CMES, vol.21, no.2, pp.93-103, 2007

A New Local Contact Search Method Using a Multi-Layer Neural Network

Atsuya Oishi[1] **and Shinobu Yoshimura**[2]

Abstract: This paper describes a new local contact search method using a multi-layer neural network and its application to smoothed contact surface consisting of Gregory patches. A contact search process consists of two phases: a global search phase for finding the nearest node-segment pair and a local search phase for finding an exact local coordinate of the contact point within the segment. In the present method, the multi-layer neural network is utilized in the latter phase. The fundamental formulation of the proposed local contact search method is described in detail, and it is applied to smoothed contact surfaces consisting of Gregory patches. Through sample analyses, it is proved that the proposed method is fast and accurate enough for practical applications. Specifically, it is several times faster than the conventional method in the case of smoothed contact surfaces.

Keyword: Contact Search, Neural Network, Finite Element Method, Sliding Interface, Curved Surface.

1 Introduction

A remarkable progress in the field of microelectronics has increased the performance of computers. This has also helped computer simulations to replace time-consuming and highly expensive experiments. For the analyses of dynamic problems using the finite element method (FEM) or other methods, such as vehicle crashworthiness and metal forming, contact-impact phenomena must inevitably be taken into account [Zhong (1993), Schweizerhof, Nilsson and Hallquist (1992), Vignjevic, De. Vuyst and Campbell (2006), Guz, Menshykov, Zozulya and Guz (2007)]. In the contact-impact analysis, a contact point between two objects or two parts of one identical object, and their interaction are analyzed step by step. Contact conditions between two objects are in general described by the following two criteria:

$$\Omega_1 \cap \Omega_2 = \phi \tag{1}$$

$$\Gamma_1 \cap \Gamma_2 \neq \phi \tag{2}$$

where, Ω_1 and Ω_2 are inner bodies of the objects, and Γ_1 and Γ_2 are boundaries of the objects. Criterion (1) means that no penetration occurs into each body. In dynamic contact-impact analyses, it must be checked at every time step whether contact occurs between any locations of the two surfaces of the objects, i.e. the contact conditions of criteria (1) and (2) are satisfied; this is called the contact search.

The contact-impact algorithms in finite element contact analyses can be divided into two phases: the contact-searching phase and the contact interface phase. First, the contact-searching algorithm locates where contact occurs. Second, contact forces are determined and applied to the contact surfaces according to the contact interface algorithm. The contact search phase often consumes most of calculation time of the contact-impact algorithm. Thus, fast and efficient contact search method has been sought.

The sliding interface algorithm [Hallquist, Goudreau and Benson (1985), Benson and Hallquist (1990)] is the most popular, and has been implemented in a well-known dynamic FEM code, DYNA3D, which is an explicit three-dimensional finite element code for analyzing dynamic response of inelastic solids and structures with large deformation. In the sliding interface algorithm, any two surfaces that may come into contact must be specified prior to the analysis. One of the two surfaces is designated

[1] Univ.Tokushima, Tokushima, JAPAN.
[2] Univ. Tokyo, Tokyo, JAPAN.

as a master surface, while the other as the slave surface. Contact searching is performed only between slave nodes, i.e. the nodes on the slave surface, and the master segments, i.e. the facets of the elements on the master surface. The contact search process can usually be divided into two phases: the global search phase and the local search phase. In the global contact search phase, a master segment that is in close proximity to a given slave node is picked up by checking all the slave nodes in the whole analysis domain. In the local contact search phase, the pair of the segment and the slave node selected in the global contact search phase is checked and the local coordinates of the contact point, if any, is determined by a time-consuming iterative method based on Newton's method.

Iterative methods, however, often become time-consuming and have difficulties in convergence, so that fast and efficient local contact search methods without any iteration have been sought. The approximating function described in literatures [Zhong and Nilsson (1996)] and [Wang and Nakamachi (1997)], which produces the approximate local coordinates of the contact point using area coordinates, is one of such attempts. In the method, the values of four kinds of shape functions for the contact point are approximated by the function of areas of four triangles made of one slave node and two nodes on the master segment. Though this method is adopted in several applications due to its fast calculation, it often shows poor performance in accuracy for the cases where related nodes are not coplanar. Pinball algorithm [Belytschko and Neal (1991)] is also a local contact search algorithm without iteration. In the pinball algorithm, a pinball, i.e. a sphere, is embedded in every element of the contact surfaces. The volume of the pinball is equal to that of the corresponding element and the center of the pinball is located at the gravity center of the element. Though this algorithm is fast, it also has some difficulties in accuracy due to its approximate representation of contact surfaces.

Artificial neural networks that simulate the neural system of mankind have been extensively developed [Haykin (1994), Hassoun (1995)]. Among various artificial neural networks, the multi-layer feed forward neural networks have the capability of simulating any continuous functions [Funahashi (1989)] and have been successfully applied to various engineering problems, such as crack/damage detection [Stavroulakis and Antes (1998), Oishi, Yamada, Yoshimura, Yagawa, Nagai and Matsuda (2001), Liu, Huang, Sung and Lee (2002), Zacharias, Hartmann and Delgado (2004), Fang, Luo and Tang (2005)], modeling of material properties [Furukawa and Yagawa (1998), Huber and Tsakmakis (2001), Lefik and Schrefler (2003), Hashash, Jung and Ghaboussi (2004)] and optimization [Papadrakakis, Lagaros and Tsompanakis (1998), Iranmanesh and Kaveh (1999), Cho, Shin and Yoo (2005)].

In this paper, we propose a new iteration-free local contact search method using a multi-layer neural network that explicitly represents the local coordinates of the contact point. Fundamental formulation of the proposed method is explained in detail. Its basic performance is demonstrated through sample analyses. Finally, the proposed method is applied to the local contact search process for smoothed contact surface consisting of Gregory patches.

2 Contact Search Algorithms in Sliding Interface Algorithm

Each segment is assumed to be a quadrilateral facet of an 8-noded hexahedral isoparametric element on a contact surface. As for shell elements, both surfaces of the element are regarded as two different segments. Figure 1 shows a configuration of the segment and the slave node, where P_s denotes the slave node and the quadrilateral $A(P_m)$ BCD does the segment. P_m usually belongs to several segments. The segment that P_s hits can be found by checking the following criteria:

$$(\vec{c}_1 \times \vec{r}) \cdot (\vec{c}_1 \times \vec{c}_2) > 0 \tag{3}$$

$$(\vec{c}_1 \times \vec{r}) \cdot (\vec{r}_\times \vec{c}_2) > 0 \tag{4}$$

As shown in Figure 1, $\vec{c}_1$ and $\vec{c}_2$ are vectors drawn from P_m to the neighboring node on the segment counterclockwise or clockwise, respectively. $\vec{r}$ is a vector defined as:

$$\vec{r} = \vec{t} - \left(\vec{t} \cdot \vec{m}\right) \cdot \vec{m} \tag{5}$$

$$\vec{m} = \frac{\vec{c}_1 \times \vec{c}_2}{\left|\vec{c}_1 \times \vec{c}_2\right|}, \tag{6}$$

where $\vec{t}$ is a vector drawn from P_m to P_s. The segment that meets the criteria (3), (4) is picked up as a target segment.

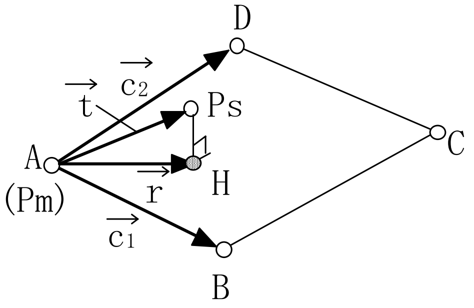

Figure 1: Local Contact Search

After the target segment is located, local coordinates (ξ, η) of the contact point on the segment has to be identified accurately. One of the method to do so is solving the following set of equations with Newton's method [Hallquist, Goudreau and Benson (1985), Benson and Hallquist (1990), Oishi, Yoshimura and Yagawa (2002)]:

$$\frac{\partial \vec{r}}{\partial \xi}(\xi_c, \eta_c) \cdot \left\{\vec{t} - \vec{r}(\xi_c, \eta_c)\right\} = 0 \tag{7}$$

$$\frac{\partial \vec{r}}{\partial \eta}(\xi_c, \eta_c) \cdot \left\{\vec{t} - \vec{r}(\xi_c, \eta_c)\right\} = 0 \tag{8}$$

Further, a penetration depth g is calculated as follows.

$$g = \vec{m} \cdot \left(\vec{t} - \vec{r}(\xi_c, \eta_c)\right) \tag{9}$$

$g > 0$ means that the node and the segment are not in contact but in proximity. If $g < 0$, contact forces acting between the slave node P_s and master nodes on the segment are calculated as

$$\vec{f_s} = -gk\vec{m} \tag{10}$$

$$\vec{f}_m^i = \phi_i(\xi_c, \eta_c) \cdot \vec{f_s}, \tag{11}$$

where $\vec{f_s}$ is the contact force vector for the slave node P_s, $\vec{f}_m^i$ is the contact force assigned to the i-th node on the target segment, k is the factor determined by geometry and material properties of the element including the target segment, and $\phi_i(\xi_c, \eta_c)$ is the shape function.

Though the above method is widely used, solving Eqs.(7) and (8) is time-consuming, and it sometimes takes much longer for some cases than for other cases depending on the relative locations of corresponding nodes. Therefore fast contact search methods have been sought.

3 Multi-Layer Neural Networks

The multi-layer neural network consists of layered units. The processing unit is modeled after a nerve cell, and takes multiple input values and outputs a single value. Taking the sigmoid function as an activating function, an input-output relation of the unit is formulated as follows:

$$O_j = f(U_j) = \frac{1}{1 + \exp(-2U_j/U_0)} \tag{12}$$

$$U_j = \sum_{i=1}^{l} W_{ji} \times I_i - \theta_j \tag{13}$$

where O_j is the output signal of the j-th unit, U_j is the internal potential of the j-th unit, $f(x)$ is the activation function, i.e. the sigmoid function here, U_0 is the constant of the sigmoid function, W_{ji} is the connection weight between the j-th unit and the i-th unit in the layer beneath, I_i is the input signal from the i-th to the j-th units, θ_j is the bias value of the j-th unit, and $l$is the number of input signals.

Figure 2 shows the multi-layer neural network. All the units are formed into multiple layers, i.e. an input layer, intermediate (hidden) layers, and an output layer. No connections exist among units in the same layer, while every two units in the neighboring layers have a connection.

The fundamental idea of training the neural network is as follows. First the following error E is defined:

$$E = E_p = \frac{1}{2}\sum_{k=1}^{n}\left(T_k^p - O_k^p\right)^2 \tag{14}$$

where E_p is the square error for the p-th training pattern, T_k^p is the teacher signal to the k-th unit

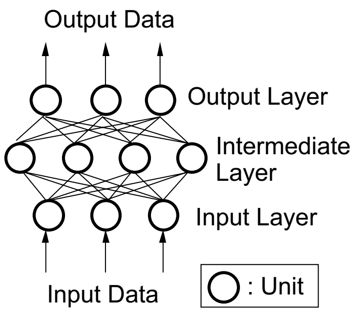

Figure 2: Multi-Layer Neural Network

in the output layer for the p-th training pattern, O_k^p is the output signal from the k-th unit in the output layer for the p-th training pattern, and n is the number of output units.

In the training process, the connection weights W_{ji} and the bias values θ_j are modified iteratively based on the steepest-descent method to minimize the above error. This training algorithm is called the back-propagation (BP). In this study, the ordinary BP algorithm combined with the moment method, which accelerates learning speed, is employed.

It is proved that the multi-layer neural network can simulate any nonlinear mapping. However, the network has some limitations in reality, due to poor convergence in training process when the numbers of hidden layers and units increase. In the present study, an appropriate size of the network is determined through trial and error as in many practical applications..

4 Local Contact Search Using Multi-Layer Neural Networks

Local coordinate values of a contact point are obtained by solving Equations (7), (8). Solving Equations (7), (8) is, in other words, finding the mapping f and g from fifteen coordinate values, $x_A, y_A, z_A, \cdots, x_D, y_D, z_D, x_S, y_S, z_S$, of the five nodes, A, B, C, D, Ps in Figure 1 to the local coordinate values ξ_c and η_c of the contact point, respectively. The mapping functions f and g are written as follows:

$$\xi_c = f(x_A, y_A, z_A, \cdots, x_S, y_S, z_S) \tag{15}$$

$$\eta_c = g(x_A, y_A, z_A, \cdots, x_S, y_S, z_S) \tag{16}$$

For input variables of the functions f and g, we should not use absolute coordinate values of nodes, but values calculated from relative positions of the nodes. This is based on the fact that the local coordinate values ξ_c and η_c do not change by affine transformation, such as translation, rotation, expansion and reduction. Referring to Figure 1, the transformation procedures adopted in this research is specifically described as follows:

(1) Node A is translated to the origin (0,0,0) and other nodes are also translated in the same manner.

(2) All the nodes are rotated and the node B is placed on the x-axis.

(3) All the nodes are rotated around the x-axis and the node D is placed on the x-y plane.

(4) Using the distance between the node A and the node B as a measure, all the coordinate values of nodes are expanded or reduced until the coordinate values of the node B is set to (1.0,0,0).

By this transformation, 15 coordinate values, $x_A, y_A, z_A, \cdots, x_D, y_D, z_D, x_S, y_S, z_S$ for input variables of the function f and g can be reduced to 8 coordinate values $x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S$, and the equations (15),(16) are rewritten as follows:

$$\xi_c = f(x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S) \tag{17}$$

$$\eta_c = g(x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S) \tag{18}$$

Though exact representations of the functions f and g are unknown, they can be approximated by a multi-layer neural network.

The procedure of the local contact search with a neural network can be summarized as follows:

(1) A lot of configurations of five nodes shown in Figure 1, i.e. one slave node Ps and four nodes on the opposite master segment, are

considered. The local coordinate values of the contact point for every configuration are calculated by Newton's method. This makes many data pairs of the coordinate values of nodes and the local coordinate values of the correspondent contact point.

(2) A multi-layer neural network is trained using a lot of data pairs obtained in the previous process: i.e. using the coordinate values of nodes as input data and the local coordinate value of the correspondent contact point as the teacher signal for the multi layer neural network. It results in the trained neural network that outputs the local coordinate values of the contact point from the input data of the coordinate values of the slave node and the nodes on the master segment.

(3) The multi-layer neural network trained in the previous process is implemented into the code for contact analyses. It replaces the conventional local contact search procedure.

5 Numerical Examples

5.1 Data Preparation

Basic performance of the proposed local contact search algorithm is tested through sample analyses of the following test problem shown in Figure 3. A slave node Ps and its corresponding four-node quadrilateral master segment ABCD are located as shown in the figure. The master node A, which is the nearest to the slave node Ps, is located on the origin of the coordinate axes, (0.0, 0.0, 0.0). The node B is located at (1.0, 0.0, 0.0). The node D is located somewhere on the $x-y$ plane, i.e. its z-coordinate is set to zero. The x, y coordinates of the nodes C, D, Ps are in the range of (-2.0, 2.0). The z coordinate of the node C and Ps is in the range of (-0.3, 0.3). Moreover, the following condition on the configuration of nodes is added: the length of the edge AD is equal to or shorter than that of AB (= 1.0). This condition reduces the number of node configurations to be considered without any loss of generality, which greatly improves the trainability of the neural network.

With the above conditions, almost 740000 patterns of configurations are obtained. Among them, 1000 patterns are randomly selected for training.

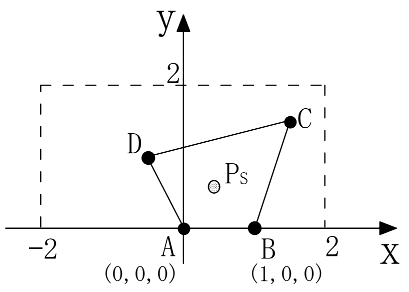

Figure 3: Configuration of nodes

5.2 Configurations and Training of Neural Network

In this example, input data to the neural network for local contact search consists of eight coordinate values of nodes: x, y coordinate values of the node D, x, y, z coordinate values of the node C and Ps. Output data from the neural network consists of two local coordinate values of the contact (or projection) point: ξ_C and η_C. This causes us to adopt the multi-layer neural network with two units in the output layer and eight units in the input layer. The optimal number of the units in the intermediate (hidden) layer is not known a priori. In this study, the number of units in the intermediate layer is set to 8 through several test analyses.

As for the training of the neural network, basic back propagation training with the momentum method is adopted. Training iterations are limited within 10000.

5.3 Results

5.3.1 Accuracy

To test accuracy in the estimation of the local coordinate values of the contact point, the trained neural network is tested for a whole, almost 740000, patterns of configurations. Figure 4 shows the distributions of approximation errors in the local coordinate values for the 740000 patterns. The vertical axis designates frequency,

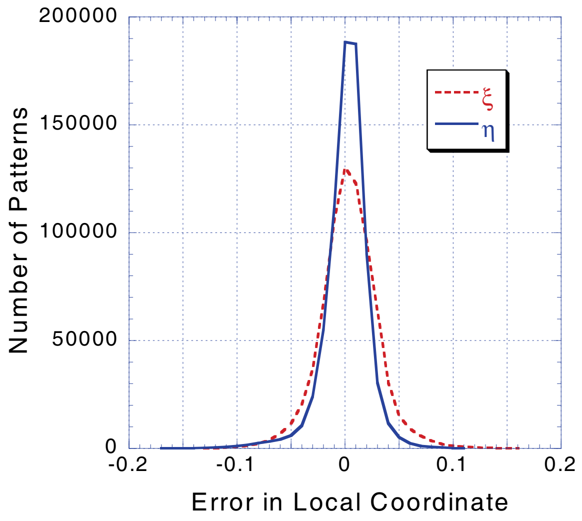

Figure 4: Accuracy of the present method

while the horizontal axis does the error, i.e. the difference between the local coordinate value obtained by the trained neural network and that by Newton's method. Figure 4 indicates that the well-trained neural network in the proposed method can produce very accurate estimation of local coordinate values of contact points.

5.3.2 Speed

CPU time to obtain local coordinate values is compared between the proposed method and the ordinary method based on Newton's method. The CPU time for the proposed method is on average equal to that of seven iterations in Newton' method. The CPU time to obtain the local coordinate values by the trained neural network is constant for all patterns of configurations, while that by Newton's method varies depending on the pattern of configurations, and more than ten iterations are sometimes required to converge in Newton's method.

6 Neural Network based Local Contact Search Method for Smoothed Contact Surface

6.1 Smoothed Contact Surface

Ordinary contact surfaces consist of facets, each of which is one face of elements exposed to the contact surface as shown in Figure 5.

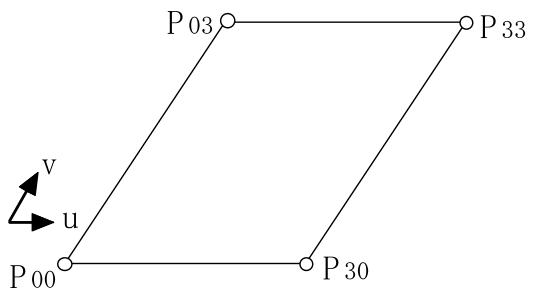

Figure 5: Patch derived from Isoparametric Element

For a bi-linear isoparametric element, the facet, or the patch, is represented by the following equation:

$$P(u,v) = \sum_{i=1}^{4} N_i(u,v)P_i \quad (-1 \le u,v \le 1) \qquad (19)$$

where P_i is the coordinate of the corner node of the patch, u and v are local coordinate values and $N_i(u,v)$ is the shape function as:

$$N_1(u,v) = \frac{1}{4}(1-u)(1-v) \qquad (20)$$

$$N_2(u,v) = \frac{1}{4}(1+u)(1-v) \qquad (21)$$

$$N_3(u,v) = \frac{1}{4}(1+u)(1+v) \qquad (22)$$

$$N_4(u,v) = \frac{1}{4}(1-u)(1+v) \qquad (23)$$

There exists discontinuity of the tangents of the facets at the boundary between two facets that arise from the bi-linear elements, and this discontinuity often causes some difficulties in numerical stability and convergence of contact analyses. Therefore, contact smoothing has been sought [Wang, Cheng and Yao (2001), Puso and Laursen (2002)].

As for surface smoothing, intensive research has been done in such fields as CAGD (Computer Aided Geometric Design) and CG (Computer Graphics) [Farin (2002)]. Bezier surfaces, which are typical smoothing ones, can be represented by the set of Bezier patches, which are quadrilateral facets made from four vertex points and additional control points. A contact surface consists of quadrilateral facets, i.e. faces of elements that reside in the surface, each of which can be regarded as the 3rd order Bezier patch if twelve control points are added. Figure 6 shows the 3rd order Bezier patch, in which $P_{11}, P_{21}, P_{22}, P_{12}$ are called inner control points and others are called boundary control points. The 3rd order Bezier patch is represented by the following equation:

$$P(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i^3(u)B_j^3(v)P_{ij} \quad (0 \le u,v \le 1) \tag{24}$$

where $B_i^n(\,)$ is the n-th order Bernstein Polynomial defined as:

$$B_i^n(u) = {}_nC_i \cdot u^i \cdot (1-u)^{n-i} \tag{25}$$

Bezier patches construct smoothing surface of only C^0 continuity at the boundary between patches, which is insufficient for numerical stability and convergence. Furthermore, it appears very difficult to apply the Bezier patches into surfaces made from unstructured mesh.

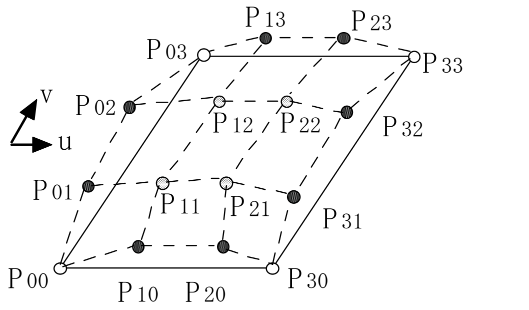

Figure 6: Control Points for Bezier Patch

In contrast to the Bezier patches, Gregory patches, which can be derived from the Bezier patches by modifying their internal control points, can construct smoothing surface of G^1 continuity at the boundary between patches, which reduces difficulties in numerical stability and convergence property, and can be applied to the contact surface arisen from unstructured meshes [Puso and Laursen (2002), Chiyokura and Kimura (1983)]. The G^1 continuity at the boundary between two patches means that they have a continuously varying tangent plane along the boundary, in other words the continuity of the unit normal vector of the patches at the boundary. Figure 7 shows the Gregory patch. Internal control points in the Bezier patch, P_{11}, P_{21}, P_{22}, P_{12}, are divided into two separate control points in the Gregory patch, $P_{110}, P_{111}, P_{210}, P_{211}, P_{220}, P_{221}, P_{120}, P_{121}$, respectively. Control points of the Gregory patch consists of 20 points. Control points other than the four corner nodes can be generated by the calculation using the coordinate values of the four corner nodes and normal vectors at the corner nodes as shown in Figure 8 [Puso and Laursen (2002)]. The node normal vector $\vec{n}_A$ at the corner node A is defined by the following equation as illustrated in Figure 9,

$$\vec{n}_A = \frac{\sum_i \left(\vec{c}_1^{(i)} \times \vec{c}_2^{(i)}\right)}{\left|\sum_i \left(\vec{c}_1^{(i)} \times \vec{c}_2^{(i)}\right)\right|} \tag{26}$$

where $\vec{c}_1^{(i)}$ and $\vec{c}_2^{(i)}$ are vectors that stem from the node along the edge of the i-th element that shares the node.

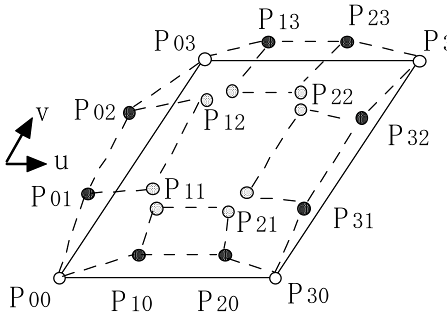

Figure 7: Control Points for Gregory Patch

The Gregory patch is represented by the same equation as the Bezier Patch as follows:

$$P(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i^3(u)B_j^3(v)P_{ij}(u,v) \tag{27}$$

where $B_i^n(\,)$ is the n-th order Bernstein polynomial and P_{11}, P_{21}, P_{22}, P_{12} are actually the linear

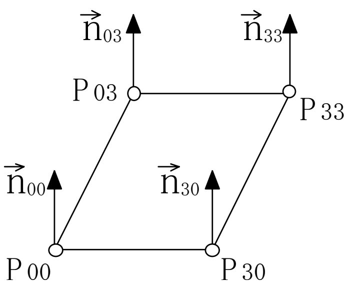

Figure 8: Data Required for Controll Points of Gregory Patch

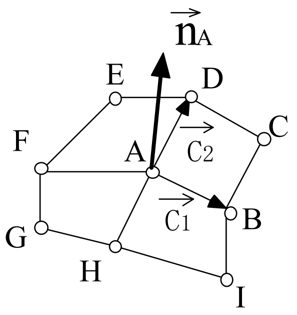

Figure 9: Nodal Normal Vector

combinations of two control points as:

$$P_{11}(u,v) = \frac{uP_{110} + vP_{111}}{u+v} \tag{28}$$

$$P_{12}(u,v) = \frac{uP_{120} + (1-v)P_{121}}{u+(1-v)} \tag{29}$$

$$P_{21}(u,v) = \frac{(1-u)P_{210} + vP_{211}}{(1-u)+v} \tag{30}$$

$$P_{22}(u,v) = \frac{(1-u)P_{220} + (1-v)P_{221}}{(1-u)+(1-v)} \tag{31}$$

Using Gregory patches for contact surface smoothing, numerical stability and convergence property are improved significantly.

6.2 Neural Network based Local Contact Search Method for Smoothed Contact Surface

In the node-segment type contact algorithm, the foot of the perpendicular from the slave node to the master segment should be calculated by solving Equations (7), (8). While the ordinary bi-linear patch that arises from a bi-linear isoparametric element is represented by Equation (19), the Gregory patch is represented by a more complicated equation, i.e. Equation (27). This makes it very time-consuming to solve Equations (7), (8) by iterative methods.

The local contact search process for the smoothed surfaces, however, can still be regarded as the mapping from the coordinate values of the related nodes to the local coordinate values of the contact point. Constructing this mapping on the multi-layer neural network can perform the local contact search for the smoothed surfaces.

The proposed method avoids the time-consuming iterative calculation of Equations (7), (8) with Equation (27) in the application phase described in Section 4, it leads to fast calculations of the contact point in the application phase.

6.3 Numerical Examples

6.3.1 Data Preparation

Performance of the proposed local contact search algorithm for smoothed contact surfaces is tested through sample analyses of the following test problem shown in Figure 3. Almost the same setting as in Section 5.1 is adopted for generating sample patterns. As for the shape of the generated segments, a condition that all interior angles of the segment should range within from 60 to 120 degrees is newly added. This condition not only omits heavily distorted segments, which are not appropriate for analyses either, but also reduces the total number of patterns to accelerate the training process significantly. On these conditions, 46631 patterns of configuration of related nodes are generated. For each pattern of the above configuration, twenty different patterns are generated. Additional condition is that the tangent of the angle between the node normal vector and the corresponding normal vector of the each segment that shares the node is below 0.2. The node normal vectors at the four corner nodes of the generated patterns are randomly set to meet this condition. Totally 835906 patterns of configurations are generated, and 83552 patterns out of them are selected at random for the test patterns for training the neural network.

6.3.2 Configuration and Training of Neural Network

In this case, input data to the neural network for local contact search consists of eight coordinate values of nodes: x, y coordinate values of the node D, x, y, z coordinate values of the node C and Ps and eight components of node normal vectors, i.e. two components for each node normal vector. Ratios of the x-component and the y-component to the z-component are adopted as the two components for the node normal vector, respectively. Output data from the neural network consist of two local coordinate values of the contact (or projection) points: u and v. Thus, we adopt the multi-layer neural network with two units in the output layer and sixteen units in the input layer. As for the number of units in the intermediate (hidden) layer, 32 and 48 are tested. Basic back propagation training with a momentum method is adopted, and training iterations are limited within 10000.

6.3.3 Results

Accuracy: To test accuracy in the estimation of the local coordinate values of the contact point, the trained neural network is tested for a whole 835906 patterns of configurations. Table 1 shows the relationship between accuracy of the trained neural networks in the estimation of the local coordinate values of the contact point and the number of units in the intermediate layer. Neuro32 and Neuro48 in Table 1 designate the neural network with 32 units in the intermediate layer and that with 48 units, respectively. The average error is defined as the average of the absolute values of the difference between the value by the trained neural network and that by Newton's method. Table 1 indicates that both Neuro32 and Neuro48 show good performance in accuracy. Figure 10 shows the distributions of estimation errors of the Neuro48 in the local coordinate values for all patterns. The vertical axis designates frequency, while the horizontal axis does the error, i.e. the difference between the local coordinate value obtained by the trained neural network and that by Newton's method. Figure 10 indicates that the well-trained neural network in the proposed method can produce very accurate estimation of local coordinate values of contact points.

Table 1: Accuracy of the Present Method

	Error			
	Average		Standard Deviation	
	u	v	u	v
Neuro32	0.00493	0.00758	0.00444	0.00902
Neuro48	0.00392	0.00479	0.00369	0.00766

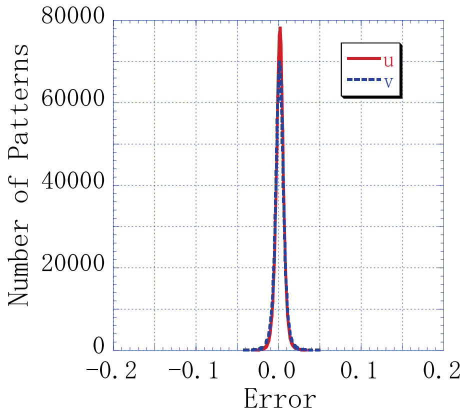

Figure 10: Distribution of Error

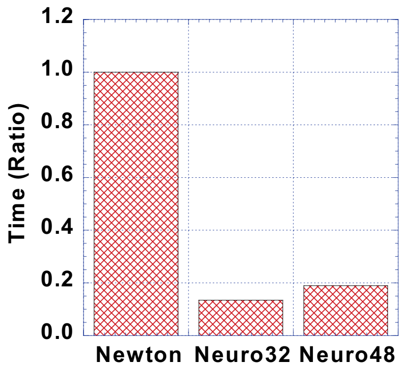

Figure 11: Speed of the Present Method

Speed: CPU time to obtain the local coordinate values is compared between the proposed method and the ordinary method based on Newton's method. Figure 11 shows the results. The vertical axis designates the ratio of the CPU time to the CPU time of the conventional Newton's method with iteration count set to five, with which most configurations showed convergence in sample analyses. The CPU time of the neural network includes the time for preprocessing described in Section 4. Figure 11 clearly shows that the proposed method is several times faster than the conventional method based on Newton's method.

7 Conclusions

A new local contact search method using a multi-layer neural network is proposed. Contact search process consists of two phases: a global search phase for finding the nearest node-segment pair and a local search phase for finding an exact local coordinate of the contact point within the segment. The local search phase can be regarded as the mapping from coordinate values of related nodes to the local coordinate values of the contact point. In the proposed method, this mapping is implemented on the multi-layer neural network as its weight values through error back-propagation training. The performance and characteristics of the proposed method are tested and its remarkable features are summarized as follows:

(1) The well-trained neural network can deliver local coordinate values of the contact point that are accurate enough for analyses.

(2) Unlike the conventional method based on Newton's method, the proposed method can deliver local coordinate values of the contact point in the same CPU time for any configuration of nodes.

The proposed method is also successfully applied to the local contact search process for smoothed contact surfaces with Gregory patches. The results are summarized as follows:

(1) The well-trained neural network can deliver accurate local coordinate values of the contact point for smoothed contact surfaces.

(2) The proposed method is significantly faster than the conventional method based on Newton's method.

References

Benson, D. J.; Hallquist, J. O. (1990): A single surface contact algorithm for the post-buckling analysis of shell structures, *Computer Methods in Applied Mechanics and Engineering*, vol.78, pp.141-163.

Belytschko, T.; Neal, M. O. (1991): Contact-impact by the pinball algorithm with penalty and Lagrangian methods, *International Journal for Numerical Methods in Engineering*, vol.31, pp.547-572.

Chiyokura, H.; Kimura, F. (1983): Design of solids with free-form surfaces, *Computer Graphics*, vol.17, pp.289-298.

Cho, J. R.; Shin, S. W.; Yoo, W. S. (2005): Crown shape optimization for enhancing tire wear performance by ANN. *Computers & Structures*, vol.83, pp.920-933.

Fang, X.; Luo, H.; Tang, J. (2005): Structural damage detection using neural network with learning rate improvement. *Computers & Structures*, vol.83, pp.2150-2161.

Farin, G. (2002): *Curves and Surfaces for CAGD: A Practical Guide 5th ed.*, Academic Press.

Funahashi, K. (1989): On the approximate realization of continuous mappings by neural networks, *Neural Networks*, vol.2, pp.183-192.

Furukawa, T.; Yagawa, G. (1998): Implicit constitutive modelling for viscoplasticity using neural networks. *International Journal for Numerical Methods in Engineering*, vol.43, pp.195-219.

Guz, A. N.; Menshykov, O. V.; Zozulya, V. V.; Guz, I. A. (2007): Contact Problem for the Flat Elliptical Crack under Normally Incident Shear Wave. *CMES: Computer Modeling in Engineering & Sciences*, vol.17, no.3, pp.205-214.

Hallquist, J. O.; Goudreau, G. L.; Benson, D. J. (1985): Sliding interfaces with contact-impact in large-scale Lagrangian computationa, *Computer Methods in Applied Mechanics and Engineering*, vol.51, pp.107-137.

Hashash, Y. M. A.; Jung, S.; Ghaboussi, J. (2004): Numerical implementation of a neural network based material model in finite element analysis. *International Journal for Numerical Methods in Engineering*, vol.59, pp.989-1005.

Hassoun, M. H. (1995): *Fundamentals of Artifi-*

cial Neural Networks, MIT Press.

Haykin, S. (1994): *Neural Networks: A Comprehensive Foundation*, Prentice-Hall.

Huber, N.; Tsakmakis, C. (2001): A neural network tool for identifying the material parameters of a finite deformation viscoplasticity model with static recovery. *Computer Methods in Applied Mechanics and Engineering*, vol.191, pp.353-384.

Iranmanesh, A.; Kaveh, A. (1999): Structural optimization by gradient-based neural networks. International Journal for Numerical Methods in Engineering, vol.46, pp.297-311.

Lefik, M.; Schrefler, B. A. (2003): Artificial neural network as an incremental non-linear constitutive model for a finite element code. *Computer Methods in Applied Mechanics and Engineering*, vol.192, pp.3265-3283.

Liu, S. W.; Huang, J. H.; Sung, J. C.; Lee, C. C. (2002): Detection of cracks using neural networks and computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, vol.191, pp.2831-2845.

Oishi,A.; Yamada, K.; Yoshimura, S.; Yagawa, G.; Nagai, S.; Matsuda, Y. (2001): Neural Network-Based Inverse Analysis for Defect Identification with Laser Ultrasonics, *Research in Nondestructive Evaluation*, vol.13, no.2, pp.79-95.

Oishi, A.; Yoshimura, S.; Yagawa, G. (2002): Domain Decomposition Based Parallel Contact Algorithm and Its Implementation to Explicit Finite Element Analysis, *JSME International*, Series A, vol.45, no.2, pp.123-130.

Papadrakakis, M.; Lagaros, N. D.; Tsompanakis, Y. (1998): Structural optimization using evolution strategies and neural networks. *Computer Methods in Applied Mechanics and Engineering*, vol.156, pp.309-333.

Puso, M. A.; Laursen, T. A. (2002): A 3D contact smoothing method using Gregory patches, *International Journal for Numerical Methods in Engineering*, vol.54, pp.1161-1194.

Schweizerhof, K.; Nilsson, L.; Hallquist, J. O. (1992): Crash-worthiness analysis in the automotive industry, *International Journal of Computer Applications in Technology*, vol.5, pp.134-156.

Stavroulakis, G.E.; Antes, H. (1998): Neural crack identification in steady state elastodynamics. *Computer Methods in Applied Mechanics and Engineering*, vol.165, pp.129-146.

Vignjevic, R.; De. Vuyst, T.; Campbell, J. C. (2006): A Frictionsless Contact Algorithm for Meshless Methods. *CMES: Computer Modeling in Engineering & Sciences*, vol.13, no.1, pp.35-47.

Wang, F.; Cheng, J.; Yao, Z. (2001): FFS contact searching algorithm for dynamic finite element analysis, *International Journal for Numerical Methods in Engineering*, vol.52, pp.655-672.

Wang, S. P.; Nakamachi, E. (1997): The inside-outside contact algorithm for finite element analysis, *International Journal for Numerical Methods in Engineering*, vol.40, pp.3665-3685.

Zacharias, J.; Hartmann, C.; Delgado, A. (2004): Damage detection on crates of beverages by artificial neural networks trained with finite-element data. *Computer Methods in Applied Mechanics and Engineering*, vol.193, pp.561-574.

Zhong, Z. H. (1993): *Finite Element Procedures for Contact -Impact Problems*, Oxford U.P.

Zhong,Z. H.; Nilsson, L. (1996): A unified contact algorithm based on the territory concept, *Computer Methods in Applied Mechanics and Engineering*, vol.130, pp.1-16.