Chin. J. Chem. Eng., **15**(4) 539—544 (2007)

Neural Network Nonlinear Predictive Control Based on Tent-map Chaos Optimization*

SONG Ying(宋莹), **CHEN Zengqiang**(陈增强)** **and YUAN Zhuzhi** (袁著祉)
Department of Automation, Nankai University, Tianjin 300071, China

Abstract With the unique ergodicity, irregularity, and special ability to avoid being trapped in local optima, chaos optimization has been a novel global optimization technique and has attracted considerable attention for application in various fields, such as nonlinear programming problems. In this article, a novel neural network nonlinear predictive control (NNPC) strategy based on the new Tent-map chaos optimization algorithm (TCOA) is presented. The feedforward neural network is used as the multi-step predictive model. In addition, the TCOA is applied to perform the nonlinear rolling optimization to enhance the convergence and accuracy in the NNPC. Simulation on a laboratory-scale liquid-level system is given to illustrate the effectiveness of the proposed method.
Keywords model-based predictive control, neural network, Tent-map, chaos optimization, nonlinear system

1 INTRODUCTION

Model-based predictive control (MPC) technique is now popular and has been implemented successfully in several industrial processes[1] owing to its special characteristics, such as the capabilities of model-based prediction, rolling optimization, and feedback tuning. Several versions of the MPC technique include dynamic matrix control (DMC)[2], generalized predictive control (GPC)[3], and so on. The above techniques are fundamentally similar since all are based on linear process modeling[4]. However, because of the assumption of linearity of the unknown system parameters, MPC encounters great difficulties when confronted with a system that has high nonlinearity and complexity. Therefore, there are several novel intelligent predictive controllers[5—7] to deal with the complex nonlinear systems.

Recently, under the receding horizon principle and MPC framework, attention has been focused on nonlinear predictive control using a nonlinear model describing the behavior of a system. As a mathematical model for the human brain, the neural network (NN) has been commonly applied in most branches of natural science, and not only the control systems field. The nonlinear modeling capability of NN is well documented[8,9]. Multi-layer feedforward neural networks are the most extensively utilized NNs in control and identification applications. For nonlinear plants, the ability of the MPC to make accurate predictions can be enhanced if a neural network is used to learn the dynamics of the plant instead of the standard nonlinear modeling techniques. The neural-network-based predictive control (NNPC) strategies have been found to be effective in controlling a wide class of nonlinear processes in the past[10—19]. In the NNPC, the neural network will be used as the prediction model of the nonlinear plant and the system performance is greatly dependent on the online optimization procedure. Several algorithms were successfully implemented in the NNPC system, such as the gradient descent method[11—15] and the Newton-Raphson method[16]. The Jacobian or Hessian matrix used for solving the optimization is normally formulated in terms of the structure of the neural network, *i.e.*, weights and biases. To reduce the computational load for a large predictive horizon, Noriega and Wang[17] presented a recursive algorithm for calculating the Jacobian matrix. However, these numerical optimization methods usually provide local optim and require the NNPC cost function which is differential and it is still a complex procedure for calculating the Jacobian or Hessian matrix even under some simplifications; hence, the intelligent algorithms are more suitable for optimizing in NNPC, such as the genetic algorithm (GA)[18] and the particle swarm optimization (PSO)[19].

Chaos is a kind of characteristic of nonlinear systems, which is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and includes infinite unstable periodic motions. A chaotic motion can traverse every state in a certain region (called the chaos space) by its own regularity, and every state is visited only once, and thus, there is no precise periodicity. Owing to the unique ergodicity and special ability to avoid being trapped in local optima, chaos has been a novel optimization technique, and the chaos optimization algorithm (COA)[20] is considerably higher than some other stochastic algorithms.

In this article, a novel neural network nonlinear predictive control strategy based on the new Tent-map chaos optimization algorithm (TCOA) is presented. The neural network, which is trained by the BP algorithm with adaptive learning rate and momentum factor (BPALM)[21], is used as the multi-step predictive model in NNPC. The Tent-map is studied in the mathematics of dynamical systems because it has several interesting properties such as chaotic orbits, simple

Received 2006-09-04, accepted 2007-03-27.
* Supported by the National Natural Science Foundation of China (No.60374037, No.60574036), the Program for New Century Excellent Talents in University of China (NCET), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No.20050055013), and the Opening Project Foundation of National Lab of Industrial Control Technology (No.0708008).
** To whom correspondence should be addressed. E-mail: chenzq@nankai.edu.cn

shape, and so on. Most importantly, the Tent-map shows the outstanding advantages and has higher iterative speed than the Logistic map, because the probability density function of the chaotic sequence for the Tent-map is a uniform function whereas the probability density function of chaotic sequence for the Logistic map is a Chebyshev-type function[22]. The TCOA is applied to perform the nonlinear optimization to enhance the convergence and accuracy. The simulation on a laboratory-scale liquid-level system shows that the method is effective.

2 NEURAL NETWORK PREDICTIVE CONTROL (NNPC)

In contrast to the neural network direct control, the NNPC is more practical. Here, the neural network will be used as the prediction model of the nonlinear plant.

Assume that the unknown nonlinear system is expressed as the input-output form by:

$$y(t)=f\left[y(t-1),\cdots,y\left(t-n_{\mathrm{a}}\right),\right.$$
$$\left.u(t-d),\cdots,u\left(t-d-n_{\mathrm{b}}\right)\right] \tag{1}$$

where, $y(t)$ and $u(t)$ are the output and input of the system, respectively; $f(\cdot)$ is the unknown nonlinear function to be estimated by a neural network; n_{a} and n_{b} are the orders of the system; d is the time delay, which is assumed to be at least one.

Since the input to the neural network is:

$$\boldsymbol{X}(t)=\left[y(t-1),\cdots,y\left(t-n_{\mathrm{a}}\right),\right.$$
$$\left.u(t-d),\cdots,u\left(t-d-n_{\mathrm{b}}\right)\right]^{\mathrm{T}} \tag{2}$$

the neural model for the unknown nonlinear system (1) can be expressed as:

$$\hat{y}(t)=\hat{f}[\boldsymbol{X}(t)] \tag{3}$$

where, $\hat{y}(t)$ is the output of the neural network and $\hat{f}$ is the estimation of f.

The purpose of the NNPC algorithm is to select the control signal $u(t)$ such that the output of the system $y(t)$ is made as close as possible to the set-point $r(t)$. A schematic illustration of the NNPC is given in Fig.1.

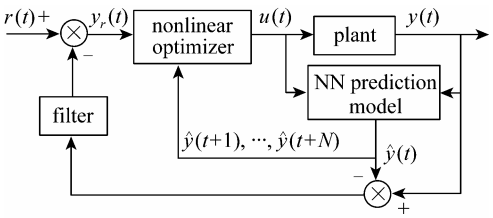

Figure 1 NNPC scheme

The process control input is calculated to minimize a criterion J at each sampling instant t,

$$J=\sum_{j=1}^{N}\left[y_{\mathrm{r}}(t+j)-y(t+j)\right]^2+\lambda\sum_{j=1}^{N_{\mathrm{u}}}\left[\Delta u(t+j-1)\right]^2 \tag{4}$$

where, N is the prediction horizon and N_{u} is the control horizon, and generally $N_{\mathrm{u}}\leqslant N$; λ is the control weighting factor; $u(t)$ is the control signal; Δ is the difference operator, $\Delta u(t)=u(t)-u(t-1)$; $\hat{y}(t+j)$ is the j-step-ahead predicted output by the network prediction model; $y_{\mathrm{r}}(t+j)$ is the j-step-ahead future reference output, which is obtained as follows:

$$\begin{cases}y_{\mathrm{r}}(t)=y(t)\\ y_{\mathrm{r}}(t+j)=\alpha y_{\mathrm{r}}(t+j-1)+(1-\alpha)r(t)\end{cases} \tag{5}$$

where, α is a soft factor, $0\leqslant\alpha<1$, and $r(t)$ represents the real set-points. The purpose of the weighting factor λ is to penalize large change in the process input and reduce actuator wear. It is usual to set λ as a positive constant.

Rewrite the criterion (4) in vector notation as follows:

$$J=\left[\boldsymbol{Y}_{\boldsymbol{r}}(t)-\hat{\boldsymbol{Y}}(t)\right]^{\mathrm{T}}\left[\boldsymbol{Y}_{\boldsymbol{r}}(t)-\hat{\boldsymbol{Y}}(t)\right]+\lambda\Delta\boldsymbol{U}(t)^{\mathrm{T}}\Delta\boldsymbol{U}(t)$$
$$=\boldsymbol{E}^{\mathrm{T}}(t)\boldsymbol{E}(t)+\lambda\Delta\boldsymbol{U}^{\mathrm{T}}(t)\Delta\boldsymbol{U}(t) \tag{6}$$

where,

$$\boldsymbol{Y}_{\mathbf{r}}(t)=\left[y_{\mathrm{r}}(t+1),y_{\mathrm{r}}(t+2),\cdots,y_{\mathrm{r}}(t+N)\right]^{\mathrm{T}}$$
$$\hat{\boldsymbol{Y}}(t)=\left[\hat{y}(t+1),\hat{y}(t+2),\cdots,\hat{y}(t+N)\right]^{\mathrm{T}}$$
$$\boldsymbol{E}(t)=\left[e(t+1),e(t+2),\cdots,e(t+N)\right]^{\mathrm{T}}$$
$$\Delta\boldsymbol{U}(t)=\left[\Delta u(t),\Delta u(t+1),\cdots,\Delta u(t+N_{\mathrm{u}}-1)\right]^{\mathrm{T}}$$
$$\boldsymbol{U}(t)=\left[u(t),u(t+1),\cdots,u(t+N_{\mathrm{u}}-1)\right]^{\mathrm{T}} \tag{7}$$

and

$$e(t+i)=y_{\mathrm{r}}(t+i)-\hat{y}(t+i)\ \text{ for }\ i=1,\cdots,N \tag{8}$$

Using the gradient decent rule, it can be obtained that

$$\boldsymbol{U}(t+1)=\boldsymbol{U}(t)-\eta_{\mathrm{c}}\frac{\partial J}{\partial\boldsymbol{U}(t)} \tag{9}$$

where, η_{c} is a learning rate,

$$\frac{\partial J}{\partial\boldsymbol{U}(t)}=-2\frac{\partial\hat{\boldsymbol{Y}}(t)^{\mathrm{T}}}{\partial\boldsymbol{U}(t)}\boldsymbol{E}(t)+2\lambda\frac{\partial\Delta\boldsymbol{U}(t)^{\mathrm{T}}}{\partial\boldsymbol{U}(t)}\Delta\boldsymbol{U}(t) \tag{10}$$

Since $\Delta u(t)=u(t)-u(t-1)$, there is

$$\frac{\partial\Delta\boldsymbol{U}(t)^{\mathrm{T}}}{\partial\boldsymbol{U}(t)}=\begin{bmatrix}1&0&0&\cdots&0\\-1&1&0&\cdots&0\\0&-1&1&\cdots&0\\\vdots&\vdots&\ddots&\ddots&\vdots\\0&\cdots&0&-1&1\end{bmatrix} \tag{11}$$

$$\frac{\partial\hat{\boldsymbol{Y}}(t)^{\mathrm{T}}}{\partial\boldsymbol{U}(t)}=\begin{bmatrix}\dfrac{\partial\hat{y}(t+1)}{\partial u(t)}&\cdots&\dfrac{\partial\hat{y}(t+1)}{\partial u(t+N_{\mathrm{u}}-1)}\\\vdots&\ddots&\vdots\\\dfrac{\partial\hat{y}(t+N)}{\partial u(t)}&\cdots&\dfrac{\partial\hat{y}(t+N)}{\partial u(t+N_{\mathrm{u}}-1)}\end{bmatrix} \tag{12}$$

It can be seen that each element in the above matrix

can be found by differentiating Eq.(3) with respect to each element in Eq.(7). As a result, the following can be obtained:

$$\frac{\partial\hat{y}(t+n)}{\partial u(t+m-1)}=\frac{\partial\hat{f}(\boldsymbol{X})}{\partial u(t+m-1)}+\sum_{i=m}^{n-1}\frac{\partial\hat{f}(\boldsymbol{X})}{\partial\hat{y}(t+i)}\left[\frac{\partial\hat{y}(t+i)}{\partial u(t+m-1)}\right] \quad (13)$$

$$\text{for}\quad n=1,2,\cdots,N;\quad m=1,2,\cdots,N_{\mathrm{u}}$$

Equation (12) is the well-known Jacobian matrix, which must be calculated using Eq.(13), and every time a new control signal has to be determined. This can result in a large computational load for a big N and N_{u}. To simplify the computation, Noriega and Wang[17] presented a recursive algorithm for calculating the Jacobian matrix. More details of the calculation procedure can be referred to Refs.[11—17].

Most of the previous works are, however, based on the nonlinear programming method, which provides local optimum values only and in addition, these values depend on the selection of the starting point. Moreover, these gradient-based methods require vast cost in the complex calculation of the Jacobian or Hessian of the criterion. To simplify and reduce the computation load of NNPC, the intelligent optimization (gradient-free) methods are more appropriate and flexible.

3 TENT-MAP CHAOS OPTIMIZATION

Chaos, an apparently disordered behavior that is nonetheless deterministic, is a universal phenomenon that occurs in several nonlinear systems. It is featured by highly unstable motion of deterministic systems in a bounded region of the phase space. High instability indicates that the distance of two nearby orbits increases exponentially with time, which is a result of the extreme sensitivity of chaotic systems to the initial conditions. The Lyapunov exponents quantify this property. The magnitude of the Lyapunov exponent represents the principal rate of the orbits' divergence in the phase space. For a one-dimensional dynamical system, $x_{i+1}=f(x_i)$, and the Lyapunov exponent (LE) is defined as the long-time average of the exponent with respect to an orbit:

$$\mathrm{LE}=\lim_{N\to\infty}\frac{1}{N}\sum_{i=0}^{N-1}\ln\left|f'(x_i)\right| \quad (14)$$

Chaos is then characterized by the boundedness of the system trajectories with a positive Lyapunov exponent, which implies that the average gradient of the map is greater than unity, and accordingly, two nearby orbits in phase space diverge at an exponential rate.

It was emphasized that the sensitivity to the initial value suggests the irregularity of the series $\{x_i\}$ generated by chaos. Consider the ith number x_i of the series. It may be possible that x_j with $j>i$ is quite close to x_i. Unless $x_i=x_j$ exactly, however, the part x_i, x_{i+1}, x_{i+2}, ⋯ is very different from the part x_j, x_{j+1}, x_{j+2}, ⋯ owing to the sensitivity to the initial difference.

Although the long-term behavior of a chaotic system shows typical stochastic properties, chaos is not equivalent to a random process. A chaotic motion can traverse every state in a certain region (called the chaos space) by its own regularity, and every state is visited only once, and therefore, there is no precise periodicity. The unique ergodicity and the irregularity of the series generated by chaos make chaotic dynamics a potential candidate in the field of global optimization, namely, the chaos optimization algorithm (COA)[20]. In fact, it has been successfully applied in improving the performance of the genetic algorithm (GA)[23] and particle swarm optimization (PSO)[24], in solving nonlinear optimization problems for the sliding mode control (SMC)[25], and so on.

Chaos variables are almost generated by the Logistic map in the literatures. However, the invariant density (also called the probability density) of the iterates for the Logistic map is:

$$\rho(x)=\frac{1}{2\pi\sqrt{x(1-x)}} \quad (15)$$

The Chebyshev-type distribution function in the interval [0, 1] is shown in Fig.2. As seen in Fig.2, the invariant density of the iterates in the small interval [0, 0.05] and [0.95, 1] is considerably higher than in the other interval [0.05, 0.95]. If the global optimum is not in the interval [0, 0.05] and [0.95, 1] but in [0.05, 0.95], the Logistic-map-based COA (LCOA) may require a large number of iterations. Thus, it affects the global search capacity and computational efficiency.

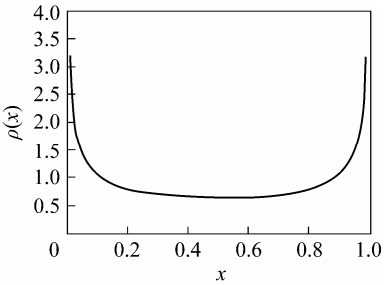

Figure 2 Probability density for the Logistic map

The invariant density of the iterates for Tent-map is:

$$\rho(x)=1 \quad (16)$$

Since the invariant density of the iterates is the uniform distribution function in the interval [0,1], the Tent-map shows outstanding advantages and higher iterative speed than the Logistic map. In this study, the Tent-map is used in chaos optimization to generate the chaotic time series. Consider the equation of Tent-map:

$$x_{i+1}=\begin{cases}2x_i, & x_i\in[0,0.5)\\ 2(1-x_i), & x_i\in[0.5,1]\end{cases} \quad (17)$$

where, x_i is the chaotic variable. Its Lyapunov exponent is:

$$\mathrm{LE}=\ln 2>0 \quad (18)$$

The chaotic evolutions can be generated by Eq.(17), and the ergodic area (*i.e.*, chaos space) is the interval (0, 1). A general procedure of chaos optimization can be found in Ref.[20].

4 NNPC BASED ON TENT-MAP CHAOS OPTIMIZATION

The core problem in NNPC is the online optimization of the criterion Eq.(4). In this study, the control actions $\boldsymbol{U}(t)=[u(t),u(t+1),\cdots,u(t+N_{\mathrm{u}}-1)]^{\mathrm{T}}$ are sought *via* Tent-map chaos optimization.

Since N_{u} components are involved in the control actions $\boldsymbol{U}(t)$, N_{u} initial chaotic variables, $x_{1,0}$, $x_{2,0},\cdots,x_{N_{\mathrm{u}},0}$, $0\leqslant x_{j,0}\leqslant 1$, $j=1,2,\cdots,N_{\mathrm{u}}$, are selected randomly, and the fixed points of the Tent-map, such as 0 and 2/3, cannot be used as initial variables. The lower bounds and upper bounds of the searched variables are denoted as l_{bd_j} and u_{bd_j}, $j=1,2,\cdots,N_{\mathrm{u}}$. J^* and $\boldsymbol{U}^*$ are assumed to be the optimal cost index and the optimizers, respectively. J_{k+1} and $\boldsymbol{U}_{k+1}$ respectively denote the cost index and the NNPC control actions in the (k+1) th iteration.

The major steps of the NNPC based on the Tent-map chaos optimization can be described as follows:

Step 1. At time step t, simultaneously sample inputs and outputs of the process [including $y(t)$].

Step 2. Start with the previously calculated control input vector, and compute the predictive output values using the neural network model, and thus, determine the cost function $J[\boldsymbol{U}(t)]$.

Step 3. Apply the Tent-map-based chaos optimization to calculate a new control input that minimizes criterion (4).

① Chaotify the variables. Substitute $x_{1,k},x_{2,k},\cdots,x_{N_{\mathrm{u}},k}$ in the Eq.(17) to generate N_{u} chaotic variables $x_{1,k+1},x_{2,k+1},\cdots,x_{N_{\mathrm{u}},k+1}$ *via* the Tent-map.

② Perform the transformation from the chaotic space to the solution space using the following formula, namely, the first carrier wave.

$$\boldsymbol{U}_{j,k+1}=l_{\mathrm{bd}_j}+\left(u_{\mathrm{bd}_j}-l_{\mathrm{bd}_j}\right)\cdot x_{j,k+1}$$
$$j=1,2,\cdots,N_{\mathrm{u}} \tag{19}$$

③ Compute the performance index J_{k+1} in Eq.(4), and assign the optima as follows: If $k=0$ or $J_{k+1}\leqslant J^*$; then, $J^*=J_{k+1}$, $\boldsymbol{U}^*=\boldsymbol{U}_{k+1}$; otherwise, do nothing.

Repeat the above ①, ②, and ③ until J^* and $\boldsymbol{U}^*$ do not improve within certain steps, and turn to the next ④.

④ Utilize the Tent-map Eq.(17) again to generate N_{u} chaotic variables $x_{1,k+1}$, $x_{2,k+1},\cdots$, $x_{1,k+1}$ and perform chaos search using second carrier wave.

$$\boldsymbol{U}_{k+1}=\boldsymbol{U}^*+\beta x_{j,k+1} \tag{20}$$

where, $\beta x_{j,k+1}$ is a chaos variable with small ergodic interval; β is an adjusting coefficient, normally chosen as a small positive constant, here, $\beta=0.01$.

⑤ Compute the performance index J_{k+1} in Eq.(4), and assign the new optima; do the same as ③.

Repeat the above ④ and ⑤ until J^* and $\boldsymbol{U}^*$ do not improve within certain steps, and turn to Step ④.

Step 4. Send the first control input $u(t)$ to the process.

Step 5. t:=t+1, turn to Step 1.

5 SIMULATION STUDY

To validate the theoretical developments, the following simulation study is presented.

Example: Control of a laboratory-scale liquid-level system.

This example was taken from Ref.[13, 14]. The model is identified from a laboratory-scale liquid level system. The system consists of a D.C. water pump feeding a conical flask, which in turn feeds a square tank, giving the system second-order dynamics. The controllable input is the voltage to the pump motor, and the plant output is the height of the water in the conical flask. The aim is that the water height must follow some demand signals, and the identified model is given as:

$$\begin{aligned}y(t)=&\,0.9722y(t-1)+0.3578u(t-1)-\\&0.1295u(t-2)-0.3103y(t-1)u(t-1)-\\&0.04228y^2(t-2)+0.1663y(t-2)u(t-2)-\\&0.03259y^2(t-1)y(t-2)-\\&0.3513y^2(t-1)u(t-2)+\\&0.3084y(t-1)y(t-2)u(t-2)+\\&0.1087y(t-2)u(t-1)u(t-2)+\\&0.2573y(t-2)e(t-1)+\\&0.2939y^2(t-2)e(t-1)+\\&0.4770y(t-2)u(t-1)e(t-1)\end{aligned} \tag{21}$$

The desired set-points $r(t)$ are switched between 2 and −2 every 100 iterations, and $e(t)$ is a zero mean white noise sequence with variance 0.1. The initial conditions are set as $y(-1)=0$, $y(-2)=0$.

A feedforward neural network with 4 input neurons, 7 hidden neurons, and 1 output neuron is used, namely 4-7-1 structure. The input signal applied to plant (21) is a finite sequence of uniformly distributed random variables with range [−2, 2]. Thus, it generates input/output samples (patterns), which will be used to train the NN. Among the samples, 100 samples are used as training NN data, while the remaining 100 samples are used as testing NN data. In conventional BP algorithm[26], the input signal is often used to choose a proper learning rate that stays fixed during the whole process of training. Therefore, its convergence tends to be very slow, and it often produces suboptimal solutions. To improve the performance of the BP algorithm, the BP algorithm with adaptive learning rate and momentum factor (BPALM)[21] is employed to train the weights and biases. During the training, the weights and biases of the NN are optimized by the BPALM, which minimize the mean

square error criterion:

$$E_{\mathrm{nn}}=\frac{1}{2p}\sum_{t=1}^{p}\left[y(t)-\hat{y}(t\,|\,\boldsymbol{W})\right]^2 \tag{22}$$

where, p represents the training samples, and $\boldsymbol{W}$ represents the optimized neural weights vector. The weights are therefore updated as follows[21]:

$$\boldsymbol{W}(t+1)=\boldsymbol{W}(t)+\eta(t)\frac{\partial E_{\mathrm{nn}}}{\partial \boldsymbol{W}(t)}+\mu(t)\left[\boldsymbol{W}(t)-\boldsymbol{W}(t-1)\right] \tag{23}$$

where, η and μ are the learning rate and the momentum factor, respectively. η and μ are adaptively adjusted at each iteration given by:

$$\eta(t+1)=\begin{cases}(1+\theta_1)\eta(t), & E_{\mathrm{nn}}(t)<E_{\mathrm{nn}}(t-1)\\ \eta(t) & E_{\mathrm{nn}}(t)=E_{\mathrm{nn}}(t-1)\\ (1-\theta_1)\eta(t), & E_{\mathrm{nn}}(t)>E_{\mathrm{nn}}(t-1)\end{cases} \tag{24}$$

$$\mu(t+1)=\begin{cases}(1+\theta_2)\mu(t), & E_{\mathrm{nn}}(t)<E_{\mathrm{nn}}(t-1)\\ \mu(t) & E_{\mathrm{nn}}(t)=E_{\mathrm{nn}}(t-1)\\ (1-\theta_2)\mu(t), & E_{\mathrm{nn}}(t)>E_{\mathrm{nn}}(t-1)\end{cases} \tag{25}$$

where, $0<\theta_1, \theta_2<1$. In this article, θ_1 and θ_2 are set to be 0.05, and the initial learning rate and the momentum factor are set as $\eta(0)=0.01$, $\mu(0)=0.8$.

The parameters of the NNPC are set as $N=7$, $N_{\mathrm{u}}=4$, $\lambda=0.05$, and $\alpha=0.2$. Figs.3(a), (b), and (c) show the control of the liquid level system by quasi-Netwon, LCOA, and TCOA, respectively. As seen in Fig.3, the proposed controller has a good tracking capability. In addition, the Tent-map chaos optimization (TCOA) method is compared with the quasi-Newton, Logistic-map chaos optimization (LCOA) methods in terms of the mean square tracking error (MSE) as shown in Table 1. It is seen that the TCOA method has smaller mean square tracking error than the quasi-Newton and LCOA methods.

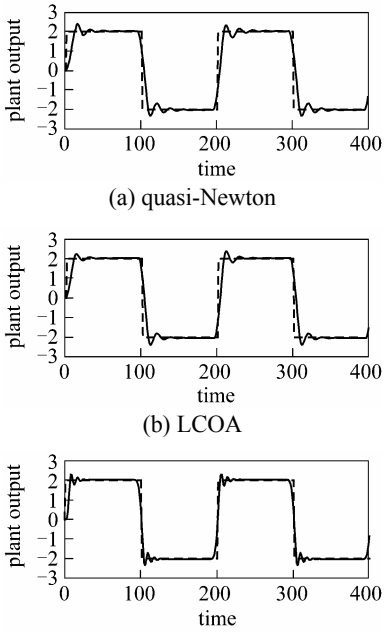

Figure 3 Control of the liquid level system

– – – – $r(t)$; —— $y(t)$

Table 1 Comparison of the mean square tracking error

MSE of quasi-Newton	MSE of LCOA	MSE of TCOA
0.0632	0.0359	0.0181

6 CONCLUSIONS

In this article, a novel nonlinear neural network predictive control (NNPC) strategy based on the new Tent-map chaos optimization algorithm (TCOA) is presented. The neural network is used as the multi-step predictive model and the TCOA is applied to perform the nonlinear rolling optimization to enhance the convergence and accuracy in the NNPC.

The disadvantages of the gradient-based techniques such as slow convergence and dependence on initial values can be addressed by the chaos optimization algorithm. The integration of TCOA with neural network not only avoids the risk of trapping in the local optimum point but also allows neglecting the error gradient information Furthermore, the TCOA can avoid calculating the complex Jacobian or Hessian matrices required in gradient-based methods and reduce the computation loads of the NNPC. The simulation results show that the method is effective.

NOMENCLATURE

d	time delay of system
$\boldsymbol{E}(t)$	output vector of system
E_{nn}	mean square error criterion of NN
$e(t)$	output error of system
$\hat{f}$	estimate of unknown function f
J	cost function of NNPC
J^*	optimal cost index
LE	Lyapunov exponent
l_{bd_j}	j-low- bound of solution space
N	prediction horizon
N_{u}	control horizon
n_{a}	degree of system output
n_{b}	degree of system input
$r(t)$	set-point
$\boldsymbol{U}^*$	optimal input vector of system
$\boldsymbol{U}(t)$	input vector of system
$u(t)$	input of system
u_{bd_j}	j-upper-bound of solution space
$\boldsymbol{W}$	weights vector of neural network
$\boldsymbol{X}(t)$	input vector to the neural network
$\{x_i\}$	series of chaos
$\boldsymbol{Y}(t)$	output vector of system
$\hat{\boldsymbol{Y}}(t)$	output vector of neural network
$\boldsymbol{Y}_{\mathrm{r}}(t)$	reference output vector of system
$y(t)$	output of system
$\hat{y}(t)$	output of the neural network
$y_{\mathrm{r}}(t)$	reference output of system
$y_{\mathrm{r}}(t+j)$	j-step-ahead future reference output
α	soft factor of the reference output
β	adjusting coefficient of chaos optimization
Δ	symbol of backward difference
η	learning rate of neural weights

η_c learning rate of NNPC
θ_1 adjusting coefficient of NN learning rate
θ_2 adjusting coefficient of NN momentum factor
λ control weight factor
μ momentum factor of neural weights
$\rho(x)$ probability density of chaotic map

REFERENCES

1 Henson, M.A., "Nonlinear model predictive control: Current status and future directions", *Computers and Chemical Engineering*, **23**(2), 187—202(1998).
2 Cutler, C.R., Ramaker B.L., "Dynamic matrix control - A computer control algorithm", In: Proc. of the American Control Conf., IEEE Press, Piscataway, NJ (1980).
3 Clarke, D.W., Mohtadi, C., Tuffs, P.S., "Generalized predictive control (Ⅰ) & (Ⅱ)", *Automatica*, **23**(2), 137—160(1987).
4 Chen, Z.Q., Mao, Z.X., Du, S.Z., Sun, Q.L., Yuan, Z.Z., "Analysis of robustness of PID-GPC based on IMC structure", *Chin. J. Chem. Eng*., **11**(1), 55—61(2003).
5 Su, B.L, Chen, Z.Q., Yuan, Z.Z., "Multivariable decoupling predictive control with input constraints and its application on chemical process", *Chin. J. Chem. Eng*., **14**(2), 216—222(2006).
6 Zhong, W.M., He, G.L., Pi, D.Y., Sun, Y.X., "SVM with quadratic polynomial kernel function based nonlinear model one-step-ahead predictive control," *Chin. J. Chem. Eng*., **13**(3), 373—379(2005).
7 Wang, Y.H., Huang, D.X., Jin, Y.H., "A hybrid model predictive control for handling infeasibility and constraint prioritization", *Chin. J. Chem. Eng*., **13**(2), 211—217(2005).
8 Hornik, K., Stinchcombe, M., White, H., "Multilayer feedforward network are universal approximators", *Neural Networks*, **2**(5), 359—366(1989).
9 Chen, T.P., Chen, H., "Approximations of continuous functionals by neural network with application to dynamics systems", *IEEE Trans. Neural Networks*, **4**(6), 910—918(1993).
10 Hussian, M.A., "Review of the applications of neural networks in chemical process control — Simulation and online implementation", *Artificial Intelligence in Engineering*, **13**(1), 55—68(1999).
11 Saint, D.J., Bhat, N., McAvoy, T.J., "Neural net based model predictive control", *Int. J. Control*, **54**(6), 1453—1468(1991).
12 Tan, Y., Cauwenberghe, A., "Nonlinear one-step- ahead control using neural networks: Control strategy and stability design", *Automatica*, **32**(12), 1701—1706(1996).
13 Ahmed, M.S., Anjum, M.F., "Neural-net-based direct self-turning control of nonlinear plants", *Int. J. Control*, **66**(1), 85—104(1997).
14 Li, X., Chen, Z.Q., Yuan, Z.Z., "Simple recurrent neural network-based adaptive predictive control for nonlinear systems", *Asian J. Control*, **4**(2), 231—239(2002).
15 Zhang, Y., Chen, Z.Q, Yang, P., Yuan, Z.Z., "Multi- variable nonlinear proportional- integral-derivative decoupling control based on recurrent neural net-works", *Chin. J. Chem. Eng*., **12**(5), 677—681(2004).
16 Soloway, D., Haley, P.J., "Neural generalized predictive control: A Newton-Raphson implementation", In: Proc. IEEE Int. Symposium on Intelligent Control, IEEE Press, Piscataway, NJ, 277—282(1996).
17 Noriega, J.R., Wang, H., "A direct adaptive neural network control for unknown nonlinear systems and its application", *IEEE Trans. Neural Networks*, **9**(1), 27-34(1998).
18 Shin, S.C., Park, S.B., "GA-based predictive control for nonlinear processes", *Electronics Letters*, **34**(20), 1980—1981(1998).
19 Wang, X., Xiao, J., "PSO-based model predictive control for nonlinear processes", In: Lecture Notes in Computer Science 3611, Springer-Verlag, Berlin, Germany, 196—203(2005).
20 Li, B., Jiang, W.S., "Optimizing complex functions by chaos search", *Cybernetics and Systems*, **29**(4), 409—419(1998).
21 Yu, C.C., Liu, B.D., "A backpropagation algorithm with adaptive learning rate and momentum coefficient", In: Proc. 2002 Int. Joint Conf. Neural Networks, IEEE Press, Piscataway, NJ, 1218—1223(2002).
22 Steeb, W.H., The Nonlinear Workbook: Chaos, Fractals, Cellular Automata, Neural Networks, Genetic Algorithms, Gene Expression Programming, Support Vector Machine, Wavelets, Hidden Markov Models, Fuzzy Logic with C++, Java and SymbolicC++ Programs, 3rd Ed. World Science Publisher, Kackensack, NJ (2005).
23 Yan, X.F., Chen, D.Z., Hu, S.X., "Chaos-genetic algorithms for optimizing the operating conditions based RBF-PLS model", *Computers and Chemical Engineering*, **27**(10), 1393—1404(2003).
24 Liu, B., Wang, L., Jin, Y.H., "Improved particle swarm optimization combined with chaos", *Chaos, Solitons and Fractals*, **25**(5), 1261—1271(2005).
25 Lu, Z., Shieh, L.S, Chen, G.R., Coleman, N.P., "Simplex sliding mode control for nonlinear uncertain systems *via* chaos optimization", *Chaos, Solitons and Fractals*, **23**(3), 747—755(2005).
26 Rumelhart, D.E., Hinton, G.E., Williams R.J., "Learning representations by back-propagating errors", *Nature*, **323**, 533—536(1986).