Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 605625, 6 pages
http://dx.doi.org/10.1155/2014/605625

Research Article

A Multiagent Transfer Function Neuroapproach to Solve Fuzzy Riccati Differential Equations

Mohammad Shazri Shahrir,[1,2] N. Kumaresan,[2] Kuru Ratnavelu,[2] and M. Z. M. Kamali[3]

[1] *Telekom Research & Development Sdn Bhd, 63000 Cyberjaya, Selangor, Malaysia*
[2] *Institut Sains Matematik (ISM), Universiti Malaya, Kuala Lumpur, Malaysia*
[3] *Pusat Asasi Sains, Universiti Malaya, Kuala Lumpur, Malaysia*

Correspondence should be addressed to N. Kumaresan; drnk2008@gmail.com

Received 18 February 2014; Revised 25 April 2014; Accepted 25 April 2014; Published 21 May 2014

Academic Editor: Weichao Sun

A numerical solution of fuzzy quadratic Riccati differential equation is estimated using a proposed new approach for neural networks (NN). This proposed new approach provides different degrees of polynomial subspaces for each of the transfer function. This multitude of transfer functions creates unique "agents" in the structure of the NN. Hence it is named as multiagent neuroapproach (multiagent NN). Previous works have shown that results using Runge-Kutta 4th order (RK4) are reliable. The results can be achieved by solving the 1st order nonlinear differential equation (ODE) that is found commonly in Riccati differential equation. Multiagent NN shows promising results with the advantage of continuous estimation and improved accuracy that can be produced over Mabood et al. (2013), RK-4, and the existing neuromethod (NM). Numerical examples are discussed to illustrate the proposed method.

1. Introduction

In optimal control theory, solving the Riccati differential equation for state space representation of a dynamical system is a central issue. Chen et al. [1] have shown that the stochastic LQR problem is well posed if there are solutions to the Riccati equation and then optimal feedback control can be obtained. In this paper, a new multiagent NN approach is proposed to solve nonlinear Riccati differential equation that is related to the LQR. Multiagent NN approach improved the results generated by Mabood et al. [2], RK-4, group method of data handling (GMDH), and neuromethod (NM).

Fundamental theories of Riccati equation can be applied to stochastic processes and diffusion problems [3], robust stabilization, network synthesis, and financial mathematics [4, 5]. Apart from the more traditional methods like RK-4 and forward Euler method, there are other nontraditional approaches for solving the aforementioned problems such as the unconditionally stable scheme by Dubois and Saïdi [6]. El-Tawil et al. [7] also have presented the usage of Adomian decomposition method (ADM) to solve the nonlinear Riccati differential equation in an analytical form. In this method, the ODE can be decomposed to a set of Adomian polynomials.

Tan and Abbasbandy [8] employed the analytic technique called homotopy analysis method (HAM) to solve a quadratic Riccati equation. This technique has been derived from perturbation theory. A modified variational iteration method was used to solve quadratic Riccati equation by Geng [9]. HAM method is the generalization of other methods. But it has some limitations to solve the differential equations.

Multiagent NN has its origins from group method of data handling (GMDH) neural networks. GMDH polynomial neural network was created by the Ukrainian scientist Aleksey Ivakhnenko. GMDH design and weights (parameters of polynomials) adjustment resembles the Kolmogorov-Gabor polynomial by using low order polynomials for every pair of the input values [10]. There are shortcomings of GMBH, firstly the inability to control contribution of subspaces to the final solution and secondly the choice of degree of polynomial sometimes that does not fit with the complexity

of the problem in hand. To overcome the limitations and shortcoming of the existing methods, the new multiagent NN approach is proposed to solve nonlinear fuzzy Riccati differential equation.

The general concept of solving ODE using neural networks is to fit the derivative model of the neural network (as given in (1)) for the given ODE:

$$\min_{w\in(-\infty,+\infty)} \frac{dy}{dx} - NN'(w), \tag{1}$$

where w represents the weights in the neural network.

One of the advantages of neural network is its nonparametric nature. Another advantage is its ability to estimate unseen or untrained points. Previous studies of the first order differential equation (FODE) and second order differential equation (SODE) solution via neural networks [10–12] have been promising. It has also been shown in [10] that neural network methods are more stable and accurate than Euler method, first order implicit method, and second order implicit method. There is no report on NN method [13–16] to solve nonlinear ODE.

Traditionally, the input is given in one form to the NN. In this paper, we propose a multiagent approach (multiagent NN) in which different forms of inputs are used and each transfer function is an agent. Each agent has different degrees of polynomial. Our paper will investigate and show the fact that the fuzzy Riccati differential equation can be solved for fuzzy control system using multiagent NN. In Section 2, the statement of the problem in hand and the corresponding Riccati differential equation are derived. In Section 3, a proposed new approach of multiagent transfer function neuromethod is defined. In Sections 4 and 5, the reader can find the results of fuzzy Riccati differential equation using multiagent NN and conclusion.

2. Statement of the Problem

Any first order ordinary differential equation that is quadratic in the unknown function is called the Riccati equation. It is usually written as

$$\dot{K}_t(t) = q_0(t) + q_1(t)K_i(t) + q_2(t)K_i^2(t). \tag{2}$$

Given the linear time-invariant fuzzy system that can be expressed in the form

$$R^i\text{: If } x_j \text{ is } T_{ji}\left(\mu_{ji}, \sigma_{ji}\right), \quad i = 1, 2, \ldots, r, \ j = 1, 2, \ldots, n, \tag{3}$$

then

$$\dot{x}(t) = A_i x(t) + B_i u(t), \quad x(0) = x_0, \tag{4}$$

where R^i indicates the ith rule of the fuzzy model, μ_{ji} and σ_{ji} are the mean and standard deviation of the Gaussian membership function, $x(t) \in \mathbb{R}^n$ is a generalized state space vector, $u(t) \in \mathbb{R}^m$ is a control variable, $A_i \in \mathbb{R}^{n\times n}$ and $B_i \in \mathbb{R}^{n\times m}$ are known as coefficient matrices associated with $x(t)$ and $u(t)$, respectively, x_0 is given initial state vector, and $m \le n$.

If all state variables are measurable, then a linear state feedback control law is given as

$$u(t) = -R^{-1}B_i^T\lambda(t). \tag{5}$$

Equation (5) can be obtained to the system from (4) and

$$\lambda(t) = K_i(t)x(t), \tag{6}$$

where $K_i(t) \in R^{n\times n}$ is a symmetric matrix such that $K_i(t_f) = S$.

To minimize both the state and control signals of the feedback control system, a quadratic performance index is minimized:

$$J = \frac{1}{2}x^T\left(t_f\right)Sx\left(t_f\right) + \frac{1}{2}\int_o^{t_f}\left[x^T(t)Qx(t) + u^T(t)Ru(t)\right]dt, \tag{7}$$

where T represents the transpose operator, $S \in \mathbb{R}^{n\times n}$ and $Q \in \mathbb{R}^{n\times n}$ are symmetric and positive definite (semidefinite), and $R \in \mathbb{R}^{m\times m}$ is a symmetric and positive definite weighting matrix for $u(t)$.

Based on standard procedure, J can be minimized by minimizing the Hamiltonian equation

$$H(x(t), u(t), \lambda(t)) = \frac{1}{2}x^T(t)Qx(t) + \frac{1}{2}u^T(t)Ru(t) + \lambda^T(t)\left[A_i(x) + B_i(t)\right]. \tag{8}$$

The necessary condition for optimality $(\partial H/\partial u)(x, u, \lambda, t) = 0$ implies that $Ru(t) + B_i^T\lambda(t) = 0$ and

$$\frac{\partial H}{\partial x} = \dot{\lambda}(t), \quad \longrightarrow \dot{\lambda}(t) = -Qx(t) - A_i^T\lambda(t), \tag{9}$$

$$\frac{\partial H}{\partial \lambda} = \dot{x}(t), \quad \longrightarrow \dot{x}(t) = A_i x(t) - B_i u(t), \tag{10}$$

and, from (5), we have

$$\dot{x}(t) = Ax(t) - B_i R^{-1}B_i^T\dot{\lambda}(t). \tag{11}$$

Equations (9) and (11) can be written in a matrix form as follows:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{\lambda}(t)\end{bmatrix} = \begin{bmatrix}A_i & -B_iR^{-1}B_i^T\\ -Q & -A_i^T\end{bmatrix}\begin{bmatrix}x(t)\\ \lambda(t)\end{bmatrix}, \tag{12}$$

where $x(0) = x_0$.

From (6), we have

$$\dot{\lambda}(t) = \dot{K}_i x(t) + K_i\dot{x}(t). \tag{13}$$

Substituting (9) and (11) into (13) will generate the following equation:

$$\begin{aligned}&\left[\dot{K}_i(t) + K_i(t)A_i + A_i^T K_i(t)\right.\\&\left.\quad + Q - K_i(t)B_i R^{-1}B_i^T K_i(t)\right]x(t) = 0.\end{aligned} \tag{14}$$

Since (14) holds for all nonzero $x(t)$, the term premultiplying $x(t)$ must be zero. Therefore, we obtain the following fuzzy Riccati differential equation for the fuzzy linear system (4):

$$\begin{aligned}&\left[\dot{K}_i(t) + K_i(t)A_i + A_i^T K_i(t) + Q - K_i(t)B_i R^{-1}B_i^T K_i(t)\right]\\&\quad = 0.\end{aligned} \tag{15}$$

Assume $m = n = 1$; then (15) becomes

$$\dot{K}_i(t) + 2A_i K_i(t) + Q - B_i^2 R K_i^2(t) = 0. \tag{16}$$

Fuzzy systems are control systems where the feedback loop is based on fuzzy logic. Fuzzy-feedback system mechanism is based on imprecise logic or "fuzzy logic." The following are the rules applied to the fuzzy system:

Fuzzy systems are control systems where the feedback loop is based on fuzzy logic. Fuzzy-feedback system mechanism is based on imprecise logic or "fuzzy logic." The following are the rules applied to the fuzzy system:

Fuzzy Rule 1:

$$\begin{aligned}&K = y, \quad Q = 1, \quad A_1 = 0, \quad B_1^2 R = 1:\\&\frac{dy}{dx} = -y^2(t) + 1, \quad y(0) = 0,\end{aligned} \tag{17}$$

Fuzzy Rule 2:

$$\begin{aligned}&K = y, \quad Q = 1, \quad A_2 = 1, \quad B_2^2 R = 1:\\&\frac{dy}{dt} = 2y(t) - y^2(t) + 1, \quad y(0) = 0.\end{aligned} \tag{18}$$

This paper investigates the performance of solving the aforementioned equations generated from Fuzzy Rules 1 and 2.

3. Neural Networks

By mapping the output of the neural network to the expected value and using the characteristics of NM (19), the numerical integration can be acquired [11]:

$$\frac{d\sigma(z_i)}{dx} = \sigma(z_i)\left(1 - \sigma(z_i)\right). \tag{19}$$

Multiagent NN differs from NM. In the present NM, only one form of input is used in the transfer functions whereas multiagent NN has given different form of inputs in the transfer functions.

Neural methods, being NM or multiagent NN, use neural networks to estimate solution for ODE. Traditionally neural network has shown performance capability as a *function estimator* [10] and *regressor* [10].

In the neuroapproach, to solve ODE [12], $\sigma(z_i) = 1/(1 + e^{-\alpha z_i})$ which is the transfer function in the neural network process. Part of NM process is to find the derivative of neural networks using (19). In our proposed multiagent NN, different agents of σ have been used in the transfer functions.

3.1. Multiagent NN Algorithm

Step 1. Feed the input vector t_j.

Step 2. Initialize the weight matrix w_{ij} and bias u_i.

Step 3. Compute $z_i = \sum_{j=1}^{n} w_{ij} t_j^p$, where p is the power of the input.

Step 4. Compute $\sigma_i(z_i) = z_i$.

Step 5. Initialize weight vector v_i hidden to output layer.

Step 6. Calculate $N_{ij} = \sum_{i=1}^{n} v_i \sigma(z_i)$.

Step 7. Compute purelin function of N_{ij}.

Step 8. Minimize the following fitness function:

$$E_r = \sum_{i,j=1}^{n}\left(\frac{d^2 y}{dx^2} - \emptyset\left(y, \frac{dy}{dx}\right)\right)^2, \tag{20}$$

where $y = A_i + t_j N_{ij}$ is the trial function for first order and r is the epoch. The proposed multiagent NN method converges when the error tends to zero as given in (21). The convergence of method usually takes around 10–15 mins on an i7 Processor, 4-Gigabyte RAM system. The method does not need to be recomputed for the untrained points within the bounds of the model or problem defined. Consider

$$\lim_{r \to \infty} E_r \longrightarrow 0. \tag{21}$$

The kth order derivative of neural networks is shown in

$$\frac{d^k N}{dx_j^k} = \sum_{i=1}^{H} v_i w_{ij}^k \sigma_i^k. \tag{22}$$

Example of derived fitness function of second order in Matlab (to solve $y'' = 2$) is as follows:

```
Error = 2.*(Weight1t2.*Weight2t3'*(SigOut1_')) +
((Input(:,cnt).*((Weight2t3'.*Weight1t2.*Weight1t2)
*SigOut1_'))) - 2.
```

The above fitness function can easily be derived using chain rule:

```
Error = (Out2_ + ((Input(:,cnt).*(Weight1t2*
(Weight2t3.*SigOut1_')))))) - TargetFunction.
```

The fitness function in (20) is simply defined by the derivatives of neural networks in the given differential equation.

Table 1: Solution of fuzzy Riccati differential equation by various methods for Fuzzy Rule 1.

x	RK-4	Mabood et al. [2]	NM	GMDH	Multiagent NN	Analytical
0	0	0	0	0	0	0
0.1	0.1	0.099668	0.095826	0.024865	0.099668	0.099668
0.2	0.199	0.197376	0.199513	0.055562	0.197375	0.197375
0.3	0.29504	0.291315	0.298841	0.093447	0.291313	0.291313
0.4	0.386335	0.379949	0.391187	0.139876	0.379949	0.379949
0.5	0.47141	0.462092	0.472797	0.19621	0.462121	0.462117
0.6	0.549187	0.53691	0.544233	0.26381	0.537077	0.53705
0.7	0.619026	0.603815	0.606699	0.344037	0.604513	0.604368
0.8	0.680707	0.662245	0.661873	0.438244	0.664640	0.664037
0.9	0.734371	0.711287	0.710854	0.547766	0.718390	0.716298
1	0.780441	0.749123	0.75499	0.673906	0.767899	0.761594
SAE	0.101748	0.019995	0.059152	1.936046	0.009178	

Table 2: Solution of fuzzy Riccati differential equation by various methods for Fuzzy Rule 2.

x	RK-4	Mabood et al. [2]	NM	GMDH	Multiagent NN	Analytical
0	0	0	0	0	0	0
0.1	0.1	0.110328	0.109895	0.062432	0.110295	0.110295
0.2	0.219	0.242273	0.237926	0.139493	0.241976	0.241977
0.3	0.358004	0.396175	0.416948	0.234568	0.395089	0.395105
0.4	0.516788	0.570231	0.652779	0.351046	0.56766	0.567812
0.5	0.693439	0.759555	0.930998	0.492328	0.755134	0.756014
0.6	0.884041	0.955094	1.200455	0.661821	0.949964	0.953566
0.7	1.082696	1.142444	1.396588	0.862943	1.141423	1.152949
0.8	1.282012	1.300569	1.519028	1.099099	1.315723	1.346364
0.9	1.474059	1.400444	1.616554	1.373660	1.456545	1.526911
1	1.651586	1.403645	1.694985	1.689896	1.546032	1.689498
SAE	0.478868	0.477505	1.008306	1.774001	0.25181	

4. Results and Discussion

The following are the generated results from the FODE of Fuzzy Rule 1 and Fuzzy Rule 2. The methods used to solve both of the FODEs are Runge-Kutta 4th order (RK-4), Mabood et al. [2], neuromethod (NM), GMDH, and the proposed new method (multiagent NN).

Revisiting Fuzzy Rule 1, $K = y, Q = 1, A_i = 0$, and $B_i^2 R = 1$:

$$\frac{dy}{dx} = -y^2(t) + 1, \quad y(0) = 0. \tag{23}$$

Revisiting Fuzzy Rule 2, $K = y$, $Q = 1$, $A_i = 1$, and $B_i^2 R = 1$:

$$\frac{dy}{dt} = 2y(t) - y^2(t) + 1, \quad y(0) = 0. \tag{24}$$

4.1. Multiagent NN Solution. In multiagent NN, one input layer, one hidden layer containing seven neurons, and one output layer are taken for training the NN (Figure 1). In the hidden layer, linear functions are assigned to the neurons. Initially the weights from input layer to hidden layer and from hidden layer to output layer are taken randomly. Then the weights are updated using gradient decent method while NN is under training the inputs. The NN solutions are compared with the solutions obtained by the existing methods. All the solutions are presented in Tables 1 and 2. All the solutions curves are displayed in Figures 2 and 3.

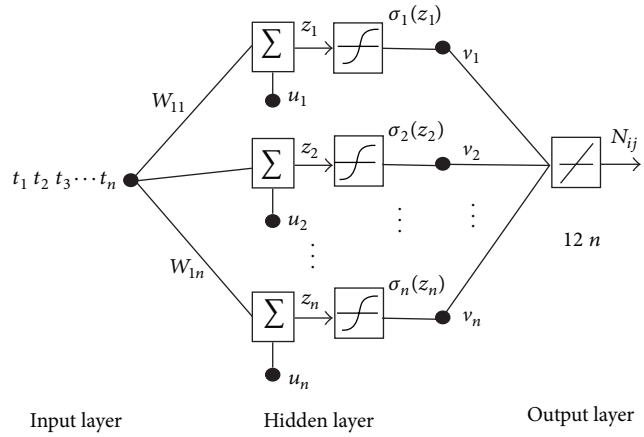

Figure 1: A neural network with arbitrary number of hidden nodes.

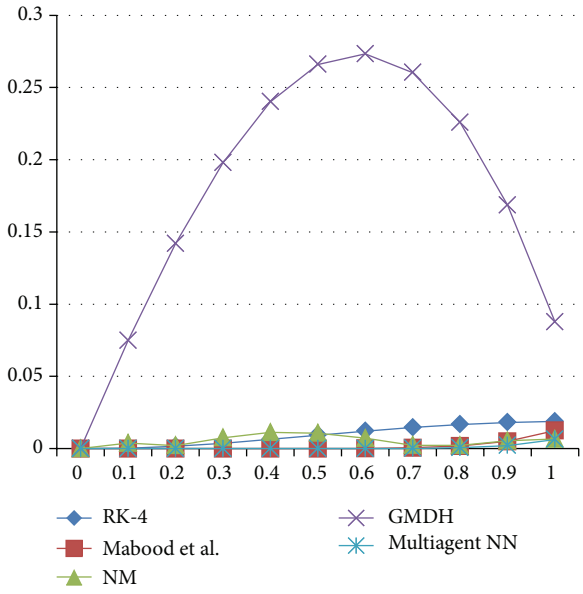

Figure 2: Solution curves by various methods for Fuzzy Rule 1.

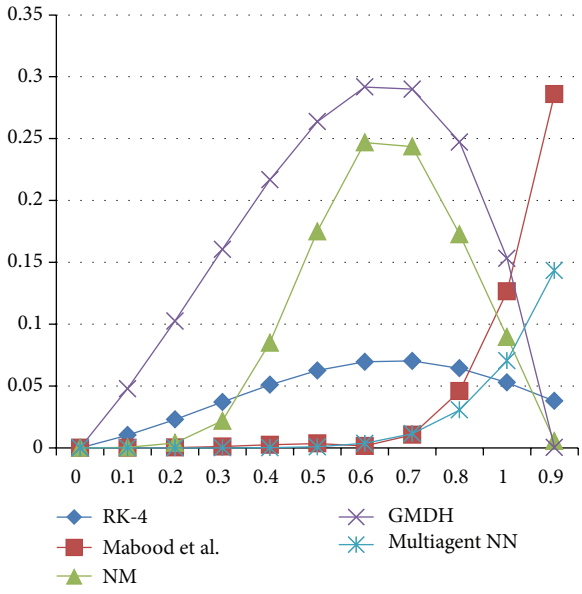

Figure 3: Solution curves by various methods for Fuzzy Rule 2.

For Fuzzy Rule 1, in Table 1, nontraditional methods (except GMDH) in all cases outperformed traditional method (RK-4).

In the case of Fuzzy Rule 2, only 2 methods, Mabood et al. [2] (only 0.3% improvement against RK-4) and multiagent NN, outperformed RK-4.

For Rule 1, multiagent NN improvement against RK-4 is 91%, Mabood et al. [2] is 54%, neuromethod (NM) is 90%, and GMDH is 99.5%.

For Rule 2, multiagent NN improvement against RK-4 is 47%, Mabood et al. [2] is 47%, neuromethod (NM) is 75%, and GMDH is 85.8%.

5. Conclusions

Multiagent neural network method has shown improved performance against RK-4, Mabood et al. [2], and neuromethod (NM). Added advantage of using neuromethod is its ability of continuous estimation. Because of this ability of continuous estimation, points, which are not explicitly trained, can be estimated without going through another phase of computational processing.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are very much thankful to the referees for their valuable comments and suggestions to improve this paper.

References

[1] S. Chen, X. Li, and X. Y. Zhou, "Stochastic linear quadratic regulators with indefinite control weight costs," *SIAM Journal on Control and Optimization*, vol. 36, no. 5, pp. 1685–1702, 1998.

[2] F. Mabood, A. I. M. Ismail, and I. Hashim, "Application of optimal homotopy asymptotic method for the approximate solution of Riccati equation," *Sains Malaysiana*, vol. 42, no. 6, pp. 863–867, 2013.

[3] W. T. Reid, *Riccati Differential Equations*, vol. 86 of *Mathematics in Science and Engineering*, Academic Press, New York, NY, USA, 1972.

[4] B. D. O. Anderson and J. B. Moore, *Optimal Control-Linear Quadratic Methods*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.

[5] I. Lasiecka and R. Triggiani, *Differential and Algebraic Riccati Equations with Application to Boundary/Point Control Problems: Continuous Theory and Approximation Theory*, vol. 164 of *Lecture Notes in Control and Information Sciences*, Springer, Berlin, Germany, 1991.

[6] F. Dubois and A. Saïdi, "Unconditionally stable scheme for Riccati equation," in *Contrôle des Systèmes Gouvernés par des Équations aux Dérivées Partielles (Nancy, 1999)*, vol. 8 of *ESAIM Proceedings*, pp. 39–52, Société de Mathématiques Appliquées et Industrielles, Paris, France, 2000.

[7] M. A. El-Tawil, A. A. Bahnasawi, and A. Abdel-Naby, "Solving Riccati differential equation using Adomian's decomposition method," *Applied Mathematics and Computation*, vol. 157, no. 2, pp. 503–514, 2004.

[8] Y. Tan and S. Abbasbandy, "Homotopy analysis method for quadratic Riccati differential equation," *Communications in Nonlinear Science and Numerical Simulation*, vol. 13, no. 3, pp. 539–546, 2008.

[9] F. Geng, "A modified variational iteration method for solving Riccati differential equations," *Computers and Mathematics with Applications*, vol. 60, no. 7, pp. 1868–1872, 2010.

[10] S. Ferrari and R. F. Stengel, "Smooth function approximation using neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 24–38, 2005.

[11] N. Kumaresan, "Solution of generalized matrix Riccati differential equation for indefinite stochastic linear quadratic singular

fuzzy system with cross-term using neural networks," *Neural Computing and Applications*, vol. 21, no. 3, pp. 497–503, 2012.

[12] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[13] M. S. Bin Shahrir, K. Ratnavely, and N. Kumaresan, "A Neural Network (NN) approach to solving a static-non-exchange scattering of electron-hydrogen," in *Proceeding of the 5th IEEE International Conference on Computational Intelligence, Modelling and Simulation*, pp. 14–16, 2013.

[14] J. Fojdl and R. W. Brause, "The performance of approximating ordinary differential equations by neural nets," in *Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '08)*, pp. 457–464, November 2008.

[15] S. Chandana and R. V. Mayorga, "Can we work around numerical methods? An insight," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06)*, vol. 2, pp. 1859–1860, July 2006.

[16] Z. Zeng, Y.-N. Wang, and H. Wen, "Numerical integration based on a neural network algorithm," *Computing in Science and Engineering*, vol. 8, no. 4, pp. 42–48, 2006.