 CMES, vol.28, no.2, pp.127-146, 2008

Genetic Approaches to Iteration-free Local Contact Search

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

Abstract: This paper describes new methods based on genetic approaches for finding approximating expressions of local coordinates of a contact point in a local contact search process. A contact search process generally consists of the following 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 contact search can be regarded as the mapping from the coordinates of nodes to the local coordinates of contact points. In this paper, two methods are proposed to find mathematical expressions that approximate the mapping in the local contact search using either the genetic algorithm (GA) or the genetic programming (GP). For both methods, parallel processing is efficiently utilized, and it enables massive search for suitable expressions in reasonable time. The GA-based method is utilized for finding approximating polynomials for the mapping in the local contact search process, while the linear GP-based method is utilized for finding approximating rational expressions. Fundamental procedures of the proposed methods are first described in detail, and next their basic performance is thoroughly tested for sample data. Finally, the proposed method using the linear GP is successfully applied to find the approximating expressions of the local coordinate value in the local contact search process for the smoothed contact surfaces with Gregory patches, and the approximating expressions obtained by the proposed method prove to be very fast in estimating the local coordinate values of the contact point.

Keyword: Contact Search, Genetic Programming, Genetic Algorithm, Finite Element Method, Sliding Interface, Curved Surface.

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

1 Introduction

Concomitant with a remarkable progress in the field of microelectronics, the performance of computers has increased dramatically. This has been promoting the replacement of time-consuming and highly expensive experiments with such computer simulations as the finite element method (FEM). As computer simulation is increasingly utilized as a tool in design and analysis, more complex models will need to be simulated.

For the analyses of dynamic problems such as vehicle crashworthiness and metal forming, contact-impact phenomena must inevitably be taken into account [Zhong (1993), Schweizerhof, Nilsson and Hallquist (1992), Guz, Menshykov, Zozulya and Guz (2007), Guz and Zozulya (2007), Keppas, Giannopoulos and Anifantis (2008), Ozaki, Hashiguchi, Okayasu and Chen (2007), Vignjevic, De Vuyst and Campbell (2006)]. In the contact-impact analyses, the 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, Ω_2 are inner bodies of the objects, Γ_1, Γ_2 are boundaries of the objects. Criterion (1) means no penetration into each body. In dynamic contact-impact analyses, it must be checked every time step whether the contact conditions, i.e. criteria (1), (2), will be met between any two surfaces of the whole analysis domain. This process is called contact search.

The contact-impact algorithms in finite element contact analyses can be divided into two phases:

the contact-searching algorithm phase and the contact interface one. First, the contact-searching algorithm locates where contact occurs. Second, contact forces are applied to the contact surfaces according to the contact interface algorithm. The contact search process often consumes the most amount of time for the contact-impact algorithm. So, a fast and efficient contact search method has been desired.

The sliding interface algorithm [Hallquist, Goudreau and Benson (1985), Benson and Hallquist (1990)] is most popular, and has been implemented in DYNA3D code, 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 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. Contact search process usually consists of the following two phases: the global search phase and the local search phase. In the global contact search phase, the 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 selected pair of the segment and the slave node is examined precisely and the local coordinates of the contact point, if any, is determined by a time-consuming iterative method such as Newton's method.

Genetic approaches, such as Genetic Algorithms (GAs) [Goldberg (1989)] and Genetic Programming (GP) [Koza (1992), Koza (1994)], have been developed as engineering algorithms that simulate biological genetic mechanism. They are very useful tools for combinatorial optimization problems and have been successfully applied to various problems in engineering fields [de Lacerda and da Silva (2006), Globus, Menon and Srivastava (2002), Mathur, Advani and Fink (2003), Mustata, Harris, Elliott, Lesnic and Ingham (2000), Rama Mohan Rao, Appa Rao and Dattaguru (2004), Singh, Mani and Ganguli (2007)].

In the present paper, we propose a new iteration-free local contact search method using an approximating function that explicitly represents the local coordinates of a contact point. The approximating functions are determined through genetic approaches, such as genetic algorithms (GA) and genetic programming (GP), on parallel computers. Fundamental formulations of the proposed methods and their implementations on parallel computers are explained in detail. Sample analyses show that the approximating functions obtained by the proposed methods are faster in estimating the contact point than the conventional method based on the Newton's iteration and accurate enough for some practical applications.

2 Contact Search Algorithms in Sliding Interface Algorithm

2.1 Fundamental 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, there are two segments in one element, which correspond to both surfaces of the element. Figure 1 shows a configuration of a segment and a slave node, where P_s is a slave node and the quadrilateral A(P_m)BCD is a segment. P_m, the master node that has been identified as the nearest one from the slave node P_s in the global search, usually belongs to multiple segments. The segment that P_s hits can be found by the following criterion:

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

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

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.

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

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

$\vec{t}$ is a vector drawn from P_m to P_s. The segment meeting these criteria (3) and (4) is picked up and denoted as a target segment.

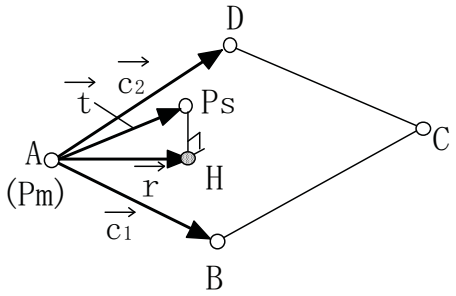

Figure 1: Local Contact Search

After the target segment is specified, the local coordinates (ξ, η) of an accurate contact point on the segment will be identified by solving the following set of equations with Newton's method [Hallquist, Goudreau and Benson (1985), Benson and Hallquist (1990), Oishi, Yoshimura and Yagawa (2002)]:

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

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

In addition, 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 the 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 a shape function.

2.2 Past Research on Local Contact Search without Iteration

Iterative solvers, such as Newton's method, have been utilized to obtain the local coordinates of the contact point in the node-segment type contact algorithm. As iterative solvers often meet difficulties in convergence and become time-consuming, local contact search methods without any iteration have been desired.

The approximating function described in literatures [Zhong and Nilsson (1996), Wang and Nakamachi (1997)], which uses area coordinates, is the best-known one of this type. The values of four shape functions for the contact point are approximated by the function containing areas of four triangles made of one slave node and two nodes on the master segment as shown in Figure 2.

In the figure, the quadrilateral segment consists of four nodes labeled from 1 to 4, and the point H is the contact point. Δ_1 is the area of the triangle ΔHP_1P_2, Δ_2 is that of the triangle ΔHP_2P_3, Δ_3 is that of the triangle ΔHP_3P_4, Δ_4 is that of the triangle ΔHP_4P_1. Each shape function value is estimated by the following equations, respectively,

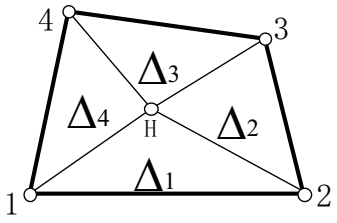

Figure 2: Area coordinates for a quadrilateral segment

$$\Phi_1 = \Delta_2\Delta_3/\Delta \tag{12}$$

$$\Phi_2 = \Delta_3\Delta_4/\Delta \tag{13}$$

$$\Phi_3 = \Delta_4\Delta_1/\Delta \tag{14}$$

$$\Phi_4 = \Delta_1\Delta_2/\Delta \tag{15}$$

where Δ is defined by the following equation:

$$\Delta = (\Delta_1 + \Delta_3) \times (\Delta_2 + \Delta_4) \tag{16}$$

In this method, the value of each shape function at the contact point is quickly obtained in an explicit form without iteration, and the local coordinates can also be obtained from the values of shape

functions if each shape function is linear. Though this method has been adopted in several applications due to its fast calculation, it often shows poor accuracy for configurations where four master nodes are not coplanar.

Pinball algorithm [Belytschko and Neal (1991)] is another 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. This algorithm is fast, but it also has some difficulties in accuracy due to its approximate representation of contact surfaces.

Recently, neural networks [Haykin (1994), Hassoun (1995)] have been applied to the local contact search process by the authors [Oishi and Yoshimura (2007)]. Neural networks can simulate arbitrary continuous functions [Funahashi (1989)] and have been applied to various engineering problems [Oishi, Yamada, Yoshimura, Yagawa, Nagai and Matsuda (2001), Papadrakakis, Lagaros and Tsompanakis (1998), Noroozi, Sewell and Vinney (2006), Furukawa and Yagawa (1998)]. Their capabilities of simulating functions have been utilized to simulate the mapping that appears in the local contact search process. Though the neural network approach is applicable to various kinds of local contact search processes, this approach is not very fast because many madd (multiply and add) operations and evaluations of the activation functions, such as sigmoid functions, used in the neural network need some computational power.

3 Searching for Optimal Polynomial Approximation Using Genetic Algorithms

3.1 Optimal Polynomial Approximation Based on Least Squares Method

Local coordinate values of the contact point are obtained by solving equations (7), (8). Solving equations (7), (8) is, in other words, finding the mapping functions f and g from the fifteen known 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, S 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 expressed as follows:

$$\xi_c = f(x_A, y_A, z_A, \cdots, x_S, y_S, z_S) \tag{17}$$

$$\eta_c = g(x_A, y_A, z_A, \cdots, x_S, y_S, z_S) \tag{18}$$

Though exact representations of the functions f and g are unknown, they can be approximated by polynomials with desired precision. In this research, approximating polynomials of f and g are searched, respectively, using genetic algorithms (GA) [Oishi, Yoshimura and Yagawa (2004)].

As input variables of f and g, we do not employ 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 along with the node A.

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

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

(4) Using the distance between the nodes A and 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 the above transformation, fifteen 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 functions f and g can be reduced to eight coordinate values $x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S$, and f and g are expressed in the following forms:

$$\xi_c = f(x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S) \tag{19}$$

$$\eta_c = g(x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S) \tag{20}$$

Assuming the N-th order perfect polynomials as a candidate, an approximating polynomial for ξ_c is represented as follows:

$$\begin{aligned}\xi_c &= f\left(x_C, y_C, z_C, x_D, y_D, x_S, y_S, z_S\right) \\ &= a_0 + \sum_{n=1}^{N} \sum_{k_1^i+\cdots+k_8^i=n}^{i} a_i x_C^{k_1^i} \cdots z_S^{k_8^i}\end{aligned} \tag{21}$$

Each coefficient a_i of equation (21) is determined so as to minimize the total sum of the square error for multiple sample data points as described in the following:

$$\sum_i \left(f\left(x_C^i, y_C^i, \cdots, y_S^i, z_S^i\right) - \xi_C^i\right)^2 \rightarrow \min \tag{22}$$

Differentiate the left hand side of equation (22) by part with respect to each a_i and set it to be zero, then simultaneous linear equations of a_i is derived. The most appropriate a_i values from the viewpoint of the least square are obtained by solving those equations, and an approximating polynomial is determined.

As described above, the selected candidate polynomial limits the degree of approximation. If each of eight variables possesses third order polynomial respectively, the highest order of the terms of the approximating function becomes twenty-fourth order polynomial and the number of its terms may be as many as 65536. The polynomial with such a huge number of terms cannot be practically used for approximating function because it takes very long computation time for evaluating the value of the polynomial. This implies that the number of terms involved in the approximating polynomial must be reduced until calculation load falls within a feasible level without badly degrading the accuracy of approximation. Thus, a key issue is to find the polynomial that consists of as few terms as possible but realizes sufficient accuracy in approximation. In this research, GAs are adopted for searching polynomials that meet the above criterion, and parallel processing is also utilized for calculation speedup.

3.2 Searching for Optimal Polynomial Approximation Using Genetic Algorithms

Genetic algorithms are a kind of artificial optimization methods that mimic the theory of evolution, i.e. natural selection [Goldberg (1989), Michalewicz (1992)]. In genetic algorithms, the fundamental unit to be optimized is called an individual, which is an array of parameters to be adjusted, which is called genes. A group of individuals that are to be optimized evolve by repeatedly suffering genetic operations that simulate the procedures of natural selection. Operations in GAs consist of the following four genetic operations:

(1) Crossover: Two individuals are picked up at random, then randomly selected sequence of several genes of both individuals are interchanged to make two new individuals.

(2) Mutation: One individual is picked up at random, and then one randomly selected gene of the individual is modified to make a new individual.

(3) Evaluation: Every individual in the population is evaluated, that is, its degree of fitness for target objectives is calculated.

(4) Selection: Every individual is tested for survival to the next generation. The higher the fitness of the one individual is, the higher the probability of its survival to the next generation is.

In this research, each individual represents a polynomial expression, and polynomials that approximate the mapping from coordinates of related nodes to the local coordinates of the contact point in the local contact search process are searched using GAs. Hereafter, to make explanation clear and compact, the implementation of perfect polynomials with two variables into the individual of GAs is explained as an example. The second-order perfect polynomial with two variables, x and y, consists of six terms as follows:

$$a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \tag{23}$$

where $a_0, \cdots, a_5$ are real-valued coefficients. Polynomials with two variables, in which the order of each term is equal to or under than two,

are implemented on the GA individual that consists of six genes where each gene corresponds to one term in the second-order perfect polynomial, respectively. When the value of a gene is one, the polynomial represented by the individual including the gene has the corresponding term, and when the value of a gene is zero, it does not have the corresponding term. For example, the individual (110001) represents the following polynomial,

$$a_0 + a_1 x + a_5 y^2 \tag{24}$$

and (100110) represents the following one,

$$a_0 + a_3 x^2 + a_4 xy \tag{25}$$

Replacing the sequence of genes in the individual (24), which are shaded in equation (26), with correspondent counterpart in the individual (25), which are also shaded in equation (27), and vice versa,

$$(110\,001)\ a_0 + a_1 x + a_5 y^2 \tag{26}$$

$$(100\,110)\ a_0 + a_3 x^2 + a_4 xy \tag{27}$$

it makes two new individuals as shown in equations (28) and (29). This is an example of the crossover.

$$(110\,110)\ a_0 + a_1 x + a_3 x^2 + a_4 xy \tag{28}$$

$$(100\,001)\ a_0 + a_5 y^2 \tag{29}$$

An example of mutation is shown below. The shaded gene in equation (30) suffers mutation such that the value of the gene is changed from 0 to 1.

$$(110\,0\,01)\ a_0 + a_1 x + a_5 y^2 \tag{30}$$

This mutation results in the following new polynomial expression:

$$(110\,1\,01)\ a_0 + a_1 x + a_3 x^2 + a_5 y^2 \tag{31}$$

Repeating crossover and mutation operations generation by generation results in creating a lot of new individuals to be evaluated.

Evaluation of each newly created individual, i.e. a polynomial expression, consists of two processes. Firstly, all the coefficients of the terms in the polynomial are determined using the least squares method as described in the previous section. Secondly the fitness of the individual with its coefficients is evaluated. For example, the $fitness$ of the individual for polynomials that approximate local coordinate ξ is defined as follows:

$$fitness = \frac{1}{\sum_i \left| f\left(x_C^i, y_C^i, \cdots, y_S^i, z_S^i\right) - \xi^i \right|} \tag{32}$$

where $f(\cdot)$ is the polynomial to be evaluated, $x_C^i \cdots z_S^i$ is the coordinate values of the related nodes in the i-th sample data and ξ^i is the local coordinate value to be approximated, i.e. the target value.

In the selection process, every individual is tested and reproduced according to its fitness value: the higher the fitness of an individual is, the more probably it will be reproduced for the next generation. In this research, roulette type selection rule is adopted.

By repeating the four operations, crossover, mutation, evaluation and selection generation by generation, approximating polynomials that fit well to the target function can be obtained out of huge amount of candidates created during many generations.

4 Searching for Optimal Approximating Function using Genetic Programming

4.1 Genetic Programming

The genetic programming (GP) was invented by Koza [Koza (1992), Koza (1994), Langdon and Poli (2002)]. It can be regarded as a variation of genetic algorithms. Major difference between GA and GP is that the latter can handle structural representation such as tree structure. Though genetic operations such as crossover, mutation, evaluation and selection, act on individuals that are a simple sequence of bits in GA, almost the same genetic operations act on individuals represented by tree structure. The tree structure can represent numerical expressions. Figure 3 shows an example of tree structure representation of a numerical expression, $x+3y$. In the figure, each circle is called

a node: the nodes that have x, 3 and y are terminal nodes, the nodes that have + and - are non-terminal nodes, and the node + is also called the root node. The tree structure can represent various mathematical expressions by setting variables and constants in the terminal nodes and setting operators or functions in the non-terminal nodes.

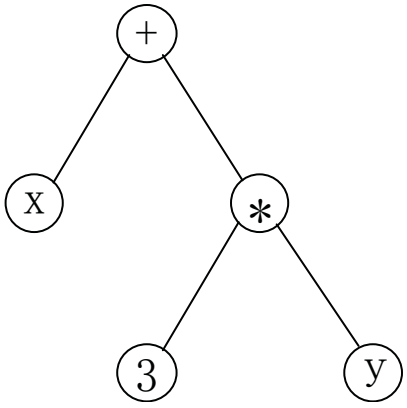

Figure 3: Tree representation of mathematical expression

Genetic operations such as crossover and mutation for tree structures in GP are also defined in the same way as those for bit sequence in GA. An example of the crossover of two tree structures is shown in Figure 4.

The crossover of two individuals that represent $y(3+x)$ and $3xy+x$ respectively creates two new individuals that represent $3xy^2$ and $3+2x$. In the crossover of tree structures, a selected branch of one individual is exchanged with a selected branch of another individual to produce new individuals. An example of the mutation is also shown in Figure 5. The individual that represents $y(3+x)$ is mutated to produce the new individual that represents $3xy$. In the mutation of the tree structure, one node is selected at random for mutation, then it is changed to different value if the selected one is a terminal node and otherwise to different operator.

A whole procedure of genetic programming is summarized as follows:

(1) Initial Population: A number of individuals with tree structure are generated at random.

(2) Evaluation: Fitness of each individual is evaluated based on a prescribed measure.

(3) Selection and Reproduction: Each individual is tested for reproduction. The higher its fitness is, the more probably it is reproduced to the next generation.

(4) Crossover: Two randomly selected individuals are crossed to produce two new individuals.

(5) Mutation: Randomly selected individual is mutated to a new individual.

(6) Go back to (2)

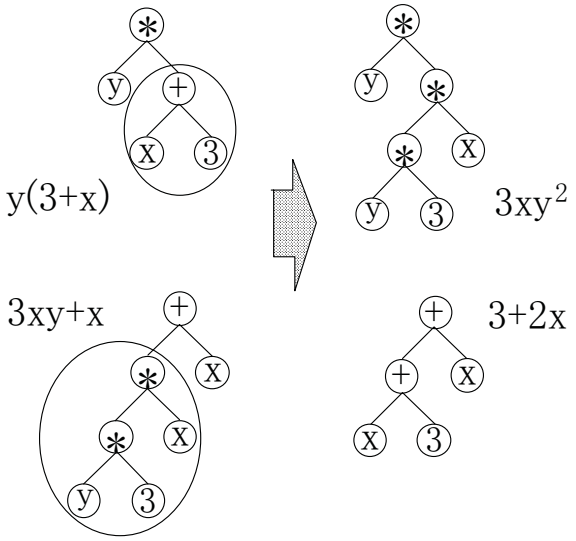

Figure 4: Crossover

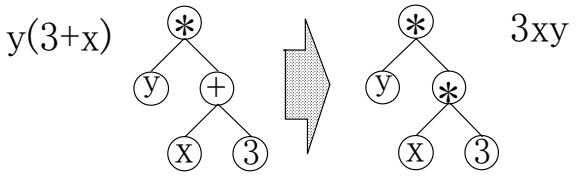

Figure 5: Mutation

4.2 Linear Genetic Programming

Tree structure of an individual in the genetic programming is usually implemented using pointers

in C language. Though it is a natural implementation where crossover between two individuals can be realized only by exchanging pointers for subtrees, it has some disadvantages in computation speed during evaluation and in implementation on a parallel processing environment. Our target application must evaluate mathematical expressions represented by individuals in GP for a vast amount of test data, so that both efficiency in evaluation and parallel processing are essential.

Linear genetic programming which uses a one-dimensional array to express mathematical expressions is faster in speed, better in search performance and more efficient in memory usage than the pointer-based genetic programming [Tokui and Iba (1999)]. In this research, linear genetic programming utilizing stack to represent tree structures is adopted. Mathematical expressions are implemented on the one-dimensional array in a prefix order, where every operator is placed before its operands. Table 1 shows some examples of mathematical expressions represented by the linear GP.

Table 1: Linear GP Representation

Mathematical Expression	Representation by Linear GP
x+3y	+x*3y
y(3+x)	*y+3x
3xy+x	+*3*xyx

Though any mathematical expression can be represented by the linear GP, two genetic operations, crossover and mutation, that are easy to be implemented on the pointer-based GP, need some special treatment. Unlike every node in the pointer-based GP automatically represents a subtree which can be crossed or mutated, in the linear GP the subtree arisen from one node, i.e. the tree structure with the node as root, cannot be automatically obtained but must be determined by checking the sequence of nodes in the array just after the node. Stack count is used for this purpose. Every node has its own stack count according to its node type: the stack count of an operator node that has two operands, such as + and *, is -1, that of an operator node that has only one operand, such as sin() and cos(), is 0 and that of variable or constant is +1. With this definition of the stack count, a total sum of stack count of the entire nodes in subtree is always 1. Once a node is specified, checking the stack count can pick up a subtree starting from the node.

As an example, let's consider the following mathematical expression:

$$3xy + y. \tag{33}$$

This is represented by the linear GP as follows:

$$[+*3*xyy] \tag{34}$$

Its stack count is also given by

$$\begin{bmatrix} -1 & -1 & 1 & -1 & 1 & 1 & 1 \end{bmatrix}. \tag{35}$$

Therefore, once the second node from the left is selected as:

$$[+ \ * \ 3 * x \ y \ y] \tag{36}$$

then the subtree starting from the selected node is picked up by

$$[+ \ * \ 3 * x \ y \ y] \tag{37}$$

because the sum of the stack count of the contiguous five genes starting from the selected gene equals to 1, as shown below:

$$\begin{bmatrix} -1 & -1 & 1 & -1 & 1 & 1 \end{bmatrix}. \tag{38}$$

This shows that the subtree starting from any given node in the one dimensional array is determined by the correspondent array of each gene's stack count. The stack count is also utilized to evaluate the fitness of each individual: each value of operators can be evaluated one after another using stack operation in interpreting the genes of the individual in the reverse order.

In this research, one dimensional integer array is adopted for representing the individual in the linear GP. Operators and variables are represented by corresponding integer numbers, and constants are represented by integer numbers that indicate corresponding locations in the real array that consists of previously selected real values for constants.

4.3 Searching for Optimal Expressions for Local Contact Search using Linear GP

Our goal is to find as compact and accurate expression as possible for the mapping in the local contact search. As described in equations (19) and (20), the mapping to be approximated is the one from eight coordinates of nodes to the local coordinate value of the contact point. Unlike the GA case, mathematical expressions to be searched by the linear GP are not restricted within polynomials.

In the case of quadrilateral segment derived from a linear hexahedral element and a slave node, the fundamental procedure of the search for mathematical expressions of the mapping in the local contact search process using the linear GP is summarized as follows:

(1) Data Preparation: A number of configurations of five nodes shown in Figure 1, i.e. one slave node Ps and four master nodes A, B, C and D that construct quadrilateral master segment, are generated at random, then the corresponding contact point H for each configuration is calculated using Newton's method. A number of data pairs, coordinates of related nodes and the local coordinates ξ, η of the corresponding contact point, are collected.

(2) Search by Linear GP: Using the linear GP, approximating functions that use coordinate values of nodes as input data and output the local coordinate value ξ or η of the contact point are searched. This process leads to finding the explicit mathematical expressions of the local coordinate ξ or η of the contact point as the functions of coordinate values of the related nodes.

(3) Application: The approximating expressions obtained in the procedure (2) are implemented in the code for local contact search.

This method basically applies to any local contact between a slave node and a master segment derived from various elements, such as four-noded tetrahedral elements, ten-noded tetrahedral elements and twenty-noded hexahedral elements.

5 Parallel Processing of Genetic Approaches

5.1 PC cluster as Parallel Processing Environment

Parallel processing is a promising technique for high performance computing [Almasi and Gottlieb (1994), Ma, Lu, Wang, Roy, Hornung, Wissink and Komanduri (2005), Ha, Seo and Sheen (2006)]. A large amount of computational workload distributed among processors may lead to the speedup proportional to the number of processors used if additional overheads due to parallel processing are negligible [Grimaldi, Pascazio and Napolitano (2006), Namilae, Chandra, Srinivasan and Chandra (2007), Trindade and Pereira (2007), Moulinec, Issa, Marongiu and Violeau (2008)]. A PC cluster is a popular parallel processing machine. It usually uses Linux OS equipped with MPI [Gropp, Lusk and Skjellum (1999), Pacheco (1997)] and Ethernet for communication among processors. PC clusters, however, usually have relatively large latency in interprocessor communication. Therefore, the larger computational grain, i.e. the larger amount of computational workload performed without any interprocessor communication, is more suitable to parallel processing in PC clusters.

5.2 Parallel Processing of Genetic Approaches

Both genetic approaches in this research, the GA and the Linear GP for searching approximating functions of the mapping in the local contact search process, are computation intensive in the evaluation process that calculates the fitness of every individual for a lot of sample data. In addition, as the computation of fitness value of each individual can be performed independently, implementing the evaluation process in the parallel processing environment is not difficult. Therefore, parallel processing is employed for the present genetic approaches.

There exist two types of implementations of parallel processing for our genetic approaches depending on the number of populations. The one consists of only one group of individuals where all the individuals belong to, and the other consists of several groups of individuals where all the in-

dividuals are divided into several groups and each group evolves independently with occasional interchanges of individuals among groups. We call the former a parallel GA /GP, the latter a parallel and distributed GA/GP.

In the present research, a parallel version of genetic algorithm, denoted as p-GA, and a parallel-distributed version of linear genetic programming, denoted as pd-LinearGP, are implemented in PC cluster. PCs in the cluster are divided into two categories, one server PC and other client PCs. In the case of p-GA, only the evaluation process is distributed among the client PCs and other processes such as crossover, mutation and selection are performed in the server PC. In the case of pd-LinearGP, a whole of individuals are first divided into several groups and each group independently goes through the whole processes of linear GP with occasional interchange of individuals among groups. The server PC manages the exchange of individuals in the latter case, so arbitrary exchange pattern is possible. In this research, we virtually place all the client PCs in a circle and one PC always sends individuals to the previously chosen PC that is assumed to be adjoining in the virtual circle. In both cases, the amount of data to be communicated among PCs is relatively small: only the individuals and their fitness values, one dimensional integer arrays and real variables, are communicated among PCs. This compactness of the data to be communicated is suitable for PC clusters where communication speed is relatively slow.

6 Basic Performances

6.1 Problem Definition

Basic performance of the proposed methods is tested through sample analyses of the following test problem. Figure 6 shows the configuration of the test problem in which a slave node Ps and its corresponding four-noded quadrilateral master segment ABCD are located. 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 be zero. The x, y coordinates of the nodes C, D, Ps are set within the range of (-2.0, 2.0) at intervals of 0.2 in both directions. The z coordinate of the node C and Ps is set within the range of (-0.3, 0.3) at intervals of 0.1. 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) and the quadrilateral ABC*D, where C* is the projection of the node C onto the x-y plane, is convex. This condition reduces the number of node configurations to be considered without any loss of generalization, which improves efficiency of the genetic approaches due to eliminating redundancy.

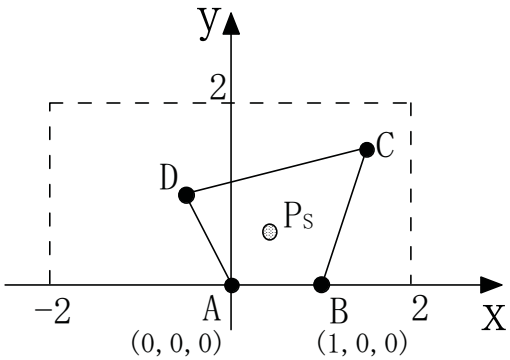

Figure 6: Configuration of nodes

With the above conditions, a number of patterns of configurations are obtained, and for each configuration the local coordinate values of the contact point are obtained by Newton's method. Data set, node configurations and corresponding local coordinates of the contact point, are utilized for test data in the evaluation process of the genetic approaches.

6.2 Performance of Genetic Algorithm Approach

6.2.1 Setting Up

Approximating polynomials for the mapping in the local contact search are searched by the proposed GA-based approach. A parallel GA code written in C using MPI is developed. The PC cluster, which consists of thirty-two Linux PCs

equipped with Pentium4 CPU and Fast Ethernet, in the Center for Advanced Information Technology of Tokushima University, is used. In this study, the polynomial expressions for ξ_c and η_c are determined by the proposed method. Out of the node configurations generated in the previous section, 28546 patterns are selected at random for test data in the fitness evaluation in this test.

In this study, the highest order of each term in the polynomials to be determined is set to be three due to the restriction of practical computational workload for evaluating the value of the polynomial. Table 2 shows the relationship between the highest order among all terms in the perfect polynomial of eight variables and the estimated computational workload, such as the multiplication counts and the number of terms, for calculating its value. The number of multiplication is counted term by term. Though it is known that polynomials with one variable can be efficiently evaluated by Horner's method (Knuth, 1969), any similar efficient algorithm is not popular for polynomials with several variables. Therefore, computational workload is roughly estimated by the number of terms included in the polynomial.

Table 2: Number of Terms in the Perfect Polynomial with Eight Variables

Highest Order	Number of terms in the polynomial	Number of Multiplications
0	1	0
1	9	8
2	45	80
3	165	440
4	495	1760
5	1287	5720

Ten test runs for each setting of parameters are executed with different initial populations and different random number sequences. The maximum number of generations, MaxGeneration, is set to 3000. The average error of the best polynomial obtained in the ten test runs is defined as follows:

$$E_\xi = \frac{1}{N}\sum_{i=1}^{N}\left|\xi_i^{approx} - \xi_i^{exact}\right| \tag{39}$$

where N is the total number of test patterns, ξ_i^{approx} is the value of the polynomial for the local coordinate ξ of the contact point for the i-th configuration pattern and ξ_i^{exact} is that obtained by the Newton's method.

To test accuracy in the estimation of the local coordinate values of the contact point, Basic performance of the proposed methods is tested through sample analyses of the following test problem.

6.2.2 Results

Accuracy of Approximating Polynomials: Table 3 shows the average error E_ξ, E_η and the number of terms N_{term} of the best approximating polynomial obtained. The average errors E_ξ, E_η of the area coordinate method described in equations (12)-(16) are 0.2361 and 0.2174, respectively. This shows that the polynomial obtained by the proposed method is superior in accuracy to the area coordinate method.

Table 3: Average Error

E_ξ	N_{term}	E_η	N_{term}
0.017445	124	0.022610	129

Accuracy vs. The Number of Terms: Though the result above shows basic superiority of the proposed method in accuracy, the polynomial consists of almost 120 terms and it leads to much heavier computational load than other methods.

To reduce the number of terms of polynomials without degradation in accuracy, a new modified fitness value f_p of the individual in the proposed system is defined as follows:

$$f_p = \begin{cases} fitness & (n \le N) \\ \frac{fitness}{100\times(n-N)} & (n > N) \end{cases} \tag{40}$$

where $fitness$ is the basic fitness defined in equation (32), n is the number of terms in the polynomial and N is the target number of terms, i.e. n should be below N. Figure 7 shows the average error of the polynomial obtained with setting N to be five different values, 10,20,40,60 and 80. The vertical axis means the average error of the obtained polynomial, while the horizontal

axis means the number of terms of the polynomial. The average error of the polynomial obtained without restricting the number of terms, where N is set to 165, is also shown for the purpose of comparison. Figure 7 shows that the polynomial obtained with N>=60 shows little degradation in accuracy. In addition, though the average error of the polynomial obtained with N<=40 increases significantly, even the average error of the polynomial with ten terms is much smaller than that of the area coordinate method. The obtained polynomials with N=10 for local coordinates ξ, η are as follows:

$$\begin{aligned}\xi =& a_0 y_D + a_1 x_S + a_2 y_S + a_3 x_C y_S + a_4 y_C y_S \\ &+ a_5 x_C y_C y_S + a_6 x_C x_S y_S + a_7 x_C^2 y_D \\ &+ a_8 x_D^2 y_S + a_9 y_D x_S^2\end{aligned} \tag{41}$$

$$\begin{aligned}\eta =& b_0 + b_1 x_C + b_2 y_D + b_3 y_S + b_4 x_C y_S \\ &+ b_5 y_C y_D^2 + b_6 y_C y_S^2 + b_7 x_C y_D y_S \\ &+ b_8 y_C x_S y_S + b_9 y_D x_S y_S\end{aligned} \tag{42}$$

With adjusting the order of operations, both polynomials can be evaluated by totally twenty-four multiplications and the total multiplication operations for the whole process including the affine preprocessing described before is almost equal to one hundred, which is double of that of the area coordinate method.

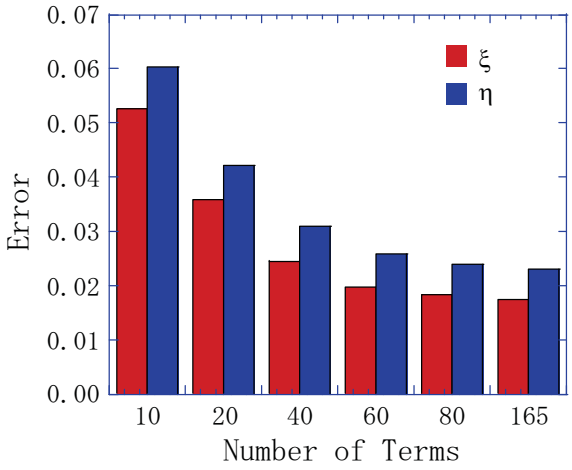

Figure 7: Average Error w.r.t. Number of Terms

Parallel Efficiency: Figure 8 shows the speedup of the proposed GA-based system in a parallel

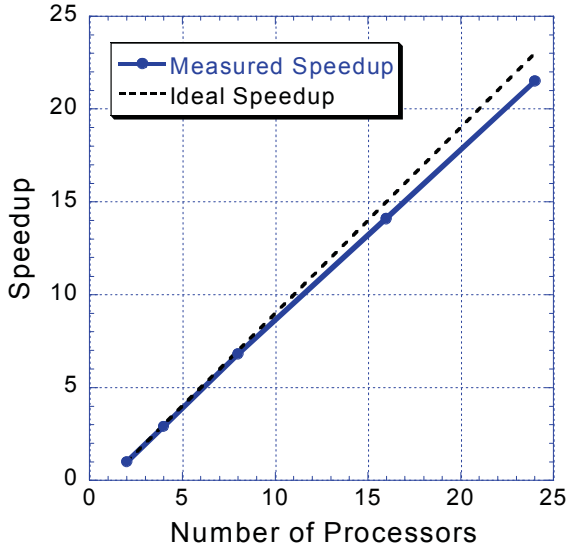

Figure 8: Speedup by Parallel Processing

processing environment. The vertical axis denotes the ratio of speedup in the parallel processing, while the horizontal axis does the total number of processors used. The ratio of speedup P_n is defined by the following equation:

$$P_n = \frac{T_1}{T_n} \tag{43}$$

where T_1 is the elapsed time for one generation using one client processor, T_n is that using n client processors. Figure 8 indicates the good scalability of the proposed GA-based system.

6.3 Performance of Genetic Programming Approach

6.3.1 Setting Up

A parallel and distributed linear GP code written in C using MPI is developed. The same PC cluster used in the GA-based case is also used. In this study, only the expressions for ξ_c are searched by the linear GP. It is clear that the same methods and results of ξ_c directly apply to the search for expressions of η_c.

Two kinds of expressions are obtained: polynomials using +, - and × operators and rational expressions using +, -, × and / operators. Constants in the expressions are to be selected from the prescribed 201 values at an interval of 0.05 within the

range of (-5.0,5.0). The number of individuals per group is set to 200, the maximum count of generations to evolve is set to 3000 and the one-tenth of individuals of each group is exchanged among groups every 10 generations. As for the maximum number of genes, MaxGeneLength, of individuals, five kinds of length, 50, 100, 150, 200 and 250, are tested. For each combination of various parameters, ten different runs are tested changing random number sequence, and each individual is evaluated according to the average error of approximation for test data.

6.3.2 Results

Parallel Efficiency: Parallel efficiency of the present parallel and distributed linear GP system is tested changing the number of groups with the number of individuals per group unchanged. CPU time is measured in the case of MaxGeneLength=50 and MaxGeneration=500. Figure 9 shows the results. The vertical axis denotes elapsed time, while the horizontal axis does the number of groups, i.e. the number of client processors. It can be seen from the figure that the elapsed time does not depend on the number of groups. It indicates very high parallel efficiency of the present code, and it directly leads to the feasibility of massive search for best expression using a lot of groups of individuals.

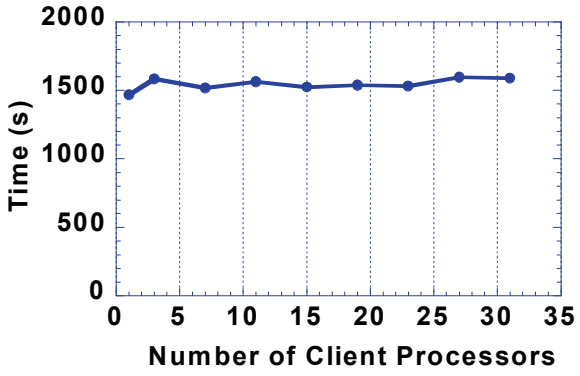

Figure 9: Parallel Performance

Number of Groups: Figure 10 shows how the performance of the best individual in accuracy depends on the number of groups for various MaxGeneLength values. The vertical axis denotes the

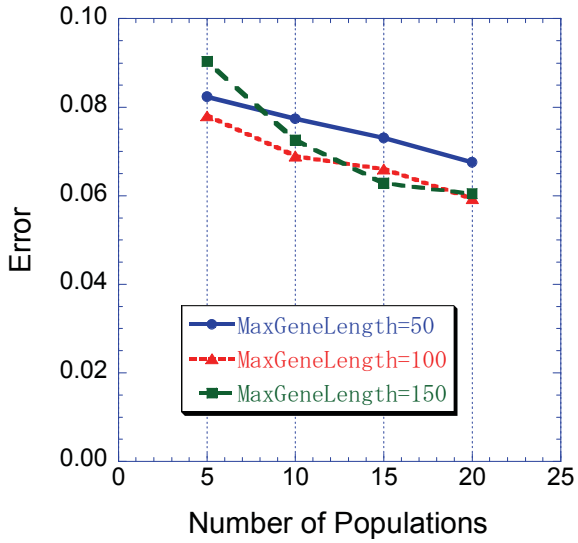

Figure 10: Average Error w.r.t. Number of Populations

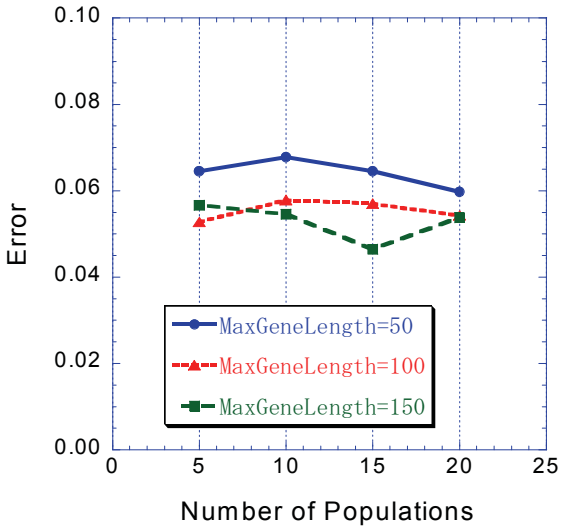

Figure 11: Minimum Error w.r.t. Number of Populations

average error of the expressions derived from the best individual obtained, while the horizontal axis does the number of groups. Twelve kinds of runs are performed under the conditions where MaxGeneLength = 50, 100 or 150 and the number of groups is set to 5, 10, 15 or 20, and the average error of the best expressions obtained from each condition is tested. Figure 10 shows the average of the average errors among ten different runs for each condition, and Figure 11 shows the min-

imum of the average errors among ten different runs for each condition. Though the average of the average errors becomes smaller when more groups are involved, the variation of the minimum error against the number of groups is much smaller than that of the average error. The computational load is almost proportional to the number of groups: the more groups are involved, the more computational time is required. Therefore, the number of groups is hereafter set to five through this research.

Length of Genes: The length of genes of individuals is apparently the most important factor in the linear GP system for the performance in accuracy of the corresponding expression. Five different values of MaxGeneLength ranging from 50 to 250 at an interval of 50 are tested. Figures 12 and 13 show the performance of the best expression for each MaxGeneLength. The left vertical axis denotes the error of the polynomial expression derived from the best individual obtained, while the right vertical axis does the count of multiplication operations of the polynomial expression in Figure 12, i.e. the count of multiplication and division operations of the rational expression in Figure 13, respectively. The horizontal axis denotes MaxGeneLength. The operation count of multiplications is used for evaluating the computational complexity of the obtained polynomial expression. Figure 12 shows that when MaxGeneLength increases, the polynomial expression derived from the individual includes more multiplication operations but shows little improvement in accuracy. As for the case of rational expression, Figure 13 shows almost the same results as those of polynomial expressions. The obtained accuracy shows little difference between polynomial expressions and rational expressions.

Computation Time vs. MaxGeneLength: The MaxGeneLength, as well as the number of individuals per group, the number of test data for evaluation of individuals and the number of generations to evolve, have impact on the required CPU time. Therefore, for five kinds of MaxGeneLength, 50, 100, 150, 200 and 250, the elapsed CPU time for one generation is measured. Figure 14 shows the results for both polynomial expressions and rational expressions. The vertical axis denotes the CPU time per generation, while the horizontal axis does MaxGeneLength. For both cases, the CPU time per generation is almost proportional to the value of MaxGeneLength. The CPU time of the rational expression case is almost 20 % longer than that of the polynomial case due to both the additional CPU cost of division over multiplication and the fact that division needs extra operations in order to prevent division by zero.

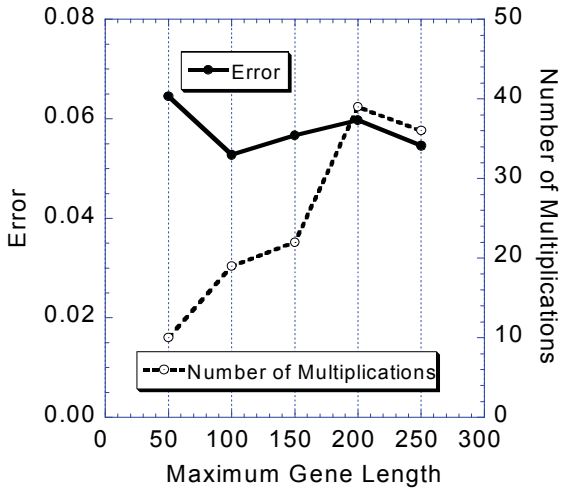

Figure 12: Accuracy of approximating polynomials

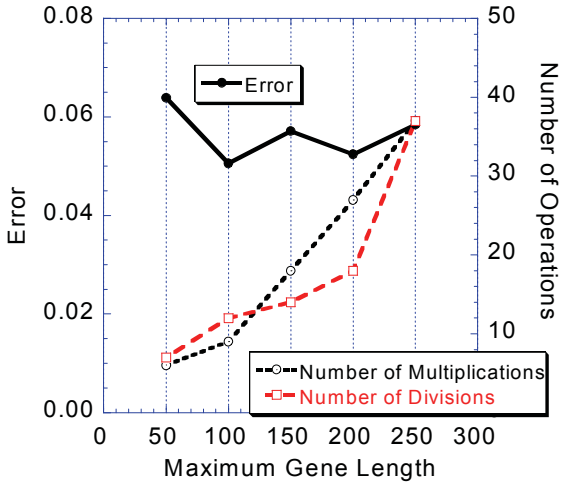

Figure 13: Accuracy of approximating rational expressions

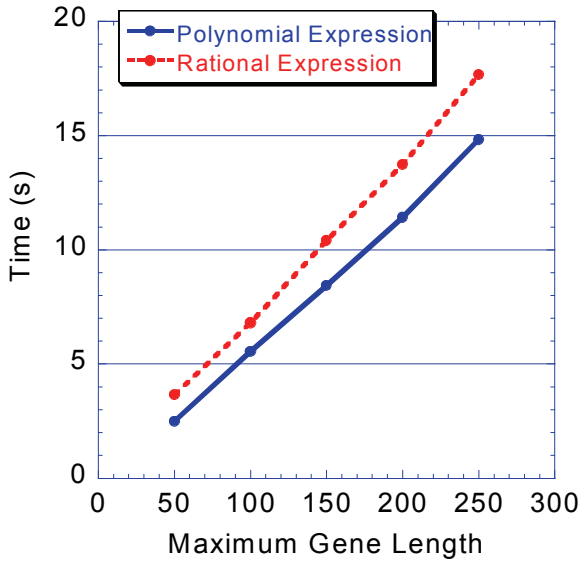

Figure 14: Computation Time

7 Application of Genetic Approach to Smoothed Contact Surface

7.1 Smoothed Contact Surface

The proposed local contact search method based on the linear GP system is applied to find the approximating functions for the mapping of the local contact search in smoothed contact surfaces.

Ordinary contact surfaces consist of facets, each of which is one face of elements exposed to the contact surface. 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 makes contact analyses difficult in numerical stability and in convergence. Therefore, contact smoothing has been desired [Wang, Cheng and Yao (2001), Puso and Laursen (2002)].

Bezier surfaces, typical smoothing surfaces, can be represented by the set of Bezier patches, i.e. quadrilateral facets made from four vertex points and additional control points [Farin (2002)]. A contact surface consists of quadrilateral facets, faces of elements that reside in the surface, and each facet can be regarded as the Bezier patch if additional control points are given. Bezier patches, however, can construct smoothing surface of only C^0 continuity at the boundary between patches, which is insufficient for numerical stability and convergence property. 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 also be applied to the contact surface arose from unstructured meshes [Puso and Laursen (2002), Chiyokura and Kimura (1983)]. Figure 15 shows the Bezier patch, and Figure 16 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 consist of twenty points. The control points other than the four corner nodes can be generated by calculation using the coordinate values of the four corner nodes and normal vectors at the corners. The node normal vector $\vec{n}_A$ at the corner node is defined by the following equation and is illustrated in Figure 17,

$$\vec{n}_A = \frac{\sum_i \left(\vec{c}_1^{(i)} \times \vec{c}_2^{(i)} \right)}{\left| \sum_i \left(\vec{c}_1^{(i)} \times \vec{c}_2^{(i)} \right) \right|} \tag{44}$$

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.

The Gregory patch is represented by the following equation:

$$P(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} B_i^3(u) B_j^3(v) P_{ij}(u,v) \tag{45}$$

where $B_i^n(\cdot)$ is the n-th order Bernstein polynomial,

$$B_i^n(u) = {}_nC_i \cdot u^i \cdot (1-u)^{n-i} \tag{46}$$

and $P_{11}, P_{21}, P_{22}, P_{12}$ are actually the linear combinations of two control points as follows.

$$P_{11}(u,v) = \frac{uP_{110} + vP_{111}}{u+v} \tag{47}$$

$$P_{12}(u,v) = \frac{uP_{120} + (1-v)P_{121}}{u + (1-v)} \tag{48}$$

$$P_{21}(u,v) = \frac{(1-u)P_{210} + vP_{211}}{(1-u) + v} \tag{49}$$

$$P_{22}(u,v) = \frac{(1-u)P_{220} + (1-v)P_{221}}{(1-u) + (1-v)} \tag{50}$$

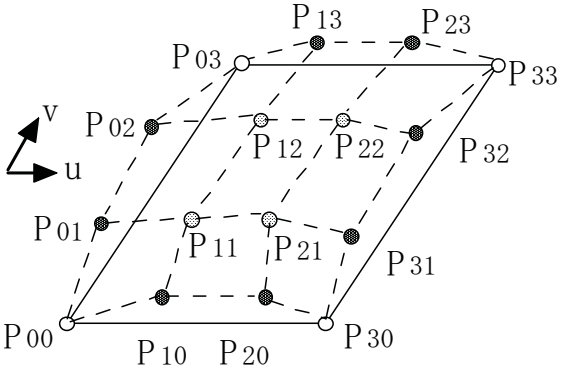

Figure 15: Control Points for Bezier Patch

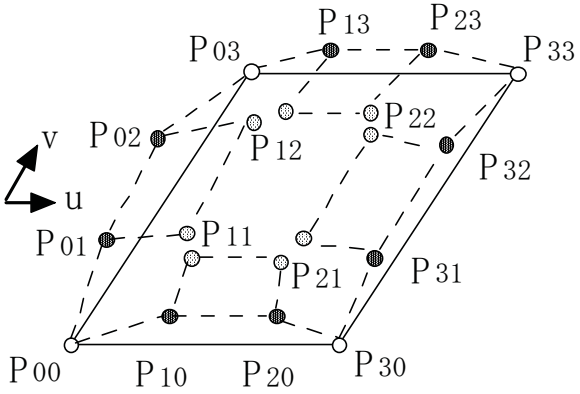

Figure 16: Control Points for Gregory Patch

7.2 Application of Linear GP System to Local Contact Search for Smoothed Contact Surface with Gregory Patches

7.2.1 Problem definition

Basic performance of the proposed method using the linear GP for finding the approximating functions that explicitly represent the mapping in the local contact search for smoothed contact surfaces with Gregory patches is tested through sample analyses of a test problem. Figure 6 again shows the configuration of the test problem, and the same patterns of configurations of nodes are considered. For each pattern of configuration, twenty patterns with different node normal vectors at corner nodes are generated. Each node normal vector is set in random directions where the tangent between the node normal vector and the normal vector at the corner node is below 0.2. With these conditions, the mapping pattern, i.e. the mappings from 16 input data to the local coordinate of the contact point are generated. Sixteen input data consist of three coordinate values of the node C, two coordinate values, x and y, of the node D, three coordinate values of the slave node Ps and eight vector component values of node normal vectors (two values of the ratio of the x- and y-component to z-component per node). Finally, 835906 patterns of configurations are generated and 83552 patterns out of them are selected as the test data for fitness evaluation in the linear GP-based system.

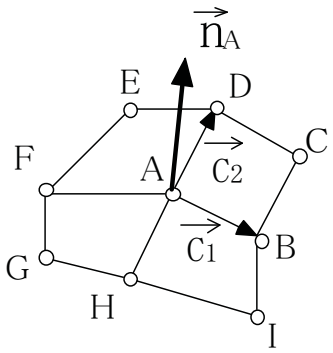

Figure 17: Nodal Normal Vector

7.2.2 Results

Five different values of MaxGeneLength ranging from 50 to 250 at an interval of 50 are tested. Figures 18 and 19 show the performance of the best expression for each MaxGeneLength. In these figures, the left vertical axis denotes the error of the expression derived from the best individual obtained, while the horizontal axis does MaxGeneLength. The right vertical axis denotes the

count of multiplication operations of the polynomial expression in Figure 18, and the count of multiplication and division operations of the rational expression in Figure 19, respectively. Figures 18 and 19 show that as MaxGeneLength increases, the expression derived from the individual includes more operations but shows little improvement in accuracy. The obtained accuracy shows little difference between polynomial expressions and rational expressions. As for the total number of multiplication and division operations in the obtained expression, it is within 50 for the polynomial expression and within one hundred for the rational expression, and both of them are much lower than that required to solve equations (7), (8) by the Newton's iteration for the patches described in equation (45).

For five kinds of MaxGeneLength, 50, 100, 150, 200 and 250, the elapsed CPU time for one generation is measured. Figure 20 shows the results for both polynomial expressions and rational expressions. The vertical axis denotes the CPU time per generation, while the horizontal axis does MaxGeneLength. For both cases, the CPU time per generation is again almost proportional to the value of MaxGeneLength.

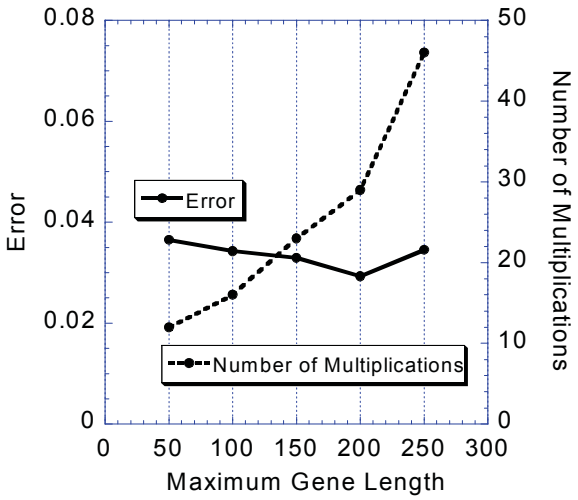

Figure 18: Accuracy of approximating polynomials

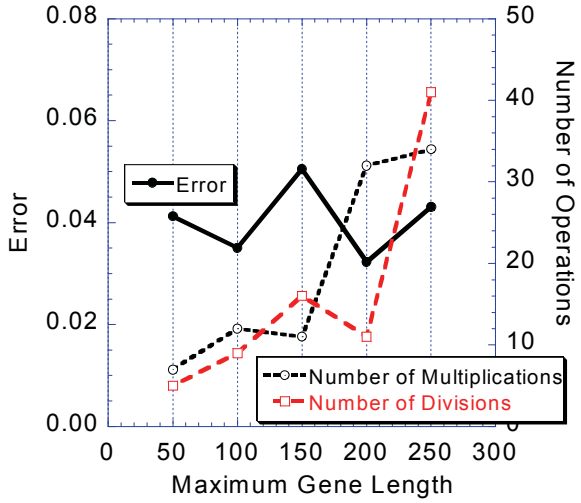

Figure 19: Accuracy of approximating rational expressions

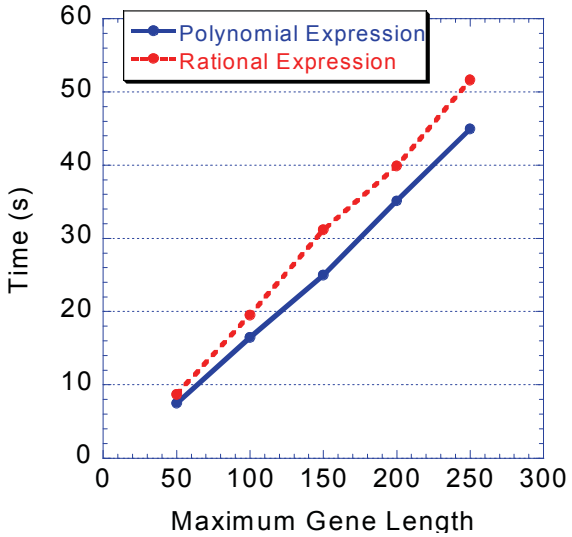

Figure 20: Computation Time

8 Conclusion

In this study, new methods based on genetic approaches for finding mathematical expressions that explicitly represent local coordinates of a contact point in a local contact search process. Two methods using either the genetic algorithm or the genetic programming, respectively, are proposed. For both methods, parallel processing is efficiently utilized.

The method based on the genetic algorithm is uti-

lized for finding approximating polynomials for the mapping in the local contact search process, and the following results are obtained.

(1) Approximating polynomials that are much superior in accuracy to the area coordinate method are obtained, while they are slightly more intensive in computation.

(2) Using parallel processing enables massive search with the proposed method.

The other method based on the linear genetic programming is utilized for finding approximating rational expressions, and the following results are obtained.

(1) Approximating expressions, either polynomials or rational expressions that are superior in accuracy to the area coordinate method are obtained, while they are also slightly more intensive in computation.

(2) Using parallel processing enables massive search with the proposed method.

(3) Both polynomials and rational expressions have achieved good accuracy in the similar level for sample data, while rational expressions are not as fast in evaluation speed as rational expressions due to division operations.

The linear GP-based method is then applied to find approximating expressions for the mapping in the local contact search process for smoothed contact surfaces using Gregory patches, and the following results are obtained.

(1) Good approximating expressions, either polynomials or rational expressions are obtained, and using these expressions enables much faster evaluation of the local coordinates of the contact point for smoothed surface than the ordinary Newton's method.

Both proposed methods can be easily applied to any contact surfaces arisen from various finite elements. In addition, the accuracy of expressions can be controlled by the length of genes. Therefore these methods are applicable to various applications.

References

Almasi, G.S.; Gottlieb, A. (1994): *Highly Parallel Computing 2nd edition*, Benjamin, Redwood City, CA.

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.

de Lacerda, L.A.; da Silva, J.M. (2006): A Dual BEM Genetic Algorithm Scheme for the Identification of Polarization Curves of Buried Slender Structures. *CMES: Computer Modeling in Engineering & Sciences*, vol.14, no.3, pp.153-160.

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*, **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.

Globus, A.; Menon, M.; Srivastava, D. (2002): JavaGenes: Evolving Molecular Force Field Parameters with Genetic Algorithm. *CMES: Computer Modeling in Engineering & Sciences*, vol.3, no.5, pp.557-574.

Goldberg, D.E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Grimaldi, A.; Pascazio, G.; Napolitano, M. (2006): A Parallel Multi-block Method for the Unsteady Vorticity-velocity Equations. *CMES: Computer Modeling in Engineering & Sciences*, vol.14, no.1, pp.45-56.

Gropp, W.; Lusk, E.; Skjellum, A. (1999): *Using MPI: Portable Parallel Programming with the Message-Passing Interface, second edition*, MIT Press, Cambridge, MA.

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.

Guz, A.N.; Zozulya, V.V. (2007): Investigation of the Effect of Frictional Contact in III-Mode Crack under Action of the SH-Wave Harmonic Load. *CMES: Computer Modeling in Engineering & Sciences*, vol.22, no.2, pp.119-128.

Ha, T.; Seo, S.; Sheen, D. (2006): Parallel iterative procedures for a computational electromagnetic modeling based on a nonconforming mixed finite element method. *CMES: Computer Modeling in Engineering & Sciences*, vol.14, no.1, pp.57-76.

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.

Hassoun, M.H. (1995): *Fundamentals of Artificial Neural Networks*, MIT Press.

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

Keppas, L.K.; Giannopoulos, G.I.; Anifantis, N. (2008): Transient Coupled Thermoelastic Contact Problems Incorporating Thermal Resistance: a BEM Approach. *CMES: Computer Modeling in Engineering & Sciences*, vol.25, no.3, pp.181-196.

Knuth, D.E. (1969): *The Art of Computer Programming Vol.2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA.

Koza, J.R. (1992): *Genetic Programming*, MIT Press, Cambridge, MA.

Koza, J.R. (1994): *Genetic Programming II*, MIT Press, Cambridge, MA.

Langdon, W.B.; Poli, R. (2002): *Foundations of Genetic Programming*, Springer-Verlag, Berlin.

Ma, R.; Lu, H.; Wang, B.; Roy, S.; Hornung, R.; Wissink, A.; Komanduri, R. (2005): Multiscale Simulations Using Generalized Interpolation Material Point (GIMP) Method And SAMRAI Parallel Processing. *CMES: Computer Modeling in Engineering & Sciences*, vol.8, no.2, pp.135-152.

Mathur, R.; Advani, S.G.; Fink, B.K. (2003):A Real-Coded Hybrid Genetic Algorithm to Determine Optimal Resin Injection Locations in the Resin Transfer Molding Process. *CMES: Computer Modeling in Engineering & Sciences*, vol.4, no.5, pp.587-602.

Michalewicz, Z. (1992): *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin.

Moulinec, C.; Issa, R.; Marongiu, J.-C.; Violeau, D. (2008): Parallel 3-D SPH Simulations. *CMES: Computer Modeling in Engineering & Sciences*, vol.25, no.3, pp.133-148.

Mustata, R.; Harris, S.D.; Elliott, L.; Lesnic, D.; Ingham, D.B. (2000): An Inverse Boundary Element Method for Determining the Hydraulic Conductivity in Anisotropic Rocks. *CMES: Computer Modeling in Engineering & Sciences*, vol.1, no.3, pp.107-116.

Namilae, S.; Chandra, U.; Srinivasan, A.; Chandra, N. (2007): Effect of Interface Modification on the Mechanical Behavior of Carbon Nanotube Reinforced Composites Using Parallel Molecular Dynamics Simulations. *CMES: Computer Modeling in Engineering & Sciences*, vol.22, no.3, pp.189-202.

Noroozi, S.; Sewell, P.; Vinney, J. (2006): The Application of a Hybrid Inverse Boundary Element Problem Engine for the Solution of Potential Problems. *CMES: Computer Modeling in Engineering & Sciences*, vol.14, no.3, pp.171-180.

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 Fi-

nite Element Analysis, *JSME International*, Series A, vol.45, No.2, pp.123-130.

Oishi, A.; Yoshimura, S.; Yagawa, G. (2004): A polynomial approximation of the solution of local contact search using a parallel genetic algorithm. *Journal of the Japan Society for Simulation Technology*, vol.23, no.4, pp.326-332. (in Japanese)

Oishi, A.; Yoshimura, S. (2007): A New Local Contact Search Method Using a Multi-Layer Neural Network. *CMES: Computer Modeling in Engineering & Sciences*, vol.21, no.2, pp.93-103.

Ozaki, S.; Hashiguchi, K.; Okayasu, T.; Chen, D.H. (2007): Finite Element Analysis of Particle Assembly-water Coupled Frictional Contact Problem. *CMES: Computer Modeling in Engineering & Sciences*, vol.18, no.2, pp.101-120.

Pacheco, P.S. (1997): *Parallel Programming with MPI*, Morgan Kaufmann, San Francisco, CA.

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.

Rama Mohan Rao, A.; Appa Rao, T.V.S.R.; Dattaguru, B. (2004): Generating optimized partitions for parallel finite element computations employing float-encoded genetic algorithms. *CMES: Computer Modeling in Engineering & Sciences*, vol.5, no.3, pp.213-234.

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.

Singh, A.P.; Mani, V.; Ganguli, R. (2007): Genetic Programming Metamodel for Rotating Beams. *CMES: Computer Modeling in Engineering & Sciences*, vol.21, no.2, pp.133-148.

Tokui, N.; Iba, H. (1999): Empirical and Statistical Analysis of Genetic Programming with Linear Genome, *Proceedings of 1999 IEEE International Conference on Systems, Man and Cybernetics*, Vol.3, pp.610-615.

Trindade, J.M.F.; Pereira, J.C.F. (2007): On the Efficiency of the Parallel-in-Time Finite Volume Calculation of the Unsteady Navier-Stokes Equations. *CMES: Computer Modeling in Engineering & Sciences*, vol.20, no.1, pp.1-10.

Vignjevic, R.; De Vuyst, T.; Campbell, J.C. (2006): A Frictionless Contact Algorithm for Meshless Methods. *CMES: Computer Modeling in Engineering & Sciences*, vol.13, no.1, pp.35-48.

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.

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.