ISSN 1392 - 1207. MECHANIKA. 2012 Volume 18(1):46-48

Real-time computation of element stiffness matrix based on BP neural networks

J.-Q. Xiong*, H.-Z. Huang, H.-Q. Li***, Z.-L. Wang****, H.W. Xu*****

**University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, E-mail: jqxiong@163.com*

***University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, E-mail: hzhuang@uestc.edu.cn*

****University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, E-mail: lihaiqing27@163.com*

*****University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, E-mail: wzhonglai@uestc.edu.cn*

******University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731, P.R. China, E-mail: xhw2211@163.com*

crossref http://dx.doi.org/10.5755/j01.mech.18.1.1279

1. Introduction

Based on serial computers, solving all the coupled problem of plasticity, damage, fatigue, and creep is complicated and extremely time consuming [1]. Therefore, it is impossible to realize real-time computation. Neural network is a complex nonlinear dynamic system, which presents high parallel computation capability. According to the theorem of the minimum potential energy, the finite element problem can be treated as a constrained nonlinear optimization problem. Using improved Hopfield network, the above problem could be mapped to a dynamic circuit, and its solution may be obtained within circuit time-constant [2]. Solving by neural networks, we suppose that the connecting weight values of networks are known. In fact, the above connecting weight matrix corresponds to structural total stiffness matrix. Because of elastic-plastic problem, crack growth and structural optimization design, stiffness matrix is not always constant. One of the keys to solve these problems real-timely is to compute the stiffness matrix in real-time.

2. BP neural networks and its function mapping capability

Neural networks consist of lots of artificial neurons connected each other. It is a large-scale complex system that is able to complete various intelligent tasks. In a simple fully connected network, each unit in a hidden layer is connected to all of the units of the previous layer and the next layer. When a network consists of more than one hidden layer, the units from these layers may be connected to the units of all the previous layers and the units of all the next layers [3-5].

In layer *j*, the input values of units are

$$net_j = \sum w_{ij} o_i \tag{1}$$

where o_i is output of unit *i* in the last layer, w_{ij} is connecting weight between the *i*-th unit in the last layer and the *j*-th unit in the current layer.

The output of the *j*-th unit in the current layer is $o_j = f(net_j)$, where f is an activation function which is commonly sigmoid function

$$f(x) = \frac{1}{1+e^{-(x-\theta)}} \tag{2}$$

where θ is a neuron threshold.

Signals flow from the input layer to the output layer. Given an import signal, an output one can be obtained. A three-layer network is able to realize the mapping of arbitrary continuous function by arbitrary exactness. The performance of mapping needs to train networks. The training process is shown as below.

1. Randomly assign initial values to all weights and neurons' thresholds.

2. Select input *x* and required output $\hat{y}$ as training samples.

3. Compute the actual output *y*.

4. Modify weight: from the output layer, error-signals backward propagate. Modify each weight in order to let the error, shown as Eq. (3), be minimum

$$e = \frac{1}{2}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2 \tag{3}$$

Weight modification:

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} = w_{ij}(t) + \eta \delta_{pj} y_{pj}$, where η is learning factor, *p* denotes the *p*-th sample. If *j* is an output unit, $\delta_{p\hat{y}} = (\hat{y}_{pj} - y_{pj}) f_j'(net_{pj})$. If *j* is a hidden unit, $\delta_{pj} = f_j'(net_{pj}) \sum_k (\delta_{pk} w_{kj})$. If an inertia term is added, $\Delta w_{ij}(t+1) = \eta \delta_j y_i + \alpha \Delta w_{ij}(t)$. *t*+1 refers to the (*t*+1)-th iteration. α refers to a rate factor.

5. If error exactness is satisfied, learning stops; otherwise, return to step 2.

3. Analysis of element stiffness matrix

Connecting weights must be given before using the improved Hopfield network to perform Finite Element Analysis (FEA) of the structure real-timely [6]. These weights are elements of corresponding structural total

stiffness matrix. The changing of structure element dimensions and structure material properties will result in weight changing. Therefore, for the issues such as elastic-plastic problem, crack growth and adaptive element discreteness, the performance of real-time computation of element stiffness matrices (ESM) is critical. ESM computation is actually a mapping issue of element dimensions and material properties to element stiffness. BP networks can perform computation completely. The components of ESM must be re-analyzed in order to reduce the input/output parameter number of BP networks and decrease the learning time of BP networks. Suppose the shape function is $N_i(x, y, z)\ (i = 1, 2, ..., m)$, ESM is $\boldsymbol{B} = [[B_1]\ [B_2] ... [B_m]]$.

$$[B_i] = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \end{bmatrix}, \quad i = 1,\ 2,\ ...,\ m. \tag{4}$$

The ESM can be given by

$$[K]^e = \iiint [B]^T [D][B] dxdydz =$$
$$= \iiint [[B_1][B_2] ... [B_m]]^T [D] [[B_1][B_2] ... [B_m]] dxdydz \tag{5}$$

$$[k_{ij}]^e = \iiint [B_i]^T [D][B_j] dxdydz \tag{6}$$

$$\sum_{j=1}^{m} [k_{ij}]^e = \sum_{j=1}^{m} \iiint [B_i]^T [D][B_j] dxdydz =$$
$$= \iiint [B_i]^T [D] \sum_{j=1}^{m} [B_j] dxdydz \tag{7}$$

For shape function, $\sum_{j=1}^{m} N_j(x, y, z) = 1$, therefore, $\sum_{j=1}^{m} [B_j] = 0$, and $\sum_{j=1}^{m} [k_{ij}]^e = 0$.

From the above analysis, we know: (1) ESM is a symmetric square matrix; (2) the sum of each row of the element in ESM is zero. By assembly process of the total stiffness matrix, we have

$$[k_{ij}]^z = \sum_{e=1}^{n} [k_{ij}]^e \tag{8}$$

where n is the structure element number. $[k_{ii}]^z$ along the diagonal line are the sum of partitioned matrices of all elements sharing node i. $[k_{ij}]^z$ are the sums of partitioned matrices of all elements sharing edge ij, $i \neq j$. Apparently, the number of share-nodded elements is much more than that of share-edged elements. We take

$$[k_{ii}]^e = -\sum_{j=1,\ j \neq i}^{m} [k_{ij}]^e \tag{9}$$

Therefore, we only need to calculate $[k_{ij}]^e$ (i=1, 2, …, m–1; j=i+1, …, m) other than $[k_{ii}]^e$. For example, as for triangular elements, the number of share-edged elements is no more than 2, but that of share-nodded elements may be more. In addition, as seen from the physical background of ESM, an ESM is only related to the relative coordinates of every element node, material properties, and shape function. It is unrelated to the absolute coordinates of nodes.

4. Element stiffness matrix computation by BP neural networks

For linear elastic problems, the computation of ESM is a mapping of the relative coordinates of element nodes to the elements of ESM. This mapping can be implemented by BP neural networks, which has two stages, learning stage and working stage. Suppose the element numbers are i, j, and m, respectively. The total coordinates are (x_i, y_i), (x_j, y_j), (x_m, y_m). Now we should set up a local coordinate system, whose origin is i. The directions of local coordinate axes are the same as those of total coordinate axes, respectively, then

$$\begin{cases} \Delta x_j = x_j - x_i \\ \Delta y_j = y_j - y_i \end{cases} \tag{10}$$

$$\begin{cases} \Delta x_m = x_m - x_i \\ \Delta y_m = y_m - y_i \end{cases} \tag{11}$$

According to the above analysis, we only need to solve $[k_{ij}]^e$, $[k_{im}]^e$, and $[k_{jm}]^e$ for triangular elements, then ESM can be determined. We take ten elements of the above three partitioned matrices as outputs. In practice, we may bisect the biggest change of the element node's relative coordinates, which forms many sets of input samples. On the basis of these samples, we can use the computation formula of ESM to compute the above ten elements taken as the corresponding goal outputs. Repeatedly train the network till it converges.

5. An example

Given a triangular element 1-2-3, take node 1 as the origin to set up a local coordinate system. The local coordinates of nodes 2 and 3 are $\Delta x_2, \Delta y_2, \Delta x_3$, and Δy_3, respectively. Each coordinate variation range is [0, 1]. Compute the corresponding elements of the ESM and take them as goal outputs. The unit number of the input and output layers of BP networks are equal to 4 and 10, respectively. Set up two hidden layers with 16 and 20 units, respectively, η = 0.8. Inertia term is 0.9, $e = 1.0 \times 10^{-3}$. Train network 15000 times till convergence. Table provides a comparison of the computation results obtained by FEA and BP networks, respectively.

As seen in Table, when BP networks method is used to solve ESM, the solution error is less than 2%, and

its exactness can be given arbitrarily. In learning stage, BP networks take a bit more time to train. Once the network is convergent, it works real-timely.

Table

A comparison of the computation results of element stiffness matrix

FEA	BP networks
-0.5	-0.505
-0.25	-0.247
-0.25	-0.245
0.0	0.001
-0.25	-0.242
-0.25	-0.249
-0.5	-0.490
0.0	-0.005
0.0	0.015
0.0	-0.029

6. Conclusions

1. It is practical to use BP networks to compute the element stiffness matrix. BP networks provide a possible method to use the finite element method in solving the issues changed structures or changed elements real-timely. BP networks' solution exactness can be given arbitrarily. But with more solution exactness, the training is great time consuming. Therefore, the set exactness should not be greater than the project needs in practice. The learning time of BP networks is a bit long, and off-line learning can be used for practical engineering structures. The working time of trained network is circuit time-constant (ns), namely real-time in working stage.

2. According to the analysis, element stiffness matrix is symmetric square matrix. In the matrix the partitioned matrices along diagonal line are the negative algebraic sums of all the same row/column partitioned matrices, and amount of computation will be decreased greatly.

3. As for the same linear elastic materials, changes of element stiffness matrix only depend on the relative coordinates of element nodes. Therefore, we only take the relative coordinates of nodes as the inputs of networks.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under the contract number 50775026.

References

1. **Cook, R.D.; Malkus, D.S.; Plesha, M.E.** 1989. Concepts and Applications of Finite Element Analysis. New York: John Wiley & Sons.
2. **Theocaris, P.S.; Panagiotopoulos, P.D.** 1993. Neural network for computing in fracture mechanics, methods and prospects of applications, Computer Methods in Applied Mechanics and Engineering 106(1-2): 213-228. http://dx.doi.org/10.1016/0045-7825(93)90191-Y
3. **Topping, B.H.V.; Bahreininejad, A.** 1997. Neural Computing for Structural Mechanics. Edinburgh: Saxe-Coburg Publications.
4. **Li, H.B.; Duan, W.; Huang, H.Z.** 2010. Neurocomputing method based on structural finite element analysis of discrete model, Neural Computing & Applications 19(6): 875-882. http://dx.doi.org/10.1007/s00521-010-0353-0
5. **Li, H.B.; Huang, H.Z.** 2008. Neurocomputing for fuzzy finite element analysis of structures based on fuzzy coefficient programming, Journal of Multiple-Valued Logic and Soft Computing 14(1-2): 191-204.
6. **Huang, H.Z.; Li, H.B.** 2005. Perturbation finite element method of structural analysis under fuzzy environments, Engineering Applications of Artificial Intelligence 18(1): 83-91. http://dx.doi.org/10.1016/j.engappai.2004.08.033

J.-Q. Xiong, H.-Z. Huang, H.-Q. Li, Z.-L. Wang, H.W. Xu

REALAUS LAIKO ELEMENTO STANDUMO MATRICOS SKAIČIAVIMAI REMIANTIS BP NEURONINIAIS TINKLAIS

R e z i u m ė

Tampriai plastiniai skaičiavimai yra daugelio kompleksinių mechaninių reiškinių, tokių kaip nuovargio, pažeidimo ir suirimo, fundamentali analizė. Naudojantis serijiniais kompiuteriais, negalima realiu laiku atlikti tampriai plastinių skaičiavimų. Šiame straipsnyje analizuojama elemento standumo matrica. Pasiūlytas metodas realaus laiko elemento standumo matricai skaičiuoti naudojant BP neuroninius tinklus. Elemento standumo matrica skaičiuojama atitinkamai atliekant baigtinių elementų analizę ir naudojant BP neuroninius tinklus. Palyginti realaus laiko skaičiavimų, naudojant BP neuroninius tinklus, ir baigtinių elementų analizės rezultatai.

J.-Q. Xiong, H.-Z. Huang, H.-Q. Li, Z.-L. Wang, H.W. Xu

REAL-TIME COMPUTATION OF ELEMENT STIFFNESS MATRIX BASED ON BP NEURAL NETWORKS

S u m m a r y

Elastoplasticity computation is the analysis foundation of many complex mechanical behaviours such as fatigue, damage, and fracture. It is impossible to realize real-time computation of elastoplasticity based on serial computer. The form of the element stiffness matrix is analyzed. A method that may realize the real-time computation of element stiffness matrix based on BP neural networks is proposed. The element stiffness matrix is computed using finite element analysis and BP neural networks, respectively.

Keywords: element stiffness matrix, BP neural network.

Received December 28, 2010
Accepted January 25, 2012