Design and Analysis of a General Recurrent Neural Network Model for Time-Varying Matrix Inversion

Yunong Zhang, *Member, IEEE,* and Shuzhi Sam Ge, *Senior Member, IEEE*

***Abstract*—Following the idea of using first-order time derivatives, this paper presents a general recurrent neural network (RNN) model for online inversion of time-varying matrices. Different kinds of activation functions are investigated to guarantee the global exponential convergence of the neural model to the exact inverse of a given time-varying matrix. The robustness of the proposed neural model is also studied with respect to different activation functions and various implementation errors. Simulation results, including the application to kinematic control of redundant manipulators, substantiate the theoretical analysis and demonstrate the efficacy of the neural model on time-varying matrix inversion, especially when using a power-sigmoid activation function.**

***Index Terms*—Activation function, implicit dynamics, inverse kinematics, recurrent neural network (RNN), time-varying matrix inversion.**

I. Introduction

THE problem of finding the inverse of a time varying matrix online arises in numerous fields of science, engineering, and business. It is usually an essential part of many solutions, e.g., as preliminary steps for optimization [1], signal processing [2], electromagnetic systems [3], and robot kinematics [4]. Since the mid-1980s, efforts have been directed toward computational aspects of fast matrix inversion and many algorithms have been proposed [5]–[8]. For many numerical methods, the minimal arithmetic operations are usually proportional to the cube of the matrix dimension [9], and consequently such algorithms performed on digital computers may not be efficient enough in large-scale online applications. In view of this, parallel computation schemes have been investigated for matrix inversion.

The dynamical approach is one of the important methods for solving matrix inversion problems [4], [10], [11]. Recently, due to the in-depth research in neural networks, numerous dynamic solvers based on recurrent neural networks (RNNs) have been developed and investigated [2], [12]–[20]. Particularly, a simple neural network was proposed to solve linear programming problems in real time and implemented on analog circuits [12]. The neural approach is now regarded as a powerful alternative for online computation because of its parallel distributed nature and convenience of hardware implementation.

However, almost all the aforementioned numerical, dynamical and neural computation schemes were designed intrinsically for constant matrices rather than time-varying ones. They are in general related to the gradient descent method in optimization [1], where a scalar-valued cost function is first constructed such that its minimum point is the matrix inverse, and then an algorithm is designed to evolve along a descent direction of this cost function until a minimum is reached. The typical descent direction is the negative gradient. Since the matrix to be inverted online is usually time varying, such a gradient-based method only works approximately and with appreciable residual errors, as shown in [21, Figs. 4 and 5]. Moreover, the gradient-based method also requires much faster convergence in comparison with the time scale of time-varying matrices or imposes very stringent restrictions on design parameters [22].

In this paper, following the idea of using first-order time derivatives [21], a general RNN model with implicit dynamics is developed and analyzed for solving the problem of time-varying matrix inversion. Neural dynamics are elegantly introduced by defining the matrix-valued error-monitoring function rather than the usual scalar-valued cost function such that the computation error can be made decreasing to zero globally and asymptotically. As noted, nonlinearity and errors always exist. Even if a linear activation function is used, the nonlinear phenomenon may appear in its hardware implementation. For superior convergence and better robustness, different kinds of activation functions (linear, sigmoid, power functions, and/or their variants, e.g., power-sigmoid function) are investigated. Theoretical and simulation results both demonstrate the efficacy of the proposed neural approach. To the best of our knowledge, there is little work dealing with such a problem in the literature at present stage, except some preliminary results presented in [21]. The main contributions of the paper are as follows:

i) the establishment of a general nonlinear RNN model is introduced for time-varying matrix inversion by using monotonically increasing odd functions as activation functions;
ii) the utilization of time derivative of given time-varying matrices plays an important role for efficient online matrix inversion;
iii) the computational deviation caused by possible differentiation and implementation errors can be made arbitrarily small by increasing the design parameters through robustness analysis; and

Manuscript received March 11, 2003; revised April 17, 2005.

Y. Zhang was with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore. He is now with the Hamilton Institute, National University of Ireland, Maynooth, Ireland (e-mail: ynzhang@ieee.org).

S. S. Ge is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore (e-mail: elegesz@nus.edu.sg).

Digital Object Identifier 10.1109/TNN.2005.857946

iv) the proposed scheme is applied to inverse kinematic control of redundant manipulators via online solution of time-varying pseudoinverse.

The remainder of this paper is organized as follows. Section II presents the problem formulation and preliminaries. In Section III, the general RNN model with implicit dynamics is described in detail. Corresponding to different activation functions, convergence and robustness results are studied in Section IV. Section V presents an illustrative example of solving time-varying matrix inverse in real time. In Section VI, the proposed neural model is applied to inverse kinematic control of redundant manipulators.

II. Problem Formulation

Consider a smooth time-varying matrix $A(t) \in R^{n\times n}$. We are to find $X(t) \in R^{n\times n}$ such that the following matrix equation holds

$$A(t)X(t) - I = 0, \quad t \in [0, +\infty) \tag{1}$$

where $I \in R^{n\times n}$ is the identity matrix. The objective of this paper is to invert time-varying matrices in real time and in an error-free manner.

Without loss of generality, $A(t)$ and its time derivative $\dot{A}(t)$ are assumed to be known or measurable. As a basis of discussion, the existence of the inverse $A^{-1}(t)$ at any time instant t is also assumed. To facilitate the convergence and robustness analysis, the following condition is introduced to guarantee the existence of $A^{-1}(t)$:

Invertibility Condition: There exists a positive real number $\alpha > 0$ such that

$$\min_{\forall i\in\{1,\cdots,n\}} |\lambda_i(A(t))| \geqslant \alpha, \quad \forall t \geqslant 0 \tag{2}$$

where $\lambda_i(\cdot)$ denotes the ith eigenvalue of matrix $A \in R^{n\times n}$.

If the invertibility condition holds, it is clear that there exists a unique solution to (1). Let $\|A\|_F = \sqrt{\mathrm{tr}(A^T A)}$ denote the Frobenius norm. Then, the invertibility condition leads to the following Lemma on the boundedness of $\|A(t)\|$ and $\|A^{-1}(t)\|$.

Lemma 1: If $A(t)$ satisfies the invertibility condition (2) with its norm uniformly upper bounded by β (i.e., $\|A(t)\|_F \leqslant \beta$, $\forall t \geqslant 0$), then $\|A^{-1}(t)\|_F$ is uniformly upper bounded by a scalar $\varphi(\alpha, \beta, n) = \sum_{i=0}^{n-2} C_n^i \beta^{n-i-1}/\alpha^{n-i} + n^{3/2}/\alpha$, where $C_n^i := n!/(i!(n-i)!)$.

Proof: See the Appendix for details. □

It is worth mentioning that the invertibility condition, parameter α in (2), and Lemma 1 are used only for analytic purposes, and that there is no need to know the exact value of α. Instead, in a practical application via the proposed neural network model, we could check the invertibility condition by simply monitoring whether or not the value of $\|AX - I\|$ is becoming very small (near zero) after network started. For example, in view of the existence of exponential convergence, after $4/\gamma$ seconds, $\|AX - I\|$ should be less than $1.85\%(\approx e^{-4})$ of $\zeta\|AX(0) - I\|$ [22], where γ is to be defined as the standard convergence rate and also a design parameter of the neural network, and constant $\zeta > 0$. If $\gamma = 10^6$, such an exponential-convergence time is around $4 \sim 6$ μs. Otherwise, the matrix could be singular.

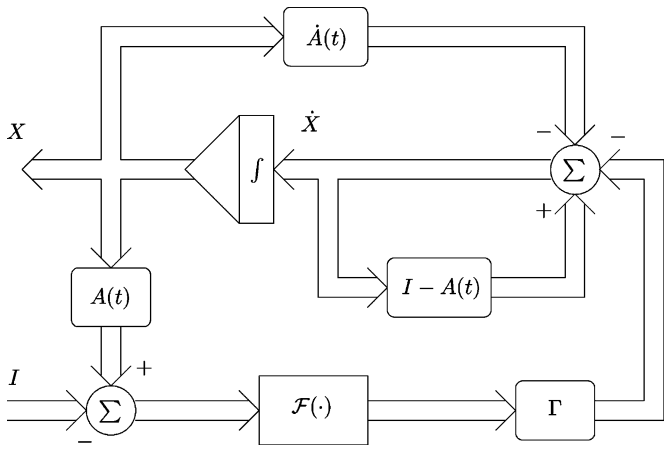


Fig. 1. Block diagram of general RNN model (4) for online matrix inversion.

III. A General RNN Model

In the literature, traditional gradient-based neural network approaches [2], [10], [13], [15], [17], [19] have been developed to compute the inverse of a time-invariant matrix. For the time-varying case, much faster convergence rate of the gradient-based networks is usually required in comparison with the time scale of time-varying matrices. Otherwise, the gradient-based networks often yield relatively large computational errors. To solve time-varying Sylvester equations online, an elegant RNN was first proposed in [21] by utilizing the time derivatives of the coefficient matrices. The following general RNN design method is an important extension of the aforementioned neural network approach [21] to a time-varying one with general nonlinear activation functions and various implementation errors considered.

To monitor the matrix-inversion process, the following matrix-valued error function is defined instead of the usual scalar-valued cost functions

$$E(X(t), t) := A(t)X(t) - I.$$

The error-function derivative $\dot{E}(X(t), t)$ should be made such that every entry $e_{ij}(t), i, j = 1, \cdots, n$, of $E(X(t), t)$ converges to zero. Specifically, $\dot{E}(X(t), t)$ can be described in the general form

$$\frac{dE(X(t), t)}{dt} = -\Gamma\mathcal{F}(E(X(t), t)) \tag{3}$$

where Γ is in general a positive-definite matrix used to scale the convergence rate of the solution, and $\mathcal{F}(\cdot) : R^{n\times n} \to R^{n\times n}$ denotes a matrix mapping of neural networks in the context of this paper. The RNN design formula (3) leads to the following implicit dynamic equation of the generalized neural model

$$A(t)\dot{X}(t) = -\dot{A}(t)X(t) - \Gamma\mathcal{F}(A(t)X(t) - I) \tag{4}$$

where $X(t)$, staring from an initial condition $X(0) = X_0 \in R^{n\times n}$, is the activation state matrix corresponding to the theoretical solution $X^*(t)$ of (1).

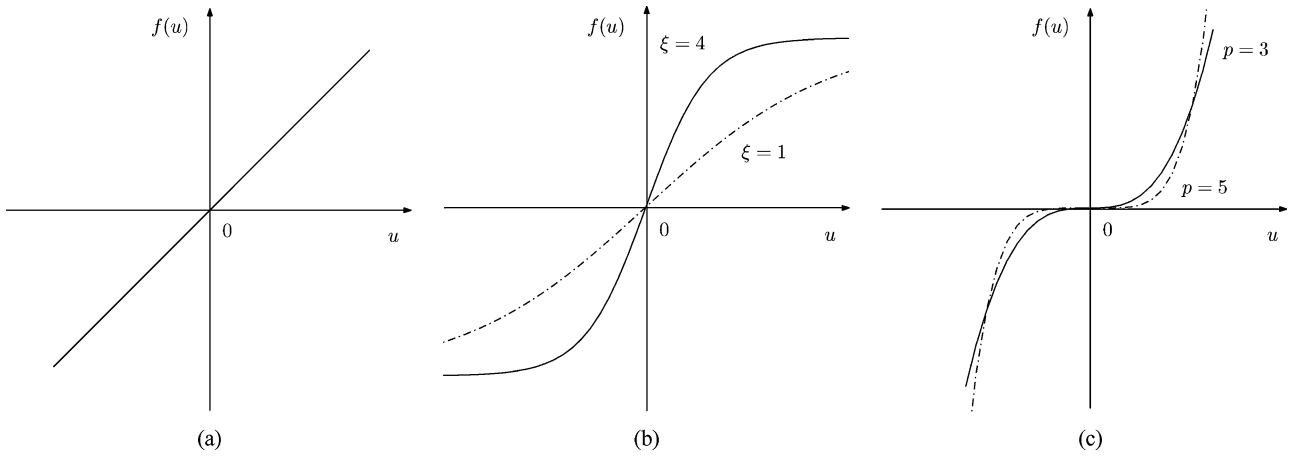


Fig. 2. Activation functions $f(\cdot)$ for the proposed neural network model (4). (a) Linear function. (b) Sigmoid function. (c) Power function.

Similar to the usual neural approaches, the matrix parameter Γ in (4), like a set of inductance parameters or reciprocals of capacitive parameters, is set as large as the hardware permits (e.g., in analog circuits or VLSI [23]) or selected appropriately for experimental/simulative purposes. In comparison with the classical Hopfield-type RNN (termed as differential neural network in [24]) described by an explicit set of differential equations, the proposed general neural model is described by an implicit dynamic equation which arises frequently in analog electric circuits and systems due to Kirchhoff's rules. In addition, the model methodically exploits the time derivative of matrix $A(t)$ during its inverting process.

The block diagram realization of the neural model (4) is shown in Fig. 1 in the most general form. In view of (4) and Fig. 1, different choices for Γ and $\mathcal{F}$ will lead to different performance. In general, any monotonically increasing odd activation function $f(\cdot)$, being the ijth element of matrix mapping $\mathcal{F}$, can be used for the construction of the neural network. To demonstrate the main ideas and results of the paper, however, only three basic types of activation functions are focused on: linear function $f(u) = u$, bipolar sigmoid function $f(u) = (1-\exp(-\xi u))/(1+\exp(-\xi u))$ with $\xi \geqslant 1$, and power function $f(u) = u^p$ with integer $p \geqslant 3$ as shown in Fig. 2. New activation-function variants can thus be generated readily based on these three basic types.

IV. Convergence and Robustness

While Section III only gives a general framework to solve this kind of problem, detailed design consideration and main theoretical results are given in this section. To keep the differential equation well conditioned, it is desired to keep Γ well conditioned, i.e., its eigenvalues of Γ are in the same scale. For simplicity, in this paper, we assume $\Gamma = \gamma I$. As a consequence, we would be able to keep every $e_{ij}(t)$, $i, j = 1, \cdots, n$, converge at the same rate and at the same time, to simplify the network design and analysis.

A. Convergence Analysis

The following theoretical results on global convergence and exponential convergence are presented.

Theorem 1: Given time-varying matrix $A(t) \in R^{n\times n}$ satisfying the invertibility condition (2), if a monotonically increasing odd function array $\mathcal{F}(\cdot)$ is used, then the state matrix $X(t)$ of neural system (4) starting from any initial state $X_0 \in R^{n\times n}$ converges to the time-varying theoretical inverse $X^*(t)$ of matrix $A(t)$. In addition, the neural system (4) possesses

i) exponential convergence with the rate γ if using linear activation function;
ii) exponential convergence with the rate $\xi\gamma/2$ for error range $e_{ij}(t) \in (-\ln 2/\xi, +\infty)$ if using bipolar sigmoid function; and
iii) superior convergence for error range $|e_{ij}(t)| > 1$ to cases (i) and (ii), if using power activation function.

Proof: Let $\tilde{X}(t) := X(t) - X^*(t)$ denote the difference between the solution $X(t)$ generated by the proposed neural network and the theoretical solution $X^*(t)$ of (1). By using the identity $\dot{A}(t)X^*(t) + A(t)\dot{X}^*(t) = 0$ which is the time derivative of (1), it follows that $\tilde{X}(t)$ is the solution to the ensuing dynamics with the initial state $\tilde{X}(0) = X_0 - X^*(0)$

$$A(t)\dot{\tilde{X}}(t) + \dot{A}(t)\tilde{X}(t) = -\gamma\mathcal{F}\left(A(t)\tilde{X}(t)\right). \tag{5}$$

Since $E(t) = A(t)\tilde{X}(t)$, (5) can be rewritten as $\dot{E}(t) = -\gamma\mathcal{F}(E(t))$, which is a compact matrix form of the following set of n^2 equations

$$\dot{e}_{ij}(t) = -\gamma f\left(e_{ij}(t)\right), \quad \forall i, j \in \{1, 2, \cdots, n\}. \tag{6}$$

Clearly, we can define a Lyapunov function candidate $v_{ij} = e_{ij}^2/2 \geqslant 0$ for the ijth subsystem (6) with its time derivative

$$\frac{dv_{ij}}{dt} = e_{ij}\dot{e}_{ij} = -\gamma e_{ij} f(e_{ij}). \tag{7}$$

Because monotonically increasing odd functions are used as activation functions, we have $f(-u) = -f(u)$, and

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

which guarantees the negative definiteness of $\dot{v}_{ij}$; i.e., $\dot{v}_{ij} < 0$ for $e_{ij} \neq 0$ and $\dot{v}_{ij} = 0$ for $e_{ij} = 0$. By the Lyapunov

stability theory [25], $e_{ij}(t)$ globally converges to zero for any $i,j \in \{1,\cdots,n\}$. Thus, in view of $E(t) = A(t)\tilde{X}(t)$ and (2), we have $\tilde{X}(t) \to 0 \in R^{n\times n}$ as $t \to \infty$, i.e., the neural state $X(t)$ is globally convergent to the theoretical inverse $X^*(t)$. The proof on global convergence is thus complete.

In view of $E = A\tilde{X}$, (2) and Lemma 1, we have $\|X(t) - X^*(t)\|_F \leqslant \|A^{-1}\|_F\|E\|_F \leqslant \varphi(\sum_i^n \sum_j^n e_{ij}^2)^{1/2} \leqslant n\varphi \max_{1\leqslant i,j\leqslant n}|e_{ij}|$, which implies the computation error and the network convergence can be estimated by those of maximum entry error $e_{ij}(t)$ in (6) [22], [26]. We now come to prove the additional convergence properties corresponding to specific kinds of activation function $f(\cdot)$.

(i) For the simple linear case, $\dot{e}_{ij} = -\gamma e_{ij}$, and the entry error is $e_{ij}(t) = \exp(-\gamma t)e_{ij}(0)$. Thus there exists a constant $\zeta(\alpha,\beta,n,e_{ij}(0)) > 0$ such that $\|X(t) - X^*(t)\|_F \leqslant \zeta\exp(-\gamma t)$. This means that neural network (4) possesses the exponential convergence with rate γ when using linear activation function $f(u) = u$.

(ii) For the bipolar sigmoid function $f(u) = (1 - \exp(-\xi u))/(1 + \exp(-\xi u))$, define the constant $\eta = (\exp(-\xi e_{ij}(0)) - 1)^2/\exp(-\xi e_{ij}(0))$. The solution to (6) is given as $e_{ij}(t) = -\ln(1 + z(t))/\xi$ where

$$z(t) = \frac{\eta\exp(-\xi\gamma t)}{2} - \operatorname{sgn}(e_{ij}(0))\sqrt{\left(1+\frac{\eta\exp(-\xi\gamma t)}{2}\right)^2 - 1}. \tag{8}$$

Using the Taylor series expansion formula $\ln(1 + z) = z - z^2/2 + z^3/3 - + \cdots$ and the algebraic formula $z + z^2 + z^3 + \cdots = z/(1 - z)$ for $|z| < 1$, we have

$$\begin{aligned} |e_{ij}(t)| &\leqslant \left|-\frac{z}{\xi} + \frac{z^2}{(2\xi)} - \frac{z^3}{(3\xi)} + \cdots\right| \\ &\leqslant \frac{(|z| + |z|^2 + |z|^3 + \cdots)}{\xi} \\ &= \frac{|z|}{(\xi\,(1 - |z|))}. \end{aligned}$$

It follows from the formulas' requirement, $|z| < 1$, that the error range is defined as $e_{ij}(t) > -\ln 2/\xi$. From (8), we have

$$\begin{aligned} |z(t)| &\leqslant \frac{\eta\exp(-\xi\gamma t)}{2} + \sqrt{\frac{\eta^2\exp(-2\xi\gamma t) + 4\eta\exp(-\xi\gamma t)}{4}} \\ &\leqslant \frac{\eta}{2}\exp(-\xi\gamma t) + \frac{\sqrt{\eta^2+4\eta}}{2}\exp\left(-\frac{\xi\gamma t}{2}\right) \\ &\leqslant \frac{\eta + \sqrt{\eta^2+4\eta}}{2}\exp\left(-\frac{\xi\gamma t}{2}\right). \end{aligned}$$

In view of the global convergence of $e_{ij}(t)$ and $z(t)$ to zero, there exists $z_0 = |z(t)| < 1$ such that

$$|e_{ij}(t)| \leqslant \frac{1}{\xi(1 - z_0)}|z| \leqslant \frac{\eta + \sqrt{\eta^2+4\eta}}{2\xi(1 - z_0)}\exp\left(-\frac{\xi\gamma t}{2}\right)$$

which means the neural network (4) possesses the exponential convergence with rate $\xi\gamma/2$ for $e_{ij}(t) > -\ln 2/\xi$ when using the sigmoid activation function.

(iii) For the pth power activation function $f(u) = u^p$, (6) becomes $\dot{e}_{ij} = -\gamma e_{ij}^p$ and its general solution is

$$e_{ij}(t) = e_{ij}(0)\left\{(p-1)e_{ij}^{p-1}(0)\gamma t + 1\right\}^{-\frac{1}{p-1}}.$$

Specifically, when $p = 3$, the entry error $e_{ij}(t) = e_{ij}(0)/\sqrt{2e_{ij}^2(0)\gamma t + 1}$. Clearly, as $t \to \infty$, $e_{ij}(t) \to 0$. Review the Lyapunov function $v_{ij} = e_{ij}^2(t)$ and its time derivative $\dot{v}_{ij} = -\gamma e_{ij}f(e_{ij})$ in (7). For the error range $|e_{ij}| \gg 1$, we have $e_{ij}^{p+1} \gg e_{ij}^2 \gg |e_{ij}|$. In other words, the deceleration magnitude of power activation function is much greater than those of linear function and sigmoid function. This means when using power function, much faster convergence is achieved by the network for $|e_{ij}(t)| > 1$ in comparison with cases (i) and (ii). □

Remark 1: It follows from the above theorem that to achieve superior convergence, a high-performance neural network can be developed by switching power activation function to sigmoid or linear activation function at the switching points $|e_{ij}(t)| = 1$, $i,j \in \{1,\cdots,n\}$. For example, the following power-sigmoid activation function is preferable if the hardware permits

$$f(u) = \begin{cases} u^p, & \text{if } |u| \geqslant 1 \\ \frac{1+\exp(-\xi)}{1-\exp(-\xi)} \cdot \frac{1-\exp(-\xi u)}{1+\exp(-\xi u)}, & \text{otherwise} \end{cases} \tag{9}$$

with suitable design parameters ξ and p. One more advantage of using the sigmoid function over the linear function lies in the extra parameter ξ, which is a multiplier of the exponential convergence rate. When there is an upper bound on γ due to hardware implementation, the parameter ξ will be another effective factor expediting the network convergence. For example, the convergence for nonlinear activation functions could be much faster than that for linear functions when using the same level of ξ and p for the power and sigmoid activation function as that of design parameter γ for linear function (e.g., $10 \sim 100$).

Remark 2: Nonlinearity always exists, which is one of the main motivations for us to investigate different activation functions. Even if the linear activation function is used, the nonlinear phenomenon may appear in its hardware implementation; e.g., in the form of saturation and/or inconsistency of the linear slope, and in digital realization due to truncation and round-off errors [14], [23]. The investigation of different activation functions (like the sigmoid function and the power function) gives much insights into the problems and effects of imperfections/nonlinearities existing in implementing linear activation functions.

For graphical interpretation, the convergence characteristics of entry errors $e_{ij}(t)$ is illustrated in Fig. 3 for using different activation functions where $\gamma = 10^3$. Note that to draw all the curves in the same one plot, the minimal values of design parameters of nonlinear activation functions (like $\xi = 4$, $p = 3$, or $p = 5$) are used.

B. Robustness Analysis

In the realization of neural networks, there may be some errors involved. Regarding the linear network for solving time-varying Sylvester equation [21], robustness results were given for coefficient matrix perturbation only. For matrix-inversion,

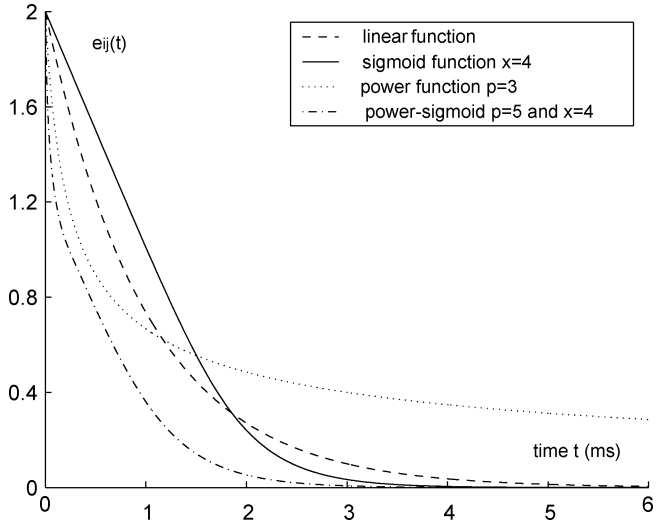


Fig. 3. Convergence characteristics of entry errors for different kinds of activation functions.

if $A(t)$ in (4) is perturbed with an additive term $\Delta_A(t)$ where $\|\Delta_A(t)\|_F \leqslant \varepsilon_1$ for any $t \in [0,\infty)$, then the following lemma can be similarly generalized from [21].

Lemma 2: Consider the perturbed RNN model $\hat{A}\dot{X} = -\hat{A}X - \gamma\mathcal{F}(\hat{A}X - I)$ with $\hat{A} = A + \Delta_A$. The steady-state residual error $\lim_{t\to\infty}\|X(t) - X^*(t)\|_F$ is uniformly upper bounded by $\varphi(\hat{\alpha}, \beta + \varepsilon_1, n) + \varphi(\alpha, \beta, n)$, provided that the invertibility condition (2) still holds with $\hat{\alpha} = (\sqrt{\alpha} - \varepsilon_1)^2$ instead of α for matrix $\hat{A}$.

Proof: It can be easily obtained by following the proof of Theorem 2 in [21] and taking into account the inequality $\sum_{i=1}^{n}|\lambda_i(A)|^2 \leqslant \|A\|_F^2$, Lemma 1, and Theorem 1 of this paper. □

Lemma 2 guarantees that the coefficient perturbation will not derail the neural network if the invertibility condition still holds true. However, such a result does not cover the differentiation error and the model-implementation error, which may appear more frequently than matrix perturbation does in neural network realization [23]. In the remainder of this subsection, the following dynamics are considered for the general robustness properties of the proposed neural system (4)

$$A\dot{X} = -(\dot{A} + \Delta_B)X(t) - \gamma\mathcal{F}(A(t)X(t) - I) + \Delta_C \quad (10)$$

where $\Delta_B(t) \in R^{n\times n}$ and $\Delta_C(t) \in R^{n\times n}$ denote respectively the differentiation error and the model-implementation error, which result from truncating/roundoff errors in digital realization or high-order residual errors of circuit components in analog realization.

Theorem 2: Consider the general RNN model with implementation errors Δ_B and Δ_C in (10). If $\|\Delta_B(t)\|_F \leqslant \varepsilon_2$ and $\|\Delta_C(t)\|_F \leqslant \varepsilon_3$ for any $t \in [0,\infty)$, then the computation error $\|X - X^*\|_F$ is bounded with steady-state residual error as $n\varphi(\varepsilon_3 + \varepsilon_2\varphi)/(\gamma\rho - \varepsilon_2\varphi)$ under the design-parameter requirement $\gamma > \varepsilon_2\varphi/\rho$, where the parameter $\rho > 0$ is defined between $f(e_{ij}(0))/e_{ij}(0)$ and $f'(0)$. Furthermore, as the design parameter γ tends to positive infinity, the steady-state residual error can be diminished to zero.

Proof: By defining the error matrix $E(t) = A(t)(X(t) - X^*(t))$ with $X^*(t) = A^{-1}(t)$, (10) is finally reformulated as $\dot{E} = -\gamma\mathcal{F}(E) - \Delta_B X^* E + (\Delta_C - \Delta_B X^*)$. This is equivalent to the following vector form [27]

$$\dot{e}(t) = -\gamma\mathcal{F}(e(t)) + B(t)e(t) + c(t) \quad (11)$$

where $e := \mathrm{vec}(E) \in R^{n^2\times 1}$ denotes a column vector obtained by stacking all column vectors of E together, and the activation-function array $\mathcal{F}$ is here of dimension $n^2 \times 1$ due to the vectorization. In addition, $B := I \otimes (-\Delta_B X^*) \in R^{n^2\times n^2}$ and $c := \mathrm{vec}(\Delta_C - \Delta_B X^*) \in R^{n^2\times 1}$ with the symbol $\otimes$ denoting the Kronecker product; i.e., $P \otimes Q$ is a large matrix made by replacing the ijth entry p_{ij} of P with the matrix $p_{ij}Q$. For detailed properties of Kronecker product, see [21], [27].

Define the Lyapunov function candidate $v = e^Te/2 = \sum_{i=1}^{n}\sum_{j=1}^{n}e_{ij}^2(t)/2 \geqslant 0$ for the error dynamics (11). The time derivative of v is

$$\begin{aligned}\frac{dv}{dt} &= e^T(t)\dot{e}(t)\\ &= e^T(t)\{-\gamma\mathcal{F}(e(t)) + B(t)e(t) + c(t)\}\\ &= -\gamma e^T\mathcal{F}(e) + e^TBe + e^Tc\\ &= -\gamma e^T\mathcal{F}(e) + e^T\frac{B+B^T}{2}e + e^Tc.\end{aligned}$$

It follows from the inequality $\sum_{i=1}^{n}|\lambda_i(A)|^2 \leqslant \|A\|_F^2$ and Lemma 1 that

$$\begin{aligned}e^T\frac{B+B^T}{2}e &\leqslant e^Te\max_{1\leqslant i\leqslant n^2}\left|\lambda_i\left(\frac{B+B^T}{2}\right)\right|\\ &= e^Te\max_{1\leqslant i\leqslant n^2}\left|\lambda_i\left(\frac{I\otimes\left(\Delta_BX^*+(\Delta_BX^*)^T\right)}{2}\right)\right|\\ &= e^Te\max_{1\leqslant i\leqslant n}\left|\lambda_i\left(\frac{\Delta_BX^*+(\Delta_BX^*)^T}{2}\right)\right|\\ &\leqslant e^Te\left\|\frac{\Delta_BX^*+(\Delta_BX^*)^T}{2}\right\|_F\\ &\leqslant e^Te\|\Delta_B\|_F\|X^*\|_F\\ &\leqslant e^Te\,\varepsilon_2\varphi.\end{aligned}$$

Similarly, it follows from $\|A\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2$ that $|c_i| \leqslant \|\Delta_C - \Delta_BX^*\|_F \leqslant \|\Delta_C\|_F + \|\Delta_BX^*\|_F = \varepsilon_3 + \varepsilon_2\varphi$, $\forall i \in \{1,\cdots,n^2\}$. Thus, $e^Tc \leqslant (\varepsilon_3 + \varepsilon_2\varphi)\sum_{i=1}^{n}\sum_{j=1}^{n}|e_{ij}|$. In view of the above facts and the symmetry property of $f(\cdot)$, we finally have

$$\begin{aligned}\frac{dv}{dt} &\leqslant -\gamma e^T\mathcal{F}(e) + (\varepsilon_2\varphi)e^Te + e^Tc\\ &= -\sum_{i=1}^{n}\sum_{j=1}^{n}|e_{ij}|(\gamma f(|e_{ij}|) - \varepsilon_2\varphi|e_{ij}| - \varepsilon_3 - \varepsilon_2\varphi). \quad (12)\end{aligned}$$

During the time evolution of $e_{ij}(t)$, the above equation falls into two situations: $\gamma f(|e_{ij}|) - \varepsilon_2\varphi|e_{ij}| - \varepsilon_3 - \varepsilon_2\varphi \geqslant 0$, $\forall i$, $j \in \{1,\cdots,n\}$ or $\gamma f(|e_{ij}|) - \varepsilon_2\varphi|e_{ij}| - \varepsilon_3 - \varepsilon_2\varphi < 0$, $\exists i$, $j \in \{1,\cdots,n\}$.

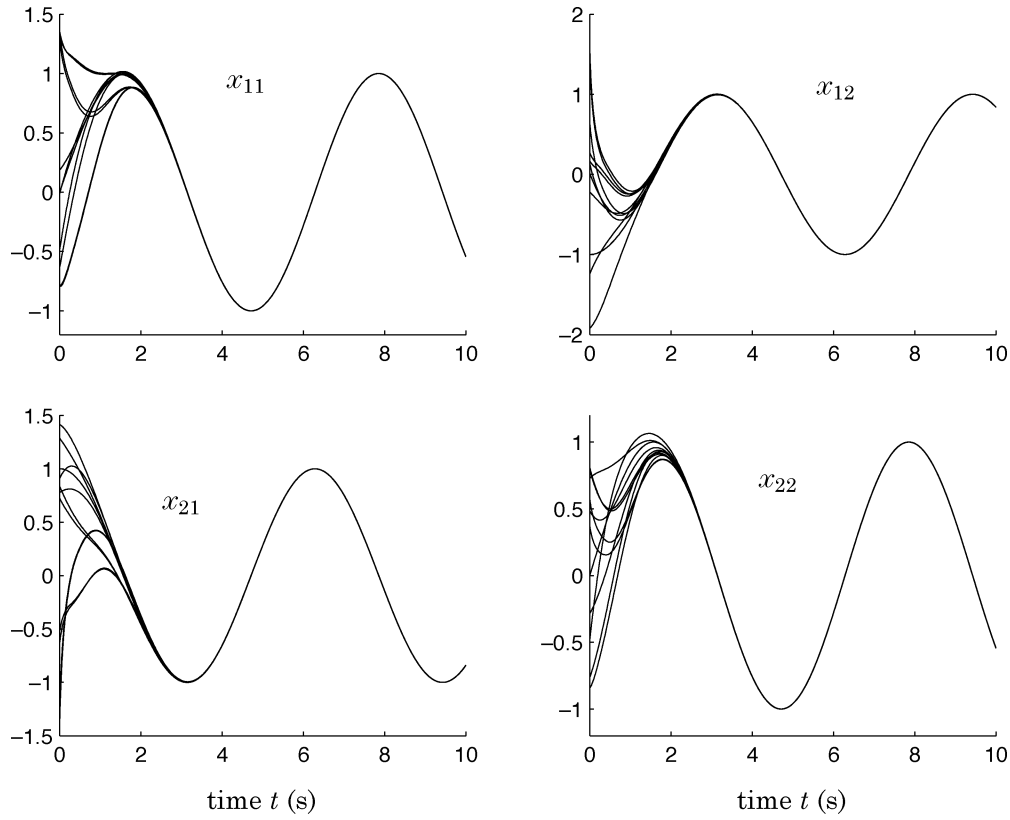


Fig. 4. Online inversion of $A(t)$ using power-sigmoid function (9).

If in the time interval $[t_0, t_1)$ the trajectory of the system (10) is in the first situation, $\dot{v} \leqslant 0$ and (12) implies $X(t)$ converges to $X^*(t)$ as time evolves.

For any time t that the trajectory falls into the second situation, the distance between $X(t)$ and $X^*(t)$ may not decrease again. But, even in the worst case, the entry error $|e_{ij}(t)|$ is also upper bounded by the steady-state entry residual error $\bar{e}_{ij} = (\varepsilon_3 + \varepsilon_2\varphi)/(\gamma\rho - \varepsilon_2\varphi)$ [24], where the insensitivity parameter $\rho > 0$ is defined between $f(e_{ij}(0))/e_{ij}(0)$ and $f'(0)$, and the design parameter γ is required as

$$\gamma > \frac{\varepsilon_2\varphi}{\rho}. \tag{13}$$

Thus, it follows that

$$\lim_{t\to\infty} \|X(t) - X^*(t)\|_F \leqslant \frac{n\varphi(\varepsilon_3 + \varepsilon_2\varphi)}{(\gamma\rho - \varepsilon_2\varphi)} \tag{14}$$

and evidently, this steady-state residual error caused by implementation errors can be made arbitrarily small as design parameter γ increases. □

Corollary 2.1: In addition to the general robustness results in Theorem 2, the imprecisely constructed general RNN model (10) possesses the following properties that

i) if linear activation function used, then the entry residual error $\bar{e}_{ij} = (\varepsilon_3 + \varepsilon_2\varphi)/(\gamma - \varepsilon_2\varphi)$ under the requirement $\gamma > \varepsilon_2\varphi$;
ii) if sigmoid activation function used, then the steady-state residual error can be made smaller by increasing γ or ξ, and superior robustness property exists for $\bar{e}_{ij} \leqslant \epsilon$, $\exists \epsilon > 0$, as compared to linear function; and
iii) if power activation function used, then the design-parameter requirement (13) always holds true for any positive γ and superior robustness property exists for $\bar{e}_{ij} > 1$, as compared to linear function.

Proof: It follows from (12) that $\forall i, j \in \{1, \cdots, n\}$

$$\gamma f(|e_{ij}|) - \varepsilon_2\varphi|e_{ij}| - \varepsilon_3 - \varepsilon_2\varphi \geqslant 0 \tag{15}$$

is a sufficient condition for ensuring $\dot{v} \leqslant 0$.

(i) For the linear activation function, the insensitivity parameter $\rho = f(|e_{ij}|)/|e_{ij}| \equiv 1$ and the design-parameter requirement on γ becomes $\gamma > \varepsilon_2\varphi$. From Theorem 2 and its proof, we have the steady-state entry residual error $\lim_{t\to\infty} |e_{ij}(t)| \leqslant (\varepsilon_3 + \varepsilon_2\varphi)/(\gamma - \varepsilon_2\varphi)$.

(ii) For the bipolar sigmoid function, as $|e_{ij}(t)| \to 0$, the value $f(e_{ij}(t))/e_{ij}(t)$ changes smoothly from $f(e_{ij}(0))/e_{ij}(0)$ to ξ. In view of $f(e_{ij}(0))/e_{ij}(0) < 1$ and $\xi > 1$, there exists $t_m(\xi)$ such that $f(e_{ij}(t_m))/e_{ij}(t_m) = 1$. Denoting $|e_{ij}(t_m)|$ as $\epsilon(\xi)$, we have the insensitivity parameter $\rho(\xi) > 1$ for the error range $|e_{ij}(t)| \leqslant \epsilon(\xi)$, which, compared to the linear case $\rho \equiv 1$, means easier satisfaction of (15) and a smaller steady-state residual error in (14), i.e., $\bar{e}_{ij} = (\varepsilon_3 + \varepsilon_2\varphi)/(\gamma\rho(\xi) - \varepsilon_2\varphi)$. In addition, the design-parameter requirement (13) is also relaxed by a factor of $\rho(\xi)$. Especially, when the implementation error $\varepsilon_3 + \varepsilon_2\varphi$ is small (e.g., when the steady-state entry residual error is near zero), the insensitivity parameter ρ is evaluated as $f'(0) = \xi$, which means much superior robustness property to the linear/power activation function cases.

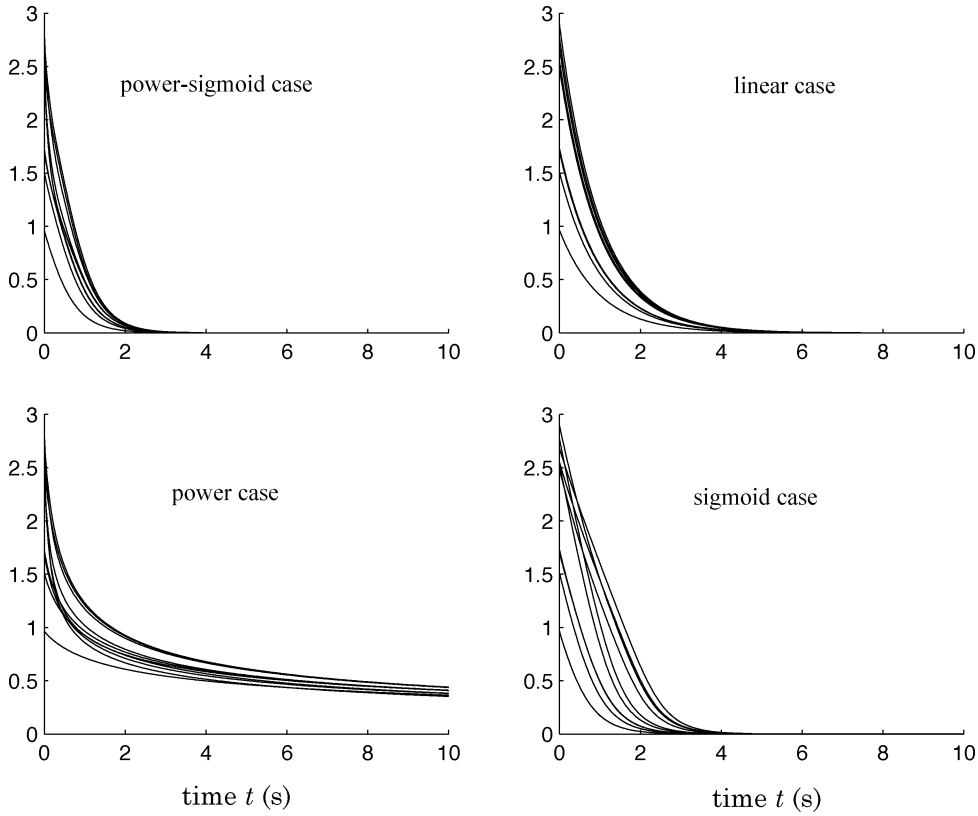


Fig. 5. Convergence comparison of $\|X(t) - A^{-1}(t)\|_F$ using different activation functions.

(iii) For the power-function case, (15) becomes $(\gamma|e_{ij}|^{p-1} - \varepsilon_2\varphi)|e_{ij}| - \varepsilon_3 - \varepsilon_2\varphi \geqslant 0$. Clearly, there always exists $\zeta > 1$ such that both the previous equation and $\gamma|e_{ij}|^{p-1} - \varepsilon_2\varphi > 0$ hold true for $|e_{ij}(t)| \geqslant \zeta$. Thus, the design-parameter inequality (13) always holds and such a requirement can be removed in this case. In the situation that the differentiation/implementation error $\varepsilon_3+\varepsilon_2\varphi$ is so large that $\bar{e}_{ij} > 1$, the insensitivity parameter $\rho = e_{ij}^{p-1} > e_{ij}^2 > 1$. This means easier satisfaction of (15) and smaller steady-state residual error (14), as compared to linear or sigmoid case under the same design specification. □

Remark 3: From the above theoretical analysis, for large matrix-inversion error, using the power activation function has much better convergence and robustness than using the linear function. This is because for large entry error (e.g., $|e_{ij}| > 1$), the power activation function could amplify the signal ($e_{ij}^{p-1} > \cdots > e_{ij}^2 > 1$), automatically eliminate the insensitivity condition (13), and also expedite the network convergence. On the other hand, for small matrix-inversion error, using a sigmoid activation function has much better convergence and robustness than using the linear function. This is because of the larger slope of the sigmoid function near the origin, which implies the stronger insenstivity/robustness of the sigmoid-based neural model (4), as compared to the linear ones. It follows from Remark 1 and the above analysis that for superior convergence and robustness, the power-sigmoid activation function in (9) might be a better choice than other functions. Details about activation functions implementation can be found in [4]–[6], [10]–[18], [23], [24].

V. Illustrative Example

For illustration and comparison, let us consider the same time-varying matrix as in [21]:

$$A(t)=\begin{bmatrix}\sin t & \cos t\\ -\cos t & \sin t\end{bmatrix},\quad A^{-1}(t)=A^T=\begin{bmatrix}\sin t & -\cos t\\ \cos t & \sin t\end{bmatrix}.$$

The RNN (4) is thus in this specific form

$$\begin{aligned}&\begin{bmatrix}\sin t & \cos t\\ -\cos t & \sin t\end{bmatrix}\begin{bmatrix}\dot{x}_{11} & \dot{x}_{12}\\ \dot{x}_{21} & \dot{x}_{22}\end{bmatrix}\\ &\quad=-\begin{bmatrix}\cos t & -\sin t\\ \sin t & \cos t\end{bmatrix}\begin{bmatrix}x_{11} & x_{12}\\ x_{21} & x_{22}\end{bmatrix}\\ &\qquad-\gamma\mathcal{F}\left(\begin{bmatrix}\sin t & \cos t\\ -\cos t & \sin t\end{bmatrix}\begin{bmatrix}x_{11} & x_{12}\\ x_{21} & x_{22}\end{bmatrix}-\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)\end{aligned}$$

where the array $\mathcal{F}$ is constituted by the power-sigmoid activation function as in (9) with $\xi = 4$ and $p = 3$, and the design parameter $\gamma := 1$ for illustration purposes.

A. Convergence Discussion

As seen from Figs. 4 and 5, starting from any initial states randomly selected in $[-2,2]$, state matrices of the proposed neural model all converge to the theoretical inverse $A^{-1}(t)$. The convergence can be expedited by increasing γ. For example, if γ is increased to 10^6, the convergence time is within 6 μs. Note that, for the case of using pure power activation function shown in the

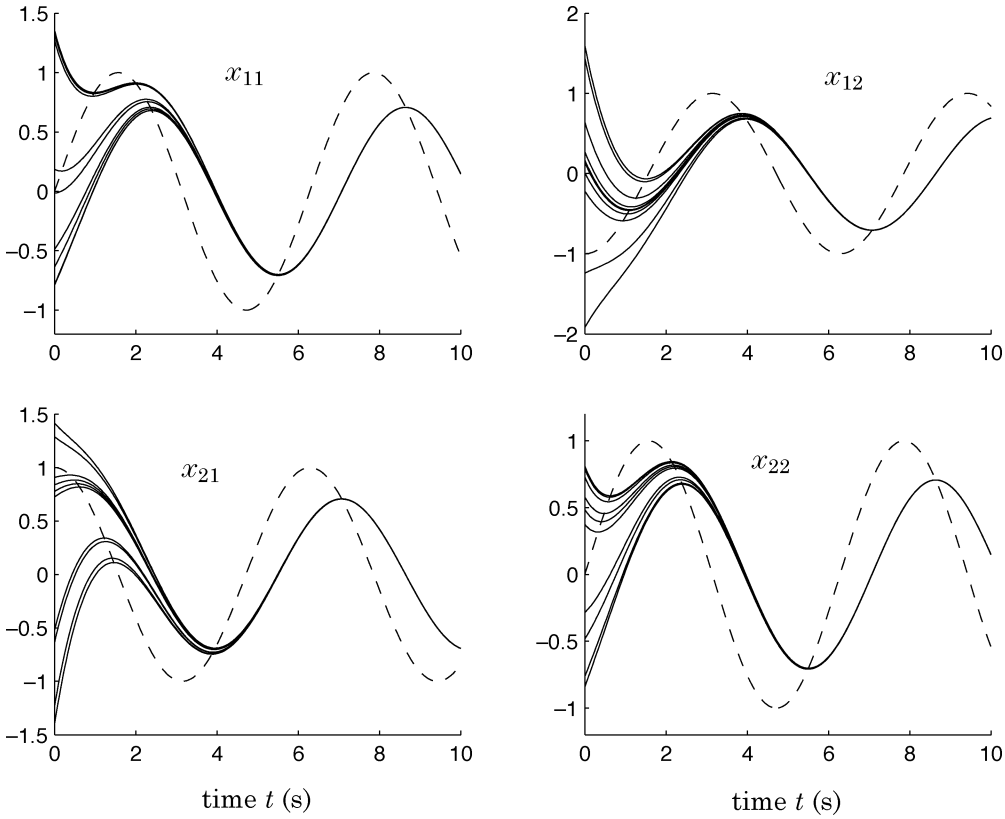


Fig. 6. Performance of gradient-based neural network methods, where dashed lines denote entries of the exact inverse.

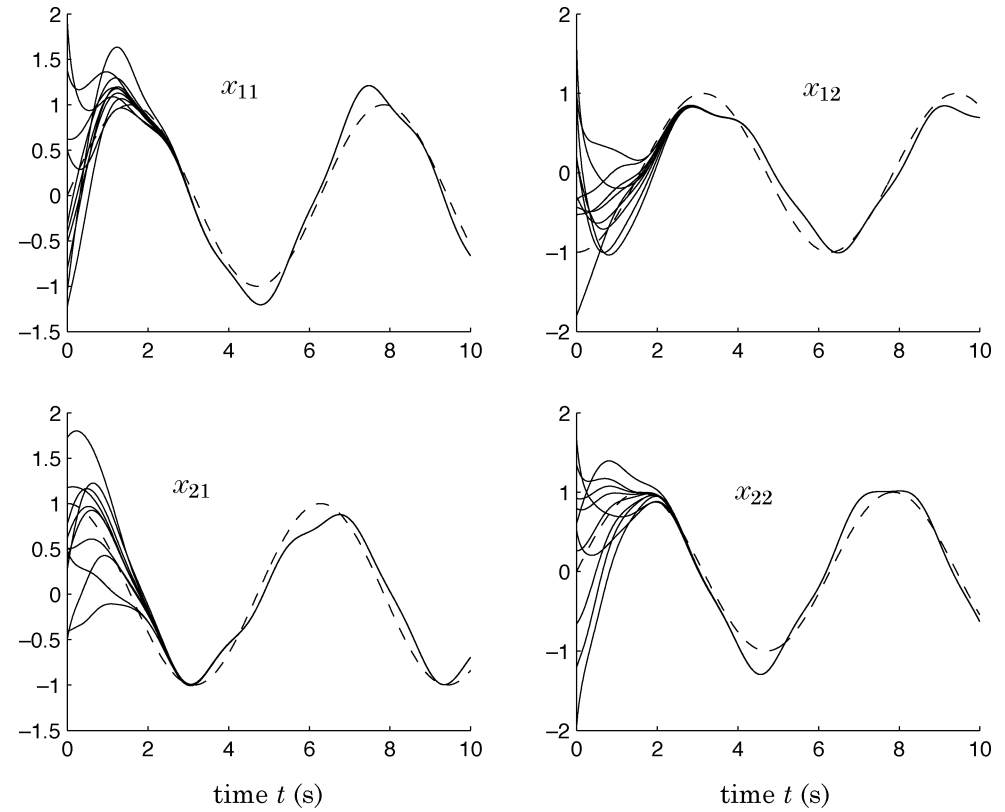


Fig. 7. Online inversion of $A(t)$ with $\gamma = 1$ and using power-sigmoid activation function in the presence of differentiation and model-implementation errors.

third subplot of Fig. 5, the convergence is only asymptotic and thus apparently slower than that for other cases which possess exponential convergence.

For comparison with traditional neural approaches, the gradient-based neural network [17], [21], $\dot{X}(t) = -\gamma A^T(t)A(t)X(t) + \gamma A^T(t)$, is also simulated for

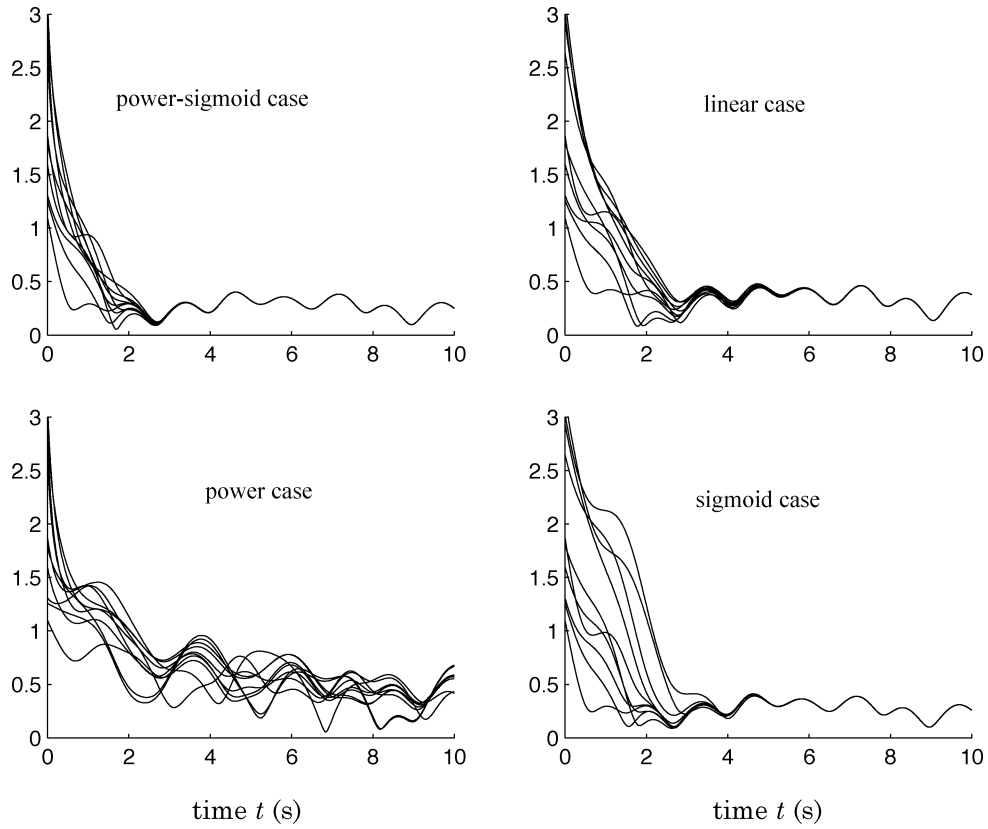


Fig. 8. Computing errors $\|X(t) - A^{-1}(t)\|_F$ with $\gamma = 1$ in the presence of differentiation and model-implementation errors.

time-varying matrix inversion. Its performance is depicted in Fig. 6 under the same design parameters and initial states as in Fig. 4. Clearly, the steady-state error of the solution computed by the traditional neural model is considerably large. This is because the $\dot{A}(t)$ information has not been utilized in the traditional gradient-based scheme.

B. Robustness Discussion

To show the robustness characteristics of the proposed neural model (4), the following differentiation error and model-implementation error are considered in a higher-frequency sinusoidal form

$$\Delta_B(t) = \varepsilon_2 \begin{bmatrix} \cos 3t & -\sin 3t \\ \sin 3t & \cos 3t \end{bmatrix}, \quad \Delta_C(t) = \varepsilon_3 \begin{bmatrix} \sin 3t & 0 \\ 0 & \cos 3t \end{bmatrix}$$

with $\varepsilon_2 = \varepsilon_3 = 0.5$.

As can be seen from Figs. 7 and 8, even with large differentiation and implementation errors, the computing error $\|X(t) - A^{-1}(t)\|_F$ synthesized by the neural model (4) is bounded and very small, and using power-sigmoid function or sigmoid function has smaller steady-state residual error than using linear or pure power function. Moreover, as the design parameter γ increases from 1 to 10, the convergence is expedited and the steady-state computation error is decreased, which is shown in Figs. 9 and 10. It is observed from other simulation data that when using power-sigmoid activation function, the maximum steady-state residual error is only 6×10^{-3} and 6×10^{-4} respectively for $\gamma = 100$ and $\gamma = 1000$. Clearly, compared to linear or pure power function case, superior performance can be achieved by using power-sigmoid or sigmoid activation functions under the same design specification. These simulation results have confirmed the theoretical analysis presented in the previous sections.

VI. Application to Robot Kinematic Control

This section presents the application of the proposed neural model (4) to kinematic control of redundant manipulators via online solution of time-varying pseudoinverse.

A. Preliminaries on Inverse Kinematics

Consider a redundant manipulator of which the end-effector position/orientation vector $r(t) \in R^m$ in Cartesian space is related to the joint-space vector $\theta(t) \in R^n$ through the following forward kinematic equation

$$r(t) = \phi(\theta(t)) \tag{16}$$

where $\phi(\cdot)$ is a continuous nonlinear mapping function with a known structure and parameters for any given manipulator. The inverse kinematics problem is to find the joint variable $\theta(t)$ for any given $r(t)$ through the inverse mapping of (16), i.e., $\theta(t) = \phi^{-1}(r(t))$.

Unfortunately, it is usually impossible to find an analytic solution of ϕ^{-1} due to the serious nonlinearity. The inverse kinematics problem is thus usually solved at the velocity level. Differentiating (16) with respect to time yields a linear relation between velocities $\dot{r}$ and $\dot{\theta}$:

$$J(\theta(t))\,\dot{\theta}(t) = \dot{r}(t) \tag{17}$$

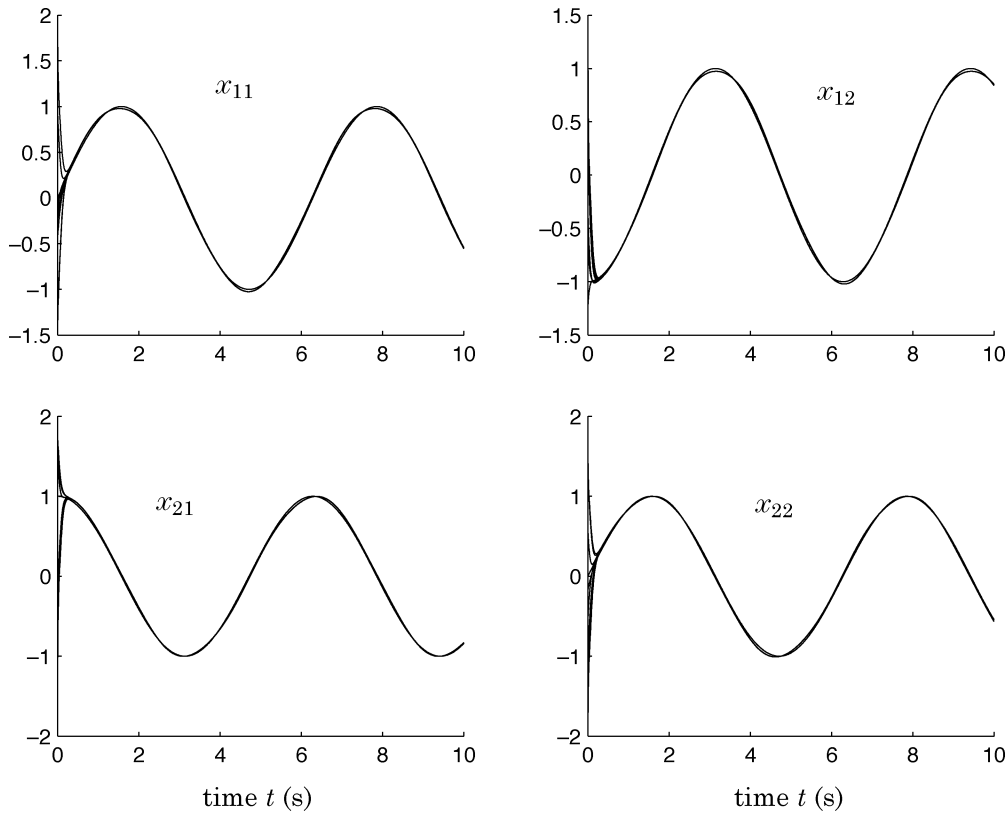


Fig. 9. Online inversion of $A(t)$ with $\gamma = 10$ and using power-sigmoid activation function in the presence of differentiation and model-implementation errors.

where $J(\theta) \in R^{m\times n}$ is the Jacobian matrix defined as $J(\theta) = \partial\phi(\theta)/\partial\theta$. Since $m < n$ in a redundant manipulator, (17) is underdetermined and may admit an infinite number of solutions.

The pseudoinverse/nullspace-type solution to (17), widely used by most of the current researchers, is generally formulated as a minimum-norm particular solution plus a homogeneous solution [25], [28]

$$\dot{\theta}(t) = J^{+}(t)\dot{r}(t) + \left(I - J^{+}(t)J(t)\right) z(t) \tag{18}$$

where $J^{+}(t) \in R^{n\times m}$ denotes the pseudoinverse of $J(t)$. The vector $z(t) \in R^{n}$ is arbitrary and can be chosen as the negative gradient of a performance index to be minimized, e.g., the optimization criteria of avoiding joint limits, singularity, and/or obstacles.

In the sense of Moore-Penrose generalized inverse, J^{+} is defined as $J^{T}(JJ^{T})^{-1}$ if J is of full row rank [27]. However, like solving matrix inverse, the usual numerical ways for solving $J^{+}(t)$ are in general computationally intensive and with large relative computational error because of no information of $\dot{J}(t)$ considered. In addition, with multiple tasks and constraints included by (18) via z, the heavy-burden J^{+}-computing procedure may hinder on-line applications, especially in high-DOF sensor-based robotic systems [28], [29].

Defining $A(t) = J(t)J^{T}(t)$, we could re-exploit the neural model (4) to solve $J^{+}(t)$ in the parallel manner so as to expedite the computation process and achieve better control precision. That is, $J^{+}(t) = J^{T}(t)X(t)$ where $X(t)$ is the state matrix generated online by the proposed RNN (4).

B. Simulation Based on PUMA560 Robot Arm

The Unimation PUMA560 robot arm has six joints [30]. When we consider only the positioning of the end-effector, the PUMA560 becomes a redundant manipulator with the dimensionality of J being 3×6 [31].

In this subsection, the proposed neural model (4) is first applied to the PUMA560 robot arm with the design parameter $\gamma = 10^{3}$ and $\dot{A}(t) = \dot{J}(t)J^{T}(t) + J(t)\dot{J}^{T}(t)$. The desired motion of the end-effector is a circle of radius $r = 10$ cm and with the revolute angle about x-axis being $-\pi/6$. The motion duration is 10 seconds, and the initial joint variables $\theta(0) = [0, 0, 0, 0, 0, 0]^{T}$ in radians. Fig. 11(a) illustrates the simulated motion of the robot arm in the 3-dimensional work space, which is sufficiently close to the desired one. Specifically, as shown in Fig. 12, the maximal Cartesian position and velocity tracking errors at the end-effector are respectively less than 3×10^{-7} m and 7×10^{-9} m/s. The corresponding joint variables and joint velocities are depicted in the subplots (b) and (c) of Fig. 11. The network states $X(t)$ of the proposed neural model (4) is shown in Fig. 13.

For comparison, the traditional gradient-based neural network, mentioned in Section V-A, is also simulated for solving $J^{+}(t)$ online. Under the same design condition $\gamma = 10^{3}$, the end-effector position and velocity errors in this case are considerably large (i.e., $\geqslant 1.0$ cm and 1.5 cm/s, respectively). In addition, to achieve the similar precision of the proposed neural model, the traditional gradient-based neural network requires $\gamma \geqslant 10^{9}$, which is a very stringent restriction on system design, either analog or digital. To facilitate the comparison

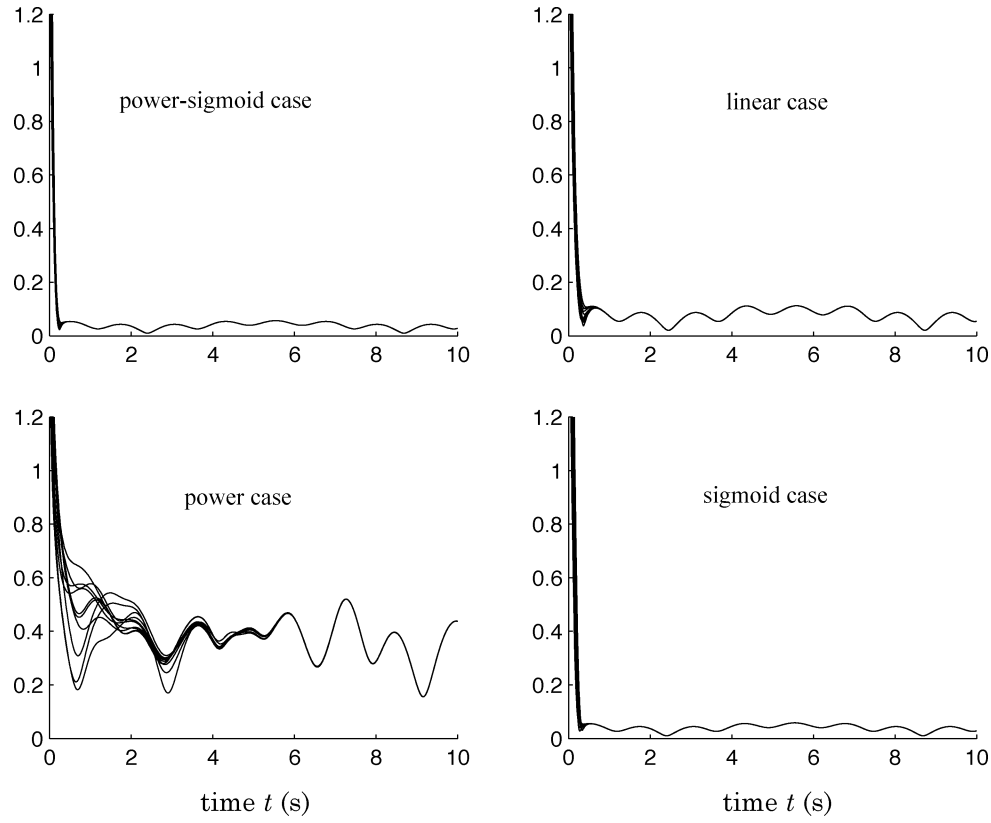


Fig. 10. Computing errors $\|X(t) - A^{-1}(t)\|_F$ with $\gamma = 10$ in the presence of differentiation and model-implementation errors.

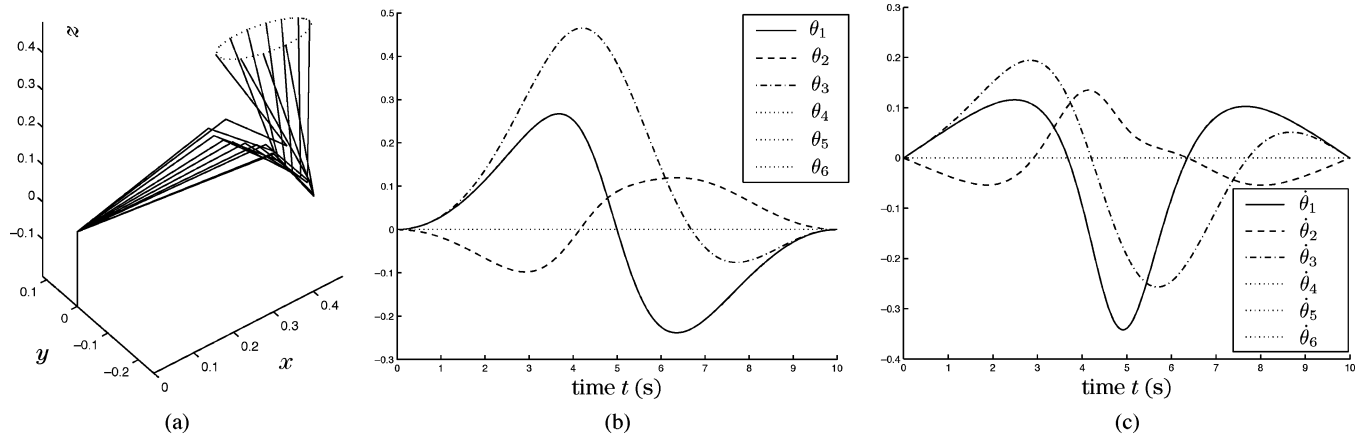
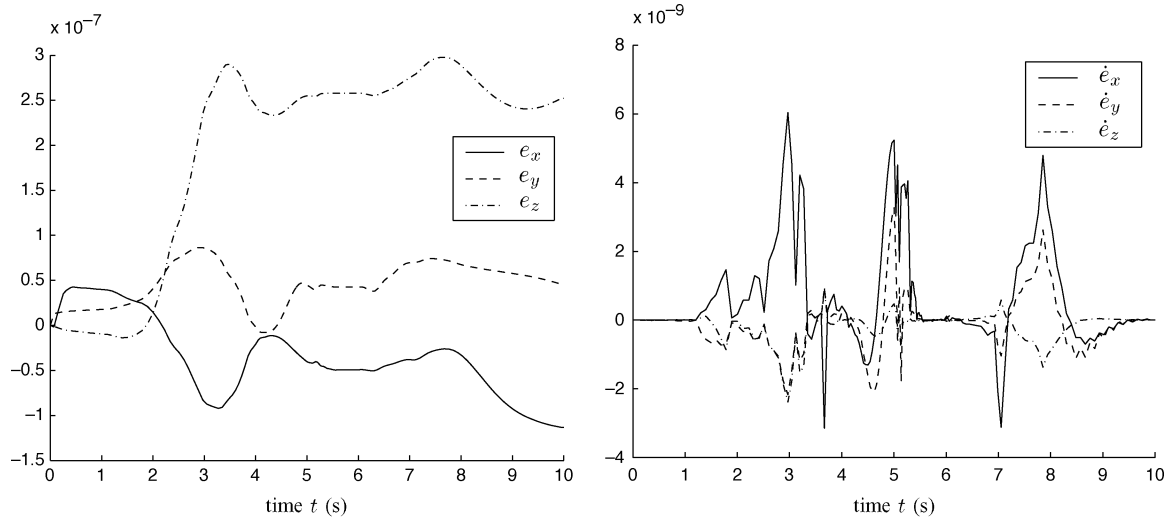


Fig. 11. Motion trajectories of the PUMA560 manipulator synthesized by the proposed neural model (4). (a) Simulated motion. (b) Joint variables in rad. (c) Joint velocities in rad/s.

with the proposed model, the states $X(t)$ of the traditional gradient-based neural network are also depicted, i.e., in Fig. 14. Clearly, in the solution generated by traditional neural network, the state x_{ij} does not equals x_{ji} violating the symmetric property of $(JJ^T)^{-1}$, and the settling time-period from initial states to steady states is relatively long (about 0.6 s). These lead to the considerably larger Cartesian positioning error at the PUMA560 end-effector as in Fig. 15. The reason is also that the $\dot{A}(t)$ information has not been utilized in the traditional gradient-based neural approach.

The above simulation results based on PUMA560 robot arm substantiate the advantages of the proposed neural model (4) over traditional gradient-based approaches in terms of computational accuracy and convergence rate. Before ending this numerical study, it is also worth mentioning the availability and acquisition of the derivative information, $\dot{A}(t)$. In some situations such as in Sections V and VI, the analytical form of $\dot{A}(t)$ could be given directly or derived from $A(t)$. If unknown, $\dot{A}(t)$ could be measured from $A(t)$ by using analog differentiation circuits or finite-difference technique in digital realization [14], [23]. The possible $\dot{A}(t)$-measuring error (termed differentiation error in Section IV-B) is actually a motivation for us to provide the robustness analysis. Theorem 2 shows that even in the presence of the $\dot{A}(t)$-measuring error, the inverse-computing error is still bounded and could be diminished to zero as design parameter γ tends to positive infinity.



(a) (b)

Fig. 12. End-effector errors of the PUMA560 manipulator tracking a 10-cm-radius circle. (a) Position error in m. (b) Velocity error in m/s.

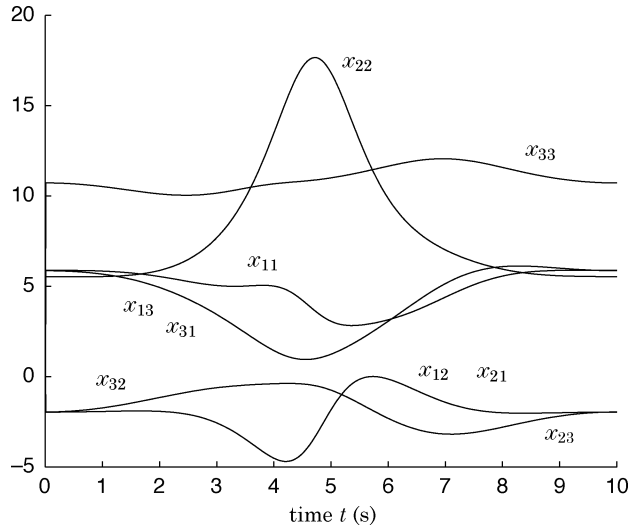


Fig. 13. States of the proposed neural model (4) for kinematic control of the PUMA560, where $x_{ij} = x_{ji}$ shows the symmetry.

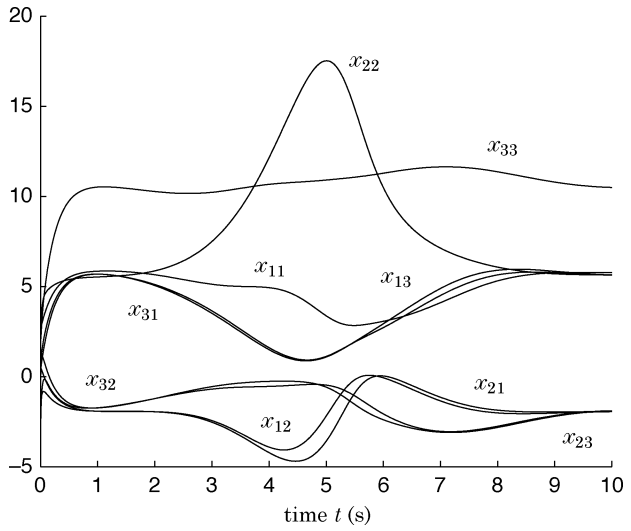


Fig. 14. States of the traditional gradient-based neural network for kinematic control of the PUMA560.

VII. Concluding Remarks

A general RNN model with implicit dynamics has been presented in this paper for solving the inverse of time-varying matrix in real time. Different from the traditional gradient-based methods used in time-varying cases, the proposed neural approach fully and methodically utilizes the time-derivative information of the matrix to be inverted. It is thus able to guarantee the global exponential convergence of the proposed neural model to the exact inverse of such a given time-varying matrix. Moreover, it has been shown that superior convergence and robustness can be achieved by using sigmoid or power-sigmoid activation functions. Simulation results including kinematic control of the PUMA560 manipulator have demonstrated the effectiveness and efficiency of the proposed RNN model. Further efforts are to be directed at the design and analysis of discrete-time neural networks, numerical algorithms, and electronic circuits for solving the inverse of time-varying matrix.

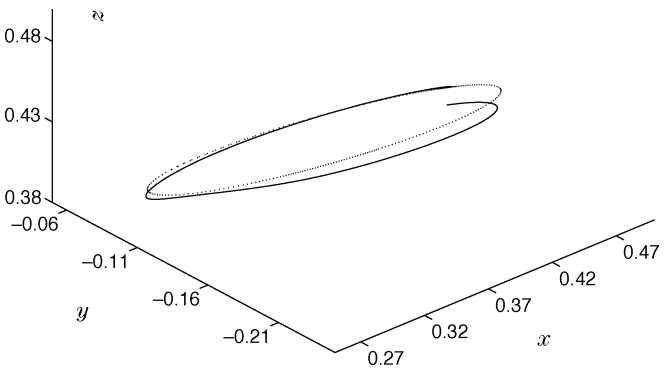


Fig. 15. End-effector motion trajectory of the PUMA560 synthesized by the traditional gradient-based neural network, where the dotted curve denotes the desired circular path.

APPENDIX

Proof of Lemma 1

By the Cayley-Hamilton Theorem [27], matrix A satisfies its characteristic polynomial

$$A^n + \omega_{n-1}A^{n-1} + \cdots + \omega_{n-i}A^{n-i} + \cdots + \omega_0 I = 0 \quad (19)$$

where ω_i are coefficients defined below in terms of the eigenvalues of A

$$\begin{cases} \omega_{n-1} = \sum_{j_1=1}^{n} (-\lambda_{j_1}), \\ \cdots \\ \omega_{n-i} = \sum_{j_i>\cdots>j_2>j_1=1}^{n} (-\lambda_{j_1})(-\lambda_{j_2})\cdots(-\lambda_{j_i}), \\ \cdots \\ \omega_0 = (-\lambda_1)(-\lambda_2)\cdots(-\lambda_n). \end{cases}$$

Specifically, $\omega_0 \neq 0$ and $|\omega_0| \geqslant \alpha^n$ due to the invertibility condition (2). It follows from (19) that

$$A^{-1} = -(\upsilon_n A^{n-1} + \upsilon_{n-1}A^{n-2} + \cdots + \upsilon_{n-i}A^{n-i-1} + \cdots + \upsilon_1 I) \quad (20)$$

where the coefficients υ_j are

$$\begin{cases} \upsilon_n = \frac{1}{\omega_0} = \frac{1}{(-\lambda_1)(-\lambda_2)\cdots(-\lambda_n)}, \\ \upsilon_{n-1} = \frac{\omega_{n-1}}{\omega_0} = \sum_{j_1=1}^{n} \frac{1}{(-\lambda_1)\cdots(-\lambda_{j_1-1})(-\lambda_{j_1+1})\cdots(-\lambda_n)} \\ \cdots \\ \upsilon_{n-i} = \frac{\omega_{n-i}}{\omega_0} = \sum_{j_i>\cdots>j_2>j_1=1}^{n} \frac{(-\lambda_{j_1})(-\lambda_{j_2})\cdots(-\lambda_{j_i})}{(-\lambda_1)(-\lambda_2)\cdots(-\lambda_n)} \\ \cdots \\ \upsilon_1 = \frac{\omega_1}{\omega_0} = \sum_{j_1=1}^{n} \frac{1}{(-\lambda_{j_1})}. \end{cases}$$

In view of (2), by defining the operator $C_n^i = n!/(i!(n-i)!)$ for any $i \in \{1, \cdots, n\}$, we have

$$\begin{cases} |\upsilon_n| \leqslant \frac{1}{\alpha^n}, \\ |\upsilon_{n-1}| \leqslant \frac{n}{\alpha^{n-1}}, \\ \cdots \\ |\upsilon_{n-i}| \leqslant \frac{C_n^i}{\alpha^{n-i}}, \\ \cdots \\ |\upsilon_1| \leqslant \frac{n}{\alpha}. \end{cases}$$

It follows from the above inequalities and (20) that

$$\begin{aligned} \|A^{-1}\|_F &\leqslant |\upsilon_n|\|A\|_F^{n-1} + |\upsilon_{n-1}|\|A\|_F^{n-2} + \cdots \\ &\quad + |\upsilon_{n-i}|\|A\|_F^{n-i-1} + \cdots + |\upsilon_1|\|I\|_F \\ &\leqslant \frac{\beta^{n-1}}{\alpha^n} + \frac{n\beta^{n-2}}{\alpha^{n-1}} + \cdots + \frac{C_n^i\beta^{n-i-1}}{\alpha^{n-i}} + \cdots + \frac{n\sqrt{n}}{\alpha} \\ &= \frac{\sum_{i=0}^{n-2} C_n^i \beta^{n-i-1}}{\alpha^{n-i}} + \frac{n^{\frac{3}{2}}}{\alpha} \end{aligned}$$

which completes the proof of Lemma 1. □

REFERENCES

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming—Theory and Algorithms*. New York: Wiley, 1993.

[2] R. J. Steriti and M. A. Fiddy, "Regularized image reconstruction using SVD and a neural network method for matrix inversion," *IEEE Trans. Signal Process.*, vol. 41, no. 10, pp. 3074–3077, 1993.

[3] T. Sarkar, K. Siarkiewicz, and R. Stratton, "Survey of numerical methods for solution of large systems of linear equations for electromagnetic field problems," *IEEE Trans. Antennas Propag.*, vol. 29, no. 6, pp. 847–856, 1981.

[4] R. H. Sturges Jr, "Analog matrix inversion (robot kinematics)," *IEEE J. Robot. Automat.*, vol. 4, no. 2, pp. 157–162, 1988.

[5] K. S. Yeung and F. Kumbi, "Symbolic matrix inversion with application to electronic circuits," *IEEE Trans. Circuits Syst.*, vol. 35, no. 2, pp. 235–238, 1988.

[6] A. El-Amawy, "A systolic architecture for fast dense matrix inversion," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 449–455, 1989.

[7] V.-E. Neagoe, "Inversion of the Van der Monde matrix," *IEEE Signal Process. Lett.*, vol. 3, no. 4, pp. 119–120, 1996.

[8] Y. Q. Wang and H. B. Gooi, "New ordering methods for space matrix inversion via diagonaliztion," *IEEE Trans. Power Syst.*, vol. 12, no. 3, pp. 1298–1305, 1997.

[9] C. K. Koc and G. Chen, "Inversion of all principal submatrices of a matrix," *IEEE Trans. Aerosp. Electr. Syst.*, vol. 30, no. 1, pp. 280–281, 1994.

[10] R. K. Manherz, B. W. Jordan, and S. L. Hakimi, "Analog methods for computation of the generalized inverse," *IEEE Trans. Autom. Contol*, vol. 13, no. 5, pp. 582–585, 1968.

[11] N. C. F. Carneiro and L. P. Caloba, "A new algorithm for analog matrix inversion," in *Proc. 38th Midwest Symp. Circuits Syst.*, vol. 1, 1995, pp. 401–404.

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

[13] J. Jang, S. Lee, and S. Shin, "An optimization network for matrix inversion," in *Neural Inform. Process. Comput.*. New York: Amer. Inst. of Phys., 1988, pp. 397–401.

[14] J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*. Cambridge: The MIT Press, 1988.

[15] F. L. Luo and B. Zheng, "Neural network approach to computing matrix inversion," *Appl. Math. Comput.*, vol. 47, pp. 109–120, 1992.

[16] A. Cichocki and R. Unbehauen, "Neural network for solving systems of linear equations and related problems," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 124–138, 1992.

[17] J. Wang, "A recurrent neural network for real-time matrix inversion," *Applied Math. Comput.*, vol. 55, pp. 89–100, 1993.

[18] ——, "Recurrent neural networks for computing pseudoinverses of rank-deficient matrices," *SIAM J. Sci. Comput.*, vol. 18, pp. 1479–1493, 1997.

[19] J. Song and Y. Yam, "Complex recurrent neural network for computing the inverse and pseudo-inverse of the complex matrix," *Applied Math. Comput.*, vol. 93, pp. 195–205, 1998.

[20] S. S. Ge and C. C. Hang, "Structural network modeling and control of rigid body robots," *IEEE Trans. Robot. Autom.*, vol. 14, no. 5, pp. 823–827, 1998.

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

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

[23] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[24] S. Poznyak, E. N. Sanchez, and W. Yu, *Differential Neural Networks for Robust Nonlinear Control (Identification, State Estimation and Trajectory Tracking)*. Singapore: World Scientific, 2001.

[25] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*. London, U.K.: World Scientific, 1998.

[26] Y. Zhang, P. A. Heng, and A. W. C. Fu, "Estimate of exponential convergence rate and exponential stability for neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1487–1493, 1999.

[27] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*. Cambridge: Cambridge University Press, 1991.

[28] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. London, U.K.: Springer-Verlag, 2000.

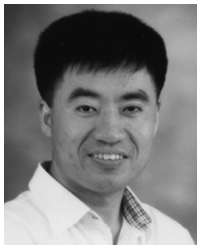
[29] Y. Zhang, J. Wang, and Y. Xu, "A dual neural network for bi-criteria kinematic control of redundant manipulators," *IEEE Trans. Robot. Automat.*, vol. 18, no. 6, pp. 923–931, 2002.

[30] P. I. Corke and B. Armstrong-Helouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, 1994, pp. 1608–1613.

[31] Y. Zhang and J. Wang, "A dual neural network for constrained joint torque optimization of kinematically redundant manipulators," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 32, no. 5, pp. 654–662, 2002.

Yunong Zhang (S'02–M'03) received the B.E. degree from the Huangzhong University of Science and Technology (HUST), China, in 1996, the M.E. degree from the South China University of Technology (SCUT) in 1999, and the Ph.D. degree from the Chinese University of Hong Kong (CUHK) in 2002.

He was a Research Fellow with both the National University of Singapore (NUS) and the University of Strathclyde, U.K., in 2003 and 2004, respectively. Since 2005, he has been with the National University of Ireland, Maynooth, as a Research Scientist. His current research interests are recurrent neural networks and their hardware/circuits implementation, redundant robot manipulators and the related biomechanics research, and scientific computing and optimization such as Gaussian process regression.



Shuzhi Sam Ge (S'90–M'92–SM'00) received the B.Sc. degree from Beijing University of Aeronautics and Astronautics (BUAA), China, in 1986, and the Ph.D. degree and the Diploma of Imperial College (DIC) from Imperial College of Science, Technology and Medicine, in 1993.

He has been with the Department of Electrical and Computer Engineering, the National University of Singapore since 1993, where he is now a Full Professor. He has authored and coauthored more than 200 international journal and conference papers, three monographs, and coinvented three patents. His current research interests are control of nonlinear systems, neural/fuzzy systems, robotics, hybrid systems, sensor fusion, and system development. He serves as a technical consultant for local industry.

Dr. Ge has served as a Member of the Technical Committee on Intelligent Control since 2000. In 2004, he was a Member of the Board of Governors (BOGs), IEEE Control Systems Society. He served as an Associate Editor on the Conference Editorial Board of the IEEE Control Systems Society in 1998 and 1999. He currently serves as Editor of the *International Journal of Control, Automation, and Systems*, Corresponding Editor for Asia and Australia of the IEEE *Control Systems Magazine*, and an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS, the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, *The Automatica*, and the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.