681.3:621

Paper No. 149 -13

Direct Simulation of Engineering Problems with a Fast Array Computer*

(1st Report, Basic Concept of Direct Simulation and
Parallel Computation Algorithms using Jacobi Method)

By Kunikazu HAYASHI** and Masaaki YOKOYAMA***

The finite difference or finite element method, which has been applied successfully to the numerical analyses of many engineering problems, consists of synthesizing linear equations for nodal points of mesh of continuous solids and solving these simultaneous linear equations by a proper algorithm using a conventional computer, which has one or a few arithmetic units (AU).

If an array computer is designed, which has hundreds of AUs connected in a network fashion, the time-consuming finite difference or finite element computations can be executed very fast, provided each AU is attached to each nodal point and parallel computations are performed at each AU according to common timing diagrams.

This report deals with the development of parallel computation algorithms using Jacobi method and discusses the computation efficiencies, aiming at the development of an array computer which can be used profitably to solve many engineering problems.

1. Introduction

Recently engineering designers in various industrial branches have been faced with large sized numerical computations, since the problems of structural design and of strength design to be solved have become complex and large, and the solution of them with high accuracy have come to be needed. Usually the time consumed for these computations is very long and the reduction of it is increasing in importance.

The most important problem of numerical analyses in engineering is large sized simultaneous linear equations, and the solution of it with high efficiency has been of increasing importance in recent years in conjunction with the developments of computational algorithms based on the finite difference or finite element method. If an efficient computational algorithm for the solution of this problem is established, it must be very useful and profitable. But in reality such an algorithm has not yet been found.

Considering these situation, this study aims to design and construct a new type computer system, which with high efficiency can solve simultaneous linear equations derived from finite difference or finite element method. This first report deals with the basic concept of this computer system, parallel computation algorithm using Jacobi method and their computational efficiencies.

* Received 3rd September, 1974.

** Professor.

*** Associate, Graduate School of Science and Engineering, Tokyo Institute of Technology, 4259 Nagatsuda Midori-ku, Yokohama.

In finite difference or finite element analysis, a structure or a continuous solid is at first divided into small elements and then linear equations of physical quantities are made at all nodal points. In ordinary method these equations, being synthesized as a system of simultaneous linear equations, are then solved by a conventional computer, which has only one or a few arithmetic units. From another point of view, each physical quantity of nodal point is regarded as an averaged value of those of neighboring nodal points. So if arithmetic units, each of which consists of simple arithmetic processor and memory, are attached to all nodal points and parallel iterative computations are performed at all arithmetic units, it can be expected that the computations of complex and large design problems are executed very fast and their programs become more simple. In this study a computer system, which is intended to be used in design computations and has hundreds of arithmetic units arranged into network fashion of two or three dimensional array, is called an array computer and a simulation method based on the iterative method using the array computer is named direct simulation. There have been constructed many computers similar to this array computer, one of which is the famous SOLOMON COMPUTER(1). This computer is a large system having 1K units of arithmetic processors and 16 KBit/unit memory, and seems to have been constructed mainly for the high speed processing of radar data and large matrices.

This first report as mentioned earlier, deals with the basic system of the array computer, validity of parallel computation algorithms using Jacobi method and their computational efficiencies by simulating

the procedure with a conventional computer.

2. Basic concept of direct simulation

Direct simulation with an array computer consists in performing parallel computations at all arithmetic units, attaching an arithmetic unit to each nodal point. In more details, linear equations, the number of which is the same as that of unknown physical quantities, are satisfied at each nodal point, and each arithmetic unit computes the values of these unknown quantities in sequence and iterates this process. For example, at i -th nodal point

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i \quad (1)$$

is assumed to be satisfied. In reality almost all of coefficients a (italicised symbols means a set of coefficients) are zero, and only a few have non-zero values. An arithmetic unit attached to this nodal point iterates the computations of the value of

$$x_i = (b_i - a_{i1}x_1 - a_{i2}x_2 - \dots - a_{in}x_n) / a_{ii} \quad (2)$$

receiving data x from neighboring arithmetic units and multiplying a by these data. If the number of unknown physical quantities is two, i.e. for example x and y , an arithmetic unit computes x at first and y in the next. All arithmetic units basically execute same operations and same computations simultaneously according to common timing diagrams. For example, if an arithmetic unit receives a datum from left arithmetic unit and multiplies it by coefficient a , all arithmetic units receive data from left arithmetic units simultaneously and multiply them by coefficients at the same time.

In direct simulation arithmetic units are attached to all nodal points, and accordingly the array computer must have many arithmetic units arranged into a two or three dimensional array. Further each arithmetic unit must transfer data directly to its neighboring arithmetic units, so it must be connected to its neighbors via shiftregisters. Each arithmetic unit must have its own memory to store coefficients a and b and other values. Basic system of this array computer is shown in Fig. 1. This system consists of three kinds of arithmetic units, i.e. AU, $\bar{A}U$ and $\tilde{A}U$. AU is a small sized unit, attached to each nodal point performing basic arithmetic operations in parallel manner. $\bar{A}U$ computes the sum of data in column direction. $\tilde{A}U$ is of medium sized, controls the total system and computes the sum of $\bar{A}U$'s data in row direction. RAM is a random access memory which stores data such as values of physical quantities and of coefficients. RAM of $\tilde{A}U$ is given enough capacity to store computation programs. SR is a shiftregister, which enables data transmission between neighboring AUs. CU is a control unit, which controls the total system generating timing

pulses. Three kinds of arithmetic units do not execute computations at the same time. ROM is read only memory and I/O is an input-output apparatus. The array computer which is intended to be constructed in this study is a system having 1K units of AUs and 100 words/unit RAM. As described earlier SOLOMON COMPUTER is similar to this array computer. And advanced system of SOLOMON COMPUTER is ILLIAC IV(2). This computer is a large system having 64 units of high speed arithmetic processors and 64 Bit-2K words/unit memory, and seems to have been constructed mainly for the high speed processing of radar data and large matrices, and for the analysis of weather forecasting models.

3. Parallel computation method using Jacobi method

Suppose that n nodal points are derived by the division of continuous solid into elements and following linear equations are obtained at each nodal point by the application of finite difference or finite element method;

$$\begin{aligned} \text{1st nodal point} & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ \text{2nd nodal point} & a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ & \dots \dots \dots \\ \text{n-th nodal point} & a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{aligned} \quad (3)$$

In reality almost all of a are zero. In parallel computation method using Jacobi method, AUs attached to nodal points iterate the computations of the following equations

$$\begin{aligned} \text{1st AU } x_1^{(v+1)} &= (b_1 - a_{12}x_2^{(v)} - \dots - a_{1n}x_n^{(v)}) / a_{11} \\ \text{2nd AU } x_2^{(v+1)} &= (b_2 - a_{21}x_1^{(v)} - \dots - a_{2n}x_n^{(v)}) / a_{22} \end{aligned} \quad (4)$$

Fig. 1 Basic system of array computer

4. Parallel computation method using modified Jacobi method

Results of the preceding chapter show that parallel computation method using Jacobi method is feasible if a relaxation factor is chosen appropriate. But as everybody knows Jacobi method is the most basic and its convergence speed is slow compared with that of SOR (successive over relaxation) method. This means the slow convergence speed of the parallel computation method using Jacobi method. Hence a parallel computation method, whose convergence speed is faster than that of Jacobi method has to be established. In this report a parallel computation method, which partly uses SOR method, is proposed. This method is named a parallel computation method using modified Jacobi method. In this

chapter the validity of this method is numerically experimented.

Suppose that eq. (3) is derived at each nodal point by the application of the appropriate digitizing method. In the parallel computation method using Jacobi method, the values in the right side are all v -th approximations as in eq. (4). But in the method using modified Jacobi method, a few values are $(v+1)$ -th approximations. For example, AUs compute the following equations

$$\left. \begin{aligned} \text{1st AU } x_1^{(v+1)} &= (b_1 - a_{12}x_2^{(v)} - \dots - a_{1n}x_n^{(v)})/a_{11} \\ \text{2nd AU } x_2^{(v+1)} &= (b_2 - a_{21}x_1^{(v+1)} - \dots - a_{2n}x_n^{(v)})/a_{22} \end{aligned} \right\} \dots (7)$$

Fig. 4 Relation between number of iterations v before convergence and relaxation factor ω (problem (D))

Fig. 5 Relation between relaxation factor ω and convergence-divergence condition (problems (B) and (C))

clear that the mode of convergence of iteration is affected a little by ϵ .

From above described numerical experiments, it is clear that the parallel computation methods using Jacobi method and modified Jacobi method are feasible if the relaxation factor is chosen appropriate.

In the next place, computational efficiency of array computer is discussed. Computation speed of array computer is ideally N (number of used AUs) times faster than that of conventional computer, but in reality the ratio is less than N due to the difference of computation methods, which can be applied to both computers. This ratio is given by eq. (8),

$$\frac{\text{comp. speed of conve. computer}}{\text{comp. speed of array computer}} = \frac{\text{number of iterat. by conve. computer } v_N}{\text{number of iterat. by array computer } v_A} \times \text{number of used AUs} \quad (8)$$

The numbers of iterations v_N and v_A are the values at optimum relaxation factors. Table 1 shows the calculated results of this ratio for the cases that the numbers of nodal points are 144 and 768 and for $\epsilon < 10^{-4}$. From this table it is seen that the ratio defined by eq. (8) is $1/5 \sim 1/2$ of the number of used AUs. The reduction of this ratio is due to the fact that the convergence speed of parallel method is slower than that of SOR method as shown by v_A/v_N in Table 1. This result leads to a conclusion that the usefulness of array computer lies not in dealing with small sized problems, in which N and v_A/v_N are comparable, but in dealing with large sized problems, in which N is large compared with v_A/v_N . As the total number of arithmetic operations in parallel computation method using modified Jacobi method is 30% increased compared with that of SOR method, the ratio v_N/v_A is in reality 30% decreased.

5. Choice of relaxation factors

Direct simulation consists in performing parallel computations at all AUs. The most difficult problem in computation is the choice of appropriate relaxation factor which makes iteration converge very fast. Up to the present a choosing method for this factor has not yet been given except for a system of linear equations whose coefficients matrix has Young's property A(3).

In this chapter one direction in the choice of relaxation factors is suggested.

Execution of parallel computation method using Jacobi method or modified Jacobi method with a conventional computer is equivalent to Jacobi method (or simultaneous displacement method). So the optimum relaxation factor for parallel Jacobi or modified Jacobi method is the same as that for Jacobi method. Regarding eq. (3) as a system of linear equations and assuming $a_{11}=a_{22}=\dots=1$, eq. (4) is written as

$$\mathbf{x}^{(v+1)} = \mathbf{b} + \mathbf{E} \mathbf{x}^{(v)} \quad (9)$$

This does not lose generality. $\mathbf{E}$ is shown in Fig. 8. Let the matrix $\mathbf{A}$ be an all coefficients matrix. Substituting eq. (9) into eq. (5),

$$\mathbf{x}^{(v+1)} = \omega \mathbf{b} + \omega (\mathbf{I} - \mathbf{A}) \mathbf{x}^{(v)} \quad (10)$$

is obtained. Let the matrix of second term in the right side be $\mathbf{M}$, then the necessary and sufficient condition for the convergence of sequence $[\mathbf{x}]$ is given by (4)

$$0 < \rho(\mathbf{M}) < 1 \quad (11)$$

$\rho(\mathbf{M})$ is the spectral radius of the matrix $\mathbf{M}$. This is the same as absolute value of the maximum absolute eigenvalue. As an equation

$$\rho(\mathbf{M}) \leq \|\mathbf{M}\|_S \quad (12)$$

is satisfied between spectral norm $\|\mathbf{M}\|_S$ and $\rho(\mathbf{M})$, the sufficient condition for the validity of eq. (11) is

$$\|\mathbf{M}\|_S < 1 \quad (13)$$

From the properties of norm

$$\begin{aligned} \|\mathbf{I}\|_S - \omega \|\mathbf{A}\|_S &\leq \|\mathbf{M}\|_S \\ \omega \|\mathbf{A}\|_S - \|\mathbf{I}\|_S &\leq \|\mathbf{M}\|_S \end{aligned} \quad (14)$$

are satisfied. Substituting eq. (13) into eq. (14) leads to

$$0 \leq \omega \leq 2 / \|\mathbf{A}\|_S \quad (15)$$

Inverse condition can be shown easily using the fact that the eigenvalues of $\mathbf{M}$ are the same as that of $\mathbf{I} - \omega \mathbf{A}$. If a relaxation factor is so chosen as to satisfy eq. (15), Jacobi method, and accordingly the parallel Jacobi or modified Jacobi method converges necessarily, even if any initial vectors $\mathbf{x}^{(0)}$ and constants $\mathbf{b}$ are given. Let the transposed matrix of matrix $\mathbf{A}$ be $\mathbf{A}^T$, following are satisfied,

$$\|\mathbf{A}\|_S^2 = |\text{maximum absolute eigenvalue of } \mathbf{A}^T \mathbf{A}|$$

Table 1 Comparison of computational efficiencies

problem	computation method	number of used AUs N	number of iterations v_N	number of iterations v_A	$\frac{v_A}{v_N}$	$\frac{v_N^N}{v_A}$
(C)	Jacobi	144	9126	45560	4.99	28.8
	Modified Jacobi			21303	2.33	61.7
(D)	Jacobi	768	972	2959	3.04	252
	Modified Jacobi			1747	1.80	427

$$\leq \|A^T A\| \leq \|A^T\| \|A\| \dots \dots \dots (16)$$

In these equations, $\|A^T\|$ and $\|A\|$ are arbitrary norms, chosen from $\|A^T\|_1$, $\|A\|_1$, $\|A^T\|_\infty$, $\|A\|_\infty$, $\|A^T\|_s$, $\|A\|_s$ and Euclidian norms $\|A^T\|_E$, $\|A\|_E$. Hence the following choice

$$\|A^T\| = \max(\|A^T\|_1, \|A^T\|_\infty) \dots \dots \dots (17)$$

$$\|A\| = \max(\|A\|_1, \|A\|_\infty)$$

do not lose generality. As in the real matrix

$$\|A^T\|_1 = \|A\|_\infty, \quad \|A^T\|_\infty = \|A\|_1 \dots \dots \dots (18)$$

is satisfied, eq. (19) is derived.

$$\|A^T\| \|A\| = [\max(\|A\|_1, \|A\|_\infty)]^2 \dots \dots \dots (19)$$

Replacing the terms in the right side of eq. (19) by the sums of coefficients, and substituting the result into eq. (16),

$$\|A\|_s \leq \max(\max_i \sum_j |a_{ij}|, \max_j \sum_i |a_{ij}|) \dots \dots \dots (20)$$

is obtained. From eq. (15) and eq. (20)

$$0 \leq \omega \leq 2 / \max(\max_i \sum_j |a_{ij}|, \max_j \sum_i |a_{ij}|) \dots \dots \dots (21)$$

is obtained. This equation is very important in actual usage, because if a relaxation factor, satisfying eq. (21), is chosen, iteration converges necessarily. In this case the optimum relaxation factor, which makes iteration converge fastest, is given by

$$\omega_1 = 2 / \max(\max_i \sum_j |a_{ij}|, \max_j \sum_i |a_{ij}|) \dots \dots \dots (22)$$

Comparison of ω_1 with ω_{opt} derived by numerical experiments is given in Table 2. ω_1 is estimated smaller than ω_{opt} . To correct this estimation, the following ω_2 is adopted as the optimum relaxation factor

$$\omega_2 = 2 / \text{mean}(\sum_{i \text{ or } j} |a_{ij}|) \dots \dots \dots (23)$$

Eq. (23) is derived from eq. (22) and the following equation

$$\text{mean}(\sum_{i \text{ or } j} |a_{ij}|) \leq \max(\max_i \sum_j |a_{ij}|, \max_j \sum_i |a_{ij}|) \dots \dots \dots (24)$$

Calculated results of ω_2 are listed in Table 2, which shows that ω_2 is closer to ω_{opt} than ω_1 .

So far is the case of Jacobi, and accordingly parallel Jacobi method, in which the number of unknown physical quantities is one. In the case of the number of these quantities being more than two or in the case of parallel modified Jacobi method, the $(v+1)$ -th approximations $x^{(v+1)}$ appear in the right sides of equations. Elimination of these terms leads to a change of the matrix A. Let A be partitioned as shown in Fig. 9. -G are diagonal elements newly added. Eq. (8) is rewritten as

$$x^{(v+1)} = b + (E + G) x^{(v)} \dots \dots \dots (25)$$

and eqs. (9)~(24) are valid. But the sum of

Fig. 8 Partition of coefficients matrix

Fig. 9 Partition of coefficients matrix

Table 2 Results of numerical experiments

No.	computation method	problem	number of nodal points	ω_{opt}	ω_1	ω_2	ω_3	digitizing method	mesh spacing
1	Jacobi method	(A)	4	0.79	0.522	0.754		finite difference method	equal
2			12	0.73	0.522	0.751			
3		(B)	8x8	0.73	0.625	0.735			
4		(C)	4x5	0.63	0.575	0.626			
5			4x9	0.63	0.575	0.635			
6			6x12	0.62	0.575	0.630			
7			8x18	0.62	0.575	0.629			
8	Modified Jacobi method	(D)	4x12	1.04	0.656	0.801	0.845	finite element method	unequal
9			8x24	1.00	0.626	0.696	0.816		
10			16x48	1.01	0.626	0.673	0.803		
11			6x12	1.03	0.574	0.759	0.862		
12		(A)	4	1.60	0.500	0.762	0.762	finite difference method	equal
13			12	1.15	0.500	0.783	0.783		
14		(B)	8x8	1.14	0.625	0.807	0.832		
15		(C)	4x5	0.87	0.587	0.615	0.647		
16			4x9	0.82	0.587	0.626	0.659		
17			6x12	0.78	0.587	0.657	0.694		
18			8x18	0.77	0.587	0.665	0.702		
19		(D)	4x12	1.40	0.614	0.682	0.909	finite element method	unequal
20			8x24	1.35	0.570	0.649	0.870		
21			16x48	1.30	0.570	0.633	0.851		
22			6x12	1.20	0.484	0.664	0.897		

coefficients is replaced by $\Sigma a_{ij} + g_{ij}$, in which g_{ij} are the elements of G . Calculated results of ω_1 and ω_2 are listed in Table 2. ω_2 is closer to ω_{opt} than ω_1 , but approximation is not good. This is due to the fact that the sum of coefficients increases with an increase of the number of substitutions of $x^{(v+1)}$. To make the sum of coefficients decrease, the following equation

$$\frac{\text{mean}(\sum_{i \text{ or } j} |a_{ij}| - \sum_{i \text{ or } j} |h_{ij}| + |g_{ij}|)}{\sum_{i \text{ or } j} |a_{ij}| + \sum_{i \text{ or } j} |h_{ij}| + |g_{ij}|} \dots (26)$$

must be used. h_{ij} are the coefficients of eliminated terms, and a_{ij} are other coefficients. The term in the right side of eq. (24) and that of eq. (26) are equal. From eq. (24) and eq. (26)

$$\omega_3 = 2 / \text{mean}(\sum_{i \text{ or } j} |a_{ij}| - \sum_{i \text{ or } j} |h_{ij}| + |g_{ij}|) \dots (27)$$

is derived. ω_3 can be expected to be closer to ω_{opt} than ω_1 . Calculated results of ω_3 meets the expectation. Eq. (27) is applicable to actual computation, even if ω_3 is calculated smaller than ω_{opt} .

6. Conclusive remarks

To shorten the time consumed for the design computations in many engineering branches, the authors intend to use a new type computer system, i.e. an array computer, which has hundreds of arithmetic units

connected in a two- or three-dimensional network fashion. This report has dealt with the basic system of array computer, the parallel computation methods using Jacobi and modified Jacobi methods, and discussed the validity of the methods and computational efficiencies by simulating the procedures with a conventional computer. The results of the simulation show that the parallel computation methods are feasible if a relaxation factor is chosen appropriate and they can compute engineering problems efficiently. Further, a direction is suggested in the choice of appropriate relaxation factors, and then empirical formulae are proposed.

The authors would like to thank Dr. K. Umezawa, Associate Professor of the Research Laboratory of Precision Machinery and Electronics of the Tokyo Institute of Technology, for his invaluable suggestions in computer simulations.

References

- (1) Slotnik, D. L., 1962 Fall Joint Computer Conf., AFIPS Conf. Proc., (1962), 97.
- (2) Bouknight, W. J., et al., Proc. IEEE, 60-4 (1972), 369.
- (3) Varga, R. S., Matrix Iterative Analysis, (1962), 99, Prentice-Hall.
- (4) 13 page of (3).