Comparison on Zhang Neural Network and Gradient Neural Network for Time-Varying Linear Matrix Equation $AXB = C$ Solving

Yunong Zhang$^1$ and Ke Chen$^2$

$^1$School of Information Science and Technology
$^2$Software School
Sun Yat-Sen University, Guangzhou 510275, China
Email: ynzhang@ieee.org

Abstract—For solving online the linear matrix equation $AXB = C$ with time-varying coefficients, this paper presents a special kind of recurrent neural networks by using a design method recently proposed by Zhang et al. Compared with gradient neural networks (abbreviated as GNN, or termed as gradient-based neural networks), the resultant Zhang neural network (termed as such and abbreviated as ZNN hereafter for presentation convenience) is designed based on a matrix-valued error function, instead of a scalar-valued error function. Zhang neural network is deliberately developed in the way that its trajectory could be guaranteed to globally exponentially converge to the time-varying theoretical solution of given linear matrix equation. In addition, Zhang neural network is described by an implicit dynamics, instead of an explicit dynamics usually describing recurrent neural networks. Convergence results of Zhang neural network are presented to show the neural-network performance. In comparison, we develop and simulate the gradient neural network as well, which is exploited to solve online the time-varying linear matrix equation. Computer-simulation results substantiate the theoretical efficacy and superior performance of Zhang neural network for the online solution of time-varying linear matrix equation, especially when using a power-sigmoid activation function.

Index Terms—Recurrent neural network; Gradient neural network; Time-varying linear matrix equation; Matrix-valued error function; Implicit dynamics.

I. INTRODUCTION

The problem of linear matrix equations solving (including matrix-inverse problems as a sub-topic) is considered to be a very fundamental problem widely encountered in science and engineering. It could usually be an essential part of many solutions; e.g., in control system design [1][2] and image-processing [3]. In view of these, we consider in this paper the following general problem formulation of linear matrix equation: $AXB = C$, where coefficient matrices $A \in R^{m \times m}$, $B \in R^{n \times n}$ and $C \in R^{m \times n}$, while $X \in R^{m \times n}$ is the unknown matrix to be found. Evidently, when $B = C = I$ and $m = n$, the problem reduces to the matrix-inversion problem.

There are two general types of solutions to the problem of linear matrix equations. One is the numerical algorithms performed on digital computers (i.e., on our today’s computers). Usually, such numerical algorithms are of serial-processing nature and may not be efficient enough for large-scale online or real-time applications. Being the second general type of solution, many parallel-processing computational methods have been developed, analyzed, and implemented on specific architectures [3]-[13].

The dynamic-system approach is one of such important parallel-processing methods for solving linear matrix equations. Recently, because of the in-depth research in neural networks, numerous dynamic and analog solvers based on recurrent neural networks (RNN) have been developed and investigated [3][6][10]-[13]. The neural dynamic approach is thus now regarded as a powerful alternative to online computation of matrix problems because of its parallel distributed nature and convenience of hardware implementation [10][14].

Different from gradient neural networks for constant problems solving [2][3][6][10][15]-[17], a special kind of recurrent neural networks has recently been proposed by Zhang et al [10]-[12] for real-time solution of time-varying problems solving. In other words, in our context of $AXB = C$, coefficient matrices $A$, $B$ and $C$ could be $A(t)$, $B(t)$ and $C(t)$, time-varying ones. The design method of Zhang neural network is completely different from that of gradient neural networks. In this paper, we generalize such a design method to solving online the time-varying linear matrix equation, $A(t)X(t)B(t) = C(t)$ over time $t \in [0, +\infty)$. Theoretical and simulation results both demonstrate the efficacy of the proposed ZNN neural approach. To the best of our knowledge, there is little work dealing with such a time-varying problem in the literature at present stage, except some preliminary results presented in [11]-[13]. The main contributions of the paper are thus as follows.

1) In our paper, we propose a special kind of recurrent neural network to solve the time-varying linear matrix equation in real-time. As far as we know (with 10-year research experience on neural networks), there are almost no other papers working on these time-varying...
problems. Almost all researchers except us work on the constant linear matrix equation problem.

2) Our paper investigates not only the formulation of the model but also the theoretical results. In other words, our paper could be considered as a complete systematic approach for solving a set of time-varying problems. Thus, other linear matrix problems (including matrix-inversion problems) could be solved based on the theoretical analysis of our paper as well. Evidently, it follows from point 1) that this work is of theoretical significance and application values as well.

3) In order to show the characteristics of the new kind of recurrent neural networks, we also discuss the standard gradient-based neural network for solving the time-varying problems for comparative purposes. Computer-simulation results substantiate the efficacy of ZNN model and show the less favorable property of gradient neural network on the time-varying problems solving.

The remainder of this paper is organized in four sections. Section II presents the problem formulation and design methods of two dynamic-system solvers, i.e., Zhang neural network and gradient neural network. In Section III, corresponding to different kinds of activation functions, the convergence property of Zhang neural network for solving online time-varying linear matrix equation is investigated. Section IV presents an illustrative computer-simulation example about the online solution of time-varying equation \( A(t)X(t)B(t) = C(t) \). Conclusions are given in Section V.

II. PROBLEM FORMULATION AND SOLVERS

Consider a time-varying linear matrix equation,

\[
A(t)X(t)B(t) - C(t) = 0, \quad 0 \leq t < +\infty
\]

(1)

where \( A(t) \in \mathbb{R}^{m \times m} \), \( B(t) \in \mathbb{R}^{n \times n} \) and \( C(t) \in \mathbb{R}^{n \times n} \) are smooth time-varying coefficient matrices, which, together with their derivatives, are assumed to be known or could be estimated accurately. \( X(t) \in \mathbb{R}^{m \times n} \) is the unknown matrix to be obtained.

To lay a basis for further discussion, the following unique-solution condition is assumed.

**Unique-solution condition.** Linear matrix equation (1) is uniquely solvable, if all eigenvalues of matrices \( A(t) \in \mathbb{R}^{m \times m} \) and \( B(t) \in \mathbb{R}^{n \times n} \) are nonzero at any time instant \( t \in [0, +\infty) \).

It follows from Kronecker-product and vectorization techniques [13][18] that time-varying linear matrix equation (1) is equivalent to

\[
(B^T(t) \otimes A(t))\operatorname{vec}(X(t)) = \operatorname{vec}(C(t)).
\]

(2)

Symbol \( \otimes \) denotes the Kronecker product; i.e., \( P \otimes Q \) is a large matrix made by replacing the \( ij \)th entry \( p_{ij} \) of \( P \) with the matrix \( p_{ij}Q \). Operator \( \operatorname{vec}(X) \in \mathbb{R}^{mn} \) generates a column vector obtained by stacking all column vectors of \( X \) together.

If the unique-solution condition holds true, then it is evident that, at any time instant \( t \), the \( mn \)-dimensional square matrix \( B^T(t) \otimes A(t) \) is nonsingular with each eigenvalue being nonzero [18]. Thus, a unique solution to (1) exists. Moreover, the unique-solution condition equals that a positive real number \( \alpha > 0 \) exists such that

\[
(B^T \otimes A)^T(B^T \otimes A) \geq \alpha I, \quad \forall t \geq 0,
\]

(3)

where \( I \) denotes the identity matrix (with its dimension being \( mn \) in the context of this equation).

A. Zhang neural network

To solve online the time-varying linear matrix equation (1), we could develop a recurrent neural network by using the following design method by Zhang et al [10]-[12].

Firstly, to monitor the equation-solving process, a matrix-valued error function \( e(t) \) is defined below, instead of a scalar-valued error function usually associated with gradient-based neural network.

\[
e(t) := A(t)X(t)B(t) - C(t) \in \mathbb{R}^{m \times n}.
\]

(4)

Secondly, the time derivative of error function \( e(t) \in \mathbb{R}^{m \times n} \), i.e., \( \dot{e}(t) \in \mathbb{R}^{m \times n} \), should be chosen such that each element \( e_{ij}(t) \) of \( e(t) \) converges to zero, \( \forall i = 1, \ldots, m \) and \( j = 1, \ldots, n \). Specifically, \( \dot{e}_{ij}(t) \) is chosen such that

\[
\lim_{t \to +\infty} e_{ij}(t) = 0, \quad \forall i,j. \quad (5)
\]

A general form of \( \dot{e}(t) \) is

\[
\frac{de(t)}{dt} = -\Gamma F(e(t)),
\]

(5)

where design parameter \( \Gamma \) and activation-function mapping \( F(\cdot) \) are described as follows.

- \( \Gamma \in \mathbb{R}^{m \times m} \) is a positive-definite matrix used to scale the convergence rate of the solution. \( \Gamma \) could simply be \( \gamma I \) with scalar \( \gamma > 0 \in \mathbb{R} \). \( \Gamma \) (or \( \gamma \)), being a set of inductance parameters or reciprocals of capacitive parameters, should be set as large as the hardware permits (e.g., in analog circuits or VLSI), or selected appropriately for simulative and experimental purposes.

- \( F(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n} \) denotes an activation-function matrix mapping (or to say, an activation-function matrix array) of recurrent neural networks. A simple example of activation-function mapping \( F(\cdot) \) could be the linear one, i.e., \( F(e) = e \). In general, any monotonically increasing odd activation function \( f(\cdot) \), being the \( ij \)th element of \( F(\cdot) \), can be used for the construction of the neural network. Four types of activation function \( f(\cdot) \) are discussed in this paper:

  - linear activation function \( f(e_{ij}) = e_{ij} \),
  - bipolar sigmoid activation function
  \[
  f(e_{ij}) = \frac{1 - \exp(-\xi e_{ij})}{1 + \exp(-\xi e_{ij})}
  \]

with design parameter $\xi \geq 2$.
- power activation function $f(e_{ij}) = e_{ij}^p$ with odd integer $p \geq 3$, and
- power-sigmoid activation function
\[
f(e_{ij}) = \begin{cases} 
  e_{ij}, & \text{if } |e_{ij}| > 1 \\
  \frac{1 - \exp(-\xi e_{ij})}{1 + \exp(-\xi e_{ij})}, & \text{otherwise} 
\end{cases}
\]
with suitable design parameters $\xi \geq 2$, $p \geq 3$, and $\delta := (1 + \exp(-\xi))/(1 - \exp(-\xi)) > 1$.

Thirdly, expanding the design formula (5) leads to the following implicit dynamics of the resultant Zhang neural network for the online solution of time-varying linear matrix equation (1):
\[
\dot{A}XB = -\dot{A}XB - AX\dot{B} + \dot{C} - \gamma F(AXB - C),
\]
where $X(t)$, staring from an initial state $X(0) = X_0 \in \mathbb{R}^{m \times n}$, is the activation state matrix corresponding to theoretical solution $X^*(t)$ of (1). We use $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{n \times n}$ and $C \in \mathbb{R}^{m \times n}$ to denote the known analytical forms or measurements of the time derivatives of matrices $A$, $B$ and $C$, respectively. In addition, when using the linear activation-function mapping $F(e) = e$, Zhang neural network (7) reduces to the following linear one:
\[
\dot{A}XB = -\dot{A}XB - AX\dot{B} - \gamma AXB + (\dot{C} + \gamma C),
\]
where argument $t$ is dropped for presentation convenience, and so is in (7) and equations afterwards.

B. Gradient neural network

For comparison, it is worth mentioning here that we can develop a gradient-based neural network to solve online linear matrix equation (1). However, similar to almost all the numerical algorithms and neural-dynamic computational schemes [2][3][6][10][15]-[17], the gradient neural network is designed intrinsically for constant coefficient matrices $A$, $B$ and $C$. It generally belongs to the gradient descent method in optimization [16][17][19] and can be designed by the following procedure.

- Firstly, a scalar-valued error function, such as $\|AXB - C\|_F^2/2$ with Frobenius norm $\|C\|_F := \sqrt{\text{trace}(C^TC)}$, is constructed such that its minimum point is the solution of equation (1).
- Secondly, an algorithm is designed to evolve along a descent direction of this error function until the minimum is reached. The typical descent direction is the negative gradient of $\|AXB - C\|_F^2/2$, i.e.,
\[
\frac{\partial (\|AXB - C\|_F^2/2)}{\partial X} = -A^T(AXB - C)B^T.
\]
- Thirdly, by using the above negative gradient to construct the neural network, we could have a linear GNN model, $\dot{X} = -\gamma A^T(AXB - C)B^T$, and a general nonlinear GNN model as the following
\[
\dot{X}(t) = -\gamma A^TF(AXB - C)B^T,
\]
where convergence results could be achieved only for the situation of using constant $A$, $B$ and $C$ [2][10][16].

C. Model comparison

In this subsection, we would like to compare the two design methods and models of Zhang neural network (7) and gradient neural network (9), which are exploited for the online solution of time-varying linear matrix equation (1). The differences lie in the following facts.

Firstly, Zhang neural network (7) is designed based on the elimination of every entry of the matrix-valued error function $e(t) = A(t)X(t)B(t) - C(t)$. In contrast, gradient neural network (9) is designed based on the elimination of the norm-based scalar-valued error function $\|AXB - C\|_F^2/2$ (note that here $A$, $B$ and $C$ could only be constant in the design and analysis of gradient-based neural networks).

Secondly, Zhang neural network (7) methodically and systematically exploits the time-derivative information of coefficient matrices $A(t)$, $B(t)$ and $C(t)$ during its real-time solving process. This is the reason why Zhang neural network (7) could globally exponentially converge to the exact solution of a time-varying problem. In contrast, gradient neural network (9) has not exploited such important information, and thus may not be effective in solving such a time-varying problem.

Thirdly, Zhang neural network (7) is depicted in an implicit dynamics, i.e., $A(t)\dot{X}(t) = B(t)$. In contrast, gradient neural network (9) is depicted in an explicit dynamics, i.e., $\dot{X}(t) = \cdots$, which is usually associated with classic Hopfield-type recurrent neural networks.

Before ending this subsection, following the above third point, we would like to mention that the implicit dynamic equations (or to say, implicit systems) frequently arise in analog electronic circuits and systems due to Kirchhoff’s rules. Furthermore, implicit systems have higher abilities in representing dynamic systems, as compared to explicit systems. The implicit dynamic equations could preserve physical parameters in the coefficient matrices, e.g., $A(t)$ and $B(t)$ on the left-hand side of (7). They could describe the usual and unusual parts of a dynamic system in the same form. In this sense, implicit systems might be much superior to the systems represented by explicit dynamics. Besides, the implicit dynamic equations (or implicit systems) could be mathematically transformed to explicit dynamic equations (or explicit systems) if needed.

III. THEORETICAL RESULTS

While Section II gives a general description of Zhang neural network (7) to solve the time-varying problem (1),
detailed design consideration and main theoretical results about its global exponential convergence are given in this section. The following Theorems 1 through 3 (with proof omitted due to space limitation) and Remarks 1 through 3 might be important.

**Theorem 1:** Consider smooth time-varying coefficient matrices $A(t) \in R^{m \times m}$, $B(t) \in R^{n \times n}$ and $C(t) \in R^{m \times n}$ of linear matrix equation (1). If the unique-solution condition (3) is satisfied, then state matrix $X(t)$ of Zhang neural network (8), starting from any initial state $X_0 \in R^{m \times n}$, converges exponentially to the theoretical solution $X^*(t)$ of time-varying linear matrix equation (1). In addition, the exponential convergence rate of equation (8) is $\gamma$. \hfill \Box

If different activation functions (other than the aforementioned linear one) are exploited to construct the neural network, we could have the following theorems on global convergence of ZNN (7).

**Theorem 2:** Consider smooth time-varying coefficient matrices $A(t) \in R^{m \times m}$, $B(t) \in R^{n \times n}$ and $C(t) \in R^{m \times n}$ of linear matrix equation (1), with unique-solution condition (3) satisfied. If a monotonically-increasing odd activation-function array $F(\cdot)$ is used, then state matrix $X(t)$ of Zhang neural network (7), starting from any initial state $X_0 \in R^{m \times n}$, converges to the theoretical solution $X^*(t)$ of time-varying linear matrix equation (1). \hfill \Box

**Theorem 3:** In addition to Theorem 2, Zhang neural network (7) possesses the following properties. If linear activation function $f(e_{ij}) = e_{ij}$ is used, then global exponential convergence with rate $\gamma$ could be achieved for (7). Compared to the linear situation, superior convergence could be achieved for (7) if power-sigmoid function (6) is used. \hfill \Box

**Remark 1:** Evidently, we could readily analyze the situation of using bipolar sigmoid activation function (or power activation function) from Theorem 3. The results are as follows. If the bipolar sigmoid activation-function is used, then the superior convergence can be achieved for (7) for error range $e_{ij}(t) \in [-\varrho, \varrho]$, where $\varrho > 0$, as compared to the linear situation. If the power activation function is used, then the superior convergence can be achieved for (7) for error range $|e_{ij}(t)| > 1$, as compared to the linear situation.

**Remark 2:** One more advantage of using the power-sigmoid or sigmoid activation function over the linear function lies in the extra parameter $\gamma$, which is a multiplier of the exponential convergence rate. When there is an upper bound on $\gamma$ due to hardware implementation, the parameter $\gamma$ will be another effective factor expediting the ZNN convergence. The convergence for nonlinear activation functions could be much faster than that for linear activation functions, when using the same level of design parameters $\xi$, $p$ and $\gamma$ (e.g., 10 ~ 100).

**Remark 3:** Nonlinearity always exists, which is one of the main motivations for us to investigate different activation functions in the construction of Zhang neural network (7). 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, or due to truncation and round-off errors of digital realization [4][10][14][20]. The investigation of different activation functions may give more insights into the imprecise-implementation problem of neural networks. That is, if a monotonically-increasing odd activation function $f(\cdot)$ is implemented imprecisely, then Zhang neural network (7) will still be able to solve the time-varying problem (1) asymptotically and globally.

![Fig. 1. Online solution of time-varying linear matrix equation (1) by Zhang neural network (7) using power-sigmoid activation function (6) with design parameters $\xi = 4$ and $p = 3$, where theoretical solution $X^*(t)$ is denoted by dotted lines in red](image-url)
Fig. 2. Online solution of time-varying linear matrix equation (1) by gradient neural network (9) with \( \gamma = 1 \), where theoretical solution \( X^\ast(t) \) is denoted by dotted lines in red

### IV. ILLUSTRATIVE EXAMPLES

In this section, computer-simulation results are presented to demonstrate the characteristics and verify the theoretical results of Zhang neural network (7), which is also compared with gradient neural network (9).

Now let us consider the real-time solution of linear matrix equation \( A(t)X(t)B(t) - C(t) = 0 \) with time-varying coefficients defined as follows:

\[
A(t) = \begin{bmatrix}
\sin t & \cos t \\
-\cos t & \sin t
\end{bmatrix} \in \mathbb{R}^{2 \times 2},
\]

\[
B(t) = \begin{bmatrix}
2 & \sin t & \cos t \\
-\cos t & 2 & \cos t \\
\sin t - 2 & 2 - \sin t & 2 - \sin t
\end{bmatrix} \in \mathbb{R}^{3 \times 3},
\]

and \( C(t) = [2 + \sin t \cos t - 2 \cos t, 3 \cos t + \sin t - \sin t \cos t, \sin t + 2 \cos t - \sin t \cos t; -\cos t + \sin^2 t - 2 \sin t, 2 + 2 \sin t - \sin^2 t, \cos t + 2 \sin t - \sin^2 t] \in \mathbb{R}^{2 \times 3}. \)

Simple algebraic manipulations can verify that the theoretical solution \( X^\ast(t) \) to linear matrix equation (1) with the above coefficients is

\[
X^\ast(t) = \begin{bmatrix}
\sin t & -\cos t & 0 \\
\cos t & \sin t & 1
\end{bmatrix} \in \mathbb{R}^{2 \times 3},
\]

which is given just for comparative purposes. That is, to check the correctness of neural solutions.

#### A. Model comparison

To construct Zhang neural network (7), the time derivatives of the above coefficient matrices are assumed to be measurable or known as follows:

\[
\dot{A}(t) = \begin{bmatrix}
\cos t & -\sin t \\
\sin t & \cos t
\end{bmatrix} \in \mathbb{R}^{2 \times 2},
\]

\[
\dot{B}(t) = \begin{bmatrix}
0 & \cos t & -\sin t \\
\sin t & -\cos t & \sin t \\
\cos t & \sin t & -\cos t
\end{bmatrix} \in \mathbb{R}^{3 \times 3},
\]

and \( \dot{C}(t) = [-\sin^2 t + \cos^2 t + 2 \sin t, -3 \sin t + \cos t + \sin^2 t - \cos^2 t, \cos t - 2 \sin t + \sin^2 t - \cos^2 t; \sin t + 2 \cos t \sin t - 2 \cos t - 2 \cos t - 2 \cos t \sin t] \in \mathbb{R}^{2 \times 3} \). Then, Zhang neural network (7) is simulated. As seen from Fig. 1, starting from any initial states randomly selected in \([-2, 2]^6\), state \( X(t) \) of the proposed Zhang neural network (7) all converges to the theoretical solution \( X^\ast(t) \) to (1). It is worth mentioning that the maximum steady-state residual errors \( \lim_{t \to \infty} \|X(t) - X^\ast(t)\|_F \) synthesized by Zhang neural network (7) are about \( 1 \times 10^{-15} \) for \( \gamma = 1 \) and \( 1.1 \times 10^{-14} \) for \( \gamma = 10 \), respectively. Note that such a maximum steady-state residual error should theoretically be zero but are numerically nonzero because of the finite-arithmetic simulation performed on finite-memory digital computers with floating-point relative accuracy being \( 2.22 \times 10^{-16} \) [21].

In comparison, gradient neural network (9) is simulated as well for this equation-solving task under the same design parameters. Its performance is depicted in Fig. 2, where the maximum steady-state residual error of the GNN solution is 0.7, considerably large. The reason for this less favorable performance is evidently that the time-derivative information of coefficients \( A(t), B(t) \) and \( C(t) \) has not been utilized in the traditional gradient-based scheme.

#### B. Activation-function comparison

Besides the above comparative results, in this subsection we simulate Zhang neural network (7) using linear activation function \( f(e_{ij}) = e_{ij} \). The results are shown in Fig. 3(a)(b), where computational errors \( \|X(t) - X^\ast(t)\|_F \) of Zhang neural network decrease rapidly to 0 as well.

However, by comparing Fig. 3(a) with Fig. 3(c) (i.e., the situation with \( \gamma = 1 \)), we can see that the typical convergence time of ZNN (7) using power-sigmoid activation function is roughly 3 seconds, two times faster than that using linear activation function (i.e., roughly 6 seconds). This observation appears to be the same in the situation with \( \gamma = 10 \), by comparing Fig. 3(b) with Fig. 3(d). As a result, compared to the linear-function case, superior performance can be achieved for ZNN (7) by using the power-sigmoid function under the same design parameters. These have substantiated the theoretical analysis presented in the previous section, i.e., Theorems 1 through 3.

#### C. Design-parameter comparison

In this subsection, by summarizing the figures presented before, we would like to compare the effects of using
different values of design parameter \( \gamma \). It is worth mentioning that, in Figs. 1 and 3, \( \gamma = 1 \) and \( \gamma = 10 \) are used just for illustrative purposes, so that the readers could see easily the transient behavior of the neural network. As seen from Figs. 1 and 3, when design parameter \( \gamma \) is increased from 1 to 10, the convergence is expedited about ten times faster. This, in turn, has substantiated the proved exponential convergence of Zhang neural network (7), i.e., Theorems 1 and 3.

As a general conclusion for ZNN (7) using power-sigmoid activation function, if \( \gamma \) is increased to \( 10^3 \), the convergence time is within 3 milliseconds; while, if \( \gamma \) is increased to \( 10^5 \), the convergence time is within 3 microseconds. On the other hand, for ZNN (7) using linear function, if \( \gamma \) is increased to \( 10^6 \), the convergence time is within 6 microseconds.

V. CONCLUSIONS

By following the design method recently proposed by Zhang et al, a recurrent neural network has been developed and analyzed for the online solution of time-varying linear matrix equation. The neural network could globally exponentially converge to the exact solution of such a time-varying linear matrix equation. Simulation results have substantiated the theoretical analysis. Future research directions may lie in circuit implementation and discrete-time model/algorithm development of such a Zhang neural network.

ACKNOWLEDGEMENTS

Before joining Sun Yat-Sen University in 2006, the corresponding author, Yunong Zhang, had been with National University of Ireland, University of Strathclyde, National University of Singapore, Chinese University of Hong Kong, since 1999. Continuing the line of this research, he has been supported by different research fellowships/assistantship. His web-page is now available at http://www.ee.sysu.edu.cn/teacher/detail.asp?sn=129.

REFERENCES
