Neural Comput & Applic (2010) 19:875–882
DOI 10.1007/s00521-010-0353-0

ORIGINAL ARTICLE

Neurocomputing method based on structural finite element analysis of discrete model

Haibin Li · Wei Duan · Hong-Zhong Huang

Received: 25 May 2007 / Accepted: 25 February 2010 / Published online: 19 March 2010

Abstract Based on structural finite element analysis of discrete models, a neurocomputing strategy is developed in this paper. Dynamic iterative equations are constructed in terms of neural networks of discrete models. Determination of the iterative step size, which is important for convergence, is investigated based on the positive definiteness of the finite element stiffness matrix. Consequently, a method of choosing the step size of dynamic equations is proposed and the computational formula of the best step size is derived. The analysis of the computing model shows that the solution of finite element system equations can be obtained by the method of neural network computation efficiently. The proposed method can be used for parallel computation of structural finite element in a large-scale integrated circuit (LSI).

Keywords Finite element · Neural computation · Discrete model · Structural analysis · Large-scale integrated circuit (LSI)

H. Li · W. Duan
College of Science, Inner Mongolia University of Technology, 010051 Huhhot, China
e-mail: lhbnm2002@163.com

W. Duan
e-mail: duan_3@163.com

H.-Z. Huang (✉)
School of Mechanical, Electronic, and Industrial Engineering, University of Electronic Science and Technology of China, 611731 Chengdu, Sichuan, China
e-mail: hzhuang@uestc.edu.cn

1 Introduction

Finite element method has been developed rapidly during the last century and has become one of the most effective structural analysis tools today. However, when structural problems to be analyzed are large-scale and complex, finite element computation method based on the traditional serial computer with limited memory capacity and speed becomes inefficient. To overcome this drawback, researchers have committed themselves to develop a more efficient finite element analysis method.

Parallel computing was first introduced in 1970s, and its idea was originated from the parallelism of many practical application procedures [1]. In finite element analysis, sub-structure method is an effective and common application of parallel computing. Sub-structure method transforms the system equations of entire structure into a few interface equations of the sub-structure connecting area. According to the results from the crunode, each interior region can be divided into many sub-regions for solution evaluation. Consequently, the analysis results of the entire structure can be obtained by parallel computing executed on different computers. The advantage of this approach is that the size of system equations of the interface crunode is less than that of the original structure. Therefore, the computational time will be reduced greatly [2]. However, with the increasing numbers of computers needed for parallel computing of complicated structural analysis, the communication cost is also increased. This will reduce the effectiveness of the parallel computing and greatly affect the application of the finite element parallel computational method.

Neural network is a complex nonlinear system and has highly parallel computational capability. In 1980s, the technology of the artificial neural networks was revived, and its application domain has been expanded unceasingly to optimized computation, association memory, automatic control, matrix algebra, computational mechanics, etc. In 1991, Hajela and Berke described the mapping of neural network among nodal displacement, nodal stress of truss and the area of section using BP network. Instead of using finite element method, the whole process was completed using mapping of neural network, and the approximate analysis method of structure based on neural network [3]. Lee explained the approximate analysis method of structure based on neural networks. He discussed how to choose the quantity of network training samples, the studying error, the number of hidden layer elements, the style of neural activation function and learning rate [4]. In the above-mentioned applications, neural network is treated as a black box system. The focus is on the relationship between stimulation and response, but neglects the merge of existing structural analysis method. As a result, the accuracy of these methods is low. On the other hand, combining the finite element method with neural network forms a promising approach. A neural network based on global flexibility simulating (GFS) is discussed in Ref. [5]. In this method, the mapping between vector of nodal load and vector of nodal deformation was simulated using neural network. A well-trained neural network can be a solver of finite element equations, and the speed of solving equations could be improved. A neural network with fixed weights was constructed according to finite element equations, and the calculation process was considered as a pattern recognition problem [6]. The input variables of the network are adjusted by boundary condition and reverse training algorithm. When the training converges, the input of network is the result. This method was applied to solve the poisson equation of circuit problem. In Ref. [7], the conjugate-gradient method was combined with neural network calculation, and the precision and efficiency were improved. The element stiffness matrix was simulated by sub-neural network in Ref. [5], and then sub-neural networks were combined to a network system based on the matrix transformation. The network should be calculated using conjugate-gradient method, which is called the neural network method based on element stiffness simulating (ESS). In Ref. [8], a structure controllable multi-layer forward neural network (SCNN) was applied to solve finite element linear equations. All the above-mentioned methods use trained forward neural networks. However, attaching the local minimum easily and weak generalization are two inevasible deficiencies. Because of this, the above-mentioned methods have rarely been applied to finite element analysis.

Some researchers treated the finite element problems as optimization problems and solved them using feedback neural network. In 1995, a method solving the structural finite element using feedback interconnection neural network was proposed by Zeng [9]. In 1996, Genki and Yagawa proposed a finite element neural computation method of Poisson's equation. They used a feedback neural network, which was discussed in Ref. [10]. Lou and Perez [11] and Sun et al. [12] proposed structural analysis neural networks using the comparability of finite element method and neural networks. Gao et al. [13] developed a structural finite element method based on the feedback neural network proposed by Cohen and Grossberg (CG neural network). Li et al. [14] proposed a computational finite element method based on a linear saturated system model (LSSM) neural network. All these feedback neural networks are based on continuous optimization computing systems. Although they can be translated into large-scale analog circuit, it is still very difficult to take LSI, which has precision resistance, capacitance and operational amplifier into practice due to technical limitations. When comparing with the analog circuit, digital circuit is much easier because it can process binary information. The computational system of the finite element neural networks based on digital circuit is the potential direction of the finite element parallel calculation. Therefore, neural network computing method is studied using the discrete model in this paper.

2 Theoretical analysis

Digital circuits process numerical information. The characteristic of the digital signal in the time scale or size scale is discrete. Therefore, the corresponding neural network system is also a discrete system. Because the finite elements are assembled by the unit combination, the total potential energy and the restraint of the system can be written in the following form [7]:

$$\begin{aligned} &\min_{x\in\Omega} \prod = \frac{1}{2}\delta^T \mathbf{K}\delta - \mathbf{q}^T\delta \\ &\text{s.t } \mathrm{A}\delta = \bar{\delta}(\text{on } S_u) \end{aligned} \tag{1}$$

where $\mathbf{K}$ is the global stiffness matrix, δ is the solution vector, $\mathbf{q}$ is the nodal loading array, A is the constraint matrix, $\bar{\delta}$ is the displacement constraint array and S_u is the constrained surface.

If the constraint is taken out, Eq. 1 is transformed to the following unconstrained optimization problem [12]:

$$\min_{x\in\Omega} \prod = \frac{1}{2}\delta^T \mathbf{K}'\delta - \mathbf{q}'^T\delta \tag{2}$$

where $\mathbf{K}'$ is the global stiffness matrix, and $\mathbf{q}'$ is the loading matrix. Here, the constraint has been removed. As the finite element theory, $\mathbf{K}'$ is a symmetric positive definite matrix, and the superscript T denotes the transpose operator.

Taking the right-hand side of Eq. 2 as the neural network energy function, the finite element solution could be obtained by constructing the following discrete dynamic equation of neural networks.

$$\delta^{k+1} = \delta^k + (-\mathbf{K}'\delta^k + \mathbf{q}')\Delta t \tag{3}$$

where k indicates the kth iteration and Δt is step size.

2.1 Stability analysis

The study of stability is important in neural computation. It is useful when the state of the neural computation of the whole neural network changes in the prospective direction. Because of this, the stability analysis of the constructed neural networks is necessary. Running neural networks in Eq. 3, and taking Π^k as the energy function of the kth iteration step, the increment of the energy function $\Delta\Pi^k$ can be expressed as:

$$\begin{aligned}\Delta\Pi^k &= \frac{1}{2}(\delta^{k+1})^T\mathbf{K}'(\delta^{k+1}) - \mathbf{q}'^T(\delta^{k+1}) - \frac{1}{2}(\delta^k)^T\mathbf{K}'\delta^k + \mathbf{q}'^T\delta^k \\ &= \frac{1}{2}(\delta^k)^T\mathbf{K}'\Delta\delta + \frac{1}{2}\Delta\delta^T\mathbf{K}'\delta^k + \frac{1}{2}\Delta\delta^T\mathbf{K}'\Delta\delta - \mathbf{q}'^T\Delta\delta\end{aligned} \tag{4}$$

where $\Delta\delta = \delta^{k+1} - \delta^k$.

Because $\mathbf{K}'$ is a positive definite matrix, Eq. 4 could be simplified as:

$$\begin{aligned}\Delta\Pi^k &= (\delta^k)^T\mathbf{K}'\Delta\delta + \frac{1}{2}\Delta\delta^T\mathbf{K}'\Delta\delta - \mathbf{q}'^T\Delta\delta \\ &= ((\delta^k)^T\mathbf{K}' - \mathbf{q}'^T))\Delta\delta + \frac{1}{2}\Delta\delta^T\mathbf{K}'\Delta\delta\end{aligned} \tag{5}$$

It follows from Eq. 3 that

$$-\mathbf{K}'\delta^k + \mathbf{q}' = \Delta\delta/\Delta t \tag{6}$$

Plugging Eq. 6 into Eq. 5, Eq. 7 can be derived as

$$\begin{aligned}\Delta\Pi^k &= \Delta\delta^T\frac{1}{\Delta t}\Delta\delta + \frac{1}{2}\Delta\delta^T\mathbf{K}'\Delta\delta \\ &= \Delta\delta^T\left(-\frac{E}{\Delta t} + \frac{1}{2}\mathbf{K}'\right)\Delta\delta\end{aligned} \tag{7}$$

where E is a unit matrix.

As a positive definite matrix, $\mathbf{K}'$ can be written as

$$\mathbf{K}' = \mathrm{U}^T \cdot \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \cdot \mathrm{U} \tag{8}$$

where λ_i is the ith eigenvalue of $\mathbf{K}'$ and U is a orthogonal matrix.

Plugging Eq. 8 into Eq. 7, we have

$$\Delta\Pi^k = (\mathrm{U}\Delta\delta)^T\left(\begin{bmatrix}\frac{\lambda_1}{2} - \frac{1}{\Delta t} & 0 & \cdots & 0 \\ 0 & \frac{\lambda_2}{2} - \frac{1}{\Delta t} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{\lambda_n}{2} - \frac{1}{\Delta t}\end{bmatrix}\right)\mathrm{U}\Delta\delta \tag{9}$$

We know from Eq. 9 that if $1/\Delta t > \lambda_n/2$, for discretional U and $\Delta\delta$, $\Delta\Pi^k < 0$. In other words, the energy function of the neural networks has the descendent monotony when Eq. 10 is contented with the step size Δt.

$$\Delta t < 2/\lambda_n \tag{10}$$

From the structural finite element theory, the whole stiffness matrix with a constrained boundary is a positive definite matrix [12]. Because the eigenvalues of positive definite matrix are greater than zero, the result of Eq. 10 are not infinite.

Since $\Delta\Pi^k < 0$ and the energy functions Π^k are bounded, the stable state of the networks dynamic equations in Eq. 3 corresponds to the minimum of the energy of the neural networks. Equation 2 is a convex optimization problem so the neural networks have a unique minimum. So, the stable result of the network is the digital solution to the finite element system of equations.

2.2 Computational speed

The process converging to the theoretical value of the neural networks is the minimizing descendent process of the energy function of neural network. For each iteration, the faster the neural network converges, the faster the function of the neural network declines. From Eq. 9, we see that $\Delta\Pi^k$ is a function of Δt. It is necessary to analyze the effect of Δt on $\Delta\Pi^k$ in order to make the energy function of networks $\Delta\Pi^k$ converge to the minimum fast. Equation 3 shows that:

$$\delta^{k+1} - \delta^k = (-\mathbf{K}'\delta^k + \mathbf{q}')\Delta t \tag{11}$$

Plugging Eq. 11 into Eq. 9, the following equation can be obtained:

$$\Delta\Pi^k = a(k)\Delta t^2 - b(k)\Delta t \tag{12}$$

where

$$a(k) = (\mathrm{U}(-\mathbf{K}'\delta^k + \mathbf{q}'))^T\begin{bmatrix}\frac{\lambda_1}{2} & 0 & \cdots & 0 \\ 0 & \frac{\lambda_2}{2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\lambda_n}{2}\end{bmatrix}$$
$$(\mathrm{U}(-\mathbf{K}'\delta^k + \mathbf{q}'))$$

$b(k) = (\mathrm{U}(-\mathbf{K}'\delta^k + \mathbf{q}'))^{T'}(\mathrm{U}(-\mathbf{K}'\delta^k + \mathbf{q}'))$

The theory of extreme value of function indicates that when the step size

$$\Delta t = \frac{b(k)}{2a(k)}$$

$\Delta\Pi^k$ has the maximum value of negative sense (maximal descendent amount) in an iteration, and the neural networks has the fastest convergent rate. So Δt is called the steepest descent energetic step size (the steepest descent step size).

Because choosing the steepest descent step size is very computationally costly, the upper limit of the maximum eigenvalue may be used with the Gershgorin disk theorem [13]. The step size, Δt_a, can also be obtained by Eq. 10:

$$\Delta t_a = \frac{2}{\max\limits_{i\in\{1,2,\ldots n\}}\left\{|a_{ii}| + \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|\right\}} \tag{13}$$

where n is the general stiffness matrix order.

3 Emulating computation

Example 1 Suppose that there is a triangle plate with thickness $t = 0.1$ mm, elastic modulus of the material $E = 1000$ Pa, the Poisson's ratio $\mu = 0$ and $F = 10$ kN. Figure 1 shows the dimension and the obligatory case. The triangle plate is meshed into four triangular elements, totaling six nodes in Fig. 1. Without considering the effect of the gravity, the routine method of finite element and the computational method of the neural network of the finite element are adopted to solve the displacement of every node separately.

First, the element stiffness matrix of every triangular element are calculated, and then the general manipulative equations are obtained after combining the element, introducing the boundary conditions and correcting the general stiffness matrix.

$$\begin{bmatrix} 0.5 & -0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ -0.5 & 1.5 & -0.25 & -0.5 & 0.25 & 0.0 \\ 0.0 & -0.25 & 1.5 & 0.25 & -0.5 & 0.0 \\ 0.0 & -0.5 & 0.25 & 1.5 & -0.25 & 0.0 \\ 0.0 & 0.25 & -0.5 & -0.25 & 1.5 & -0.5 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.5 & 0.5 \end{bmatrix} \begin{Bmatrix} \mathrm{u}_1 \\ \mathrm{u}_2 \\ \mathrm{u}_3 \\ \mathrm{u}_3 \\ \mathrm{u}_5 \\ \mathrm{u}_6 \end{Bmatrix} = \begin{Bmatrix} -1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{Bmatrix}$$

According to Eq. 3 and constructing neural network as shown in Fig. 2, the system of neural network could be obtained.

In Fig. 2, u_i (variables to be obtained) is the output, and q_i (load vector) is the input. In one iteration, u_i are fed back parallel to computing elements laid in ith row, and then 22 multiply operations are processed. Here, k_{ji} is an element in the ith row, jth column of global stiffness matrix. After doing accumulation in six elements named 'f' (multiple input and single output), we obtained the results. Z^{-1} is time delay.

First of all, the value of the step size should be considered. According to Eq. 10, the time step $\Delta t < 0.7803$. Using Eq. 13, 2/3, which is the value of step size, can be calculated. Corresponding to the different Δt (the neuro-computation can be operated) after taking the origin of coordinate as the initial point of iterative computation and taking $\|e\|_2 < 0.01$ ("e" is the error) as the condition of convergence. Different step sizes versus different Δt are listed in Table 1, which shows that the neural network

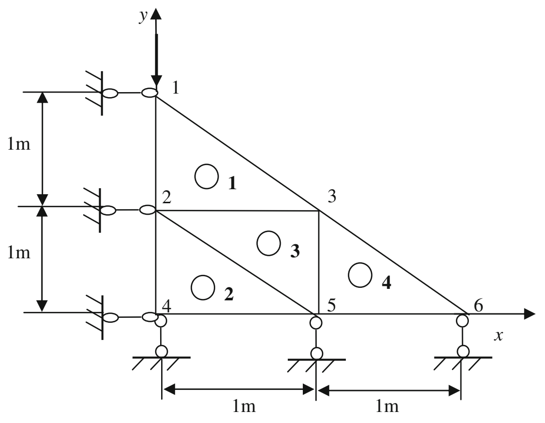

Fig. 1 Loading and finite element of triangle sheet

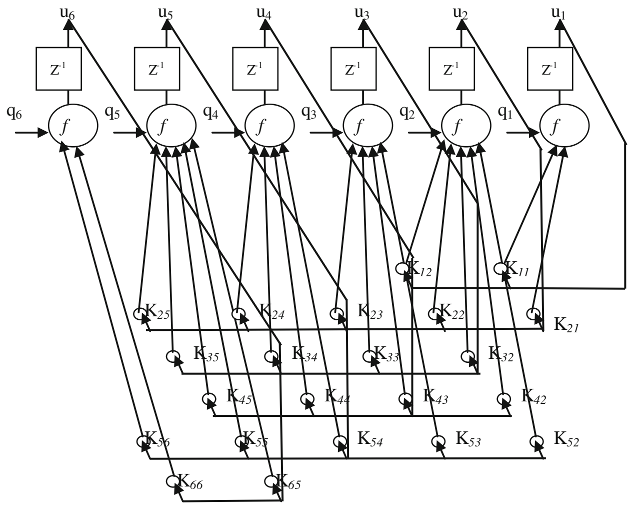

Fig. 2 The structural diagram of neural computation

Table 1 Relationship between the time of convergence (the number of iterative steps) and step size (Δt)

The step size Δt	0.1	0.3	2/3	0.75	Steepest descent	0.80
The number of iterative steps	235	77	33	31	29	Divergent

Table 2 Comparison between theoretical values and the outputs of 100th step of iteration

Initial points	u_1	u_2	u_3	u_4	u_5	u_6
x_1	−3.2525	−1.2539	−0.0868	−0.3725	0.1747	0.1761
x_2	−3.2525	−1.2536	−0.0871	−0.3728	0.1749	0.1760
x_3	−3.2528	−1.2526	−0.0880	−0.3737	0.1760	0.1758
x_4	−3.2529	−1.2522	−0.0884	−0.3741	0.1763	0.1757
x_5	−3.2529	−1.2521	−0.0885	−0.3742	0.1765	0.1757
x_6	−3.2527	−1.2529	−0.0877	−0.3734	0.1756	0.1759
Theoretical value	−3.2527	−1.2527	−0.0879	−0.3736	0.1758	0.1758

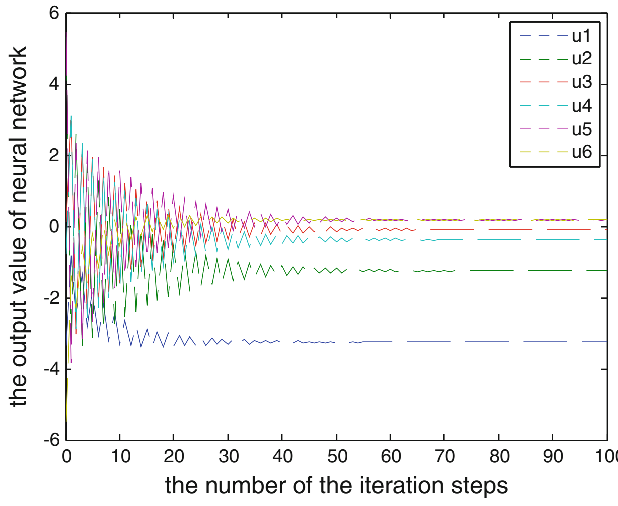

Fig. 3 Convergent curve of starting point x_1

converges to the effective value when $\Delta t < 0.7803$. In this extension, the number of iterations decreases along with the accretion of Δt, and the iterative process is faster. But, if $\Delta t > 0.7803$ (for example $\Delta t = 0.8$), the neural network could not converge to the effective value. In addition, according to Table 1, comparing with Δt, which is chosen using the energy steepest descent algorithm, the iteration times of choosing Δt using Eq. 13 do not increase obviously, so Eq. 13 can be used in practical calculations.

And then, digital experiments could be conducted by choosing different initial points. Six groups of initial points are given to make them typical. The vectors, which are formed by initial points and origin of coordinates, are orthogonal. The six groups of initial points are:

$x_1 = (-3.3346,\ 5.3281,\ 0.1176,\ -0.7961,\ 5.4552,\ -5.4850)$

$x_2 = (-2.7896,\ -3.1493,\ -7.2920,\ -5.1820,\ 1.3809,\ 0.6059)$

$x_3 = (-5.0548,\ 3.2908,\ 0.2993,\ -1.8711,\ -7.6719,\ -1.0826)$

$x_4 = (-4.6158,\ -5.4101,\ 6.2856,\ -2.6640,\ 1.6553,\ -0.2815)$

$x_5 = (-3.6230,\ -3.7163,\ -2.6794,\ 7.4256,\ -0.6697,\ -3.2088)$

$x_6 = (-4.5953,\ 2.7899,\ -0.1892,\ 2.6031,\ 2.5075,\ 7.6158)$

The six groups of orthogonal points are the initial inputs of the neural network. When $\Delta t = 2/3$, we run the neural networks, and the results of the 100th iteration are listed in Table 2. Figures 3 and 4 show the convergent process of neural networks, which takes the point x_1, x_2 as the initial input separately. Table 2 shows that the neural network computing system could obtain the approximate solution that satisfies the need of the error in finite steps, no matter what starting point of the neural networks is used. The maximum relative error of every component does not exceed 1.5%. From the stability analysis of neural network computing mentioned earlier, if iteration step meets Eq. 10, the neural network computing can converge to the efficient solution, and the initial iteration point can be randomly selected.

Two traditional computational methods are used to solve the above-mentioned example, and then the results could be compared with the result using the method presented in the paper. The first method is the Gauss–Newton for the optimality calculation, and the second is using Gauss–Seidel for the iterated operation directly on the finite element controlling equations. The convergent processes using two traditional methods are compared with the process using the proposed method. The convergent processes of u1 and u6 are showed in Figs. 5 and 6, respectively.

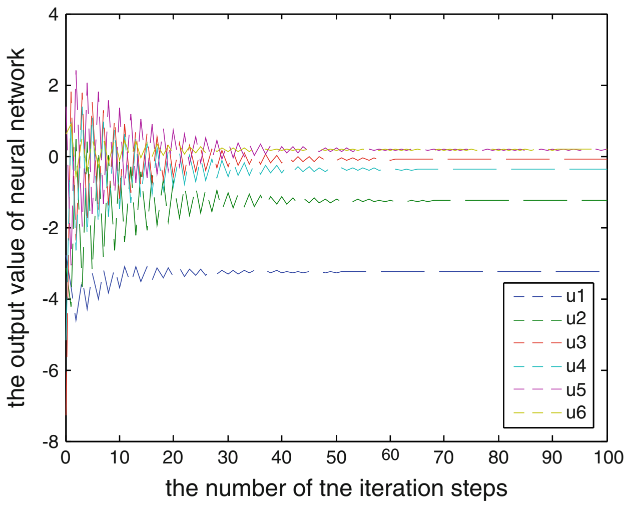

Fig. 4 Convergent curve of starting point x_2

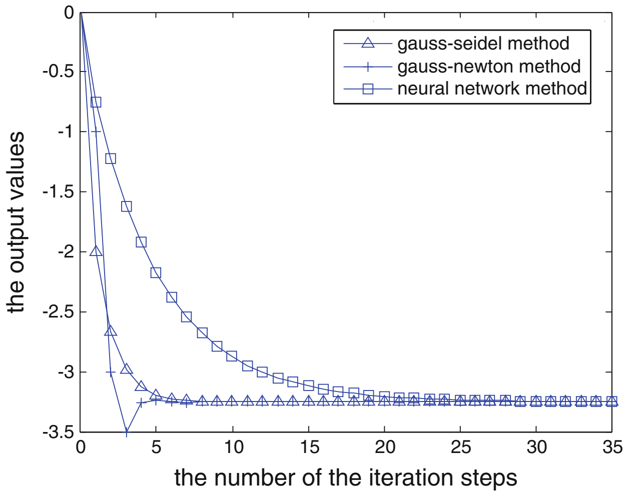

Fig. 5 Convergent process schematic diagram of u1

From the two figures, after nine iterations, the computing process using Gauss–Seidel is in the convergent range, 10 times for the Gauss–Newton method and 33 times for the developed method. Obviously, the two traditional methods take less steps in calculation.

But the total computing time of a concrete methods is $t = t_s n$, n is number of iterations and t_s is computing time of single step. t_s depends on computational cost of single step. The single-step computational cost of the example in this paper is shown in Table 3. From Table 3 and iteration times of each computing method, Gauss–Seidel method has less total computing cost than Gauss–Newton method. The Speedup η of the method in this paper to method of Gauss–Seidel is $\eta = \frac{9(35n_a+36n_m)}{33(n_a+n_m)} > 9$. Here, n_a is the computing time of doing one addition, and n_m is the computing time of doing one multiplication. From this, less computing time can be spent using the proposed method than the traditional methods.

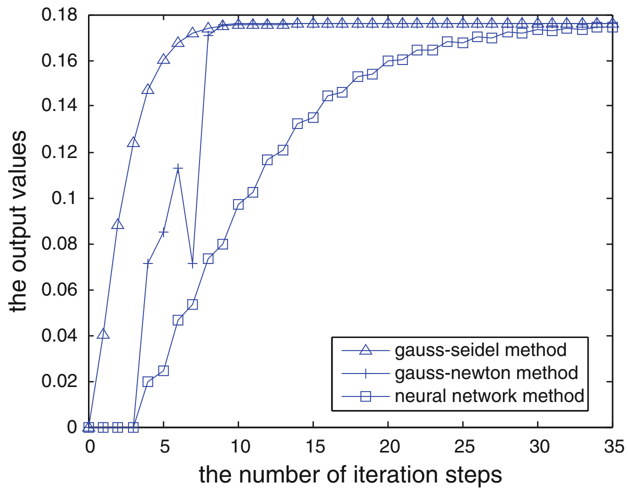

Fig. 6 Convergent process schematic diagram of u6

For comparing this method with two methods (ESS and GFS) given in Ref. [5], the relative errors of calculation results, which are iterated for 35 steps in the method of this paper, are listed in Table 4 in component model, and the results of ESS and GFS calculation are also listed in the same Table 4. It is obvious that the proposed method is better than the methods in Ref. [5] in calculation precision. The training times of neural network are 20000 in methods of Ref. [5], so the methods of Ref. [5] are less efficient than the method proposed in this paper.

Example 2 There is a steel frame (Fig. 7) with rectangular sections. Its column has $b_1 \times h_1 = 0.5$ m $\times$ 1.0 m section. Its beam has $b_2 \times h_2 = 0.5$ m $\times$ 1.26 m section. Uniform load q = 1kN/m. Suppose the elastic module $E = 1{,}000$ Pa. The internal forces of steel frame are to be evaluated.

The system equations can be obtained after meshing frame to three plane beam elements (Fig. 7), introducing boundary conditions and modifying the stiffness matrix.

$$\begin{bmatrix} 54.81 & 0 & -6.94 & -52.5 & 0 & 0 \\ 0 & 83.88 & 3.47 & 0 & -0.58 & 3.47 \\ -6.94 & 3.47 & 55.6 & 0 & -3.47 & 13.9 \\ -52.5 & 0 & 0 & 54.81 & 0 & -6.94 \\ 0 & -0.58 & -3.47 & 0 & 83.88 & -3.47 \\ 0 & 3.47 & 13.9 & -6.94 & -3.47 & 55.6 \end{bmatrix} \begin{Bmatrix} u_A \\ v_A \\ \theta_A \\ u_B \\ v_B \\ \theta_A \end{Bmatrix} = \begin{bmatrix} 3 \\ 0 \\ -3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The calculation system of neural network is constructed based on Eq. 3, and the step size $\Delta t = 114.25$ based on

Table 3 Comparison of one iteration step computing cost using different methods

Method name	Gauss–Seidel		Gauss–Newton			The proposed method	
Computation types	Addition	Multiplication	Addition	Multiplication	Matrix inversion	Multiplication	Addition
Computation times	35	36	5×36	6×36	1	1	1

Table 4 Comparing of relative calculation error between methods

	$u1$	$u2$	$u3$	$u4$	$u5$	$u6$
ESS (%)	0.75	0.79	4.86	3.60	1.71	7.56
GFS (%)	0.59	0.42	6.03	0.64	4.43	4.43
The paper's method (%)	−0.0571	−0.21	1.65	0.27	0.44	0.90

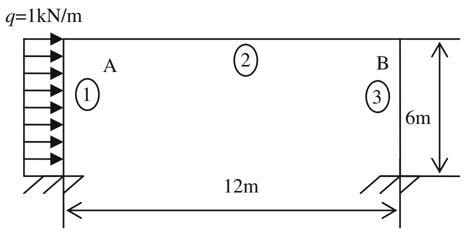

Fig. 7 Force and meshed figure of steel frame

Eq. 13. Six groups of orthogonal vectors are chosen randomly below as initial points of the calculation:

$x_1 = (4.7584, 0.0151, 4.2573, -6.0975, 4.4955, 1.3576)$
$x_2 = (4.4404, 4.2079, 1.2793, 2.9686, -3.7605, 6.1632)$
$x_3 = (4.2293, -2.5020, -2.7094, 5.7814, 5.9232, 0.0529)$
$x_4 = (2.8317, 4.6095, -7.3952, -3.4491, -0.0728, -2.0353)$
$x_5 = (3.3750, 2.8432, 3.9779, 2.4762, -1.7397, -7.4527)$
$x_6 = (4.5104, -6.8339, -1.5441, -1.5984, -5.2471, -0.6950)$

The results from 250 iterations are shown in Table 5. The units are international standard. It can be seen that after limited calculation, the neural network could always get the approximate solution fitting error limit. The max relative error is less than 2%. Then, the finite element-governing equations are calculated using Gauss–Seidel, successive over relaxation method (SOR method, here relaxation factor $k = 1.2$) and the method in this paper after choosing the origin of coordinate and taking $\|e\|_2 < 0.01$ as the convergence condition. The comparison between different calculation methods shows that equations converge after 169 iterations, while Gauss–Seidel has 69 iterations, and SOR has 54 iterations. The computational complexity of every calculating step of each method are shown in Table 6. It can be seen that the calculation

Table 5 The comparison between neural network input and output after 250 iterative steps

Initial point	u_A	v_A	θ_A	u_B	v_B	θ_B
x_1	0.8480	−0.0051	0.0284	0.8244	0.0051	0.0964
x_2	0.8517	−0.0052	0.0288	0.8281	0.0052	0.0968
x_3	0.8524	−0.0052	0.0289	0.8288	0.0052	0.0969
x_4	0.8476	−0.0051	0.0284	0.8241	0.0051	0.0964
x_5	0.8506	−0.0052	0.0287	0.8270	0.0052	0.0967
x_6	0.8494	−0.0051	0.0286	0.8259	0.0051	0.0966
Theoretical value	0.8490	−0.0051	0.0285	0.8254	0.0051	0.0965

efficiency SOR is higher than that of Gauss–Seidel. The Speedup η of this paper's method relative to SOR is:

$$\eta = \frac{54(30n_a + 36n_m)}{169(n_a + n_m)} > 9$$

Here, n_a and n_m have the same meaning as mentioned previously. From this, it can be seen that using the proposed method could save more calculating time than traditional method obviously.

4 Conclusion

A neurocomputing strategy has been developed to construct the energy function of structural finite element equations. The process of deformation under stress is the same process of minimization of energy function by relating the energy function of neural networks to the energy function of structural deformation, structuring discrete neural network dynamic equations and choosing the logical step size. In each step of iteration, the greater the values of energy function of neural networks decline, the faster the solving would be. The method of choosing the steepest descent steps of the energy function of neural networks and a simple and practical method of choosing iteration step, in which the one-dimensional search is not necessary, have been proposed.

Simulation and quantitative analysis show that the computational efficiency of the new method is better than the traditional methods. The accuracy of this method is better than those of the two methods discussed in Ref. [5]. The special advantage of the proposed method is breaking down the finite element computation to parallel

Table 6 Comparison between one iterative step of different methods

Method name	Gauss–Seidel		Sor		The paper's method	
Computation types	Addition	Multiplication	Addition	Multiplication	Addition	Multiplication
Computation times	30	36	30	36	1	1

computation of neural elements. Because every computing element is composed of addition and multiplication, the finite computing system based on neural networks is system, which is composed of addition and multiplication. All simulations in this paper, including neural network methods and traditional methods, are performed in serial computers. The theoretical feasibility of the new method is proved, but the real neural network should be calculated in digital circuit.

The technology of summator and multiplier based on LSI has been a very mature field. These will provide the realization of the computation of finite element equations based on neural networks with hardware. In comparison with the traditional methods of concurrent finite element computation, the proposed method needs fewer computers and communication equipments. The whole computing system can be integrated on one circuit board and the computation can be finished in finite iterations. This will provide a new parallel computing method to further improve the efficiency of structural finite element analysis.

Acknowledgments This research was partially supported by the National Natural Science Foundation of China under the contract number 50775026 and Higher education research projects in Inner Mongolia of the contract number NJZY08058. Constructive suggestions and comments from the referees and editors are very much appreciated. The authors would also like to acknowledge constructive revision suggestions and help with English by Professor Hae Chang Gea, Department of Mechanical and Aerospace Engineering, Rutgers, The State University of New Jersey and Dr. Christopher Hoyle, Department of Mechanical Engineering, Northwestern University, USA.

References

1. Li X, Mo Z, Hu Q (2004) The design and analysis of expandable parallel computation. National Defence Industry Press, Beijing
2. Liu Y, Zhou W, Liu F (2005) Parallel computing analysis of arch dam and foundation system using 3-D FEM with element-by-element approach. J Tsinghua Univ 45(6):772–775
3. Hajela P, Berke L (1991) Neurobiological computational models in structural analysis and design. Comput Struct 41(4):657–667
4. Lee SC, Park SK, Lee BH (2001) Development of the approximate analytical model for the stub-girder system using neural networks. Comput Struct 79:1013–1025
5. Li S (2000) Global flexibility simulation and element stiffness simulation in finite element analysis with neural network. Comput Methods Appl Mech Engrg 186:101–108
6. Takeuchi J, Kosugi Y (1994) Neural network representation of finite element method. Neural Netw 7(2):389–395
7. Consolazio Gary R (2000) Iterative equation solver for bridge analysis using neural networks. Comput-Aided Civ Infrastr Eng 15:107–119
8. Li S (2001) SCNN solver for finite element analysis with neural network elements. Proceedings of the sixth international conference on application of artificial intelligence to civil & structural engineering, pp 33–34
9. Zeng P (1996) Chinese mechanics tending towards 21st century: reports of 9th "The Forum of Young Scientist" of China association for science and technology. Beijing, Tsinghua University Press
10. Yagawa G, Okuda H (1996) Finite element solutions with feedback network mechanics through direct minimization of energy functionals. Int J Numer Methods Eng 39(5):867–883
11. Lou KN, Perez PA (1995) A novel application of artificial neural networks to structural analysis. Artif Intell Eng 9(3):211–219
12. Sun D, Hu Q, Xu H (2000) A neurocomputing model for the elastoplasticity. Comput Methods Appl Mech Eng 182(1–2):177–186
13. Gao HS, Zhang J, Qin WY (2008) On applying CG network to structural analysis. Proceedings of 4th international conference on natural computation, ICNC 2: 560–563
14. Li HB, Huang HZ (2004) Finite element analysis of structures based on linear saturated system model. Lect Notes Comput Sci 3174:820–825