

Error Evaluation in the Design of a Special-Purpose Processor That Calculates Nonbonded Forces in Molecular Dynamics Simulations

TAKASHI AMISAKI* and TAKAJI FUJIWARA

Department of Computer and Information Science, Faculty of Science, Shimane University, 1060 Nishikawatsu, Matsue 690, Japan

AKIHIRO KUSUMI

Department of Pure and Applied Sciences, The University of Tokyo, Komaba 3-8-1, Meguro-ku, Tokyo 153, Japan

HIROO MIYAGAWA and KUNIHIRO KITAMURA

Research Center, Taisho Pharmaceutical Co., Ltd., 1-403 Yoshino-cho, Ohmiya, Saitama 330, Japan

Received 26 July 1994; accepted 15 December 1994

ABSTRACT

Special-purpose parallel machines that are plugged into a workstation to accelerate molecular dynamics (MD) simulations are attracting a considerable amount of interest. These machines comprise scalable homogeneous multiprocessors for calculating nonbonded forces (Coulombic and van der Waals forces), which consume more than 99% of the central processing unit (CPU) time in standard MD simulations. Each processor element in the machine has a pipeline architecture to calculate the total nonbonded force exerted on a particle by all of the other particles using information regarding the coordinates, the electric charge, and the species of each particle broadcast by the host computer. The processor then sends the calculated force back to the host computer. This article addresses the precision of the calculated nonbonded forces in the design of a processor LSI with minimal complexity. The precision of the arithmetic inside the processor that is required to calculate forces for MD simulations using Verlet's procedure was critically evaluated. Forward and

*Author to whom all correspondence should be addressed at Department of Computer and Information Science, Faculty of Science, Shimane University, 1060 Nishikawatsu, Matsue 690, Japan.

backward error analysis, coupled with numerical MD experiments on one-dimensional systems, was performed, and the following results were obtained: (1) Each element of the position vector which the processor receives from the host computer should have a precision of at least 25 bits; and (2) the pairwise forces should be calculated using floating point numbers with at least 29 bits of mantissa in the processor. Calculation of a pairwise force, which involves second-order polynomial interpolation using a table-driven algorithm, requires a key which contains a duplicate of at least 11 most significant bits of mantissa of the squared pairwise distance. The final result was that (3) the total force that acts on a particle, which is obtained by summing the forces exerted by all of the other particles, should be calculated using an accumulator that has a mantissa of at least 48 bits. © 1995 by John Wiley & Sons, Inc.

Introduction

Molecular dynamics (MD) simulations have been used widely to understand the physical properties of condensed matter¹ at the atomic level ever since Alder and Wainwright's first trial² in 1959. Over the past decade, MD simulations have rapidly penetrated into the fields of biomolecular sciences³ involving enzyme kinetics,⁴ protein engineering,⁵ drug design,⁶ and refinements of structures based on X-ray and nuclear magnetic resonance (NMR) experiments.⁷ However, the use of MD simulations in studies of large and complex systems is severely limited by the computational complexity of MD calculations, $O(N^2)$ (N = the number of particles in the system), which is governed by the number of nonbonded interactions to be evaluated. Recently, massively parallel computers have been applied to MD simulations.⁸ Such machines employ many processors (N_p) and reduce the computational complexity to the level of $O(N^2/N_p)$. Another approach to this problem is to produce special-purpose machines, such as FASTRUN⁹ and GRAPE-2A,^{10,11} that have pipelines of digital signal processors to calculate nonbonded forces. The success of these machines encouraged design of an application-specific integrated circuit (ASIC), which will lead to production of scalable parallel machines. By taking advantage of recent cost reductions in the production of ASIC chips, which are now employed in everything from kitchenware to ballistic missiles, parallel machines that are more efficient than general-purpose parallel machines, in terms of both calculation time and cost, can be produced.

One useful concept is for the machine to be designed as an accelerator that is interfaced to a

host workstation. It receives the coordinates, the electric charge, and the species of each particle in the system from the host computer, calculates all of the nonbonded forces, and then sends the calculated values back to the host computer. In this manner, the remaining MD calculation, which is complex but only consumes a limited percentage of computation time, can be left to the host computer, and the ASIC design can be dramatically simplified (i.e., its sole task is to calculate the nonbonded forces acting on a particle from all of the other particles in the system).

In the design of ASIC, the circuit should be optimized for sufficient accuracy with minimal complexity to achieve high cost performance without compromising the accuracy of the simulation. In MD simulations of large and complex systems, approximation methods, most often a cutoff distance of the nonbonded forces, have been employed to reduce the computational complexity. However, it has been well known that cutting off the long-range nonbonded forces makes the simulation unreliable structurally and energetically.¹² Therefore, it is important to calculate long-range nonbonded interactions precisely without using the cutoff scheme. However, to our knowledge, there has not yet been a systematic study of the accuracy of nonbonded forces required in stable MD simulations.

In the present study we have carefully evaluated the arithmetic accuracy in the calculation of nonbonded forces. We employed forward and backward error analysis and performed empirical error evaluations. Forward error analysis is useful only in special cases, as will be discussed later. On the other hand, backward error analysis is generally useful for numerical error analysis. This method can validate the computational procedure when the evaluated backward errors of input data are within their own intrinsic errors, such as errors

in rounding. In our problem, coordinates of particles correspond to input data. However, since the required precision for coordinates is not known, we empirically evaluated the precision required for coordinates by performing many MD simulations. The required arithmetic accuracy inside the processor was estimated by assuming Verlet's method,¹³ which is an algorithm commonly used in MD simulations.

Architecture of the Processor Element

In this section we describe the architecture of the special-purpose machine whose design is under development in our laboratories. Knowledge of the organization of the machine, which is a scalable parallel computer, would help readers understand the subject matter of this article. However, the error evaluation given in this article is not dependent on a particular architecture.

The special-purpose machine is comprised of homogeneous processor elements (PEs) connected to a Versa Module Europe (VME) bus (Fig. 1), which calculate nonbonded forces. A PE consists of a custom processor LSI that is responsible for all the necessary arithmetic for calculation of forces and the local memories. The machine is plugged into a workstation as a hardware accelerator. To avoid the overhead of complicated communication and synchronization between PEs, as is the case with usual parallel computers, calculation of a force in a PE is driven by a host computer rather than by the other PEs. At the beginning of a new cycle of force calculation, the coordinates of all particles are transferred from the host to the local memory of every PE, and each PE starts calculating the force acting on a particle upon receipt of a trigger signal from the host computer. After calculating the total force acting on the particle, PE sends the calculated force back to the host computer. The electric charge and the species of each particle are also stored in the local memories as well as the coefficients of polynomial interpolation for calculation of forces (as a function of the squared pairwise distance).

Table I shows the steps in the calculation of the total force acting on the i th particle from all other particles, F_i . These stages are implemented as hard-wired functional blocks in a pipeline. First, the vector Δr_{ji} is calculated by subtracting the position vector of the j th particle r_j from that of the i th particle r_i . The obtained vector is used in

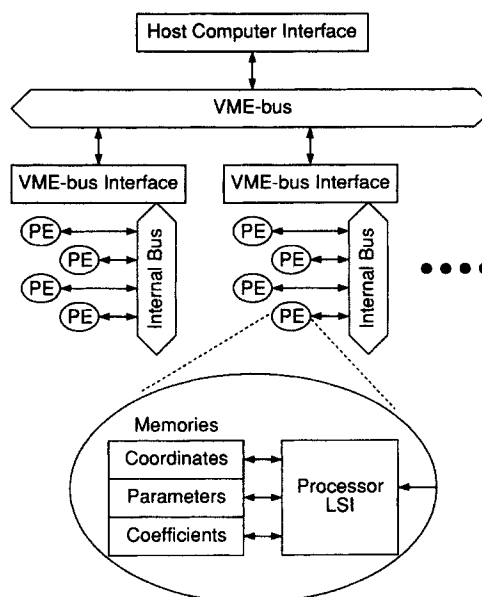


FIGURE 1. The organization of a special-purpose accelerator machine for MD simulations which is plugged into a workstation. Each processor element (PE) consists of a custom processor LSI, which is responsible for all of the necessary arithmetic to calculate the pairwise forces, and local memories, in which the coordinates of the particles, the force field parameters, and the coefficients for polynomial interpolation are stored.

the subsequent calculation of the squared distance between two particles, s . The pairwise force, f_{ji} , is calculated by

$$f_{ji} = g_0 g_{ji} \Delta r_{ji} \quad (1)$$

where g_0 is a parameter of energy. The value of g_{ji} is given by the function of the squared distance, $g(s)$. The function for Lennard-Jones interaction can be written as

$$g(s) = s^{-7} - 1/2s^{-4} \quad (2)$$

and that for Coulombic interaction as

$$g(s) = s^{-3/2} \quad (3)$$

These functions are evaluated by k th-order polynomial interpolation with a table-driven (a look-up table) algorithm using the coefficients of interpolation stored in the distributed memory. This evaluation method makes the processor simple and flexible, as compared with the one that directly evaluates $g(s)$ using a hard-wired circuit. Thus any pairwise interactions can be evaluated by setting the polynomial coefficients for that function in the local memory. The stepsize for polynomial interpolation is taken to be $2^{\lceil \log_2 s \rceil} / 2^m$, where

TABLE I.
Steps for Calculating the Force Acting on the i th Particle.^a

Stage	Calculation
Stage 0	Coordinates of all of the particles, parameters of force field, and coefficients of interpolation are broadcast by the host computer and stored in each distributed memory. The coordinates of the i th particle, \mathbf{r}_i , for which forces from all of the other particles are calculated, are loaded into the internal register of the processor LSI. A trigger signal starts the calculation process.
Stage 1	Coordinates of the j th particle, \mathbf{r}_j , are loaded into the internal register.
Stage 2	The vector from the j th particle to the i th particle is calculated as $\Delta\mathbf{r}_{ji} = \mathbf{r}_i - \mathbf{r}_j$.
Stage 3	The squared distance between the two particles is calculated as $s = \Delta\mathbf{r}_{ji}^2$.
Stage 4	A temporary value, g_{ji} , to be used for subsequent calculation of a pairwise force, is calculated by k th-order polynomial interpolation, g_{ji} is given by a function of s , $g(s)$, where $g(s) = s^{-3/2}$ for Coulombic interaction, or $g(s) = s^{-7} - 1/2s^{-4}$ for Lennard-Jones interaction. The stepsize in interpolation is taken to be $2^{\lfloor \log_2 s \rfloor} / 2^m$ to avoid decreases in accuracy for small values of s .
Stage 5	The pairwise force \mathbf{f}_{ji} is calculated by multiplying the force field parameter g_0 , which is stored in local memory, and the vector $\Delta\mathbf{r}_{ji}$ obtained in stage 2: $\mathbf{f}_{ji} = g_0 g_{ji} \Delta\mathbf{r}_{ji}$.
Stage 6	The pairwise force is added on the accumulator to obtain total force \mathbf{F}_i .

^aStages 1 to 6 are carried out over all particles j ($j \neq i$) to obtain the total force acting on the i th particle as $\mathbf{F}_i = \sum_{j \neq i} \mathbf{f}_{ji}$.

$\lfloor x \rfloor$ is the floor of x , meaning the greatest integer less than or equal to x . This can be easily implemented as a table-driven algorithm with a key to look up, which can be directly constructed from the m most significant bits of mantissa of s and several bits of the exponent of s ($= \lfloor \log_2 s \rfloor$).

The aforementioned calculations are performed for all of the particles, except the i th particle itself, using floating point arithmetic with p bits of mantissa. Finally, all of the pairwise forces are added on the accumulator with q bits of mantissa to obtain the total force acting on the i th particle. The purpose of this article is to determine the minimal values of k , m , p , and q , which can provide an accuracy sufficient for practical MD simulations.

Numerical Evaluation of the Precision Required for Position and Force

In MD simulations, the Newtonian equations of motion for N particles are integrated in a numerical manner. If the position of the i th particle at time t , $\mathbf{r}_i(t)$ is known, then the position after a short time interval, Δt , is given by Taylor's theorem with explicit remainder terms:

$$\mathbf{r}_i(t + \Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \Delta t) + \Delta t^2 \frac{\mathbf{F}_i(t)}{m_i} + \mathbf{R}_i \quad (4)$$

where

$$\mathbf{R}_i = \frac{\Delta t^4}{4!} \left\{ \frac{d^4 \mathbf{r}_i(t + \theta_1 \Delta t)}{dt^4} + \frac{d^4 \mathbf{r}_i(t - \theta_2 \Delta t)}{dt^4} \right\}, \quad 0 \leq \theta_1, \theta_2 \leq 1 \quad (5)$$

and $\mathbf{F}_i(t)$ is the force acting on the particle at time t , and m_i is the mass of the particle. Parameters θ_1 and θ_2 have certain but unknown values. By Verlet's method, the equation of motion is integrated by assuming that the remainder term \mathbf{R}_i in eq. (4) is zero. The extent of the computational error caused by the processor should be less than that caused by truncating the remainder term. Since it is impossible to evaluate \mathbf{R}_i analytically, we empirically evaluated the precision required for positions and forces by carrying out numerical experiments.

Microcanonical MD simulations were performed for 32 particles in a one-dimensional space using a periodic boundary condition considering Lennard-Jones interaction in either the presence or absence of additional Coulombic interaction. The parameters for Lennard-Jones potential and mass were those of the argon atom. Positive unit charges were given to 16 particles and negative unit charges to the remaining 16 particles for Coulombic interactions. The ratio of Coulombic and Lennard-Jones energies was taken to be 14.4. The reduced number

density was 0.8. The unit of time, τ , corresponds to 0.109 ps or 1.09 ps with or without Coulombic interactions, respectively. Under these conditions, Δt is generally taken to be on the order of $1 \times 10^{-2}\tau$. According to eqs. (4) and (5), as Δt becomes smaller, more accuracy is required for the position and force. To derive reasonable criteria for the precision, simulations were carried out with three different Δt 's (i.e., 1×10^{-2} , 10^{-3} , and 10^{-4}), which cover almost the entire range of the timestep used in standard MD simulations. Each simulation was begun under identical conditions. The set of equations of motion was integrated by Verlet's method, using ordinary double-precision numbers and exact calculation of $g(s)$ instead of interpolated values from the table-driven algorithm. The temperature of the system was equilibrated at 646 K and 98.1 K with and without Coulombic interaction, respectively. To determine the precision required for position $r_i(t)$ and force $F_i(t)$, the stability of the simulation was studied as a function of the precision of $r_i(t)$ or $F_i(t)$. The stability was evaluated in terms of the magnitude of the relative root mean square (rms) fluctuation in the total energy E , namely,

$$\frac{(\langle E^2 \rangle - \langle E \rangle^2)^{1/2}}{|\langle E \rangle|} \quad (6)$$

The average of $20,000\Delta t$ steps was taken after $20,000\Delta t$ steps of equilibration. The precision of $r_i(t)$ and $F_i(t)$ was adjusted by rounding their mantissas toward the nearest even number.¹⁴ Underflowed results were flushed to zero rather than treating subnormal numbers with gradual underflow.¹⁴ To mimic the actual behavior of the machine, the precision of $r_i(t)$ was adjusted before stage 2 in Table I.

The effect of the precision of the position on the stability of the simulation is shown in Figures 2a and 2b (with and without Coulombic interaction, respectively). The floor portion of each curve indicates the intrinsic stability with the particular timestep, which is dominated by the accuracy of the numerical integration using Verlet's method. For example, in the case of $\Delta t = 1 \times 10^{-4}$ in Figure 2a, simulations with a precision of 29 bits or greater give the same relative rms fluctuation (3.8×10^{-10}) as the intrinsic stability. In other words, the simulation with $\Delta t = 1 \times 10^{-4}$ reaches its intrinsic stability where the precision of the position is 29 bits or greater. In the case of $\Delta t = 1 \times 10^{-2}$, rms fluctuation changes slightly with changes in

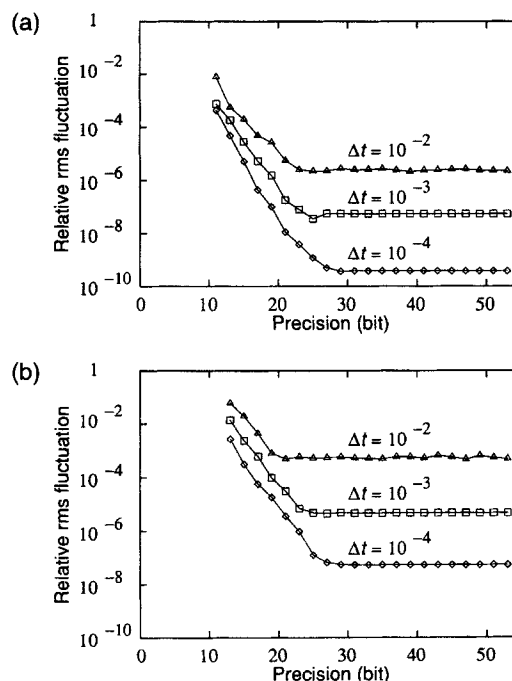


FIGURE 2. Effect of the precision of the position on the stability of one-dimensional MD simulations. The precision of the position shown on the abscissa is expressed in terms of the number of bits of the mantissa. In ordinary double-precision arithmetic, this number is 53. The stability of the simulation is shown by the relative rms fluctuation of the total energy as defined in the text. Results of simulations with timesteps of 1×10^{-2} , 10^{-3} , and 10^{-4} are shown. (a) Coulombic and Lennard-Jones interactions. (b) Lennard-Jones interaction only.

the precision of the position, even in the floor portion, which indicates that simulations with this timestep are not as intrinsically stable as those with time steps of 1×10^{-4} or 1×10^{-3} . Since many MD simulations have been performed with a timestep on the order of 1×10^{-2} , we ignore such slight fluctuations and consider the precision at which the rms fluctuation reaches the floor portion in each curve as the minimum required precision (i.e., 23, 25, and 29 bits with $\Delta t = 1 \times 10^{-2}$, 10^{-3} , and 10^{-4} , respectively). Since most MD simulations are performed with a timestep of $\Delta t = 1 \times 10^{-3}$ or greater ($\Delta t = 1 \times 10^{-4}$ is too small for practical MD simulations), we choose a precision of 25 bits for the particle position.

The effect of the precision of the force on the stability of the simulation is shown in Figures 3a and 3b (with and without Coulombic interaction, respectively). For a timestep of 1×10^{-2} , the results are similar to those in Figure 2, and simulations with this timestep are slightly unstable.

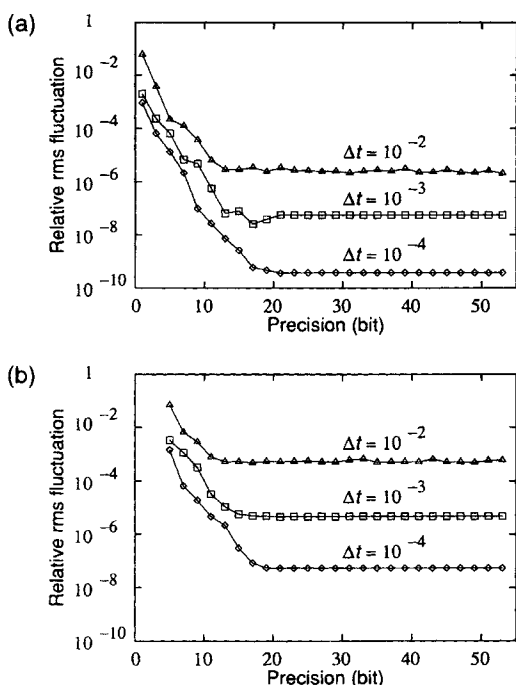


FIGURE 3. Effect of the precision of the force on the stability of one-dimensional MD simulations. The precision of the force shown on the abscissa is expressed in terms of the number of bits of the mantissa. The stability of the simulation is shown by the relative rms fluctuation of the total energy as defined in the text. Results of the simulations with timesteps 1×10^{-2} , 10^{-3} , and 10^{-4} are shown. (a) Coulombic and Lennard-Jones interactions. (b) Lennard-Jones interaction only.

Therefore, to perform stable simulations with $\Delta t = 1 \times 10^{-3}$, a minimum of 17–21 bits are required in the precision of the force.

In the simulation to generate Figure 3a, the sum of all pairwise Lennard-Jones forces and that of all pairwise Coulombic forces were rounded separately. The two sums were then added to yield $F_i(t)$. The other way may be to add the two kinds of forces first and then to round the value of $F_i(t)$. We present here the former results because actual calculation with such a machine is expected to be performed in the former scheme (Table I). We performed simulations using the later rounding scheme and found that the results were almost the same. We also found that the order of addition and rounding for the two terms in the Lennard-Jones force function [s^{-7} and s^{-4} of eq. (2)] does not affect the results presented in Figures 3a and 3b.

Polynomial Interpolation

Inside each processor element, polynomial interpolation is used with varied stepsizes of $\delta = 2^{\lfloor \log_2 s \rfloor} / 2^m$ to evaluate $g(s)$ in eqs. (2) and (3). To determine the order of the interpolation, k , we evaluated the accuracy of such an approximation as a function of m . Interpolated values were calculated using the coefficients obtained from the Vandermonde equation.¹⁵ “True” values were directly calculated by evaluating eqs. (2) and (3) using ordinary double-precision arithmetic. For 33 intervals of s , $[2^i, 2^i + \delta)$ at $i = -8, -7, \dots, \text{and } 24$, seventeen relative errors were calculated in each interval and the errors within each interval were averaged. We determined the accuracy of interpolation in terms of bits by taking a negative logarithm of the largest averaged error among 33 intervals. The accuracies were evaluated for various values of k and m .

Figures 4a and 4b show the accuracy of the polynomial interpolation for functions $g(s)$ in eqs. (3) and (2), respectively. The order k is critical in the circuit design (i.e., it is directly related to the memory size, computational speed, and accuracy). In the k th-order polynomial interpolation, the number of coefficients which have to be stored in the distributed memory is $2^m(k + 1)$. The required number of coefficients rapidly decreases as k increases, as expected from Figure 4. However, $k = 0$ is best for computational speed since the evaluation of $g(s)$ requires k multiplications, which limits the total computational speed of the processor.

The memory size must be kept as small as practical, and the number of calculation steps required for interpolation should be minimal. Since a force should have at least 17–21 significant bits and the value of $g(s)$ should be more precise than the force, it is appropriate to take k as either 2 or 3. For the speed of the processor, we have chosen $k = 2$ [second-order polynomial interpolation to evaluate $g(s)$] for subsequent discussion in this article.

Error Analysis

When rounding a real number into a floating point number with a mantissa of p bits without overflow or underflow, the rounded number has a relative error of $|\epsilon| \leq 2^{-p}$.^{16,17} The results of float-

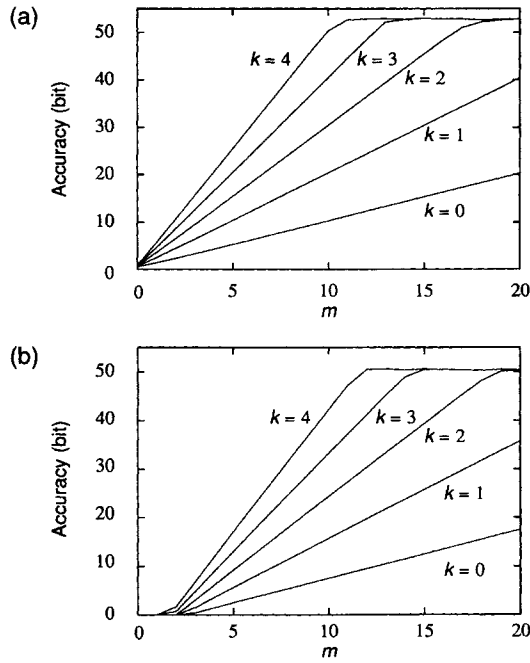


FIGURE 4. Accuracy of the function evaluation using polynomial interpolation shown as a function of m and k , where k is the order of the polynomial interpolation and m is related to the stepsize of the interpolation δ by $\delta = 2^{\lfloor \log_2 s \rfloor} / 2^m$. The accuracy on the ordinate was calculated as follows: An error was defined by the difference between the interpolated value and that calculated directly using ordinary double-precision numbers. Relative error was calculated by dividing this error by the later value. Seventeen relative errors were calculated and averaged for each interval of s , $[2^i, 2^{i+d}]$ at $i = -8, -7, \dots, 24$. The accuracy of interpolation was defined as the largest of the averaged errors among the 33 intervals. (a) Coulombic force function. (b) Lennard-Jones force function.

ing point arithmetic operations also contain a relative error of $|\epsilon_a| \leq 2^{-p}$.¹⁷ Since floating point arithmetic has such inherent inaccuracy, we use “round” symbols \oplus , \ominus , and \otimes to express floating point addition, subtraction and multiplication, respectively, to distinguish them from the true operations.

Using floating point arithmetic, the squared distance s between $\mathbf{r}_i^t = (x_i, y_i, z_i)$ and $\mathbf{r}_j^t = (x_j, y_j, z_j)$ (t indicates “transposed”) can be calculated by

$$s = (x_i \ominus^{(1)} x_j) \otimes^{(4)} (x_i \ominus^{(1)} x_j) \oplus^{(7)} (y_i \ominus^{(2)} y_j) \otimes^{(5)} (y_i \ominus^{(2)} y_j) \oplus^{(8)} (z_i \ominus^{(3)} z_j) \otimes^{(6)} (z_i \ominus^{(3)} z_j) \quad (7)$$

where superscripts over operators indicate the order of the operations. The resulting approximate value is given by the exact arithmetic of infinite precision

$$s = (1 + \eta_x) x_i^2 - 2(1 + \eta_x) x_i x_j + (1 + \eta_x) x_j^2 + (1 + \eta_y) y_i^2 - 2(1 + \eta_y) y_i y_j + (1 + \eta_y) y_j^2 + (1 + \eta_z) z_i^2 - 2(1 + \eta_z) z_i z_j + (1 + \eta_z) z_j^2 \quad (8)$$

where

$$1 + \eta_x = (1 + \epsilon_1)^2 (1 + \epsilon_4) (1 + \epsilon_7) (1 + \epsilon_8) \quad (9)$$

$$1 + \eta_y = (1 + \epsilon_2)^2 (1 + \epsilon_5) (1 + \epsilon_7) (1 + \epsilon_8) \quad (10)$$

$$1 + \eta_z = (1 + \epsilon_3)^2 (1 + \epsilon_6) (1 + \epsilon_8) \quad (11)$$

and ϵ_l ($l = 1, 2, \dots, 8$) are the relative errors at the l th floating point operation. For example, consider the second term on the right-hand side of eq. (8), $2(1 + \eta_x) x_i x_j$. This term can be considered as an exact product of x_i which involves a relative error of η_x and an exact value of $2x_j$ [i.e., the factor $(1 + \eta_x)$ expresses the deviation of x_i from the actual value and η_x is a backward error with respect to x_i]. Taking all terms in eq. (8) with eqs. (9)–(11) into consideration, the maximum backward error η_r with respect to the position vectors is given by

$$(1 - 2^{-p})^5 \leq 1 + \eta_r \leq (1 + 2^{-p})^5 \quad (12)$$

Next, we evaluate the error of polynomial (second-order) interpolation. Both the error from floating point approximation and the error caused by ignoring the higher-order terms in polynomial expansion are considered by the forward error analysis. On this analysis, we make several approximations for simplicity. As will be discussed later, however, the finally obtained upper bound of the error magnitude is strictly correct under reasonable assumptions without these approximations. For evaluation of the error in floating point operations, the squared distance s is decomposed into s_0 and Δs ($0 \leq \Delta s < 2^{\lfloor \log_2 s \rfloor} / 2^m$). The approximate value g_{ji} obtained by the second-order polynomial interpolation for the function $g(s)$ can be calculated by floating point arithmetics.

$$g_{ji} = a_0 \oplus^{(12)} (a_1 \oplus^{(10)} a_2 \otimes^{(9)} \Delta s) \otimes^{(11)} \Delta s \quad (13)$$

where a_0 , a_1 , and a_2 are coefficients of the interpolation. This value g_{ji} is given by the exact arithmetic

$$g_{ji} = (1 + \eta_{a0})a_0 + (1 + \eta_{a1})a_1\Delta s + (1 + \eta_{a2})a_2\Delta s^2 \quad (14)$$

where

$$1 + \eta_{a0} = 1 + \epsilon_{12} \quad (15)$$

$$1 + \eta_{a1} = (1 + \epsilon_{10})(1 + \epsilon_{11})(1 + \epsilon_{12}) \quad (16)$$

$$1 + \eta_{a2} = (1 + \epsilon_9)(1 + \epsilon_{10})(1 + \epsilon_{11})(1 + \epsilon_{12}) \quad (17)$$

and ϵ_9 , ϵ_{10} , ϵ_{11} , and ϵ_{12} are the relative errors of each floating point operation. Since it holds or almost nearly holds that $|\eta_{a0}| < 2^{-p}$, $|\eta_{a1}| < 3 \cdot 2^{-p}$, and $|\eta_{a2}| < 4 \cdot 2^{-p}$ according to eqs. (15), (16), and (17), the maximal absolute error of g_{ji} caused by the floating point operations is

$$2^{-p}|a_0| + 3 \cdot 2^{-p}|a_1|\Delta s + 4 \cdot 2^{-p}|a_2|\Delta s^2 \quad (18)$$

On the other hand, the absolute error of g_{ji} caused by ignoring the third- and higher-order terms in the Taylor's expansion of $g(s)$ is approximately bounded by

$$\frac{1}{6}|g'''(s)|\Delta s^3 \quad (19)$$

The relative error of g_{ji} , ϵ_g , is obtained by adding the value of eq. (19) to that of eq. (18) and then dividing the sum by $|g(s)|$. By substituting a_0 , a_1 , and a_2 in eq. (18) with $g(s)$, $g'(s)$, and $g''(s)$, respectively, and by using the relation $\Delta s < 2^{-m}s$, which is ascribed to the varied stepsize, the relative error can be expressed as

$$|\epsilon_g| < 2^{-p} + 3c_12^{-p-m} + 4c_22^{-p-2m} + c_32^{-3m} \quad (20)$$

where

$$c_i = s^i|g^{(i)}(s)/g(s)|, \quad i = 1, 2, 3 \quad (21)$$

As previously mentioned, the precision required for the force was not affected by the order of addition and rounding for three terms; two terms in the Lennard-Jones force function and a Coulombic force term. If we separately evaluate respective $|\epsilon_g|$ for the three different terms of force function (s^{-7} , s^{-4} , and $s^{-3/2}$), then c_i 's are constant and independent of s . Equation (20) shows that the

upper bound for $|\epsilon_g|$ is not less than 2^{-p} for any m and that it is not less than c_32^{-3m} for any p . Therefore, we balance the memory size (m) and the complexity of the arithmetic circuit (p) by

$$3m = p + \log_2 20 \quad (22)$$

where the value of 20 is taken from the median of c_3 over three different terms of the force functions.

The second and the third terms of the right-hand side of eq. (20) are negligible when the magnitude of c_1 and c_2 is taken into account. In the case of the first term of Lennard-Jones force function (s^{-7}) (for example, in which maximum error for g_{ji} is expected), the values of c_1 , c_2 , and c_3 are 7, 28, and 84, respectively. Since the precision of the position should be equal to or greater than 25 bits ($p \geq 25$) according to the results shown in Figure 2, the second term, $3c_12^{-p-m} \approx 3 \times 7 \times 2^{-10} \times 2^{-p}$, and the third term, $4c_22^{-p-2m} \approx 4 \times 28 \times 2^{-21} \times 2^{-p}$, are considerably smaller than the other terms. Consequently, eq. (20) can be reduced to

$$|\epsilon_g| < 6 \cdot 2^{-p} \quad (23)$$

Although we have made several approximations to derive eq. (23), the relation strictly holds without any approximation if we assume $p \geq 25$ and separately discuss the three different terms of force functions. Since such forward error is comparable to that in six operations of floating point arithmetic, a backward error with respect to s , η_s , through the operations in the second-order polynomial interpolation is given by

$$(1 - 2^{-p})^6 \leq 1 + \eta_s \leq (1 + 2^{-p})^6 \quad (24)$$

For operations to produce a pairwise force \mathbf{f}_{ji} , we consider the first element f_{xji} of \mathbf{f}_{ji} as an example, since each element of the vector \mathbf{f}_{ji} can be calculated independently. The value of f_{xji} can be obtained by

$$f_{xji} = (x_i \ominus^{(1)}x_j) \otimes^{(13)}g_0 \otimes^{(14)}g_{ji} = (x_i - x_j)g_0g_{ji}(1 + \epsilon_1)(1 + \epsilon_{13})(1 + \epsilon_{14}) \quad (25)$$

The maximum deviation (backward error) of g_{ji} , η_g , is attained when all of the relative errors are involved in g_{ji} ; thus

$$(1 - 2^{-p})^3 \leq 1 + \eta_g \leq (1 + 2^{-p})^3 \quad (26)$$

The backward error analysis for operations to obtain force vector \mathbf{F}_i can be found in the literature.¹⁷ The first element of \mathbf{F}_i , F_{xi} , is given by exact

arithmetic with infinite precision as

$$F_{xi} = \sum_{j \neq i}^N f_{xji}(1 + \eta_j) \quad (27)$$

where

$$\begin{aligned} \eta_1 &= \eta_2 \\ 1 + \eta_j &= (1 + \epsilon_j)(1 + \epsilon_{j+1}) \cdots (1 + \epsilon_{i-1}) \\ &\quad \times (1 + \epsilon_{i+1}) \cdots (1 + \epsilon_N), \quad j = 2, \dots, N \end{aligned} \quad (28)$$

Therefore, the backward error at this stage η_f satisfies

$$(1 - 2^{-q})^N \leq 1 + \eta_f \leq (1 + 2^{-q})^N \quad (29)$$

where q is the size of the mantissa of the accumulator at stage 6 (Table I).

We can now apply the backward error analysis to all of the stages in Table I. According to eqs. (12), (24), (26), and (29), the total backward error η , with respect to the elements of the position vectors, can be represented by

$$(1 - 2^{-q})^N \times (1 - 2^{-p})^{14} \leq 1 + \eta \leq (1 + 2^{-q})^N \times (1 + 2^{-p})^{14} \quad (30)$$

Since each element of the position vector requires a precision of at least 25 bits, as mentioned previously, the magnitude of the upper bound of the relative backward error η cannot exceed 2^{-25} from the standpoint of backward analysis. The pairs of p and q which satisfy this requirement are as follows: for $N = 10^4$, $\{p \geq 29, q \geq 42\}$, $\{p \geq 30, q \geq 40\}$, or $\{p \geq 31, q \geq 39\}$; for $N = 10^5$, $\{p \geq 29,$

$q \geq 45\}$, $\{p \geq 30, q \geq 43\}$, or $\{p \geq 31, q \geq 42\}$; for $N = 10^6$, $\{p \geq 29, q \geq 48\}$, $\{p \geq 30, q \geq 46\}$, or $\{p \geq 34, q \geq 45\}$. For an MD simulation which considers all particle-particle interactions, N is usually less than 10^6 . In the processor, since arithmetic circuits of p bits are used more frequently than those of q bits, we should choose the pair with the smallest value of p (i.e., $p = 29$ and $q = 48$). The minimum value of m is 11 according to eq. (22).

Numerical Tests

To confirm the validity of the requirements for $k, m, p,$ and q obtained earlier, the stabilities of one-dimensional MD simulations were examined by using a software simulator for the processor element. The conditions of the MD simulations and the computational procedures, such as rounding, were the same as those discussed earlier. To evaluate the function $g(s)$ in eqs. (2) and (3), second-order polynomial interpolation ($k = 2$) was used with a varied stepsize of $2^{\lfloor \log_2 s \rfloor} / 2^m$ ($m = 11$). The simulator operated in three modes. In mode I, the simulator performed all arithmetic using double-precision numbers (i.e., $p = q = 53$). In mode II, summation of pairwise forces (stage 6 in Table I) was performed using double precision while other arithmetic operations were carried out in single precision (i.e., $p = 24, q = 53$). Mode III is the optimized case (i.e., $p = 29$ and $q = 48$). Although these values for p and q were derived for the three-dimensional case, they also apply to the one-dimensional case.

The results of the simulations are shown in

TABLE II. Stability of One-Dimensional MD Simulations Examined with a Software Simulator.^a

Δt	Mode ^b		
	I	II	III
Lennard-Jones interaction:			
10^{-2}	5.58×10^{-4}	4.68×10^{-4}	5.36×10^{-4}
10^{-3}	4.70×10^{-6}	5.51×10^{-6}	4.58×10^{-6}
10^{-4}	5.40×10^{-8}	2.28×10^{-7}	5.49×10^{-8}
Lennard-Jones and Coulombic interactions:			
10^{-2}	2.19×10^{-6}	1.92×10^{-6}	2.57×10^{-6}
10^{-3}	5.52×10^{-8}	3.98×10^{-8}	5.41×10^{-8}
10^{-4}	3.78×10^{-10}	1.61×10^{-9}	3.65×10^{-10}

^aStability is defined by eq. (6) as the normalized rms fluctuation of the total energy.

^bMode I, ordinary double-precision arithmetic (i.e., $p = q = 53$); mode II, summation of pairwise forces is in double precision and other arithmetic is in single precision (i.e., $p = 24, q = 53$); mode III, arithmetic in optimized precision (i.e., $p = 29, q = 48$).

Table II. As mentioned previously, an MD simulation with a timestep of 1×10^{-2} in unit of τ is not sufficiently stable (Figs. 2 and 3). In cases with a timestep less than 1×10^{-2} , the results in mode I agree with those in mode III, while those in mode II are less stable than those in mode III, especially with a timestep of 1×10^{-4} . These results indicate that mode III, in which floating point arithmetic with a mantissa of 29 bits is used for calculation of pairwise forces, gives stable MD simulations, while pairwise forces calculated using single-precision arithmetic are unsatisfactory.

The influence of q on the stability of MD simulations cannot be studied with the present test system because the number of particles is only 32. Since the method we used to derive the minimum value of q is a simple backward error analysis, we believe that our estimate is correct. We will examine the effect of q after actual implementation of the processor.

The arguments we have made are based on the precision required for position and force, which was obtained from the results of periodic one-dimensional simulations. Therefore, rather tight criteria were employed for the precision of position and force. Another justification for the use of one-dimensional simulations is based on the following consideration. The requirement we obtained for the precision of the force is closely related to the magnitude of Verlet's remainder term R_i , in eq. (5). Although exact values of R_i 's are not known, it is assumed that the average absolute value of R_i is nearly equal to that calculated by setting θ_1 and θ_2 zero in eq. (5). In our simulation with Coulombic interaction, which adopted a timestep of 1×10^{-3} , the average absolute value of R_i over time and all of the particles is 1.0×10^{-13} . In the same simulation, the average absolute value of the third term on the right-hand side of eq. (4), $\Delta t^2 F_i(t)/m_i$, is 8.9×10^{-8} . By making the likely assumption that errors of the term ($\Delta t^2 F_i(t)/m_i$) do not exceed those of Verlet's method (R_i) on average, it follows that at least 20 bits of the force $F_i(t)$ should be significant. This value, 20, is consistent with the result presented in Figure 3. We confirmed that the ratio of $\Delta t^2 F_i(t)/m_i$ and R_i is almost the same for an MD simulation driven by soft-core potential in three-dimensional space (data not shown).

In this article, the basic architecture of a special-purpose machine for MD simulations was presented. To optimize the memory size, computational speed, and accuracy, requirements for k , m , p , and q have been established. In conclusion, each

element of the position vector sent from the host computer to the processor should have a precision of at least 25 bits. Inside the processor, each pairwise force should be calculated using floating point arithmetic with a mantissa of at least 29 bits ($p = 29$). In the calculation of a pairwise force using a table-driven algorithm, second-order polynomial interpolation ($k = 2$) with a key which contains a duplicate of at least 11 most significant bits ($m = 11$) of mantissa of the squared distance should be used. The total force acting on a particle, which is the sum of all of the pairwise forces acting on the particle, should be calculated using an accumulator which has a mantissa of at least 48 bits ($q = 48$). With these specifications, the machine can produce forces sufficiently precise for MD simulations of a system containing up to a million particles.

Acknowledgment

This study was supported in part by Special Coordination Funds from the Science and Technology Agency of the Japanese government.

References

1. F. Yonezawa, Ed, *Molecular Dynamics Simulations, Solid-State Sciences*, Vol. 3, Springer-Verlag, Berlin and Heidelberg, 1992.
2. B. J. Alder and T. E. Wainwright, *J. Chem. Phys.*, **31**, 459 (1959).
3. (a) J. A. McCammon and S. C. Harvey, *Dynamics of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, U. K., 1987; (b) J. M. Goodfellow, *Molecular Dynamics*, CRC Press, Inc., Boca-Raton, FL, 1990.
4. A. Warshel, *Computer Modeling of Chemical Reactions in Enzymes and Solution*, John Wiley & Sons, New York, 1991.
5. C. D. Fasman, Ed., *Prediction of Protein Structure and the Principals of Protein Conformations*, Plenum Press, New York, 1989.
6. (a) S. K. Burt, D. Mackey, and A. T. Hagler, In *Computer-Aided Drug Design*, T. J. Perun and C. L. Propst, Eds., *Theoretical Aspects of Drug Design*, Marcel-Dekker Inc., New York, 1989, 55-91; (b) S. Sumiya, T. Yoneda, K. Kitamura, M. Murata, C. Yokoo, M. Tamai, A. Yamamoto, M. Inoue, and T. Ishida, *Chem. Pharm. Bull.*, **40**, 299 (1992).
7. (a) A. T. Brünger, J. Kuriyan, and M. Karplus, *Science*, **235**, 458 (1987); (b) A. T. Brünger, G. M. Clore, A. M. Gronenborn, and M. Karplus, *Proc. Natl. Acad. Sci. USA*, **83**, 3801 (1986).
8. (a) W. D. Hillis and B. M. Boghosian, *Science*, **261**, 856 (1993); (b) W. D. Hillis and L. W. Tucker, *Commun. ACM*, **36**, 31 (1993).
9. R. Fine, G. Dimmler, and C. Levinthal, *PROTEINS*, **11**, 242 (1991).

10. J. Higo, T. Ito, T. Fukushige, H. Miyagawa, J. Makino, T. Ebisuzaki, S. Endo, K. Kitamura, K. Nagayama, and D. Sugimoto, *J. Comp. Chem.*, **15**, 1372 (1994).
11. T. Ito, T. Fukushige, J. Makino, T. Ebisuzaki, S. K. Okumura, D. Sugimoto, H. Miyagawa, and K. Kitamura, *PROTEINS*, **20**, 139 (1994).
12. (a) H. Schreiber and O. Steinhauser, *Biochemistry* **31**, 5856 (1992); (b) H. Schreiber and O. Steinhauser, *Chem. Phys.*, **168**, 75 (1992); (c) J. Guenot and P. A. Kollman, *J. Comp. Chem.*, **14**, 295 (1993).
13. L. Verlet, *Phys. Rev.*, **159**, 98 (1967).
14. *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985*, Institute of Electrical and Electronics Engineers, Inc., New York, 1985.
15. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 1988.
16. D. E. Knuth, *Seminumerical Algorithms*, Vol. 2 of *The Art of Computer Programming*, 2nd ed., Addison-Wesley, Reading, MA, 1981.
17. A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., McGraw-Hill, New York, 1978.