NONLINEAR DYNAMICAL SYSTEM MODELING VIA
RECURRENT NEURAL NETWORKS AND A WEIGHTED
STATE SPACE SEARCH ALGORITHM

LEONG-KWAN LI
Department of Applied Mathematics
The Hong Kong Polytechnic University, Kowloon, Hong Kong, China

SALLY SHAO
Department of Mathematics
Cleveland State University, Cleveland, OH 44115, USA

K. F. CEDRIC YIU
Department of Applied Mathematics
The Hong Kong Polytechnic University, Kowloon, Hong Kong, China

(Communicated by Song Wang)

Abstract. Given a task of tracking a trajectory, a recurrent neural network
may be considered as a black-box nonlinear regression model for tracking un-
known dynamic systems. An error function is used to measure the difference
between the system outputs and the desired trajectory that formulates a non-
linear least square problem with dynamical constraints. With the dynamical
constraints, classical gradient type methods are difficult and time consuming
due to the involving of the computation of the partial derivatives along the
trajectory. We develop an alternative learning algorithm, namely the weighted
state space search algorithm, which searches the neighborhood of the target
trajectory in the state space instead of the parameter space. Since there is no
computation of partial derivatives involved, our algorithm is simple and fast.
We demonstrate our approach by modeling the short-term foreign exchange
rates. The empirical results show that the weighted state space search method
is very promising and effective in solving least square problems with dynam-
ical constraints. Numerical costs between the gradient method and our the
proposed method are provided.

1. Introduction. Artificial neural networks are trainable analytic tools that at-
tempt to mimic information processing patterns in the brain. Recurrent neural
networks (RNNs) have rich dynamical structures in which they can learn extremely
complex temporal patterns. Thus, RNNs have been studied by many researchers
in last few decades and become popular both for implementation and applications.

2000 Mathematics Subject Classification. 68T05, 65K99.
Key words and phrases. Nonlinear dynamical system; recurrent neural network; state space
search.

The first author is supported by the Hong Kong Polytechnic University (PolyU A-SA63). The
third author is supported by RGC Grant PolyU. (5365/09E). The paper was presented in the 4th
International Conference on Optimization and Control with Applications (OCA2009), 6-11 June,
2009 in Harbin, China. A brief version of this paper was published in the Conference Proceedings.
Given a task of tracking a desired trajectory $y(t), t = 1, 2, ..., m$, learning in a neural network is a process of changing the network parameters $W$ so that the system output $x(t)$ will approach the desired trajectory $y(t)$. To optimizing the network performance, an error function $E(W)$ is used to measure the difference between the system outputs and the desired trajectory. Hence, a least square problem with nonlinear dynamical constraints is defined.

Gradient descent, conjugate gradient and quasi-Newton’s method have been applied to solve the least square problems for many decades. For a fully connected RNN with $n$ neurons, the connection weight $W$ is an $n$ by $n$ matrix. The classical approach was introduced in [15] where a learning algorithm for the fully connected discrete-time recurrent neural network based on the gradient descent method was developed. Since then researchers studying RNNs mainly concentrated on applications (for example, [16]). There have not been many advances made in learning algorithms. A breakthrough in the learning algorithm for RNNs was made by [1] in which the role of the parameter space $W$ and the state space of the trajectory were interchanged. Later, [2] and also [10] made some contributions in improving the RNN learning algorithms. Yet, these learning algorithms based on the gradient descent methods involve the computation of partial derivatives along a trajectory, which are time consuming and costly. Some researchers trained the RNNs with the simulated annealing algorithm (See, for example, [4] and [12]), but the efficiency and convergence are to be improved. Other attempts were suggested in [18] and [13], which for B-spline neural networks both considered general constrained optimization problems in which the hybrid simulated annealing [19] or nonlinear programming techniques were employed to obtain the network parameters. Again, gradient information is required in these approaches.

In fact, if a gradient type method is used, we need to compute the partial derivatives along the trajectory for $n^2$ parameters of $W$ per iteration. Furthermore, if the desired trajectory to be approximated is relatively long, say time $m > 100$, other numerical issues such as accuracy may be impacted significantly. It follows that the training process becomes prohibitively costly for each iteration even with the growing sophistication of computer hardware and software as well as mathematical algorithms. Hence, the growing demand for derivative free optimization methods for RNNs has triggered researchers to develop a relatively wide range of derivative-free approaches.

The main contribution of this paper is the development of a derivative-free and non-random learning algorithm—the weighted state space search algorithm (WSSSA) for the RNNs. By employing the WSSSA for the RNN learning process, we search the neighborhood of the target trajectory in the state space instead of the parameter space. Our method is based on approximating a new trajectory which is a convex combination of the system output of the RNN and the desired trajectory with a ratio $\alpha$. Since no computation of partial derivatives is involved, the WSSSA is simpler and faster. For the simple uniform weighted version of WSSSA, convergence has been established with a suitable choice of $\alpha$’s in [8]. However, for the accuracy and convergence of the trajectory, it is essential to weight data points differently at different times, with higher weights for more recent events. In view of this, we formulate the generalized version here and demonstrate how to use a systematic scheme of dynamic $\alpha$ in a weighted SSSA. From the convergence proof, one can see that the choice of the sequence of parameters depends on the choice of neural
networks, which can possibly be extended for other networks, for example, the B-spline network, and large-scale problems can be solved more efficiently.

The organization of this paper is as follows. In section 2, we highlight the leaky integrator RNN model and present some characteristics of the solutions and the stability properties of the RNN system. The WSSSA is presented in section 3. A comparison study of the proposed method and the gradient-type methods is discussed in section 4. In section 5, modeling the short-term foreign exchange rates is used as an example to illustrated our approach. Some final remarks and future research directions are given in section 6.

2. System dynamics of the leaky integrator model. Consider the continuous-time leaky integrator model of the RNN with \( n \) neurons represented by a system of nonlinear equations

\[
\frac{dx_i}{dt} = -a_i x_i + b_i \sigma \left( \sum_{j=1}^{n} w_{ij} x_i + \theta_i \right), \quad i = 1, 2, 3, \ldots n, \tag{2.1}
\]

where \( x_i \) represents the internal state of the \( i^{th} \) neuron, \( \sigma \) is a neuronal activation function that is bounded, differentiable and monotonic increasing on \([-1, 1]\). We assume further that \( \sigma(z) = \tanh(z) \) which is the symmetric sigmoid logistic function; \( \theta = [\theta_1, \theta_2, \ldots, \theta_n]^T \) is the input bias or threshold vector of the system; \( W = [w_{ij}]_{n \times n} \) is the synaptic connection weight matrix with \( w_{ij} \) being the synaptic weight of a connection from neuron \( n_j \) to neuron \( n_i \); \( A = \text{diag}[a_1, a_2, \ldots, a_n] \) and \( B = \text{diag}[b_1, b_2, \ldots, b_n] \) are diagonal matrices with positive diagonal entries. Then system (2.1) can be rewritten in the following matrix form

\[
\frac{dx}{dt} = -Ax + B\sigma(Wx + \theta), \tag{2.2}
\]

where \( \sigma = \text{diag}[\sigma_1, \sigma_2, \ldots, \sigma_n] \) with \( \sigma_i \) is defined by

\[
\sigma_i(Wx + \theta) = \sigma \left( \sum_{j=1}^{n} w_{ij} x_i + \theta_i \right), \quad i = 1, 2, \ldots, n. \tag{2.3}
\]

System (2.2) is a continuous-time model of the leaky integrator RNNs (we simply refer it as the RNNs for the rest of the paper), which has been studied in many applied areas of science (see, for example, [5], [6], [7]). In the spirit of [11], we show in the following theorems some solution behaviors and the stability property of system (2.2).

By Peano’s local existence theorem for the solution of ordinary differential equations, given any \( x_0 \in \mathbb{R}^n \), there exists a positive number \( t^*(x_0) \) such that the system (2.2) has a solution \( x(t, x_0) \in \mathbb{R}^n \) for \( t \in [0, t^*(x_0)) \), which is the maximal right existence interval of the solution \( x(t; x_0) \) satisfying \( x(0, x_0) \). Thus, the trajectory of the RNN model (2.2) is bounded for any initial point in \( \mathbb{R}^n \). With this theorem in mind, we introduce Theorem 1 as follows:

**Theorem 1.** Given an initial state \( x(0) = x_0 \), there exists a positive number \( t^*(x_0) \) such that the solution \( x(t) \in \mathbb{R}^n \) of model (2.2) is unique and bounded for all \( t \in [0, t^*(x_0)) \). In addition, the solution \( x(t) \) exists globally. In other words, the solution of system (2.2) is globally bounded.
Thus, we obtain

$$388 \text{ LEONG-KWAN LI, SALLY SHAO AND K. F. CEDRIC YIU}$$

system (2.2) is globally bounded.

where matrices $A$ and $B$ are defined as in (2.2). $v_p = (p,p,...,p)^T$ and $v_q = (q,q,...,q)^T$ are the $n$ dimensional constant vectors to be determined later, and $\theta = [\theta_1, \theta_2, ..., \theta_n]^T$ is the system input. We set $y(0) = x(0) = x_0$. We claim that: $y_i(t) \leq x_i(t) \leq z_i(t)$ for all $i = 1, ..., n$. and for all real $t$.

In fact, let $r = x_j - y_j$, then $r(0) = 0$, and

$$r' = -a_ir + g,$$

where we choose $p$ such that

$$g = b_i(\sigma_i(u_i) - p) - \theta_i > 0 \quad \text{with} \quad u_i = \sum_{j=1}^{n} w_{ij}x_i + \theta_i.$$

Then $r(t) = e^{-a_it} \int_0^t g e^{a_i s} ds > 0$ for each $i = 1, 2, ..., n$. It follows that $y_i(t) \leq x_i(t)$ for all $t \in [0, t^*(x_0)]$ with some positive number $t^*(x_0)$.

Similarly we can show that $z_i(t) \geq x_i(t)$. Hence, for any given initial state $x_0$, there exists a bounded solution $x(t)$ of system (2.2). Moreover,

$$y_i(t) = [y_i(0) - \frac{b_ip + \theta_i}{a_i}]e^{-a_it} + \frac{(b_ip + \theta_i)}{a_i}$$

and

$$z_i(t) = [z_i(0) - \frac{b_q + \theta_i}{a_i}]e^{-a_it} + \frac{(b_q + \theta_i)}{a_i}.$$

Thus, we obtain

$$[y_i(0) - \frac{b_ip + \theta_i}{a_i}]e^{-a_it} + \frac{(b_ip + \theta_i)}{a_i} \leq x_i(t) \leq [z_i(0) - \frac{b_q + \theta_i}{a_i}]e^{-a_it} + \frac{(b_q + \theta_i)}{a_i}$$

for $i = 1, 2, ..., n$.

Hence, as time $t$ tends to infinity, each component of the solution $x(t)$ of system (2.2) is asymptotically contained in an interval $[\frac{(b_ip + \theta_i)}{a_i}, \frac{(b_q + \theta_i)}{a_i}]$. This implies that system (2.2) is globally bounded.

For the uniqueness of the solution $x(t)$ of system (2.2), let $f \in C^n [R^n]$ defined by

$$f_i = -a_ix_i + b_i(\sum_{j=1}^{n} w_{ij}x_j) + \theta_i.$$  \hspace{1cm} (2.5)

and $\sigma \in C^n_{\sigma}$ for $n \geq 2$. Suppose $\sigma'$ is bounded by a positive constant $k$. Then by the mean-value theorem, $\sigma(a) - \sigma(b) = \sigma'(\phi)(a - b)$ for some $\phi \in (a, b)$, that is, $|\sigma(a) - \sigma(b)| < p|a - b|$. Hence,

$$\| f(x) - f(y) \| \leq \| A(x - y) \| + \| B(\sigma(Wx) - \sigma(Wy)) \|$$

$$\leq \| A \| \| x - y \| + \| B \| \| \sigma' \| \| Wx - Wy \|$$

$$\leq \| A \| \| x - y \| + k \| B \| \| W \| \| x - y \|$$

Therefore, $f$ is Lipschitz over $R^n$ with Lipschitz constant $\| A \| + k \| B \| \| W \|$. Thus, the uniqueness of the solution is also guaranteed by the theory of ordinary differential equations. Hence, for a fixed $W$ and arbitrary initial state $x_0$, there exists an unique solution for system (2.2). \hfill \square
Note that if the input $\theta(t)$ is a bounded function of time, the global boundedness of the system is not affected. Yet, the equilibrium state $x^* = \sigma(Wx^*) + \theta$ depends on the input. For a dynamical system, although all the real parts of the eigenvalues of the Jacobian are negative for every $x \in \mathbb{R}^n$, the dynamical system need not be globally asymptotically stable. We are interested in finding a condition for which the system is strictly globally asymptotically stable so that the output state $x(t)$ will reach the same steady state as $t$ tends to $\infty$ for an arbitrary given initial state.

One standard approach is to consider the eigenvalues of the Jacobian matrix $\frac{df(x)}{dx}$. Since $f_i = -x_i + \sigma(\sum_{j=1}^n w_{ij}x_j) + \theta_i$, we have

$$\frac{\partial f_i}{\partial x_j} = \frac{\partial}{\partial x_j}(-x_i + \sigma(\sum_{j=1}^n w_{ij}x_j) + \theta_j) = -\delta_{ij} + w_{ij}\sigma'(\sum_{j=1}^n w_{ij}x_j) = -\delta_{ij} + w_{ij}\sigma'(u_i)$$

where $\delta_{ij}$ is the Kronecker delta. Hence,

$$\frac{df(x)}{dx} = -I + \Lambda W,$$

where $\Lambda$ is a diagonal matrix with $\Lambda_{jj} = \sigma'(\sum_{k=1}^n w_{jk}x_k)$.

Now let $\Omega$ be the set of solutions of (2.2). Since $\Omega$ is bounded in $\mathbb{R}^n$, we assume that there exist constants $c, d \in \mathbb{R}^n$ such that

$$\Omega = [c_1, d_1] \times [c_2, d_2] \times \cdots \times [c_n, d_n]$$

where $c_i$ and $d_i$ are defined as in (2.7).

Before we proceed to prove Theorem 2, we need the following two lemmas.

**Lemma 1.** For $i = 1, 2, ..., n$, $\Omega^i$ is positive invariant.

**Proof.** We need to show that if the initial condition $x_0 \in \mathbb{R}^n$ satisfies

$$x_0^i = x_i(0; x_0) \in \Omega^i, \quad i = 1, 2, ..., n,$$

the $i$th component of the solution $x(t; x_0)$ has the property that $x_i(t; x_0) \in \Omega^i$ for $t \geq 0$. That is, $\Omega^i$ is positive invariant.
Let \( \hat{t}_i = \sup\{t^* \geq 0 | x_i(t; x_0) \in \Omega^i_\epsilon, \forall t \in [0, t^*]\} \geq 0. \)

We claim that: \( \hat{t}_i = \infty. \)

In fact, suppose the contrary, that is \( \hat{t}_i < \infty, \) then there exists a positive number \( \delta > 0 \) and \( x_i(t; x_0) \in \Omega^i_\epsilon \) for \( t \in [0, \hat{t}_i] \) such that

\[
x_i(t; x_0) \notin \Omega^i_\epsilon, \quad \forall t \in (\hat{t}_i, \hat{t}_i + \delta).
\]

Without loss of generality, we assume that

\[
x_i(t; x_0) < c_i - \epsilon \quad \forall t \in (\hat{t}_i, \hat{t}_i + \delta).
\]

Consider the \( i^{th} \) equation,

\[
\frac{1}{a_i} \frac{dx_i(t)}{dt} = -x_i(t) + \frac{b_i}{a_i} \sigma_i(Wx + \theta),
\]

where \( \sigma_i \) is defined in (2.3). For \( x(t) = x(t, x_0), t \geq 0, \) we have

\[
\frac{1}{a_i} \frac{dx_i(t; x_0)}{dt} = -x_i(t; x_0) + \frac{b_i}{a_i} > \epsilon > 0, \quad \text{for all} \quad t \in (\hat{t}_i, \hat{t}_i + \delta)
\]

by (2.12), the definition of \( \Omega \) and \( -1 \leq \sigma_i \leq 1. \) Since \( a_i > 0, \) we have \( \frac{dx_i(t; x_0)}{dt} > 0, \) which implies \( x_i(t; x_0) \) is strictly increasing on \( (\hat{t}_i, \hat{t}_i + \delta). \) Thus,

\[
x_i(t; x_0) > x_i(\hat{t}_i; x_0), \quad \forall t \in (\hat{t}_i, \hat{t}_i + \delta).
\]

Since \( x_i(t; x_0) \in \Omega^i_\epsilon \) for \( t \in [0, \hat{t}_i], \) together with (2.12) we have

\[
x_i(t; x_0) = c_i - \epsilon.
\]

Note that by the similarity we can obtain the same conclusion in the case of \( x_i(t; x_0) > d_i + \epsilon. \)

From (2.15) we obtain

\[
x_i(t; x_0) > x_i(\hat{t}_i; x_0) = c_i - \epsilon, \quad \forall t \in (\hat{t}_i, \hat{t}_i + \delta),
\]

which contradicts condition (2.16). Therefore, \( \hat{t}_i = \infty. \) This means that \( \Omega^i_\epsilon \) is positive invariant.

**Lemma 2.** For \( i = 1, 2, ..., n, \) if the initial condition \( x_0 \in \mathbb{R}^n \) satisfies

\[
x_{0i} \notin \Omega^i_\epsilon, \quad x_0 = [x_{01}, x_{02}, ..., x_{0n}]^T,
\]

then the solution trajectory \( x_i(t; x_0) \) will stay in \( \Omega^i_\epsilon \) after a finite period of time. That is, there exists a positive number \( \hat{t} \) such that

\[
x_i(t; x_0) \in \Omega^i_\epsilon, \quad \text{for} \quad t \geq \hat{t} > 0,
\]

which implies \( \Omega^i_\epsilon \) is an attractive set.

**Proof.** Since \( x_{0i} \notin \Omega^i_\epsilon, \) we let

\[
\hat{t}_i = \sup\{t^* > 0 | x_i(t; x_0) < c_i - \epsilon, \forall t \in [0, t^*]\} > 0.
\]

From equation (2.14), we have

\[
\frac{1}{a_i} \frac{dx_i(t; x_0)}{dt} \geq -x_i(t; x_0) + \frac{b_i}{a_i} > \epsilon > 0, \quad \forall t \in [0, \hat{t}_i).
\]

By the theory of differential inequalities, we have

\[
x_i(t; x_0) > x_{0i} + \epsilon a_i t, \quad \forall t \in [0, \hat{t}_i).
\]
Condition (2.20) together with the definition of $\hat{t}_i$, we obtain

$$\hat{t}_i \leq \frac{1}{\epsilon a_i} (c_i - \epsilon - x_0) < \infty$$  \hspace{1cm} (2.21)

and

$$x_i(\hat{t}_i; x_0) = c_i - \epsilon \in \Omega^i_\epsilon.$$  \hspace{1cm} (2.22)

This means that $\Omega^i_\epsilon$ is an attractive set.

**Theorem 2.** $\Omega$ is a positive invariant and attractive set of the RNN model (2.2).

**Proof.** By Lemma 1 and Lemma 2, for any positive number $\epsilon > 0$, $\Omega^i_\epsilon$ is a positive invariant and an attractive set. Moreover, for any solution $x(t; x_0)$ with initial point $x_0 \notin \Omega$, it will enter into $\Omega$ in a finite time. Because of the positive invariant property of $\Omega^i_\epsilon$, $x_i(t; x_0)$ will stay in $\Omega^i_\epsilon$ after a finite period of time. Let $\epsilon \to 0^+$, we have $\lim_{\epsilon \to 0^+} \Omega^i_\epsilon = \Omega$. Therefore, we can conclude that $\Omega$ is a positive invariant and attractive set of the RNN model (2.2).

Let the matrix $A = \text{diag}[a_1, a_2, \ldots, a_n] = I_n$ in system (2.2), where $I_n$ is the $n \times n$ identity matrix, then system (2.2) becomes

$$\frac{dx}{dt} = -x + B\sigma(Wx + \theta)$$ \hspace{1cm} (2.23)

or

$$\frac{dx_i}{dt} = -x_i + b_i\sigma(\sum_{j=1}^{n} w_{ij}x_j + \theta) \quad i = 1, 2, 3, \ldots, n.$$  \hspace{1cm} (2.24)

We cite without proof another result in [8] as Theorem 3 for the stability property of system (2.24) and its equivalent form (2.23). **Theorem 3.** If $W$ is invertible and the matrix $WB$ is semi-negative definite, then the RNN model (2.23) is globally exponentially stable. That is, there exists two positive constants $p \geq 1$ and $q > 0$ such that for any $x_0 \in \mathbb{R}^n$ and $t \in [0, \infty)$

$$\|x(t; x_0) - x^*\| \leq p\|x_0 - x^*\| \exp(-qt),$$  \hspace{1cm} (2.25)

where $x(t; x_0)$ is the solution of (2.23) with initial condition $x(0; x_0) = x_0$, and $x^*$ is an equilibrium point of (2.23) defined by

$$x^* = B\sigma(Wx^* + \theta).$$  \hspace{1cm} (2.26)

**Remarks.** We notice that the inequality shows the global exponential convergence rate of the continuous-time RNN model (2.24). Let the matrix $A = aI_{n \times n}$ for some positive constant $a$, we have

$$\|x(t; x_0) - x^*\| \leq p\|x_0 - x^*\| \exp(-qt)$$

with $p = (\text{cond}W)^{\frac{1}{2}} \geq 1$, $q = \frac{1}{2a}$, and $s = 1 + \frac{\lambda_{\max}(WB)^T(WB)}{2a^2}$. If $a$ is small, the RNN model (2.1) will be more sensitive to input noise and round-off errors in numerical simulations and implementation.
3. Weighted state space search learning algorithm. We have presented three important characteristics of the solutions to system (2.1) in Section 2. It is known that the continuous-time model (2.1) (and its equivalent form (2.2)) and the discrete-time system (3.1) need not share the same dynamical behavior. However, the discretized variant (3.1) of system (2.1) will inherit the same dynamics of system (2.1) when the step size is “small” (see [14]). In practice, we approximate system (2.1) by Euler’s method and obtain the dynamics of a discrete-time RNN model

\[ x(t + 1) = (I - HA)x(t) + HBS(Wx(t) + \theta), \]  

(3.1)

where \( H = \text{diag}[h_1, h_2, ..., h_n] \) is a diagonal matrix with positive entries \( h_i \) for each \( i \), and \( h_i \) is the step size of Euler’s discretization. The two systems (3.1) and (2.2) share the same dynamical behavior when \( h_i \) tends to 0. For simplicity, we let \( h_i = h \) for all \( i \).

Consider a given trajectory \( y(t) \in \mathbb{R}^n \) and \( t = 1, ..., m \), we use (3.1) to approximate the target trajectory \( y(t) \) with the error function \( E \) defined by

\[ E(W, h, \theta) = \| x(t, W, h, \theta) - y(t) \|^2 \]

\[ = \sum_{i=1}^{n} \sum_{t=1}^{m} [x_i(t, W, h, \theta) - y_i(t)]^2 \]

for some positive integer \( m \). To further simplify the notation and analysis, we fix \( h \) and let \( A = B = I_{n \times n} \), \( i = 1, 2, ..., n \). We extend the trajectory dimension by one and let \( y_{n+1}(t) = 1 \) for all \( t \) in order to absorb the variable \( \theta \) into the last column of \( W \). Without loss of generality, we still consider an \( n^2 \)-dimensional constrained least square problem

\[ E(W) = \| x(t, W) - y(t) \|^2 \]

\[ = \sum_{i=1}^{n} \sum_{t=1}^{m} [x_i(t, W) - y_i(t)]^2. \]  

(3.2)

That is

\[ \min_{W} E = \min_{W} \| x(W) - y \|^2 \]  

(3.3)

subject to RNN dynamics

\[ x(t + 1) = (1 - h)x(t) + h[\sigma(Wx(t))]. \]  

(3.4)

As we mentioned in Section 1, the gradient descent, conjugate gradient and quasi-Newton’s method have been applied to solve the above least square problems for many decades. In recent years, the growing demand for sophisticated derivative free optimization methods has triggered the development of a relatively wide range of approaches (see [3]). Note that in neural networks, learning is a process of changing the network parameters \( W \) so that the system outputs \( x(t) \) will approach to the target trajectory \( y(t) \). For a fixed \( h \), in the ideal case where the network is exactly capable, that is if \( E(W_0) = 0 \), then the optimal solution can be simply solved by \( W_0 \) where

\[ W_0 = \sigma^{-1}[\frac{C_0 - D_0}{h} - D_0D_0^T(D_0D_0^T)^{-1}, \]  

(3.5)

and matrices

\[ C_0 = [y(m) \cdots y(2)] \quad \text{and} \quad D_0 = [y(m-1) \cdots y(1)]. \]  

(3.6)
For the more general cases, naturally after we obtain \( W_0 \) from (3.5), we generate the whole trajectory set \( X(W_0) \) by
\[
x(t + 1) = x(t) + h[-x(t) + \sigma(W_0x(t))].
\] (3.7)

The basic idea of the state space search algorithm (SSSA) is that we search the class of the \( x \)-convex set \( C_x(A^+) \) in the state space \( R^{m \times n} \) instead of the parameter space of \( W \) in each iteration, where \( A^+ \) is the set of attainable points of \( x \) in \( R^n \). In the next iteration, instead of approximating the whole desired trajectory \( Y \), then we approximate the new set of trajectory \( X_1 \) defined by
\[
X_1 = \alpha Y + (1 - \alpha)X(W_0).
\] (3.8)

In fact, this new trajectory \( X_1 \) is a convex combination of the RNN system output and the desired trajectory \( Y(t) \). If \( \alpha \) is close to 1, the desired trajectory \( Y(t) \) is considered being massaged by the RNN system output. If \( \alpha \) is close to 0, \( Y \) is just a deterministic perturbation to the system output. Geometrically, \( \alpha(Y - X(W_0)) \) defines the error direction. Hence, we may vary \( \alpha \) as \( \alpha_i \) in the \( i^{th} \) learning iteration.

By the continuity of \( W \), there exists some \( \alpha^* \) with \( 0 < \alpha^* < 1 \), such that
\[
X_1 = \alpha^* Y + (1 - \alpha^*)X(W_0),
\]

where \( X_1 \) is attainable in \( C_x(A^+) \), and
\[
E(W) = \min_{\alpha} \{E(W_\alpha)\} \leq E(W_0).
\]

In practice, we only need to store the best solution for each \( \alpha \). Notice that \( X_1 \) may not be attainable even though \( X(W_0) \) is attainable, but we may repeat the process and obtain \( W_1 \) by
\[
W_1 = \sigma^{-1}[C_1 - \frac{D_1}{h} - D_1D_1^TD_1D_1^T]^{-1}
\]

where
\[
C_1 = [X_1(m) \cdots X_1(2)] \quad \text{and} \quad D_1 = [X_1(m - 1) \cdots X_1(1)].
\]

The basic idea of (3.8) assumes a uniform weight for every \( t \) and we search a reachable solution in the neighborhood of \( Y \). In many circumstances we anticipate the system output \( x(t) \) of the RNN will become closer to \( y(t) \) as time goes by. That is, if \( z(t) = \|x(t) - y(t)\|^2 \), then \( z(t) \) tends to 0 as \( t \) tends to \( m \). In view of this, we define the WSSSA by changing the \( k^{th} \) iteration \( \alpha_k \) to \( \alpha_{k,t} \), where
\[
\alpha_{k,t} = \alpha_k(1 + \frac{t}{m}).
\] (3.9)

In our numerical example, we use (3.9) as the weighting rule. Of course, we may have \( \alpha_{k,t} = \alpha_k(1 + \frac{2t}{m}) \) or other composition such as
\[
\alpha_{k,t} = \alpha_k f(t)
\]

where \( f(t) \) is an increasing function of \( t \) with the above consideration. The choice of \( \alpha_{k,t} \) is an art in WSSSA for future research and the optimal choice of \( \alpha^*_{k,t} \) is problem dependent in nature and may be approximated by a line search algorithm in each iteration.
Now, we repeat the above procedure and obtain a sequence of attainable system output \( \{X_k\} = \{X(W_{k-1})\} \) in the state space defined by
\[
\begin{align*}
X_1 &= \alpha_{1,t}^* Y + (1 - \alpha_{1,t}^*) X(W_0), \quad \text{then compute } W_1 \\
X_2 &= \alpha_{2,t}^* Y + (1 - \alpha_{2,t}^*) X(W_1), \quad \text{then compute } W_2 \\
X_3 &= \alpha_{3,t}^* Y + (1 - \alpha_{3,t}^*) X(W_2), \quad \text{then compute } W_3 \\
&\ldots \quad \ldots \quad \ldots \\
X_{k+1} &= \alpha_{k+1,t}^* Y + (1 - \alpha_{k+1,t}^*) X(W_k), \quad \text{then compute } W_{k+1},
\end{align*}
\]
(3.10)
where
\[
W_k = \sigma^{-1}\left[ \frac{C_k - D_k}{h} - D_k \right] D_k^T (D_k D_k^T)^{-1}
\]
where
\[
C_k = [X_k(m) \cdots X_k(2)] \quad \text{and} \quad D_k = [X_k(m - 1) \cdots X_k(1)].
\]
Consequently, we obtain, for each \( k \),
\[
X_{k+1} = \alpha_{k+1,t}^* Y + (1 - \alpha_{k+1,t}^*) X(W_k).
\]
(3.11)
The learning process stops if we obtain a solution \( W_N \) such that the error \( E(W_N) \) is less than a prescribed tolerance, say \( E(W_N) < 10^{-4} \). Nevertheless, in practice, we store the best results for all iterations. For the convergence and the rates of convergence of the SSSA, we cite without proof the following theorem from [9].

**Convergence Theorem.** There exists a limit point \( X^* \) such that the attainable sequence \( \{X_k\} \) of \( X_k \) converges, that is, \( \lim_{k \to \infty} X_k = X^* \). Moreover, (i) if \( \sum_i^{\infty} \alpha_i^* = M < \infty \), then \( X^* = Y + M(X(W_0) - Y) \); (ii) if \( \sum_i^{\infty} \alpha_i^* = \infty \) for nonconstant sequence \( \{\alpha_k^*\} \), then \( X^* = Y \).

The above Convergence Theorem can be extended to WSSSA since \( \sum_t \alpha_{k,t} < 2m\alpha_k \) is finite and bounded in our construction. The WSSSA performs the nonlinear optimization learning process and provides the best feasible solution for the nonlinear optimization problem (3.3). The convergence analysis shows that the network converges to the desired solution and the stability of the algorithm depends on \( \{\alpha_i\}'s \). Geometrically, for each iteration, \( X_k \) defines a homotopy with respect to \( \alpha \) between the desired trajectory \( Y \) and the system output \( X(W_{k-1}) \). Comparison between computational cost with classic gradient type method will be shown in next section. It follows from the fact that the limit point \( X^* \) of \( \{X_k\} \) will lead us to obtain the corresponding best feasible solution \( W^* \). Meanwhile, the error sequence \( \|E(W_k)\| \downarrow 0 \) when \( \lim_{k \to \infty} X_k = X \), while \( \lim_{k \to \infty} \prod_{i=1}^{k} (1 - \lambda \rho^i) = \exp[-\sum_{k=1}^{\infty} \frac{1}{k} \lambda \rho^k] < \exp[-\frac{\lambda}{1 - \rho}] < \infty \) for any \( \rho < 1 \).

Recall that our task is to minimize \( \|x(W) - y\|^2 \) subject to the RNN dynamics. In practice, we may vary \( h \) within a small range in order to find the best initial state with respect to \( h \). Then, we may improve it further by a line search algorithm in each iteration.

Furthermore, we may make a local approximation of \( E(W_{k+1}) \) with respect to \( \alpha \), since \( W_{k+1} \) and \( E(W_{k+1}) \) are relatively easy to compute. Nevertheless, it is important to have an efficient, effective and flexible learning algorithm for the optimization with dynamical constraints. From experimental examples, the WSSSA demonstrated to be a very effective tool to provide extremely promising results for.
learning the RNN dynamics. Here, RNN serves a good example to demonstrate the complexity of a nonlinear discrete-time dynamics that is extremely hard to be solved by classical gradient methods.

4. A comparison study of the SSSA and the gradient based learning algorithm. Our result demonstrate the efficiency of our learning algorithm over the gradient-type methods. In this section, we provide a comparison of the computational operations needed for the SSSA and the gradient based learning algorithm for one iteration for the discrete time RNN (4.1). Note that the complexity of the WSSSA method is same as the SSSA method. Given a sequence \( S(t) \in (-1,1)^n \), for \( 1 \leq t \leq m \), we consider a discrete time RNN similar to (3.1) with a constant step size \( h \). Both systems are equivalent if \( W \) is nonsingular and we may let \( z(t) = Wx(t) \), so that

\[
    z(t + 1) = z(t) + h[-z(t) + W\sigma(z(t))] = (1 - h)z(t) + hW\sigma(z(t)).
\]  

(4.1)

As before, we absorb \( \theta \) in \( W \) for simplicity in the following discussion. For the \( k^{th} \) component of \( z(t) \),

\[
    z_k(t + 1) = (1 - h)z_k(t) + h \sum_{r=1}^{n} W_{kr}\sigma(z_r(t)).
\]  

(4.2)

Recall that the error function

\[
    E(W) = \frac{1}{2} \sum_{t=1}^{p-1} \|z(t + 1) - S(t + 1)\|^2
\]

\[
= \frac{1}{2} \sum_{t=1}^{p-1} \sum_{k=1}^{n} [z_k(t + 1) - S_k(t + 1)]^2.
\]

For a gradient based learning algorithm, we need to compute \( \frac{\partial E}{\partial W_{ij}} \) where

\[
\frac{\partial E}{\partial W_{ij}} = \sum_{t=1}^{p-1} \sum_{k=1}^{n} \left( z_k(t + 1) - S_k(t + 1) \right) \frac{\partial z_k(t + 1)}{\partial W_{ij}}
\]

and

\[
\frac{\partial z_k(t + 1)}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \left[(1 - h)z_k(t) + h \sum_{r=1}^{n} W_{kr}\sigma(z_r(t))\right]
\]

\[
= (1 - h) \frac{\partial z_k(t)}{\partial W_{ij}} + h \sum_{r=1}^{n} \frac{\partial [W_{kr}\sigma(z_r(t))]}{\partial W_{ij}}
\]

\[
= (1 - h) \frac{\partial z_k(t)}{\partial W_{ij}} + h \left\{ \sum_{r=1}^{n} \left[ W_{kr}\sigma'(z_r(t)) \frac{\partial z_r(t)}{\partial W_{ij}} + \delta_{ki} \sigma(z_j(t)) \right] \right\}
\]

subject to initial condition

\[
\frac{\partial z_k(1)}{\partial W_{ij}} = 0
\]

for all \( 1 \leq k \leq n \) and \( 1 \leq i, j \leq n \).

Given a connection weight matrix \( W \), before applying any learning algorithm, we need to compute the entire trajectory \( z(t) \) and \( \sigma(z) \). If we store the intermediate partial derivatives in order to achieve a speedy computation, it requires extra \( np \) function evaluations to approximate \( \sigma'(z) \). For each \( W_{ij} \), it requires \( (2n + 1) \) multiplications to estimate \( \frac{\partial z_k(t + 1)}{\partial W_{ij}} \). If we neglect the computational cost of the addition operations, it takes \( nm(2n + 1) \) multiplications to estimate the partial derivatives.
∂E
∂W
ij
along the trajectory
z(t). Hence, the total computational cost for one gradient iteration is

n^2 \ [2n^2m + 2n] = 2n^3(nm + 1).

(4.4)

In other words, the order of multiplications is 2n^4m, plus nm functional evaluations of \( \sigma'(z) \) together with the trajectory information \( z(t) \) and \( \sigma(z) \).

On the other hand, for the WSSSA approach, besides the trajectory information of \( z(t) \) and \( \sigma(z) \), it takes \( n^2m \) multiplications to form the matrix equation. Hence, we use only \( 5n^3/6 \) multiplications to obtain the connection weight matrix \( W \) for the next iteration, which is much more cost effective than the gradient-based learning algorithms.

5. The empirical example. We assume that there is an implicit dynamics between the exchange rates, so that a short-term trend exists in our foreign exchange series. We try to model this by neural network techniques. Our assumption is based on the result of [17], which they show that statistically the foreign exchange series do not support the random walk hypothesis. Historical data of the foreign exchange rates are available from many web sites in the form of daily averages. We chose to retrieve the inter-bank rate data from OANDA.com. In this example, data consist of the daily exchange rates of seven major currencies, Australian dollar (AUD), Canadian dollar (CAD), Swiss Franc (CHF), Euro (EUR), Sterling (GBP), Russian Rouble (RUB) and Japanese Yen (JPY) compared to the U.S. dollar from Jan 2 to October 2009. We collected 294 daily data for each currency.

For simplicity, let matrices \( A \) and \( B \) of (3.1) be identity matrices, that is,

\[ x(t + 1) = x(t) + h[-x(t) + \sigma(Wx(t) + \theta)]. \]

(5.1)

We may consider the RNN constraint as a time series model, with this setting, if we set \( h = 0 \), the equation becomes

\[ x(t + 1) = x(t), \]

which represents the last-value forecasting of \( x(t + 1) \), that is the naive model for forecasting. When \( h = 1 \), we have

\[ x(t + 1) = \sigma(Wx(t) + \theta), \]

(5.2)

that is a nonlinear time series model. In particular, if \( \sigma \) is the identity mapping, it becomes the multi-linear time series model. These models can be used to capture the behavior of the internal mechanism of the financial market such as foreign exchange rates. Thus, we may regard the RNN as a nonlinear time series model since \( \sigma \) is nonlinear. Therefore, for \( 0 < h < 1 \), we may consider equation (5.2) as the convex combination of two forecasting models.

We use the first 284 days observations to train and validate the RNN model. After the training, we then use the resulting neural network parameters to make the out-of-sample forecasts the last 10 observations. Out-of-sample forecast errors are measured in order to judge how good our model is in terms of its prediction abilities. The learning dynamics used for the discrete-time RNN is equation (3.3).

Before the training process, the data needs to be transformed into an appropriate form for the networks. We employ a normalization for each component series in \( x(t_i) \). Since \( \text{tanh} \) is chosen as the sigmoid activation function that gives an output
in the interval \([-1, 1]\), it is necessary to normalize the data into this interval to avoid working near the asymptote of the sigmoid. This normalization is given by

\[
x(t_i) = \left[ \frac{y(t_i) - \min\{y(t_i)\}}{\text{Range of } y(t_i)} - 0.5 \right] \times 1.90
\]

\(t = 1, 2, 3, ..., 294, \ i = 1, 2, ..., 7.\) (5.3)

as in [5]. The data is then denormalized using the inverse of formula (5.3).

As we knew that this normalization will generally smooth out the extreme outliers, and guarantees \(x(t_i)\) to lie between \(-0.95\) and \(+0.95\), and therefore, they are inside the range of the neural activation function \(\sigma\). In addition, this normalization process will facilitate the computational work in the state space learning process.

Instead of using the normalized raw data to feed the system for learning, we use the moving average series \(z(t)\) of order 5, 10, 20, 50 and 100, respectively, obtained from \(x(t)\). For example, we may take a moving average of 100 terms, that is,

\[
x_i(t) = \frac{1}{100} \sum_{j=t}^{t+99} z_i(j).
\]

In general, the moving average is defined as:

\[
x_i(t) = \frac{1}{n} \sum_{j=t}^{t+n-1} z_i(j).
\]

Also, since the range of the ratios varies for different currencies, we normalize the range of the exchange rates to \([-0.95, +0.95]\). The advantage of using moving averages to model the short-term trend in foreign exchange rates can be found in [17]. We apply the same learning process to the trajectory of the 5, 10, 20, 50, 100 days moving averages of the exchange rate sequences. For simplicity, we set the external force \(J\) to be the zero vector.

In this experiment, with the 100 days moving average, the optimal connection weight matrix \(W\) is an 8 by 8 matrix generated by the WSSSA in the in-sample training process with the threshold \(\theta\) as the 8th input which is a constant vector 1. We use 4000 iterations and the different optimal step sizes \(h\) range from 0.1 to 0.15 for each of the currencies. Software MATLAB was used to conduct all computations and plot the graphs. The total forecasting errors for each of the 7 currencies in the last 10 days are displayed in the following table:

| Currencies | RNN dynamics | RNN Time Series | Linear Time Series |
|------------|--------------|-----------------|--------------------|
| AUD/US     | 0.123641915  | 0.0188989417    | 0.02637907682      |
| CAD/US     | 0.078261204  | 0.0061963128    | 0.03574454831      |
| CHF/US     | 0.082690254  | 0.0091693240    | 0.02031989936      |
| EUR/US     | 0.058972079  | 0.0040683853    | 0.02860011328      |
| GBP/US     | 0.037721388  | 0.0160630437    | 0.00610088947      |
| RUB/US     | 0.05409132   | 0.0224074275    | 0.03001978489      |
| YEN/US     | 0.070544769  | 0.0298373949    | 0.04218799861      |

The linear time series model \(x^c(t+1) = Sy(t)\) is used in the comparison in this study, where \(x^c(t+1)\) is the estimate at time \((t+1)\) and \(S\) is a 7 by 7 matrix. The RNN time series is

\[
x^c(t+1) = y(t) + h[-y(t) + \sigma(Wy(t))].
\]

From the above table, the results of the RNN time series model are usually better than the time series model because the RNN is a nonlinear regression model. On the other hand, it is reasonable that the RNN dynamics model has the largest prediction error because of the accumulation of errors over time if we did not update the information like the regression models. In the given figures, the solid line are the
100 moving average currencies series while the dotted lines are the system outputs of the RNN dynamics.

6. Concluding remarks and further researches. In this paper, we study the nonlinear dynamical system modeling via a recurrent neural network with dynamic constraints. We propose a derivative-free and non-random learning algorithm (WSSSA). We show that our algorithm is stable and convergent. A complexity analysis presented here shows that our algorithm is much simpler than the gradient based methods. We employed the proposed method to learn short term exchange rates and demonstrated its usefulness in performing the short term forecasts in the currency markets.

There are several important directions for future research in this area. Better results are expected if we use a diagonal matrix $H$ instead of a constant step size $h$. Generally speaking, for a nonlinear least-squares optimization problem with the desired trajectory $y(t)$, if our task is to solve the least-squares problem

$$\min_{W} \sum_{t} \|x(t) - y(t)\|^2$$

subject to a dynamical system of constraints,

$$x(t + 1) = F(W, x(t)),$$

where $F(W, x(t))$ is the discrete-time RNN dynamics, our approach can be easily applied to other discrete dynamic $F$ providing that the parameters $W$ of the connection weighted matrix can be obtained in a simple way as $W_0$ in (3.5). For other extensions to the prosed method, it will certainly be of interest to study the asynchronized recurrent networks and other neural network structures.

REFERENCES

[1] A. F. Atiya and A. G. Parlos, New results on recurrent network training: Unifying the algorithms and accelerating convergence, IEEE Transcations on Neural Networks, 11 (2000), 697–709.
[2] Y. Fang and T. W. S. Chow, Non-linear dynamical systems control using a new RNN temporal learning strategy, IEEE Trans on Circuit and Systems, Part II, 52 (2005), 719–723.
[3] R. A. Conn, K. Scheinberg and N. L. Vicente, “Introduction to Derivative-free Optimization,” SIAM, 2009.
[4] J. F. G. Freitas, M. Niranjan, A. H. Gee and A. Doucet, Sequential Monte Carlo methods to train neural network models, Neural Computation, 12 (2000), 955–993.
[5] L. K. Li, Learning sunspot series dynamics by recurrent neural networks, in “Advances in Data Mining and Modeling” (eds. W. K. Ching and K. P. Ng), World Science, (2003), 107–115.
[6] L. K. Li, W. K. Pang, W. T. Yu and M. D. Trout, Forecasting short-term exchange rates: a recurrent neural network approach, in “Neural Networks in Business Forecasting” (eds. G. P. Zhang), Idea Group Publishing, (2004), 195–212.
[7] L. K. Li and S. Shao, Dynamic properties of recurrent neural networks and its approximations, International Journal of Pure and Applied Mathematics, 39 (2007), 545–562.
[8] L. K. Li and S. Shao, A neural network approach for global optimization with applications, Neural Network World, 18 (2008), 365–379.
[9] L. K. Li, S. Shao and T. Zheleva, A state space search algorithm and its application to learn the short-term foreign exchange rates, Applied Mathematical Sciences, 2 (2008), 1705–1728.
[10] X. D. Li, J. K. L. Ho and T. W. S. Chow, Approximation of dynamical time-variant systems by continuous-time recurrent neural networks, IEEE Trans on Circuit and Systems, Part II, 52 (2005), 656–660.
[11] X. B. Liang and J. Wang, *A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints*, IEEE Transactions on Neural Networks, 11 (2000), 1251–1262.

[12] Z. Liu and I. Elhanany, *A Fast and Scalable Recurrent Neural Network Based on Stochastic Meta Descent*, IEEE Transactions on Neural Networks, 19 (2008), 1652–1658.

[13] S. Wang, Q. Shao and X. Zhou, *Knot-optimizing spline networks (KOSNETS) for nonparametric regression*, Journal of Industrial and Management Optimization, 4 (2008), 33-52.

[14] X. Wang and E. K. Blum, *Discrete-time versus continuous-time models of neural networks*, Journal of Computer and System Sciences, 45 (1992), 1–10.

[15] R. J. Williams and D. Zipser, *A learning algorithm for continually running fully recurrent neural networks*, Neural Computation, 1 (1989), 270–280.

[16] L. Xu and W. Liu, *A new recurrent neural network adaptive approach for host-gateway rate control protocol within intranets using ATM ABR service*, Journal of Industrial and Management Optimization, 1 (2005), 389–404.

[17] J. Yao and C. L. Tan, *A case study on using neural networks to perform technical forecasting of forex*, Neural Computation, 34 (2000), 79–98.

[18] K. F. C. Yiu, S. Wang, K. L. Teo and A. H. Tsoi, *Nonlinear system modeling via knot-optimizing B-spline networks*, IEEE Transactions on Neural Networks, 12 (2001), 1013–1022.

[19] K. F. C. Yiu, Y. Liu and K. L. Teo, *A hybrid descent method for global optimization*, Journal of Global Optimization, 28 (2004), 229–238.

Received October 2009; 1st revision July 2010; final revision January 2011.

E-mail address: malblkli@inet.polyu.edu.hk
E-mail address: S.SHAO@csuohio.edu
E-mail address: macyiu@polyu.edu.hk
Figure 1. In the figures, the solid line are the 100 moving average currencies series. The dotted lines are the system outputs of the RNN dynamics.