Journal of Zhejiang University SCIENCE A
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

Neural network method for solving elastoplastic finite element problems*

REN Xiao-qiang (任小强)†, CHEN Wu-jun (陈务军), DONG Shi-lin (董石麟), WANG Feng (王 锋)

(*Space Structures Research Center, Shanghai Jiao Tong University, Shanghai 200030, China*)

†E-mail: renxiaoqiang@sjtu.edu.cn

Received Mar. 20, 2005; revision accepted Apr. 8, 2005

Abstract: A basic optimization principle of Artificial Neural Network—the Lagrange Programming Neural Network (LPNN) model for solving elastoplastic finite element problems is presented. The nonlinear problems of mechanics are represented as a neural network based optimization problem by adopting the nonlinear function as nerve cell transfer function. Finally, two simple elastoplastic problems are numerically simulated. LPNN optimization results for elastoplastic problem are found to be comparable to traditional Hopfield neural network optimization model.

Key words: Elastoplasticity, Finite element method (FEM), Neural network
doi:10.1631/jzus.2006.A0378 **Document code:** A **CLC number:** O344.3

INTRODUCTION

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. In recent years, neural network has been widely applied in the field of engineering construction as a large dimensional nonlinear dynamic system, because of its support for parallel computing (Gao *et al*., 2000).

According to the Parametric Variation Principle (PVP) (Zhong *et al*., 1997), elastoplastic mechanics problem can be dealt with as a mathematical programming problem. Sun *et al*.(1998) transformed the solution of an elastic problem into the solution of a quadratic optimization problem with equalities constraint, which is implemented by using Hopfield neural network model. Sun *et al*.(2000) subsequently constructed a model for elastoplastic finite element problem solvable by the neural networks processing, which, however, may involve computational inaccuracy because the models above were built on a combination of Hopfield model and simulated annealing algorithm. The error may occur due to the local minima trap during annealing implementation. Moreover, there are no consistent rules for finding the parameters of the simulated annealing algorithm, e.g. initial temperature, temperature updating criteria, etc., which adds to the uncertainties of the computational results. Besides, earlier work considered using linear transfer function for solving elastoplastic finite element problems, which is inconsistent with the nonlinear character of mechanics problems.

This paper presents a Lagrange neural network model for solving finite element problems based on the basic optimization principle of artificial neural networks. Lagrange neural network model uses the Lagrange function, a kind of nonlinear function, as the transfer function of the neural network model. The stability and convergence of neural network is discussed, and the neural optimization strategy for increasing computational efficiency of the network

*Project (No. 10102010) supported by the National Natural Science Foundation of China

optimization is analyzed. Finally, the solutions of some numerical problems are presented to illustrate the validity and feasibility of neural network for solving elastoplastic finite element problems.

NEURAL NETWORK MODEL

Neural network is one kind of huge dimension nonlinear dynamic system consisting of neural elements. According to the dynamic system principle, the ultimate behavior of the system is determined by its attractors. The attractors of the dynamic system are considered to be the optimization solution of suitable energy functions (or generalized objective functions). The optimization calculation of the neural network starts from any initial condition, and reaches a stable state with the movement of the system itself. Such a stable state corresponds to a solution for the optimization. This is the principle guiding the application of the neural network model for solving optimization problems.

Considering the standard Quadratic Programming (QP) problem with inequality constraints:

$$\left.\begin{array}{ll}\min & f(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x}/2+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x}\le\boldsymbol{b}\end{array}\right\}, \tag{1}$$

where $\boldsymbol{x}=[x_1,x_2,\ldots,x_n]^{\mathrm{T}}\in\mathbb{R}^n$ is a variable vector, $\boldsymbol{G}$ is the $n\times n$ real symmetry positive definite matrix, $\boldsymbol{A}$ is an $m\times n$ matrix, $\boldsymbol{c}=[c_1,c_2,\ldots,c_n]\in\mathbb{R}^n$ and $\boldsymbol{b}=[b_1,b_2,\ldots,b_m]\in\mathbb{R}^m$ are constant vectors. After introducing the loose variable $\boldsymbol{z}=[z_1,z_2,\ldots,z_m]^{\mathrm{T}}\in\mathbb{R}^m$, Eq.(1) can be transformed into a QP problem with constraints:

$$\left.\begin{array}{ll}\min & f(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x}/2+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}=0\end{array}\right\}. \tag{2}$$

According to the theory of optimization, the Lagrange function can be defined as:

$$L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x}/2+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}), \tag{3}$$

where $\boldsymbol{y}=[y_1,y_2,\ldots,y_m]^{\mathrm{T}}\in\mathbb{R}^m$ is the Lagrange multiplier. According to classic limit theory, the necessary optimality can be expressed as a stationary point $(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)$ of $L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ over $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. That is,

$$\left.\begin{array}{l}\nabla_x L(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)=(\boldsymbol{G}\boldsymbol{x}^*+\boldsymbol{c}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}^*)=0\\ \nabla_y L(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)=(\boldsymbol{A}\boldsymbol{x}^*-\boldsymbol{b}+\boldsymbol{z}^{*\mathrm{T}}\boldsymbol{z}^*)=0\\ \nabla_z L(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)=2\boldsymbol{z}^*\boldsymbol{y}^{*\mathrm{T}}=0\end{array}\right\}. \tag{4}$$

To improve the convergency of the optimization algorithm, we may add a compensation term $\frac{k}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}\right\|$, k>0, to the Lagrange function Eq.(3). The energy function of the dynamic system can be implemented as follows:

$$\begin{aligned}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=&\boldsymbol{x}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{x}/2+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z})\\&+\frac{k}{2}\left\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z}\right\|_2^2.\end{aligned} \tag{5}$$

According to the stability theory of the differential equations, the dynamic system equations can be implemented as,

$$\left.\begin{array}{ll}\mathrm{d}\boldsymbol{u}/\mathrm{d}t=-\nabla_x E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=-(\boldsymbol{G}\boldsymbol{x}+\boldsymbol{c}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y})\\ \mathrm{d}\boldsymbol{v}/\mathrm{d}t=-\nabla_y E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=-(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{z}^{\mathrm{T}}\boldsymbol{z})\\ \mathrm{d}\boldsymbol{w}/\mathrm{d}t=-\nabla_z E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=-2\boldsymbol{z}\boldsymbol{y}^{\mathrm{T}}\\ \boldsymbol{x}_i=g(\boldsymbol{u}_i) & i=1,2,\cdots,n\\ \boldsymbol{y}_j=g(\boldsymbol{v}_j) & j=1,2,\cdots,m\\ \boldsymbol{z}_r=g(\boldsymbol{w}_r) & r=1,2,\cdots,m\end{array}\right\}, \tag{6}$$

where $\boldsymbol{b}$ and $\boldsymbol{c}$ are the network input variables, $\boldsymbol{x}\in\mathbb{R}^n$ is the network output variable. The frame of the neural network corresponding to the above formula is shown in Fig.1. $g(\boldsymbol{u})$ can be considered as a hyper-linear function $g(\boldsymbol{u})=(\mathrm{e}^{\boldsymbol{u}}-\mathrm{e}^{-\boldsymbol{u}})/2$, which serves as the neural transfer function. When the transfer function is a nonlinear function, the computational speed can be increased and the precision for solving elastoplastic

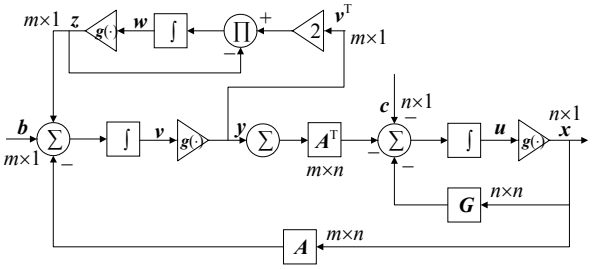

Fig.1 The frame of neural network

finite element problems can be improved because of the inherent nonlinearity of the elastoplastic problem.

STABILITY AND CONVERGENCY OF THE NEURAL NETWORK

The LPNN is globally stable and converges to the optimum solution of QP problem as stated and proved by the Appendix's Theorem.

The QP problem of Eq.(1) is a convex optimization problem (Wu and Tam, 1999). The objective function $f(\boldsymbol{x})$ is a first-order differentiable function in the open convex set $\boldsymbol{x}\in\mathbb{R}^n$. For every $\boldsymbol{x}\in\mathbb{R}^n$, the Hessian $\nabla_{xx}L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is positive definite and $f(\boldsymbol{x})$ is a strictly convex function. So, every local solution $\boldsymbol{x}^*$ of problem Eq.(1) is a global optimum solution, and the set of the global optimum solutions is convex.

The equilibrium point $(\boldsymbol{x}^*,\boldsymbol{y}^*,\boldsymbol{z}^*)$ of the LPPN satisfies a Kuhn-tucker condition of the QP problem (Wu and Tam, 1999; Zhang and Constantinides, 1992). Therefore, the neural network will always converge globally to the optimum solution from any arbitrary initial point.

ELASTOPLASTIC FINITE ELEMENT MODEL

The classic elastoplasticity theory can be simply expressed as follows:

(1) Equilibrium equation:

$$\boldsymbol{A}^{(\nabla)}\mathrm{d}\boldsymbol{\sigma}+\mathrm{d}\boldsymbol{b}=0\,. \tag{7}$$

(2) Geometry equation:

$$\mathrm{d}\boldsymbol{\varepsilon}=\boldsymbol{L}^{(\nabla)}\mathrm{d}\boldsymbol{u}\,. \tag{8}$$

(3) Boundary conditions:

$$\begin{aligned}&\boldsymbol{n}^{(\nabla)}\mathrm{d}\boldsymbol{\sigma}=\mathrm{d}\overline{\boldsymbol{p}},\\&\mathrm{d}\boldsymbol{u}=\mathrm{d}\overline{\boldsymbol{u}}.\end{aligned} \tag{9}$$

(4) Constitutive equations:

$$\left.\begin{aligned}&\mathrm{d}\boldsymbol{\sigma}=\boldsymbol{D}(\mathrm{d}\boldsymbol{\varepsilon}-\mathrm{d}\boldsymbol{\varepsilon}^P)\\&f(\boldsymbol{\sigma},\boldsymbol{\varepsilon}^P,\boldsymbol{\kappa})\le 0\\&\mathrm{d}\boldsymbol{\varepsilon}^P=(\partial g/\partial\boldsymbol{\sigma})\lambda\\&\lambda\begin{cases}\ge 0 & f=0\\ =0 & f<0\end{cases}\end{aligned}\right\}. \tag{10}$$

The meanings of the parameters are the same as that of the traditional expressions and can be found in (Zhong *et al*., 1997).

Assuming that there is only one yield function condition, the loading function can be expressed as (Taylor decomposition),

$$\begin{aligned}f=f^0+\left(\frac{\partial f}{\partial\boldsymbol{\sigma}}\right)^{\mathrm{T}}\mathrm{d}\boldsymbol{\sigma}+\left(\frac{\partial f}{\partial\boldsymbol{\varepsilon}^P}\right)^{\mathrm{T}}\mathrm{d}\boldsymbol{\varepsilon}^P\\+\left(\frac{\partial f}{\partial\boldsymbol{\kappa}}\right)\mathrm{d}\boldsymbol{\kappa}+O^2\left(\mathrm{d}\boldsymbol{\sigma},\mathrm{d}\boldsymbol{\varepsilon},\mathrm{d}\boldsymbol{\kappa}\right),\end{aligned} \tag{11}$$

here f^0 is the loading function before increment steps. If the nonce state is loading, and $f=f^0$ has been scattered by finite element method, then the potential energy of the system will become:

$$\left.\begin{aligned}&\Pi[\lambda(\cdot)]=\frac{1}{2}\mathrm{d}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{K}\mathrm{d}\boldsymbol{u}-\mathrm{d}\boldsymbol{u}^{\mathrm{T}}(\boldsymbol{\Phi}\lambda+\boldsymbol{P})\\&\boldsymbol{C}\mathrm{d}\boldsymbol{u}-\boldsymbol{U}\lambda-\boldsymbol{d}+\boldsymbol{v}=0\\&\boldsymbol{v}^{\mathrm{T}}\lambda=0,\boldsymbol{v}\ge 0,\lambda\ge 0\end{aligned}\right\}, \tag{12}$$

where,

$$\boldsymbol{K}=\sum_{e=1}^{N_E}\int_{\Omega_e}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{B}\mathrm{d}\Omega,\ \boldsymbol{\Phi}=\sum_{e=1}^{N_E}\int_{\Omega_e}\boldsymbol{B}^{\mathrm{T}}\left(\frac{\partial g}{\partial\boldsymbol{\sigma}}\boldsymbol{D}\right)^{\mathrm{T}}\mathrm{d}\Omega\,,$$

$$\boldsymbol{P}=\sum_{e=1}^{N_E}\int_{\Omega_e}(\boldsymbol{N}^{\mathrm{T}}\mathrm{d}\boldsymbol{b})\mathrm{d}\Omega+\sum_{e=1}^{N_E}\int_{S_p}(\boldsymbol{N}^{\mathrm{T}}\mathrm{d}\boldsymbol{p})\mathrm{d}S\,,$$

$$\boldsymbol{C}=\sum_{e=1}^{N_P}\int_{\Omega_e}\left[\left(\frac{\partial f}{\partial\boldsymbol{\sigma}}\boldsymbol{D}\right)\boldsymbol{B}\right]\mathrm{d}\Omega\,,$$

$$\boldsymbol{U}=\sum_{e=1}^{N_P}\int_{\Omega_e}\left[\left(\frac{\partial f}{\partial\boldsymbol{\sigma}}\right)^{\mathrm{T}}\boldsymbol{D}\left(\frac{\partial g}{\partial\boldsymbol{\sigma}}\right)-\left(\frac{\partial f}{\partial\boldsymbol{\varepsilon}^P}\right)^{\mathrm{T}}\left(\frac{\partial g}{\partial\boldsymbol{\sigma}}\right)\right]\mathrm{d}\Omega\,,$$

$$\boldsymbol{d}=\sum_{e=1}^{N_P}\int_{\Omega_e}f^0\mathrm{d}\Omega\,.$$

In the above nonlinear program problems, λ is parametric variation, and can be turned into the standard nonlinear program problems below:

$$\left.\begin{aligned}\min\quad & \boldsymbol{V}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{V}/2+\boldsymbol{V}^{\mathrm{T}}\boldsymbol{I}\\ \text{s.t.}\quad & \boldsymbol{A}\boldsymbol{V}\le\boldsymbol{J}\end{aligned}\right\}, \tag{13}$$

where,

$$\boldsymbol{G}=\begin{bmatrix}\boldsymbol{K} & -(\boldsymbol{\Phi}/2+\boldsymbol{u}\boldsymbol{Q}^{\mathrm{T}})\\ -(\boldsymbol{\Phi}/2+\boldsymbol{u}\boldsymbol{Q}^{\mathrm{T}}) & 2\boldsymbol{u}\boldsymbol{Q}\end{bmatrix},$$

$$\boldsymbol{A}=\begin{bmatrix}-\boldsymbol{Q} & \boldsymbol{U}\\ \boldsymbol{0} & \boldsymbol{E}\end{bmatrix},\quad \boldsymbol{V}=[\mathrm{d}\boldsymbol{u}\quad \lambda^{P}]^{\mathrm{T}},$$

$$\boldsymbol{I}=[-\boldsymbol{P}\quad \mu\boldsymbol{d}]^{\mathrm{T}},\ \boldsymbol{J}=[-\boldsymbol{d}\quad \boldsymbol{0}]^{\mathrm{T}}.$$

The standard optimization problem with inequality constraints, can be applied to solve the elastoplastic mechanics problems by using Lagrange neural network model.

EXAMPLES

Example 1

Flow analysis of layer rock material (Zhong *et al*., 1997). As shown in Fig.2, I and II are elastic units, and III is an elastoplastic element. The units are all isotropic. E=5×10^5, μ=0.25. The yield function and potential function can be considered as:

$$f=|\tau_{xy}|+\tan\varphi\sigma_y\le 0,\ \ g=|\tau_{xy}|+\varphi\tan\varphi\sigma_y+C_0.$$

The example has symmetry: $f_1=\sigma_y\le 0$, $g_1=f_1$. Table 1 shows that the result of the Lagrange programming neural network model is the same as that in (Zhong *et al*., 1997).

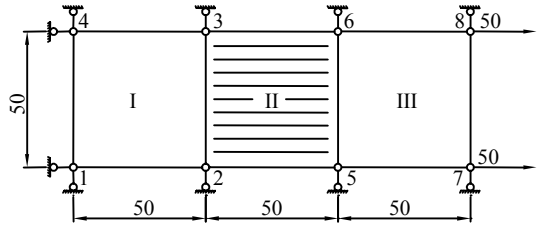

Fig.2 Layer rock units

Table 1 Displacement of unit nodes (m)

Ref.	U2	U5	U7
Zhong *et al*., 1997	1.667×10^{-4}	3.542×10^{-4}	5.208×10^{-4}
This article	1.667×10^{-4}	3.542×10^{-4}	5.208×10^{-4}

Example 2

As shown in Fig.3, the three members symmetric truss structure has a perpendicular force, P=3.0×10^5 kN. The areas of the three members are A=1000 mm^2. The three members are all ideal elastoplastic material, whose elastic modulus is E^0=2.0×10^5 MPa, and whose yield limit is σ_s=200 MPa. The loading function $f(\sigma)=\sigma_i-\sigma_s$.

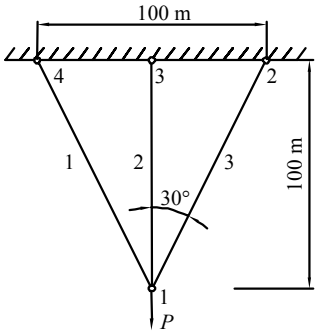

Fig.3 The truss

The results of LPNN are shown in Table 2. In this example, the result of the Lagrange programming neural network model is almost the same as that of the Hopfield neural network.

Table 2 Result of the truss

Ref.	$\sigma_{1y}\times E_0$	$\lambda_1\times E_0$	$\lambda_2\times E_0$	$\lambda_3\times E_0$
Theory	13048.9	0	0	0
Sun *et al*., 2000	13042.6	0	0	0
This article	13047.5	0	0	0

CONCLUSION

Based on the PVP, the finite element computation of elastoplastic mechanics was transformed into a quadratic programming problem with inequality constraints, which can be solved by using the Lagrange programming neural network. LPNN model is characterized by global convergence of the solution for optimization. The computation of the problem is equivalent to the dynamic stabilizing procedure of the LPNN system, the final stable equilibrium point corresponds to the solution of the elastoplastic finite element problems.

APPENDIX A

Theorem 1 The Hessian $\nabla_{xx}L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})=\boldsymbol{G}$ is positive definite everywhere in the dynamic domain of the LPNN of QP problem. The neural network is Lyapunov stable.

Proof Differentiating function $E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ with respect

to time t along the trajectory of the neural network gives:

$$\begin{aligned}\frac{\mathrm{d}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\mathrm{d}t}&=\sum_{i=1}^{n}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{x}_i}\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t}\\&\quad+\sum_{j=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{y}_j}\frac{\mathrm{d}\boldsymbol{y}_j}{\mathrm{d}t}+\sum_{k=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{z}_k}\frac{\mathrm{d}\boldsymbol{z}_k}{\mathrm{d}t}\\&=\sum_{i=1}^{n}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{x}_i}\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}\boldsymbol{u}_i}\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}+\sum_{j=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{y}_j}\frac{\mathrm{d}\boldsymbol{y}_j}{\mathrm{d}\boldsymbol{v}_j}\frac{\mathrm{d}\boldsymbol{v}_j}{\mathrm{d}t}\\&\quad+\sum_{k=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{z}_k}\frac{\mathrm{d}\boldsymbol{z}_k}{\mathrm{d}\boldsymbol{w}_k}\frac{\mathrm{d}\boldsymbol{w}_k}{\mathrm{d}t}\\&=\sum_{i=1}^{n}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{x}_i}g'(\boldsymbol{u}_i)\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t}+\sum_{j=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{y}_j}g'(\boldsymbol{v}_j)\frac{\mathrm{d}\boldsymbol{v}_j}{\mathrm{d}t}\\&\quad+\sum_{k=1}^{m}\frac{\partial E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})}{\partial \boldsymbol{z}_k}g'(\boldsymbol{w}_k)\frac{\mathrm{d}\boldsymbol{w}_k}{\mathrm{d}t}\qquad\text{(A1)}\\&=[\nabla_{\boldsymbol{x}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{u}}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t}+[\nabla_{\boldsymbol{y}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{v}}\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}\\&\quad+[\nabla_{\boldsymbol{z}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{w}}\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t}\\&=-[\nabla_{\boldsymbol{x}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{u}}\nabla_{\boldsymbol{x}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\\&\quad-[\nabla_{\boldsymbol{y}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{y}}\nabla_{\boldsymbol{y}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\\&\quad-[\nabla_{\boldsymbol{z}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})]^{\mathrm{T}}G'_{\boldsymbol{w}}\nabla_{\boldsymbol{z}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\le 0,\end{aligned}$$

where $\nabla_{\boldsymbol{x}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$, $\nabla_{\boldsymbol{y}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$, $\nabla_{\boldsymbol{z}}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is the partial derivative of $E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\boldsymbol{G}_{\boldsymbol{u}}'=\mathrm{diag}(g'(u_1), g'(u_2), ..., g'(u_n))$, $g'(u_i)>0$, $i=1,2,...,n$; $\boldsymbol{G}_{\boldsymbol{v}}'=\mathrm{diag}(g'(v_1), g'(v_2), ..., g'(v_m))$, $g'(v_j)>0$, $j=1,2,...,m$; $\boldsymbol{G}_{\boldsymbol{z}}'=\mathrm{diag}(g'(w_1), g'(w_2), ..., g'(w_m))$, $g'(w_k)>0$, $k=1,2,...,m$. The energy function $E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is always negative. Obviously, $E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is lower bounded, and when and only when $\mathrm{d}\boldsymbol{u}/\mathrm{d}t=0$ and $\mathrm{d}\boldsymbol{v}/\mathrm{d}t=0$, $\mathrm{d}\boldsymbol{w}/\mathrm{d}t=0$, we have $\mathrm{d}E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})/\mathrm{d}t=0$. So $E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is the Lyapunov function of the system, and the LPNN is Lyapunov stable.

References

Gao, H.S., Li, H.D., Ye, T.L., 2000. Neurocomputing in structural analysis and design: An overview. *Chinese Journal of Computational Mechanics*, **17**(2):223-228 (in Chinese).

Sun, D.H., Hu, Q., Xu, H., 1998. Real time neurocomputing theory and numerical simulation on elastic mechanics. *Acta Mechanica Sinica*, **30**(3):348-352.

Sun, D.H., Sun, X.F., Hu, Q., 2000. Model for solving the elastoplasticity based on neural networks. *Chinese Journal of Computational Mechanics*, **17**(3):273-277 (in Chinese).

Wu, A.I., Tam, P.K.S., 1999. A neural network methodology and strategy of quadratic optimisation. *Neural Comput. & Applic.*, **8**(4):283-289. [doi:10.1007/s005210050033]

Zhang, S.W., Constantinides, A.G., 1992. Lagrange programming neural networks. *IEEE Trans. on Circuits and Systems II Analog and Digital Signal Processing*, **39**(7):441-452. [doi:10.1109/82.160169]

Zhong, W.X., Zhang, H.W., Wu, C.W., 1997. Parametric Variation Principle and Applications in Engineering. Science Press, Beijing, p.1-123 (in Chinese).