Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 695647, 6 pages
http://dx.doi.org/10.1155/2013/695647

Research Article

Linear Simultaneous Equations' Neural Solution and Its Application to Convex Quadratic Programming with Equality-Constraint

Yuhuan Chen,[1] Chenfu Yi,[2,3] and Jian Zhong[1]

[1] *Center for Educational Technology, Gannan Normal University, Ganzhou 341000, China*
[2] *Research Center for Biomedical and Information Technology, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China*
[3] *School of Information Engineering, Jiangxi University of Science and Technology, Ganzhou, Jiangxi 341000, China*

Correspondence should be addressed to Chenfu Yi; itchenve@gmail.com

Received 7 August 2013; Accepted 18 September 2013

Academic Editor: Yongkun Li

A gradient-based neural network (GNN) is improved and presented for the linear algebraic equation solving. Then, such a GNN model is used for the online solution of the convex quadratic programming (QP) with equality-constraints under the usage of Lagrangian function and Karush-Kuhn-Tucker (KKT) condition. According to the electronic architecture of such a GNN, it is known that the performance of the presented GNN could be enhanced by adopting different activation function arrays and/or design parameters. Computer simulation results substantiate that such a GNN could obtain the accurate solution of the QP problem with an effective manner.

1. Introduction

A variety of scientific research and practical applications can be finalized as a matrix equation solving problem [1–6]. For example, the analysis of stability and perturbation for a control system could be viewed as the solution of Sylvester matrix equation [1, 2]; the stability and/or robustness properties of a control system could be obtained by the Lyapunov matrix equations solving [4–6]. Therefore, the real-time solution to matrix equation plays a fundamental role in numerous fields of science, engineering, and business.

As for the solution of matrix equations, many numerical algorithms have been proposed. In general, the minimal arithmetic operations of numerical algorithms are usually proportional to the cube of the dimension of the coefficient matrix, that is, $O(N^3)$ [7]. In order to be satisfied with the low complexity and real-time requirements, recently, numerous novel neural networks have been exploited based on the hardware implementation [2, 4, 5, 8–13]. For example, Tank and Hopfield solved the linear programming problems by using their proposed Hopfield neural networks (HNN) [9], which promoted the development of the neural networks in the optimization and other application problems. Wang in [10] proposed a kind of recurrent neural networks (RNN) models for the online solution of the linear simultaneous equations in parallel-processing circuit-implementation. In the previous work [2, 4, 5, 11], By Zhang's design method, a new type of RNN models is proposed for the solution of linear matrix-vector equation associated with the time-varying coefficient matrices in real-time.

In this paper, based on the Wang neural networks [10], we present an improved gradient-based neural model for the linear simultaneous equation, and then, such neural model is applied to solve the quadratic programming with equality-constraints. Much investigation and analysis on the Wang neural network have been presented in the previous work [10, 12, 13]. To make full use of the Wang neural network, we transform the convex quadratic programming into the general linear matrix-equation. Moreover, inspired by the design method of Zhang neural networks [2, 4, 5, 11, 12], the gradient-based neural network (GNN), that is, the Wang neural network, is improved, developed, and investigated for

the online solution of the convex quadratic programming with the usage of Lagrangian function and Karush-Kuhn-Tucker (KKT) condition. In Section 5, the computer simulation results show that, by improving their structures, we could also obtain the better performance for the existing neural network models.

2. Neural Model for Linear Simultaneous Equations

In this section, a gradient-based neural networks (GNN) model is presented for the linear simultaneous equations:

$$Ax = b, \tag{1}$$

where the nonsingular coefficient matrix $A := [a_{ij}] \in R^{n\times n}$ and the coefficient vector $b := [b_1, b_2, b_3, \ldots, b_n]^T \in R^n$ are given as constants and $x \in R^n$ is an unknown vector to be solved to make (1) hold true.

According to the traditional gradient-based algorithm [8, 10, 12], a scalar-valued norm-based energy function $\varepsilon(x) := \|Ax - b\|_2^2/2$ is firstly constructed, and then evolving along the descent direction resulting from such energy function, we could obtain the linear GNN model for the solution of linear algebraic (1); that is,

$$\dot{x} = -\gamma A^T (Ax - b), \tag{2}$$

where $\gamma > 0$ denotes the constant design parameter (or learning rate) used to scale the converge rate. To improve the convergence performance of neural networks, inspired by Zhang's neural networks [2, 4, 5, 11, 12], the linear model (2) could be improved and reformulated into the following general nonlinear form:

$$\dot{x} = -\Gamma A^T F (Ax - b), \tag{3}$$

where design parameter $\Gamma \in R^{n\times n}$ is a positive-definite matrix, which is used to scale the convergence rate of the solution. For simplicity, we can use γI in place of Γ with $\gamma > 0 \in R$ [4, 11]. In addition, the activation-function-array $F(\cdot) : R^n \to R^n$ is a matrix-valued mapping, in which each scalar-valued process unit $f(\cdot)$ is a monotonically increasing odd function. In general, four basic types of activation functions, linear, power, sigmoid, and power-sigmoid functions, can be used for the construction of neural solvers [4, 11]. The behavior of these four functions is exhibited in Figure 1, which shows that a different convergence performance could be achieved by using different activation functions. Furthermore, new activation functions could also be generated readily based on the above four activation functions. As for the neural model (3), we have the following theorem.

Theorem 1. *Consider a constant nonsingular coefficient-matrix $A \in R^{n\times n}$ and coefficient vector $b \in R^n$. If a monotonically increasing odd activation-function array $F(\cdot)$ is used, the neural state $x(t)$ of neural model* (3)*, starting from any initial state $x(0) \in R^n$, would converge to the unique solution $x^* = A^{-1}b$ of linear equation* (1).

Proof. Let solution error $\hat{x}(t) = x(t) - x^*(t)$. For brevity, hereafter argument t is omitted. Then, from (3), we have

$$\dot{\hat{x}} = -\gamma A^T F (A\hat{x}), \tag{4}$$

where $\Gamma = \gamma I$ for simplicity. Therefore, its entry-form could be written as

$$\dot{\hat{x}}_i = -\gamma \sum_{j=1}^{n} a_{ji} f\left(\sum_{j=1}^{n} \left(a_{ij}\hat{x}_j\right)\right), \quad \forall i,\ j \in \{1, 2, 3, \ldots, n\}. \tag{5}$$

Then, to analyze subsystem (5), we can define a Lyapunov candidate function as $v_i(t) = (\hat{x}_i^2)/2$. Obviously, $v_i(t) > 0$ for $\hat{x}_i \neq 0$, and $v_i(t) = 0$ only for $\hat{x}_i = 0$. Thus, the Lyapunov candidate function $v_i(t)$ is a nonnegative function. Furthermore, combining subsystem (5), we could get the time-derivative function of $v_i(t)$ as follows

$$\begin{aligned}\frac{dv_i(t)}{dt} &= \hat{x}_i\dot{\hat{x}}_i = -\gamma\hat{x}_i\sum_{j=1}^{n} a_{ji} f\left(\sum_{j=1}^{n}\left(a_{ij}\hat{x}_j\right)\right)\\ &= -\gamma\left(\sum_{j=1}^{n}\left(a_{ij}\hat{x}_j\right)\right)^T f\left(\sum_{j=1}^{n}\left(a_{ij}\hat{x}_j\right)\right) = -\gamma y^T f(y),\end{aligned} \tag{6}$$

where $y = \sum_{j=1}^{n}(a_{ij}\hat{x}_j)$. Since $f(\cdot)$ is an odd monotonically increasing function, we have $f(-u) = -f(u)$ and

$$f(u)\begin{cases} > 0 & \text{if } u > 0,\\ = 0 & \text{if } u = 0,\\ < 0 & \text{if } u < 0.\end{cases} \tag{7}$$

Therefore, $\gamma y^T f(y) > 0$ if $y \neq 0$, and $\gamma y^T f(y) = 0$ if and only if $y = 0$. In other words, the time-derivative $\dot{v}_i(t) = -\gamma y^T f(y)$ is nonpositive for any y. This can guarantee that $\dot{v}_i(t)$ is a negative-definite function. By Lyapunov theory [14, 15], each entry of solution error $\hat{x}_i$ in subsystem (5) can converge to zero; that is, $\hat{x}_i \to 0$. This means that solution error $\hat{x}(t) = x(t) - x^* \to 0$ as time $t \to \infty$. Therefore, the neural state $x(t)$ of neural model (3) could converge to the unique solution $x^* = A^{-1}b$ of linear equation (1). The proof on the convergence of neural model (3) is thus completed. □

3. Problem Formulation on Quadratic Programming

An optimization problem characterized by a quadratic objection function and linear constraints is named as a quadratic programming (QP) problem [16–18]. In this paper, we consider the following quadratic programming problem with equality-constraints:

$$\begin{aligned}&\text{minimize} \quad \frac{x^T P x}{2} + q^T x,\\ &\text{subject to} \quad Ax = b,\end{aligned} \tag{8}$$

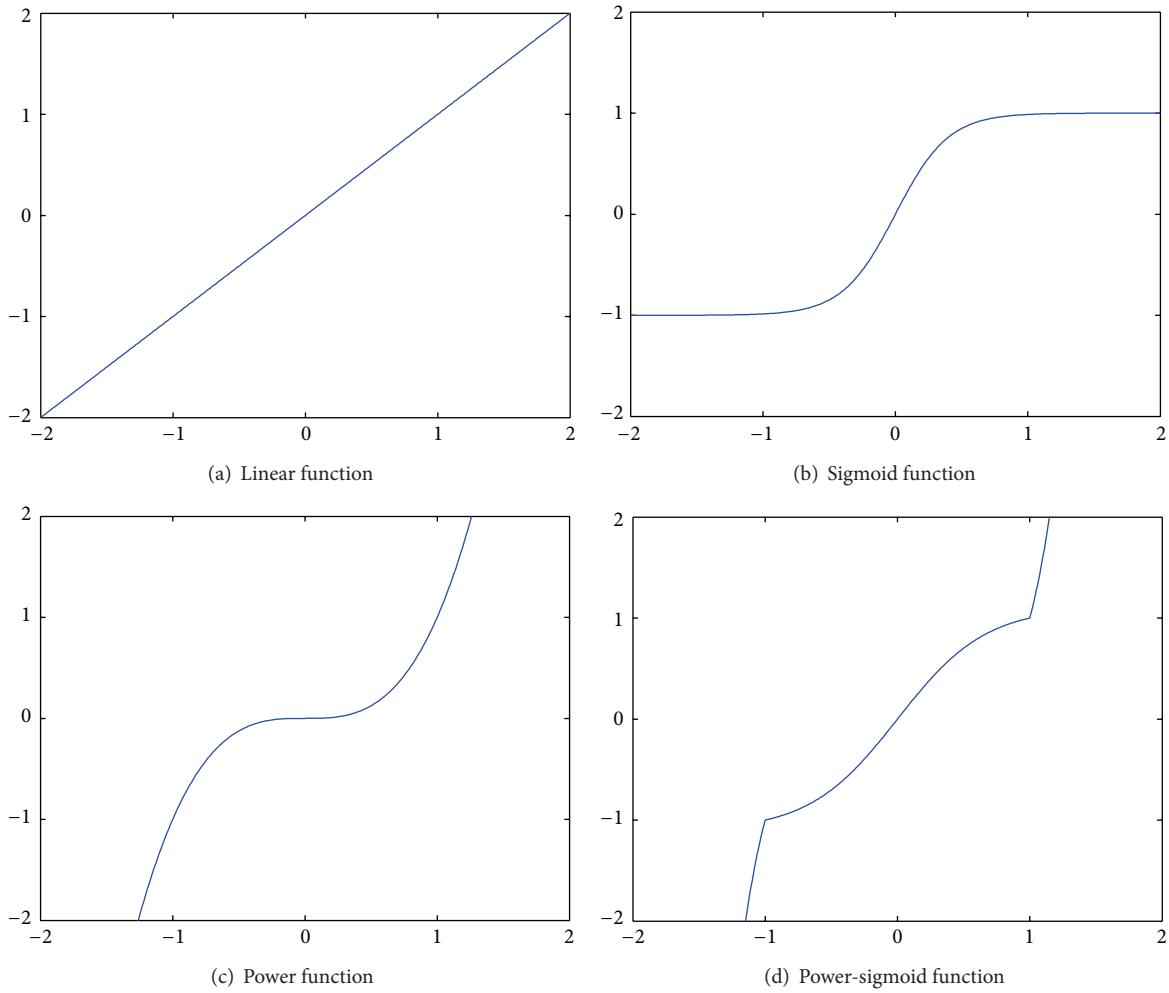

Figure 1: Behavior of the four basic activation functions.

where $P \in R^{n\times n}$ is a positive definite Hessian matrix, coefficients $q \in R^n$ and $b \in R^m$ are vectors, and $A \in R^{m\times n}$ is a full row-rank matrix. They are known as constant coefficients of the to be solved QP problem (8).

Therefore, $x \in R^n$ is unknown to be solved so as to make QP problem (8) hold true; especially, if there is no constraint, (8) is also called quadratic minimum (QM) problem. Mathematically, (8) can be written as minimize $f(x) = (1/2)x^T Px + q^T x$. For analysis convenience, let x^* denote the theoretical solution of QP problem (8).

To solve QP problem (8), firstly, let us consider the following general form of quadratic programming:

$$\begin{aligned} &\text{minimize} \quad f(x), \\ &\text{subject to} \quad h_j(x) = 0,\ j \in \{1, 2, 3, \ldots, m\}. \end{aligned} \tag{9}$$

As for (9), a Lagrangian function could be defined as

$$L(x, \lambda) = f(x) + \sum_{j=1}^{m} \lambda_j h_j(x) = f(x) + \lambda^T h(x), \tag{10}$$

where $\lambda = [\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_m]^T$ denotes the Lagrangian multiplier vector and equality constraint $h(x) = [h_1(x), h_2(x), h_3(x), \ldots, h_m(x)]^T$. Furthermore, by following the previously-mentioned Lagrangian function and Karush-Kuhn-Tucker (KKT) condition, we have

$$\begin{aligned} \frac{\partial L(x, \lambda)}{\partial x} &= Px + q + A^T \lambda = 0, \\ \frac{\partial L(x, \lambda)}{\partial \lambda} &= Ax - b = 0. \end{aligned} \tag{11}$$

Then, (11) could be further formulated as the following matrix-vector form:

$$\widetilde{P}\widetilde{x} = -\widetilde{q}, \tag{12}$$

where $\widetilde{P} := \begin{bmatrix} P & A^T \\ A & \mathbf{0}_{m\times m} \end{bmatrix} \in R^{(n+m)\times(n+m)}$, $\widetilde{x} := \begin{bmatrix} x \\ \lambda \end{bmatrix} \in R^{(n+m)}$, and $\widetilde{q} := \begin{bmatrix} q \\ -b \end{bmatrix} \in R^{(n+m)}$. Therefore, we can obtain the solution $x \in R^n$ of (8) by transforming QP problem (8) into matrix-vector equation (12). In other words, to get the solution $x \in R^n$ of (8), QP problem (8) is firstly transformed into the matrix-vector equation (12), which is a linear matrix-vector equation similar

to the linear simultaneous equations (1), and then, we thus can make full use of the neural solvers presented in Section 2 to solve the QP problem (8). Moreover, the first n elements of solution $\widetilde{x}(t)$ of (12) compose the neural solution $x(t)$ of (8), and the Lagrangian vector λ consists of the last m elements.

4. Application to QP Problem Solving

For analysis and comparison convenience, Let $\widetilde{x}^* = [x^{*^T}, \lambda^{*^T}]^T$ denote the theoretical solution of (12). Since QP problem (8) could be formulated into the matrix-vector form (12), we can directly utilize the neural solvers (2) and (3) to solve problem (12). Therefore, neural solver (2) used to solve (12) can be written as the following linear form:

$$\dot{\widetilde{x}} = -\gamma \widetilde{P}^T \widetilde{P} \widetilde{x} - \gamma \widetilde{P}^T \widetilde{q} = -\gamma \widetilde{P}^T \left(\widetilde{P} \widetilde{x} + \widetilde{q} \right). \tag{13}$$

If such linear model is activated by the nonlinear function arrays, we have

$$\dot{\widetilde{x}} = -\Gamma \widetilde{P}^T F \left(\widetilde{P} \widetilde{x} + \widetilde{q} \right). \tag{14}$$

In addition, according to model (14), we can also draw its architecture for the electronic realization, as illustrated in Figure 2. From model (14) and Figure 2, we readily know that different performance of (14) can be achieved by using different activation function arrays $F(\cdot)$ and design parameter Γ. In the next section, the previously-mentioned four basic functions are used to simulate model (14) for achieving different convergence performance. In addition, from Theorem 1 and [4, 12], we have the following theorem on the convergence performance of GNN model (14).

Theorem 2. *Consider the time-invariant strictly-convex quadratic programming problem* (8). *If a monotonically increasing odd activation-function array* $F(\cdot)$ *is used, the neural state* $\widetilde{x}(t) := [x^T, \lambda^T]^T$ *of GNN model* (14) *could globally converge to the theoretical solution* $\widetilde{x}^*(t) := [x^{*^T}, \lambda^{*^T}]^T$ *of the linear matrix-vector form* (12). *Note that, the first n elements of* $\widetilde{x}(t)$ *are corresponding to the theoretical solution* x^* *of QP problem* (8), *and the last m elements are those of the Lagrangian vector* λ.

5. Simulation and Verification

In this section, neural model (14) is applied to solve the QP problem (8) in real-time for verification. As an illustrative example, consider the following QP problem:

$$\begin{aligned} \text{minimize} \quad & x_1^2 + 2x_2^2 + x_3^2 - 2x_1x_2 + x_3, \\ \text{subject to} \quad & x_1 + x_2 + x_3 = 4, \\ & 2x_1 - x_2 + x_3 = 2, \\ & x_1 > 0,\ x_2 > 0,\ x_3 > 0. \end{aligned} \tag{15}$$

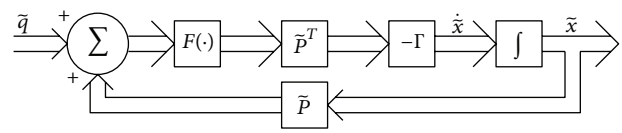

Figure 2: Block diagram of the GNN model (14).

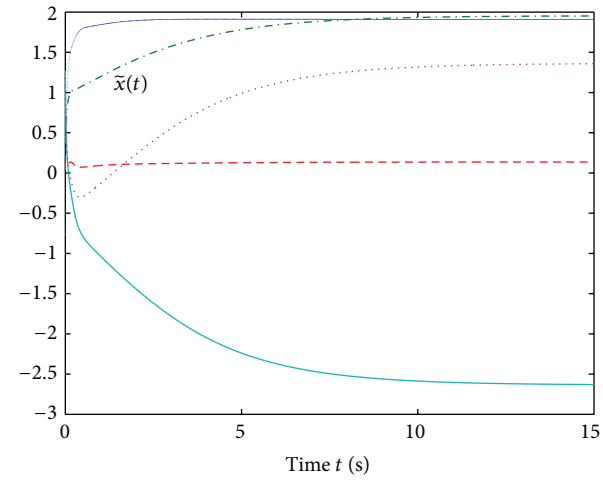

Figure 3: Neural state $\widetilde{x}(t)$ of QP problem (15) by the GNN model (14) with the usage of power-sigmoid function and design parameter $\gamma = 1$.

Obviously, we can write the equivalent matrix-vector form of QP problem (8) with the following coefficients:

$$P = \begin{bmatrix} 2 & -2 & 0 \\ -2 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad q = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 4 \end{bmatrix}. \tag{16}$$

For analysis and compassion, we can utilize the MATLAB routine "quadprog" to obtain the theoretical solution of QP (15), that is, $x^* = [1.9091, 1.9545, 0.1364]^T$.

According to Figure 2, GNN model (14) is applied to the solution of QP problem (15) in real-time, together with the usage of power-sigmoid function array and design parameter $\gamma = 1$. As shown in Figure 3, we know that, when starting from randomly-generated initial state $\widetilde{x}_0 = [-2, 2] \in R^5$, the neural state $\widetilde{x}(t)$ of GNN model (14) is fit well with the theoretical solution after 10 seconds or so. That is, GNN model (14) could achieve the exact solution. Note that the first $n = 3$ elements of neural solution are corresponding to the theoretical solution $x^* = [1.9091, 1.9545, 0.1364]^T$, while the last $m = 2$ elements are the Lagrangian multiplier vector.

In addition, the residual error $\|\widetilde{P}\widetilde{x} + \widetilde{q}\|_F^2$ could be used to track the solution-process. The trajectories of residual error could be shown in Figure 4, which is generated by GNN model (14) solving QP problem (15) activated by different activation function arrays, that is, linear, power, sigmoid, and power-sigmoid functions, respectively. Obviously, under

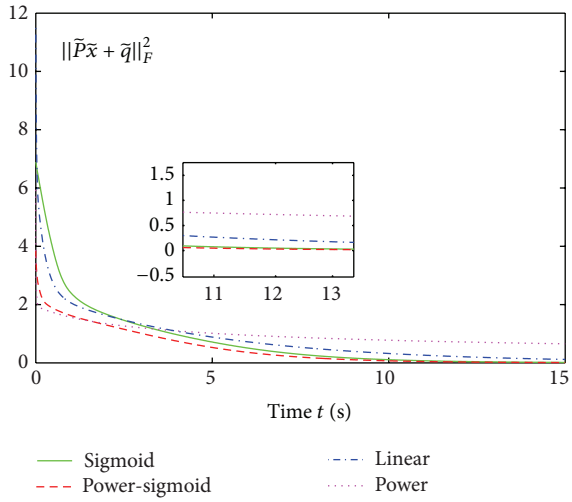

FIGURE 4: Online solution of QP problem (15) by GNN (14) with design parameter $\gamma = 1$ and the four basic activations.

TABLE 1: Performance of GNN model by using different design parameter γ and activations.

γ	GNN_{lin}	GNN_{power}	GNN_{sig}	GNN_{ps}
1	1.74×10^{-2}	0.267	2.07×10^{-2}	3.68×10^{-3}
10	1.89×10^{-3}	8.93×10^{-2}	2.56×10^{-3}	1.03×10^{-3}
100	7.24×10^{-4}	2.65×10^{-2}	9.12×10^{-4}	3.32×10^{-4}

the same simulation environments (such as, design parameter and GNN model (14)), different convergence performance could be achieved when different activation function arrays are used. As shown in Table 1, we use GNN_{lin}, GNN_{power}, GNN_{sig}, and GNN_{ps} to denote the performance of residual error obtained by GNN model (14) activated by linear, power, sigmoid, and power-sigmoid function arrays and have the following simulative results.

(i) When the same design parameter γ is used, the performance of GNN_{ps} is the best, while the residual-error of GNN_{power} is bigger. For example, when design parameter $\gamma = 1$, $(GNN_{ps} = 3.68\times10^{-3}) < (GNN_{lin} = 1.74\times10^{-2}) < (GNN_{sig} = 2.07\times10^{-2}) < (GNN_{power} = 0.267)$.

(ii) When the same activation functions are used, the performance of residual-error would be better with the increase of the value of design parameter γ. For example, when linear functions are used, the values of residual-error are 1.74×10^{-2}, 1.89×10^{-3}, and 7.24×10^{-4} corresponding to $\gamma = 1$, $\gamma = 10$, and $\gamma = 100$, respectively.

Among the four basic activation functions, we could achieve the best convergence performance when using power-sigmoid functions under the same situations. Therefore, GNN model (14) has the best convergence performance when using power-sigmoid function, while when using power function, there exist apparent residual errors between the neural state $\tilde{x}(t)$ and theoretical solution x^*. We thus generally use power-sigmoid activation function to achieve the superior convergence performance, as shown in Figure 3.

6. Conclusions

On the basis of the Wang neural network, an improved gradient-based neural network has been presented to the solution of the convex quadratic programming problem in real-time. Compared to the other three activation functions, the power-sigmoid function is the best choice for the superior convergence performance. Computer simulation results further substantiate that the presented GNN model could solve the convex QP problem with accuracy and efficiency, and the convergence performance could be obtained by using the power-sigmoid activation function.

Conflict of Interests

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

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant 61363076 and the Programs for Foundations of Jiangxi Province of China (GJJ13649, GJJ12367, GJJ13435, and 20122BAB211019) and partially supported by the Shenzhen Programs (JC201005270257A and JC201104220255A).

References

[1] P. Liu, S. Zhang, and Q. Li, "On the positive definite solutions of a nonlinear matrix equation," *Journal of Applied Mathematics*, vol. 2013, Article ID 676978, 6 pages, 2013.

[2] Y. Zhang, D. Jiang, and J. Wang, "A recurrent neural network for solving sylvester equation with time-varying coefficients," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1053–1063, 2002.

[3] Y. Li, M. Reisslein, and C. Chakrabarti, "Energy-efficient video transmission over a wireless link," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1229–1244, 2009.

[4] C. Yi, Y. Chen, and X. Lan, "Comparison on neural solvers for the Lyapunov matrix equation with stationary & nonstationary coefficients," *Applied Mathematical Modelling*, vol. 37, no. 4, pp. 2495–2502, 2013.

[5] F. Ding and T. Chen, "Gradient based iterative algorithms for solving a class of matrix equations," *IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1216–1221, 2005.

[6] M. A. Ghorbani, O. Kisi, and M. Aalinezhad, "A probe into the chaotic nature of daily streamflow time series by correlation dimension and largest Lyapunov methods," *Applied Mathematical Modelling*, vol. 34, no. 12, pp. 4050–4057, 2010.

[7] Y. Zhang and W. E. Leithead, "Exploiting Hessian matrix and trust-region algorithm in hyperparameters estimation of Gaussian process," *Applied Mathematics and Computation*, vol. 171, no. 2, pp. 1264–1281, 2005.

[8] X. Zou, Y. Tang, S. Bu, Z. Luo, and S. Zhong, "Neural-network-based approach for extracting eigenvectors and eigenvalues of

real normal matrices and some extension to real matrices," *Journal of Applied Mathematics*, vol. 2013, Article ID 597628, 13 pages, 2013.

[9] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 5, pp. 533–541, 1986.

[10] J. Wang, "Electronic realisation of recurrent neural network for solving simultaneous linear equations," *Electronics Letters*, vol. 28, no. 5, pp. 493–495, 1992.

[11] Y. Zhang, C. Yi, and W. Ma, "Simulation and verification of Zhang neural network for online time-varying matrix inversion," *Simulation Modelling Practice and Theory*, vol. 17, no. 10, pp. 1603–1617, 2009.

[12] C. Yi and Y. Zhang, "Analogue recurrent neural network for linear algebraic equation solving," *Electronics Letters*, vol. 44, no. 18, pp. 1078–1080, 2008.

[13] K. Chen, "Robustness analysis of Wang neural network for online linear equation solving," *Electronic Letters*, vol. 48, no. 22, pp. 1391–1392, 2012.

[14] Y. Zhang, "Dual neural networks: design, analysis, and application to redundant robotics," in *Progress in Neurocomputing Research*, pp. 41–81, Nova Science Publishers, New York, NY, USA, 2008.

[15] Y. Zhang and J. Wang, "Global exponential stability of recurrent neural networks for synthesizing linear feedback control systems via pole assignment," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 633–644, 2002.

[16] N. Petrot, "Some existence theorems for nonconvex variational inequalities problems," *Abstract and Applied Analysis*, vol. 2010, Article ID 472760, 9 pages, 2010.

[17] S. Burer and D. Vandenbussche, "A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations," *Mathematical Programming*, vol. 113, no. 2, pp. 259–282, 2008.

[18] Z. Dostál and R. Kučera, "An optimal algorithm for minimization of quadratic functions with bounded spectrum subject to separable convex inequality and linear equality constraints," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 2913–2938, 2010.