)^2 + \left(G_2(x_i)\right)^2 + \left(G_3(x_i)\right)^2\right] \tag{3}$$

subject to the constraints imposed by the BC's.

In the proposed approach, the trial solutions $\psi_{1t}(x, P_1)$, $\psi_{2t}(x, P_2),\ \phi_t(x, P_3)$ employ a feed forward neural network and parameters P_1, P_2, P_3 correspond to the weights and biases of the neural architecture. We choose trial functions $\psi_{1t}(x, P_1),\ \psi_{2t}(x, P_2),\ \phi_t(x, P_3)$ such that by construction satisfy the BC's. This achieves by writing it as a sum of two terms

$$\begin{aligned} \psi_{1t}(x) &= A_1(x) + F_1\left(x, N_1(x, P_1)\right) \\ \psi_{2t}(x) &= A_2(x) + F_2\left(x, N_2(x, P_2)\right) \\ \phi_t(x) &= A_3(x) + F_3\left(x, N_3(x, P_3)\right) \end{aligned} \tag{4}$$

where $N_k(x, P) = \sum_{i=1}^{H} v_i \sigma(z_i)$ and $z_i = \sum_{j=1}^{n} w_{ij} x_j + u_i$ $(k = 1,2,3)$ are single-outputs feed forward neural network with parameters P_1, P_2, P_3 and n input units fed with the input vector x. The terms $A_i(x)\ (i = 1,2,3)$ contain no adjustable parameters and satisfy the boundary conditions.

The second terms $F_i(x, N_i(x, P_i))(i = 1,2,3)$ is constructed so as not to contribute to the BC's, since $\psi_{1t}(x, P_1),\ \psi_{2t}(x, P_2),\ \phi_t(x, P_3)$ must also satisfy the BC's. These terms employ a neural network whose weights and biases are to be adjusted in order to deal with the minimization problem. Note at this point that the problem has been reduced from the original constrained optimization problem to an unconstrained one (which is much easier to handle) due to the choice of the form of the trial solution that satisfies by construction the BC's. In the next section we present a systematic way to construct the trial solution, *i.e.*, the functional forms of both A_i and F_i. We treat several common cases that one frequently encounters in various scientific fields. As indicated by our experiments, the approach based on the above formulation is very effective and provides in reasonable computing time accurate solutions with impressive generalization (interpolation) properties.

3. Neural Network for Solving Stokes Equations

To solve problem (1) with $\Omega = [0,1] \times [0,1]$, we apply the operators $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ on the first and second equations respectively. Then we obtain:

$$-\Delta\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y}\right) + \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = (f_1)_x + (f_2)_y \tag{5}$$

Using the third equation in (1), the Equation (5) may be written as:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = (f_1)_x + (f_2)_y \tag{6}$$

this is the Poisson equation, and has infinitely many solutions. By imposing some boundary conditions, we are going to obtain an appropriate solution for Equation (6) by the neural network.

The trial solution is written as

$$p_t(x,y) = A(x,y) + x(1-x)y(1-y)N(x,y,P) \quad (7)$$

where $A(x,y)$ is chosen so as to satisfy the BC, namely

$$\begin{aligned} &A(x,y) = (1-x)h_0(y) + xh_1(y) \\ &+(1-y)\{g_0(x) - [(1-x)g_0(0) + xg_0(1)]\} \\ &+y\{g_1(x) - [(1-x)g_1(0) + xg_1(1)]\}. \end{aligned} \quad (8)$$

where

$h_0(y) = p(0,y)$, $h_1(y) = p(1,y)$, $g_0(x) = p(x,0)$ and $g_1(x) = p(x,1)$.

Note that the second term of the trial solution does not affect the boundary conditions since it vanishes at the part of the boundary where Dirichlet BC's are imposed. In the above PDE problems the error to be minimized is given by

$$E(p) = \sum_i \left\{ \frac{\partial^2 p_t(x_i,y_i)}{\partial x^2} + \frac{\partial^2 p_t(x_i,y_i)}{\partial y^2} - F(x_i,y_i) \right\}^2 \quad (9)$$

where $F = (f_1)_x + (f_2)_y$ and (x_i, y_i) is a point in Ω.

By solving the optimization problem (9), the weights v_i, w_{ij}, u_i are obtained and then a trial solution for p_t is obtained. By substituting the trial solution p_t in the first equation of (1) we obtain:

$$\Delta u_1 = \frac{\partial(p_t)}{\partial x} - f_1 \quad (10)$$

which is a Poisson equation for u_1, by using (9) and by substituting u_{1t} and $\frac{\partial(p_t)}{\partial x} - f_1$ for or p and F, respectively, we can obtain a trial solution for u_{1t}. To obtain u_2, we can substitute the trial solution p_t in the second equation in (1) to obtain:

$$\Delta u_2 = \frac{\partial(p_t)}{\partial y} - f_2 \cdot \quad (11)$$

In a similar manner we can obtain u_{2t}.

4. Numerical Examples

In this section we present one example to illustrate the method. We used a multilayer perceptron having one hidden layer with five hidden units and one linear output unit. For a given input vector $x = (x_1, x_2)$ the output of the network is $N(x,P) = \sum_{i=1}^{5} v_i \sigma(z_i)$ where $z_i = \sum_{j=1}^{2} w_{ij} x_j + u_i$ and $\sigma(x) = \frac{1}{1+e^{-x}}$. The exact analytic solution is known in advance. Therefore we test the accuracy of the obtained solution.

Example. Consider the problem (1) with Ω = [0,1] × [0,1]. We choose f_1 and f_2 such that the exact solution for u_1, u_2 and p be as follows:

$$\begin{aligned} u_1 &= 10x^2 y(1-x)^2(1-3y+2y^2) \\ u_2 &= -10y^2 x(1-y)^2(1-3x+2x^2) \\ p &= 5(x^2 - y^2). \end{aligned}$$

The domain Ω is first discretized by uniform mesh of size $h = 1/3$ (4 points). This initial mesh is successively refined to produce meshes with sizes $h = 1/4$ and $h = 1/5$ (respectively 9 and 16 points). **Table 1** reports the maximum error at nodal points (Maximum error) at the training set points and the distances $\|p - p_t\|_{L_2}$ and $\|p - p_t\|_{H^{(1)}}$ between the exact solution and the training solution.

In **Figure 1** the error function $p - p_t$ for N = 25 is depicted which shows the solution is very accurate.

We used the Equation (10) and obtained the solution u_{1t}. **Table 2** reports the maximum error at the training set points and the distances $\|u_1 - u_{1t}\|_2$ and $\|u_1 - u_{1t}\|_{H^{(1)}}$ between the exact solution and the training solution.

In a similar manner we obtained u_{2t}. **Table 3** reports the maximum error at the training set points and the dis-

Table 1. Maximum error at training set points and the distances $\|p - p_t\|_2$ and $\|p - p_t\|_{H^{(1)}}$.

	N = 4	N = 9	N = 16
Maximum Error	0.0058	5.1007e-4	1.8433e-7
$\|p - p_t\|_2$	1.0025e-5	7.5171e-8	4.1647e-15
$\|p - p_t\|_{H^{(1)}}$	4.1569e-4	7.5120e-6	4.9101e-13

Table 2. Maximum error at training set points and the distances $\|u_1 - u_{1t}\|_2$ and $\|u_1 - u_{1t}\|_{H^{(1)}}$.

	N = 9	N = 16	N = 25
Maximum error	0.0372	2.0723e-8	1.733e-8
$\|u_1 - u_{1t}\|_2$	4.091e-4	4.487e-17	2.576e-17
$\|u_1 - u_{1t}\|_{H(1)}$	0.040	6.4928e-15	1.632e-15

Table 3. Maximum error at training set points and the distances $\|u_2 - u_{2t}\|_2$ and $\|u_2 - u_{2t}\|_{H^{(1)}}$.

	N = 9	N = 16	N = 25
Maximum error	0.0379	2.6141e-05	2.6507e-07
$\|u_2 - u_{2t}\|_2$	4.0684e-04	1.2722e-010	1.3238e-14
$\|u_2 - u_{2t}\|_{H^{(1)}}$	0.0407	1.8825e-008	2.7010e-12

tances $\|u_2 - u_{2t}\|_2$ and $\|u_2 - u_{2t}\|_{H^{(1)}}$ between the exact solution and the training solution.

In **Figures 2** and **3** the error functions $u_1 - u_{1t}$ and $u_2 - u_{2t}$ for N = 25 is depicted which shows the solutions are very accurate (even between training points).

In **Figures 4-7** differential of the error functions $u_1 - u_{1t}$ and $u_2 - u_{2t}$ respect to x and y, for N = 25 are depicted, respectively, which show the solutions are differentiable.

Aman and Kerayechian [8] used linear programming-methods to solve the above problem. They converted the Stokes problem to a minimization problem, then by discretizing the minimization problem. They obtained a linear programming problem and solved it. **Table 4** present the comparison between the proposed method and with Aman – Kerayechian method for 25 points.

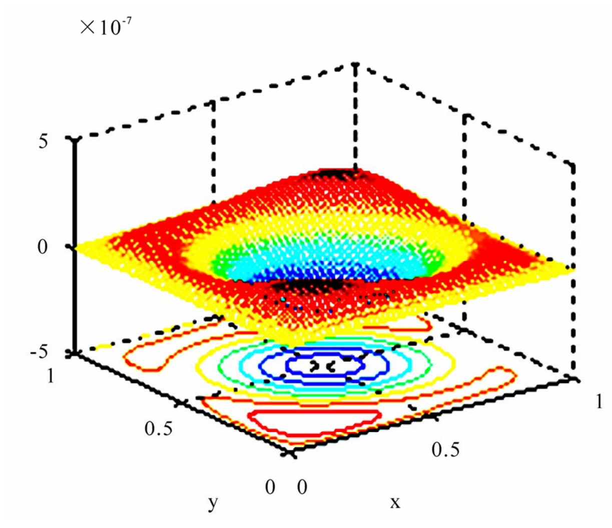

Figure 1. Error function $p - p_t$.

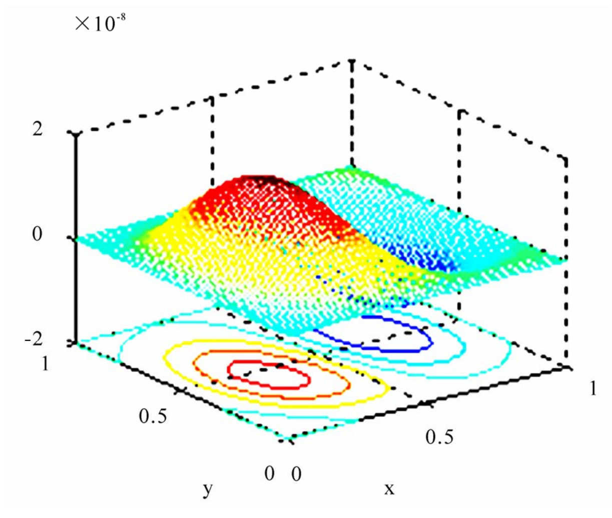

Figure 2. The error function $u_1 - u_{1t}$.

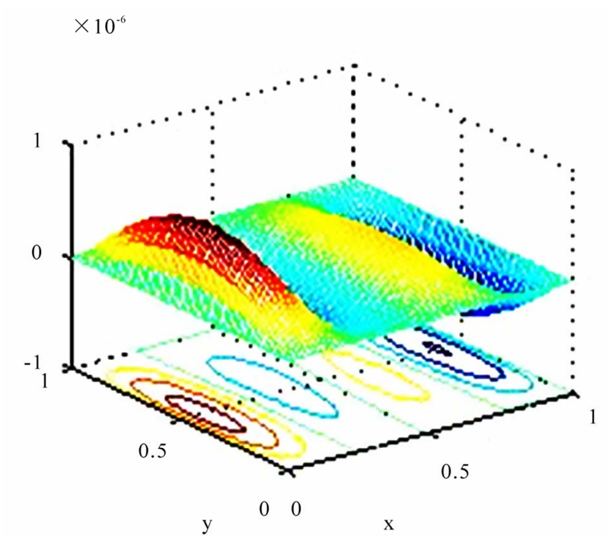

Figure 3. The error function $u_2 - u_{2t}$.

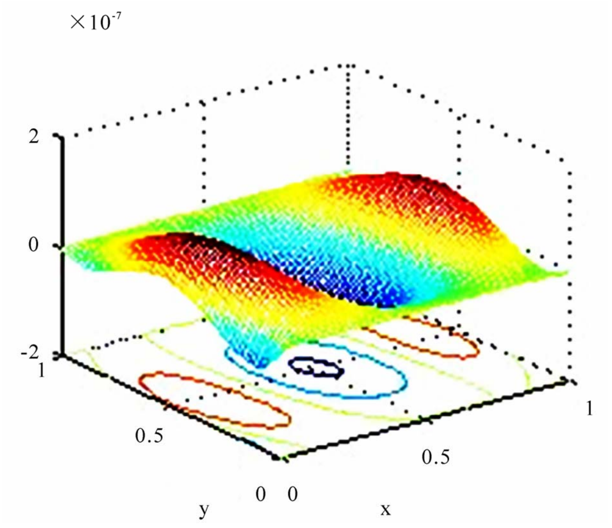

Figure 4. The differential of error function $u_1 - u_{1t}$ respect to x.

Table 4. The comparison of our proposed method with Aman – Kerayechian method.

Errors	The proposed method	Aman-Kerayechian method
Maximum error $u_1 - u_{1t}$	1.7335e-008	0.007885
$\|u_1 - u_{1t}\|_{H^{(1)}}$	1.6328e-015	0.082281
Maximum error $u_2 - u_{2t}$	2.6507e-007	0.007885
$\|u_2 - u_{2t}\|_{H^{(1)}}$	2.7010e-012	0.082281
Maximum error $p - p_t$	1.6392e-005	0.035104
$\|p - p_t\|_2$	2.2448e-011	0.027446

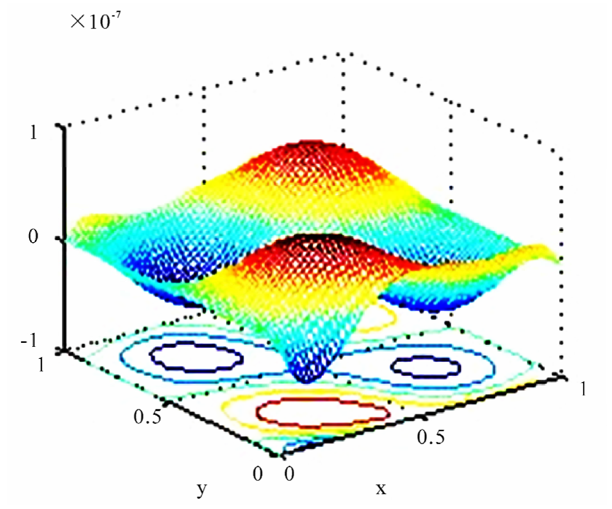

Figure 5. The differential of error function $u_1 - u_{1t}$ respect to y.

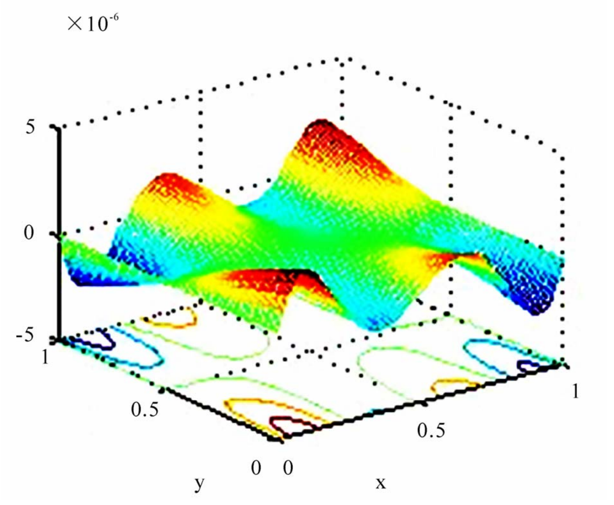

Figure 7. The differential of error function $u_2 - u_{2t}$ respect to y.

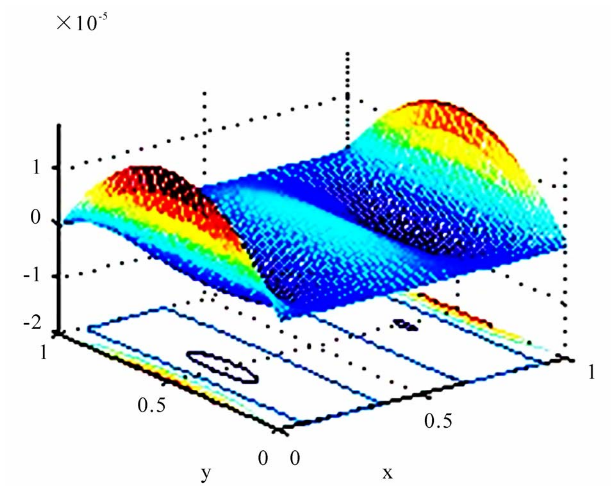

Figure 6. The differential of error function $u_2 - u_{2t}$ respect to x.

5. Conclusions

In this paper a new method based on ANN has been applied to find solution for Stokes equation. The solution via ANN method is a differentiable, closed analytic form easily used in any subsequent calculation. The neural network here allows us to obtain fast solution of Stokes equation starting from randomly sampled data sets and refined it without wasting memory space and therefore reducing the complexity of the problem.

If we compare the results of the numerical methods (see [8]) with our method, we see that our method has some small error. Other advantage of this method, the solution of the Stokes problem is available for each arbitrary point in training interval (even between training points). Indeed, after solving the Stokes problem, we obtain an approximated function for the solution and so we can calculate the answer at every point immediately.

6. References

[1] Y. C. Jin, M. Olhofer and B. Sendhoff, "A Framework for Evolutionary Optimization with Approximate Fitness Functions, Evolutionary Computation," *IEEE Transactions*, Vol. 6, No. 5, 2002, pp. 481-494.

[2] M. H. Man, R. Nateri and K. Madavan, "Aerodynamic Design Using Neural Networks," *American Institute of Aeronautics and Astronautics Journal*, Vol. 38, No. 1, 2000, pp. 173-182.

[3] S. Pierret, "Turbo Machinery Blade Design Using a Navier-Stokes Solver and Artificial Neural Network," *ASME Journal of Turbomach*, Vol. 121, No. 3, 1999, pp. 326-332.

[4] T. Ray, H. Tsai and C. Tan, "Effects of Solver Fidelity on a Parallel Search Algorithm's Performance for Airfoil Shape Optimization Problems," *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia, 2002, pp. 1816-1826.

[5] I. E. Largris and A. Likas, "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations," *IEEE transaction on neural networks*, Vol. 9, No. 5, 1998, pp. 987-1000.

[6] D. Tran-Canh and T. Tran-Cong, "Computation of Viscoelastic Flow Using Neural Networks and Stochastic Simulation," *Korea-Australia Rheology Journal*, Vol. 14, No. 4, 2002, pp. 161-174.

[7] M. Hayati and B. Karami, "Feedforward Neural Network for Solving Partial Differential Equations," *Journal of Applied Sciences*, Vol. 7, No. 19, 2007, pp 2812-2817.

[8] M. Aman and A. Kerayechian, "Solving the Stokes Problem by Linear Programming Methods," *International Mathematical Journal*, Vol. 5, No. 1, 2004, pp. 9-22.