JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

Modified Anderson Method for Accelerating 3D-RISM Calculations Using Graphics Processing Unit

Yutaka Maruyama[†,§] and Fumio Hirata*[,†,‡]

[†]Department of Theoretical and Computational Molecular Science, Institute for Molecular Science, Okazaki 444-8585, Japan

[‡]Department of Functional Molecular Science, The Graduate University for Advanced Studies, Okazaki 444-8585, Japan

ABSTRACT: A fast algorithm is proposed to solve the three-dimensional reference interaction site model (3D-RISM) theory on a graphics processing unit (GPU). 3D-RISM theory is a powerful tool for investigating biomolecular processes in solution; however, such calculations are often both memory-intensive and time-consuming. We sought to accelerate these calculations using GPUs, but to work around the problem of limited memory size in GPUs, we modified the less memory-intensive "Anderson method" to give faster convergence to 3D-RISM calculations. Using this method on a Tesla C2070 GPU, we reduced the total computational time by a factor of 8, 1.4 times by the modified Andersen method and 5.7 times by GPU, compared to calculations on an Intel Xeon machine (eight cores, 3.33 GHz) with the conventional method.

INTRODUCTION

Increasing attention has been paid to the three-dimensional reference interaction site model (3D-RISM) theory, a statistical mechanics theory of molecular liquids,[1] since it is capable of describing self-organization and molecular recognition, which are undoubtedly important processes for life phenomena. The power of the method has been demonstrated by application to a number of biomolecular processes in solution,[2−7] for example, water channels or aquaporins,[8−10] the hydration structure around DNA,[11] the B-Z transition of DNA,[12] and the stable structure of telomeric DNA,[13] etc.

Although many applications have demonstrated the superiority of the theory in evaluating solvation properties of biomolecules compared to other methods, both in terms of structure and thermodynamics, one particular shortcoming has prevented its application to solving problems in biochemistry and biophysics. This is the computational cost, since it currently takes approximately one half hour to solve the 3D-RISM equation for a single protein structure using a commodity workstation, despite great improvements achieved during the past few years to reduce the computational cost. Therefore, for the theory in its present state, it is somewhat difficult to apply it to those processes in which structural fluctuations of biomolecules play an essential role. Such processes include protein folding and unfolding, induced fitting of a ligand to a receptor, and predicting intrinsic protein disorder. These applications require the sampling of many different conformations of a protein or DNA by combining the method with a structural optimization or molecular dynamics simulation, requiring a 3D-RISM calculation for every step of sampling.[12−17] Another case in which many 3D-RISM calculations are required is that where hybridized quantum chemistry (SCF) and 3D-RISM calculations are used to evaluate the electronic structure of a protein in solution. In that case, a 3D-RISM calculation should be performed for each SCF step until overall convergence, in terms of both solvent distribution and electronic structure, is attained.[18−25]

Enhanced computational performance has thus far been achieved by increasing the number of commodity central processing units (CPUs) using OpenMP (intranode), MPI (internode), or OpenMP/MPI hybrids. The latest developments to the Peta-flop machine project in Japan, called the "K computer project," have also been achieved along this line. However, there are grave concerns among computer scientists about continuing along this direction with regard to building a high-performance computer. These concerns are two-fold: First, the cost of building such a machine is so high that the number of users that will be able to access it will inevitably be limited. Second, the electrical power consumption for operating such a machine may become intolerably large. Due to these concerns, streaming processors, such as graphics processing units (GPUs), are attracting a great deal of attention as high-performance computing devices. They are potentially several times faster than a commodity CPU for calculating single/double-precision floating-point numbers because they are specialized for computation-intensive, highly data-parallel processors. The energy consumption is also substantially lower compared to a commodity machine. The high performance of GPUs makes them an attractive alternative for accelerating scientific computation. GPUs have thus far been successfully applied to various types of applications, including molecular dynamics[26−43] and quantum chemistry.[44−60]

An advantage of GPUs is in high memory bandwidth. For example, the bandwidth of an NVIDIA Tesla C1060 or C2070 is 102 or 144 GB/s,[61,62] respectively, whereas that of an Intel CPU is only 25.6 GB/s.[63] This is of great advantage to numerical algorithms that are bound by the available access to memory. The 3D-FFT and the modified method of direct inversion in iterative subspace (MDIIS), used heavily in the 3D-RISM calculations, are two examples of such algorithms.[64] Table 1 shows, for example, the percentage of computational time in iterative steps for the 3D-RISM calculation of a DNA

Received: May 7, 2012
Published: July 23, 2012

   dx.doi.org/10.1021/ct300355r | *J. Chem. Theory Comput.* 2012, 8, 3015−3021

Table 1. Percentage (Time) of Each Subroutine in Iteration Steps

subroutine	FFT[a]	MDIIS	3D-RISM-KH	rest
%	35.8	32.8	15.8	15.6

[a]FFT routine is presented by Ooura.[65]

fragment in water. The 3D-FFT and MDIIS routines spend large portions of CPU time in an iterative process of the 3D-RISM calculation. The results suggest that a GPU is suitable for 3D-RISM calculations.

However, there is a problem that should be solved in order for GPUs to be employed for 3D-RISM calculations. The total amount of working memory on a GPU chip is not very large. The global memory on a GPU card is much smaller than that for CPUs. This limitation in memory volume is a serious drawback to the use of GPUs. A 3D-RISM calculation generally requires a large memory space. For example, it requires 9 GB of memory for a fine-grid calculation of DNA in water. On the other hand, the NVIDIA Tesla C1060 or C2070, which have the largest memories among GPUs, have only 4 or 6 GB of memory. Implementation of the 3D-RISM program on a multi-GPU processer may reduce the memory volume problem. However, the very slow data transfer between a CPU and GPU, or between a GPU and GPU, in the current stage of development worsens the performance. Therefore, the memory space required to perform a 3D-RISM calculation should be substantially reduced in order for GPUs to be effective. However, this then begs the question of how we can reduce the memory requirement of a 3D-RISM calculation. Answering this question is the main purpose of the present paper.

According to our analysis, a subroutine in the 3D-RISM program uses a major portion of the memory space, namely the MDIIS subroutine. For example, in the case of DNA in water, the MDIIS routine uses about 6 GB out of 9 GB, while the remaining routines of the calculation require only 3 GB. In short, a 3D-RISM calculation requires about 3 GB without the MDIIS subroutine in a case such as this. To run the 3D-RISM program on GPUs, we can reduce the memory used by the MDIIS method. However, this reduction drastically worsens the performance (Table 2). The iteration numbers with two (C1060) and four (C2070) subspaces drastically increase compared to the usual 10 subspaces case.

Table 2. Number of Iterations Required until Convergence[a]

number of subspaces	2	4	10
Iterations	3134	1154	278

[a]Two, four, and 10 MDIIS subspaces use 1.2, 2.4, and 5.9 GB memories, respectively, to accelerate the 3D-RISM calculation.

In the present paper, we propose a method of convergence for iterative solutions of 3D-RISM, replacing MDIIS. The method, referred to as a "modified Anderson method," updates the basis vectors at every iteration of a 3D-RISM calculation. The method uses less memory and converges faster compared to MDIIS. The performances of the new algorithm on CPUs and GPUs are compared with those of the original Anderson[66] and MDIIS methods.

METHOD

3D-RISM Theory. The 3D-RISM integral equation for the 3D solute−solvent total and direct correlation functions, $h_\gamma^{uv}(r)$ and $c_\gamma^{uv}(r)$, is written as

$$h_\gamma^{uv}(r) = \sum_{\gamma'} c_{\gamma'}^{uv}(r) * (\omega_{\gamma'\gamma}^{vv}(r) + \rho^v h_{\gamma'\gamma}^{vv}(r)) \tag{1}$$

where $\omega_{\gamma'\gamma}^{vv}(r) = \delta(r - l_{\gamma'\gamma}^{vv})$ is the intramolecular matrix of solvent molecules with site separations $l_{\gamma'\gamma}^{vv}$; γ and γ' are interaction sites of solvent molecules; ρ_v is the number density of the solvent and $*$ refers to the convolution integral in direct space.

The radial site−site correlation functions of the pure solvent, $h_\gamma^{vv}(r)$, are independently obtained from conventional 1D-RISM theory. This provides a proper description of the structure of the solution. The 3D solute−solvent site correlation functions are specified on a 3D linear grid, and the convolutions in eq 1 are handled by means of the 3D Fast Fourier transform.

In order for the 3D-RISM equation to be solved, it should be complemented by another equation that relates $h_\gamma^{uv}(r)$ to $c_\gamma^{uv}(r)$. We have recently proposed and tested a new closure which we refer to as the Kovalenko−Hirata (KH) approximation.[67−69] This closure takes the form

$$h_\gamma^{uv}(r) = \begin{cases} \exp(\chi) - 1 & \text{for } \chi \le 0 \\ \chi & \text{for } \chi > 0 \end{cases}$$
$$\chi = -\beta u_\gamma^{uv}(r) + h_\gamma^{uv}(r) - c_\gamma^{uv}(r) \tag{2}$$

where $\beta = 1/(k_B T)$, the interaction potential $u_\gamma^{uv}(r)$ between solvent site γ.

For the pair potentials between interaction sites of every species, we employ the common model that consists of the Lennard-Jones (LJ) and electrostatic interaction terms:

$$u_{\alpha\gamma}(r) = 4\varepsilon_{\alpha\gamma}\left[\left(\frac{\sigma_{\alpha\gamma}}{r}\right)^{12} - \left(\frac{\sigma_{\alpha\gamma}}{r}\right)^{6}\right] + \frac{1}{4\pi\varepsilon_0}\frac{q_\alpha q_\gamma}{r} \tag{3}$$

where $\varepsilon_{\alpha\gamma}$ and $\sigma_{\alpha\gamma}$ are the LJ energy and size parameters for a pair of solute sites α and γ, respectively, ε_0 is the dielectric constant of the vacuum, and q_α is the partial charge on site α.

The hydration free energy (HFE), $\Delta\mu$, is calculated by using the 3D extension to the Singer−Chandler formula:[70,71]

$$\Delta\mu = \rho k_B T \sum_\alpha \int_{V_{cell}} dr\left[\frac{1}{2}(h_\gamma(r))^2\Theta(-h_\gamma(r)) - c_\gamma(r) - \frac{1}{2}h_\gamma(r)\, c_\gamma(r)\right] \tag{4}$$

Figure 1 shows a flowchart that illustrates the process behind a 3D-RISM calculation. The initialization process consists of reading parameters and calculating potentials between the solute and solvent. The iteration procedure spans the range between the calculation concerning the KH-closure and the that of accelerating convergence. In the numerical procedure, 3D-FFT and a method of accelerating the convergence of the iterative procedure play important roles in the 3D-RISM calculation.

Modified Anderson Method. In the conventional program concerning 3D-RISM, the MDIIS algorithm has been used to accelerate the convergence of the iterative procedure. The algorithm, however, is not suited to GPUs because it requires a large memory space. In general, more than

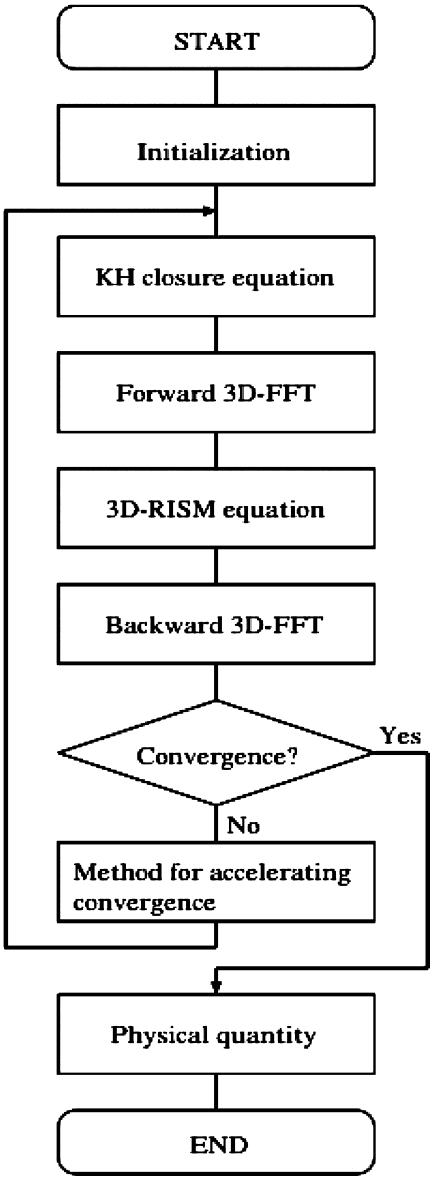

Figure 1. A flowchart illustrating the 3D-RISM calculation process.

60% of the total memory for the calculation is spent by MDIIS. On the other hand, reduction of the MDIIS subspace to save memory slows down the rate of convergence.

The Anderson method was proposed in 1965 as an iterative procedure for nonlinear integral equations.[66] The method is one kind of mixing method. The Anderson method uses a dynamic mixing parameter calculated every step whereas a basic mixing method uses a static one. The Anderson method, which has been devised to accelerate the convergence of quantum calculations, is considered to be a good candidate for an alternative to MDIIS[72] since the algorithm features stable convergence. However, the convergence of the original Anderson method turns out to be very slow for 3D-RISM calculations. Here, we propose a modified Anderson (MA) method to accelerate the convergence of 3D-RISM calculations.

We define $t^{(n)}(r)$ and $\tau^{(n)}(r)$ as functions computed by the nth time iteration of the 3D-RISM calculation and by the MA method, respectively. The physical meanings of these are identical, but the calculation processes are different. We calculate $t^{(n)}(r) = h(r) - c(r)$ after calculating the 3D-RISM equation and performing a backward 3D-FFT (Figure 1). On the other hand, we calculate $\tau^{(n)}(r)$ by the following process.

We define the residual vector $r^{(n)}(r)$ as the difference between $t^{(n)}(r)$ and $\tau^{(n)}(r)$:

$$r^{(n-1)}(r) = t^{(n)}(r) - \tau^{(n-1)}(r) \tag{5}$$

Define an inner product of two vectors with weight factor, w,

$$(u(r), v(r)) = \sum_r u(r)\, v(r) w \tag{6}$$

We choose the inverse of the number density, $1/\rho^v$, as the weight factor. In some cases, it is convenient to generalize (eq 6) in the usual way to include a nondiagonal, positive-definite metric component. Define

$$u^{(n-1)}(r) = \tau^{(n-1)}(r) + \sum_{j=1}^{M} (\theta_j^{(n-1)} + s_j)(\tau^{(n-j-1)}(r) - \tau^{(n-1)}(r)) \tag{7}$$

$$v^{(n)}(r) = t^{(n)}(r) + \sum_{j=1}^{M} (\theta_j^{(n-1)} + s_j)(t^{(n-j)}(r) - t^{(n)}(r)) \tag{8}$$

for $i = 1, 2, ..., M$. Here, we introduce a new tuning parameter, s_j, as the modification. If $s_j = 0$, then the above equations revert to the original Anderson method. This parameter rebalances the contributions of previous distribution functions and adapts the original Anderson method for accelerating the convergence of 3D-RISM calculations.

Choose the free parameters θ_j so as to minimize the linearized residual $R^{(n)}$ defined by

$$R^{(n)} = \frac{1}{2}(v^{(n)}(\mathbf{r}) - u^{(n-1)}(\mathbf{r}),\ v^{(n)}(\mathbf{r}) - u^{(n-1)}(\mathbf{r})) \tag{9}$$

This yields

$$\frac{\partial R^{(n)}(r)}{\partial \theta_j} = (r^{(n)}(r) - r^{(n-1)}(r),\ v^{(n)}(r) - u^{(n-1)}(r)) = 0 \tag{10}$$

by solving the simultaneous equations for θ_j,

$$\sum_j^M (R^{(n-1)}(r) - R^{(n-i-1)}(r),\ R^{(n-1)}(r) - R^{(n-j-1)}(r)) \times \theta_j^{(n-1)} = (R^{(n-1)}(r),\ R^{(n-1)}(r) - R^{(n-i-1)}(r)) \tag{11}$$

for $i = 1, 2, ..., M$. Then, we can get the new $\tau^{(n)}(r)$ by

$$\tau^{(n)}(r) = u^{(n-1)}(r) + b(v^{(n)}(r) - u^{(n-1)}(r)) \tag{12}$$

where b is a mixing parameter. We refer to the algorithm so defined as the extrapolation algorithm. Usually, the choice $b = 1$ is most appropriate; the choice $b = 0$ is excluded for reasons to be discussed below. The extrapolation algorithm resembles, in some respects, various search or descent methods for root finding; indeed, a "least-squares solution" can be found if no true solution exists. As in such methods, it is often useful to "under-relax" the iteration by choosing $0 < b < 1$, but the optimum b must be determined empirically.

dx.doi.org/10.1021/ct300355r | J. Chem. Theory Comput. 2012, 8, 3015−3021

We describe the difference between MDIIS and the Anderson method here. According to the above symbols, MDIIS is expressed by

$$\tau^{(n)}(r) = t^{(n)} + \sum c_i(b\tau^{(n-i)}(r) - t^{(n-i)}(r)) \quad (13)$$

where c_i are coefficients which are decided by the Lagrange multiplier technique and b is a mixing parameter. A new trial function consists of $t^{(n)}$ and a linear combination of previous error vectors. The MDIIS method therefore requires a large memory space to store typically 10 error vectors from previous steps. On the other hand, the expression of the Anderson method is the mixing form in eq 12, which uses a vector from just two previous steps. This enables us to save memory space to a large extent without sacrificing the convergence rate.

Calculation Condition. To test the MA method, we examined the hydration structure around B-DNA. For DNA, we employed the parameters from the Amber-02 force field.[73] Fragments of B-DNA with 12 base pairs d-(5′CGCGCGCGCGCG3′), totaling 754 atoms, were prepared as the solute. We used the SPC/E model[74] with the hydrogen Lennard-Jones term (σ = 1.0 Å and ε = 0.046 kcal/mol) for the water sites.[75] The LJ parameter between unlike atomic sites was determined from the standard Lorentz−Berthelot combination rules. All calculations were carried out for ambient water at temperature T = 298.15 K. The 3D-RISM/KH equations were solved on a grid of 256^3 points in a cubic cell of size 64 Å^3. In this case, a grid space of 0.25 Å is sufficiently fine to produce the solvation free energy without significant numerical errors. To converge the equations, we used the MDIIS method (10 subspaces),[64] the Anderson,[66] and the MA methods (two subspaces). One subspace requires 512 MB for water (two components; oxygen and hydrogen) on a grid of 256^3. In addition, each routine uses memory for working space. Consequently, the MDIIS routine required 5.9 GB memory in the calculation, while the MA routine used just 1.1 GB.

RESULTS AND DISCUSSION

Modified Anderson Method. We implemented the 3D-RISM program with the MA method on GPUs using CUDA 4.0 with compile options "-arch sm_20 -Iinclude LIBS = -lcufft -lcuda -Llib64". The host computer had an Intel Core i7 (2.67 GHz) CPU, 24 GB of memory, and a Tesla C1060 or C2070 GPU card. The GNU g++ compiler (version 4.4.2) was used to compile the program for the CPU. The compiler options "-O3 -fomit-frame-pointer" were used for GNU g++. The calculations were executed with double-precision accuracy.

First, we examined the efficiency of the MA method, adopted here for solving the 3D-RISM-KH equations. Table 3 shows the dependence of the number of iterations on the mixing parameter b. We found that the optimal value of the mixing parameter is 0.6 for both Anderson methods. The MA method (s_1 = 0.2 and s_2 = 0.0) shows faster convergence compared to the conventional Anderson method for the whole value of b.

Table 3. Number of Iterations and Their Dependence upon the Mixing Parameter b

b	0.5	0.6	0.7	0.8	0.9
Anderson	1645	1381	1506	1752	2041
modified Anderson[a]	305	213	255	233	245

[a]The weighting parameters (s_1, s_2) are set to (0.2, 0.0) for the modified Anderson method.

The acceleration achieved by the modification is more than six times. This improvement is remarkable, particularly when one considers the low computational cost of the modification. The results of differences among a set of s_i with b = 0.6 are shown in Table 4. The best set is that when s_1 = 0.2 and s_2 = 0.0. The modification with tuning parameters accelerates the convergence of a 3D-RISM calculation.

Table 4. The Dependence of Convergence on the Turning Parameters[a]

	s_1 = 0.0	0.1	0.2	0.3	0.4	0.5
s_2 = 0.0	1381	295	213	222	263	333
0.1	535	278	269	229	230	576
0.2	270	239	218	252	233	1019
0.3	215	227	221	237	235	856

[a]The mixing parameter b is set at 0.6.

Figure 2 presents the root-mean-square (RMS) value of the residual

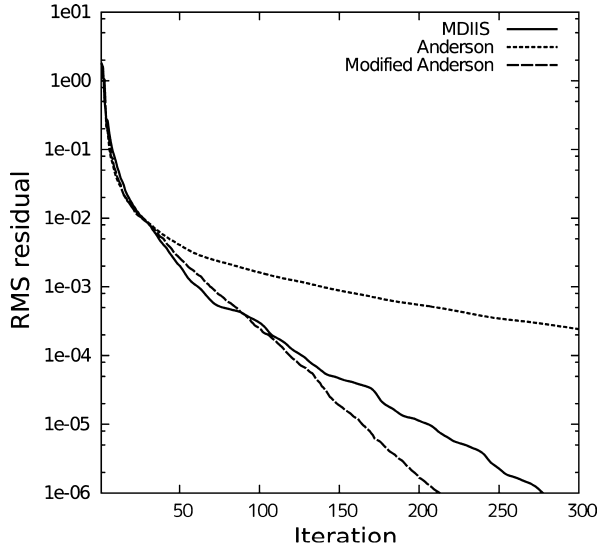

Figure 2. Root mean square residual against the number of iteration steps for the calculation of 3D site distribution profiles of water around a DNA molecule by 3D-RISM and KH closure equations. Solid, dotted, and dashed lines are for MDIIS, Anderson, and modified Anderson algorithms, respectively.

$$\left[\frac{1}{N^3}\sum_{r}^{N^3} |R^{(n-1)}(r)|^2\right]^{1/2}$$

versus the number of iterations performed. The Anderson method shows slower convergence compared to the MDIIS and MA methods, though it does show stable convergence. The MDIIS method shows faster convergence than the Anderson method. The figure shows the improvement obtained in acceleration and stability of convergence by the MA method compared to the other two methods. This demonstrates that the MA method works effectively for the convergence of the 3D-RISM equations with KH closure.

The hydration free energy (HFE) versus the number of iterations is shown in Figure 3. The arrows indicate the number of iterations within which the MA method and MDIIS

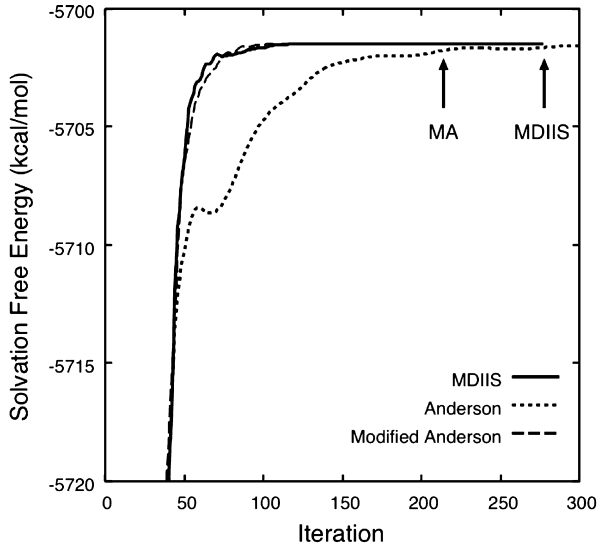

Figure 3. Hydration free energy against the number of iteration steps for the calculation of 3D site distribution profiles of water around a DNA molecule by 3D-RISM and KH closure equations. Solid, dotted, and dashed lines are for MDIIS, Anderson, and modified Anderson algorithms, respectively.

converge. All of the methods reach the same value of HFE, but the MDIIS and MA methods certainly accelerate the 3D-RISM iteration. Similar to that seen for the RMS value, the MA method shows the most stable convergence among the three methods studied.

Our main results are shown in Table 5 in which the computational time of the iteration process by each method is

Table 5. Computation Times (s) of the Iteration Steps[a]

	MDIIS	Anderson	modified Anderson
Intel i7 (CPU)	2546.1	10566.1	1797.1
C1060 (GPU)		857.1	132.7
C2070 (GPU)		391.2	60.6

[a]The mixing parameter b is set to 0.6 for the Anderson and modified Anderson methods. The weighting parameters (s1, s2) are set to (0.2, 0.0) for the modified Anderson method.

listed. We mention that the 3D-RISM calculation cannot be performed with the MDIIS method on GPUs because of the limitation in its memory space. The Anderson method is 4 times slower than the MDIIS method in terms of CPU time, but the MA method is faster than the other two. Of course, the number of iterations dominates the calculation time. The MA method on C1060/C2070 is 19/42 times faster than the original program with the MDIIS method on CPUs. The MA method on C1060/C2070 is about 14/30 times faster than the same method on CPUs. This result proves that GPUs accelerate the 3D-RISM calculation.

Table 6 shows the computation time that each subroutine takes in an iteration process and its percentage contribution to the total time. All of the subroutines are well-balanced in terms of computation time, except for the FFT routine (CUFFT), which is provided by NVIDIA. The CUFFT routine itself cannot be tuned because of the absence of its source code. The routines for the MA method, 3D-RISM, and KH closure equations are well suited to GPUs. An improvement to the 3D-

Table 6. Computation Time and the Percentage of Each Subroutine on GPU

	FFT	modified Anderson	3D-RISM-KH	rest
C1060	68.4 (51.5)	19.0 (14.3)	22.1 (16.7)	23.2 (17.5)
C2070	29.9 (49.3)	7.8 (12.9)	11.9 (19.6)	11.0 (18.2)

FFT library for a GPU is required to accelerate the 3D-RISM program further.

Depicted in Figure 4 are the total times for a 3D-RISM calculation of a DNA molecule in water using a Xeon (eight

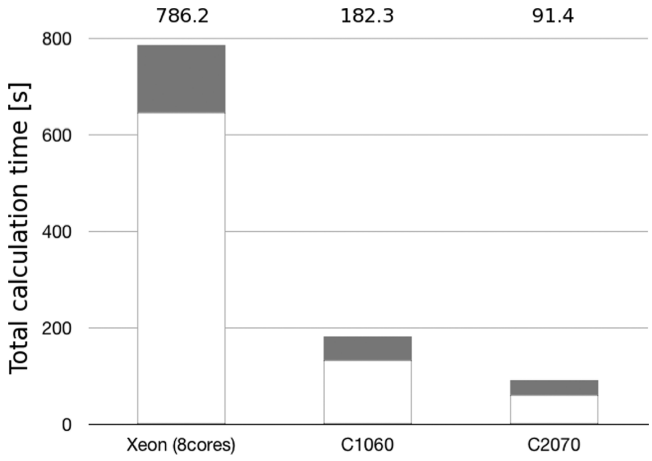

Figure 4. The total calculation time for a DNA molecule in water on a Xeon (eight cores, OpenMP) machine, and using C1060 and C2070. Black and white regions indicate the initialization and iterations, respectively.

cores, 3.33 GHz) machine, and using C1060 and C2070. The 3D-RISM program on the Xeon (eight cores) is parallelized using OpenMP and compiled by a PGI compiler (version 11.1). We used "-fastsse -O4 -Mprefetch=distance:8,nta -Minline=size:50 -mp -Mfixed -mcmodel=medium" as compiler options. The computation times on C1060/C2070 are 4/8 times shorter than on Xeon (eight cores). Similar to the iteration process, C2070 is twice as fast as C1060 for the initialization process.

CONCLUSIONS

We presented a modified Anderson method and applied it to the integral equation theory of liquids to accelerate convergence of a solution to 3D equations. We updated the Anderson basis vectors at every iterative step by using the approximate residual obtained in the Anderson extrapolation. The modified Anderson method on GPUs provides greatly accelerated convergence when compared to the MDIIS method on CPUs. 3D-RISM calculations on GPUs are good for hybrid methods, such as MD/3D-RISM, and for structure optimizations with 3D-RISM. These methods require many 3D-RISM calculations.

It is also useful to replace the 3D-FFT routine for GPUs with one that is more effective, for example, the NUFFT developed by Nukada.[76,77] These accelerations to 3D-RISM will allow its application to hybrid methods, such as MD/3D-RISM, MC/3D-RISM, etc.

The weak point of a GPU is small memory, which restricts the applicability of 3D-RISM calculations by this means. We

therefore aim to develop 3D-RISM code for multi-GPUs to avoid this restriction in the future.

AUTHOR INFORMATION

Corresponding Author

*E-mail: hirataf@fc.ritsumei.ac.jp.

Present Address

[§]Institute for Protein Research, Osaka University, Suita, Osaka 565-0871, Japan.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank Dr. Yoshida for fruitful discussions. This work is supported by a grant from Grand Challenges in Next-Generation Supercomputing Project, Nanoscience Program, and the Grant-in-Aid for Scientific Research on Innovative Areas "Molecular Science of Fluctuations toward Biological Functions" from the MEXT in Japan.

REFERENCES

(1) Hirata, F.; Sato, H.; Kinoshita, M.; Kovalenko, A.; Chong, S.-H. *Molecular Theory of Solvation*; Hirata, F., Ed.; Kluwer Academic Publishers: Dordrecht, The Netherlands, 2003.

(2) Imai, T.; Hiraoka, R.; Kovalenko, A.; Hirata, F. *J. Am. Chem. Soc.* **2005**, *127*, 15334.

(3) Imai, T.; Hiraoka, R.; Kovalenko, A.; Hirata, F. *Proteins* **2007**, *66*, 804.

(4) Yoshida, N.; Phongphanphanee, S.; Maruyama, Y.; Imai, T.; Hirata, F. *J. Am. Chem. Soc.* **2006**, *128*, 12042.

(5) Yoshida, N.; Phongphanphanee, S.; Hirata, F. *J. Phys. Chem. B* **2007**, *111*, 4588.

(6) Ikuta, Y.; Maruyama, Y.; Matsugami, M.; Hirata, F. *Chem. Phys. Lett.* **2007**, *433*, 403.

(7) Maruyama, Y.; Matsugami, M.; Ikuta, Y. *Condens. Matter Phys.* **2007**, *10*, 315.

(8) Phongphanphanee, S.; Yoshida, N.; Hirata, F. *Chem. Phys. Lett.* **2007**, *449*, 433.

(9) Phongphanphanee, S.; Yoshida, N.; Hirata, F. *J. Am. Chem. Soc.* **2008**, *130*, 1540.

(10) Phongphanphanee, S.; Yoshida, N.; Hirata, F. *J. Mol. Liq.* **2009**, *147*, 107.

(11) Yonetani, Y.; Maruyama, Y.; Hirata, F.; Kono, H. *J. Chem. Phys.* **2008**, *128*, 185102.

(12) Maruyama, Y.; Yoshida, N.; Hirata, F. *J. Phys. Chem. B* **2010**, *114*, 6464.

(13) Maruyama, Y.; Matsushita, T.; Ueoka, R.; Hirata, F. *J. Phys. Chem. B* **2011**, *115*, 2408.

(14) Miyata, T.; Hirata, F. *J. Comput. Chem.* **2008**, *29*, 871.

(15) Miyata, T.; Ikuta, Y.; Hirata, F. *J. Chem. Phys.* **2010**, *133*, 044114.

(16) Luchko, T.; Gusarov, S.; Roe, D. R.; Simmerling, C.; Case, D. A.; Tuszynski, J.; Kovalenko, A. *J. Chem. Theory Comput.* **2010**, *6*, 607.

(17) Genheden, S.; Luchko, T.; Gusarov, S.; Kovalenko, A.; Ryde, U. *J. Phys. Chem. B* **2010**, *114*, 8505.

(18) Kovalenko, A.; Hirata, F. *J. Chem. Phys.* **1999**, *110*, 10095.

(19) Sato, H.; Kovalenko, A.; Hirata, F. *J. Comput. Chem.* **2000**, *112*, 9463.

(20) Kovalenko, A.; Hirata, F. *J. Mol. Liq.* **2001**, *90*, 215.

(21) Yoshida, N.; Hirata, F. *J. Comput. Chem.* **2006**, *27*, 453.

(22) Gusarov, S.; Ziegler, T.; Kovalenko, A. *J. Phys. Chem. A* **2006**, *110*, 6083.

(23) Casanova, D.; Gusarov, S.; Kovalenko, A.; Ziegler, T. *J. Chem. Theory Comput.* **2007**, *3*, 458.

(24) Malvaldi, M.; Bruzzone, S.; Chiappe, C.; Gusarov, S.; Kovalenko, A. *J. Phys. Chem. B* **2009**, *113*, 3536.

(25) Li, Q. B.; Gusarov, S.; Evoy, S.; Kovalenko, A. *J. Phys. Chem. B* **2009**, *113*, 9958.

(26) Stone, J. E.; Phillips, J. C.; Freddolino, P. L.; Hardy, D. J.; Trabuco, L. G.; Schulten, K. *J. Comput. Chem.* **2007**, *28*, 2618.

(27) Yang, J. K.; Wang, Y. J.; Chen, Y. F. *J. Comput. Phys.* **2007**, *221*, 799.

(28) Anderson, J. A.; Lorenz, C. D.; Travesset, A. *J. Comput. Phys.* **2008**, *227*, 5342.

(29) van Meel, J. A.; Arnold, A.; Frenkel, D.; Zwart, S. F. P.; Belleman, R. G. *Mol. Simul.* **2008**, *34*, 259.

(30) Liu, W. G.; Schmidt, B.; Voss, G.; Muller-Wittig, W. *Comput. Phys. Commun.* **2008**, *179*, 634.

(31) Harvey, M. J.; Giupponi, G.; De Fabritiis, G. *J. Chem. Theory Comput.* **2009**, *5*, 1632.

(32) Chen, F. G.; Ge, W.; Li, J. H. *Sci. China, Ser. B: Chem.* **2009**, *52*, 372.

(33) Friedrichs, M. S.; Eastman, P.; Vaidyanathan, V.; Houston, M.; Legrand, S.; Beberg, A. L.; Ensign, D. L.; Bruns, C. M.; Pande, V. S. *J. Comput. Chem.* **2009**, *30*, 864.

(34) Hardy, D. J.; Stone, J. E.; Schulten, K. *Parallel Comput.* **2009**, *35*, 164.

(35) Narumi, T.; Yasuoka, K.; Taiji, M.; Hofinger, S. *J. Comput. Chem.* **2009**, *30*, 2351.

(36) Harvey, M. J.; De Fabritiis, G. *J. Chem. Theory Comput.* **2009**, *5*, 2371.

(37) Eastman, P.; Pande, V. S. *J. Comput. Chem.* **2010**, *31*, 1268.

(38) Cickovski, T.; Chatterjee, S.; Wenger, J.; Sweet, C. R.; Izaguirre, J. A. *J. Comput. Chem.* **2010**, *31*, 1345.

(39) Schmid, N.; Botschi, M.; Van Gunsteren, W. F. *J. Comput. Chem.* **2010**, *31*, 1636.

(40) Bauer, B. A.; Davis, J. E.; Taufer, M.; Patel, S. *J. Comput. Chem.* **2010**, *32*, 375.

(41) Sunarso, A.; Tsuji, T.; Chono, S. *J. Comput. Phys.* **2010**, *229*, 5486.

(42) Jha, P. K.; Sknepnek, R.; Guerrero-Garcia, G. I.; de la Cruz, M. O. *J. Chem. Theory Comput.* **2010**, *6*, 3058.

(43) Zhmurov, A.; Dima, R. I.; Kholodov, Y.; Barsegov, V. *Proteins* **2010**, *78*, 2984.

(44) Yasuda, K. *J. Comput. Chem.* **2008**, *29*, 334.

(45) Yasuda, K. *J. Chem. Theory Comput.* **2008**, *4*, 1230.

(46) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. *J. Phys. Chem. A* **2008**, *112*, 2049.

(47) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222.

(48) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 1004.

(49) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 2619.

(50) Luehr, N.; Ufimtsev, I. S.; Martnez, T. J. *J. Chem. Theory Comput.* **2011**, *7*, 949.

(51) Isborn, C. M.; Luehr, N.; Ufimtsev, I. S.; Martnez, T. J. *J. Chem. Theory Comput.* **2011**, *7*, 1814.

(52) Genovese, L.; Ospici, M.; Deutsch, T.; Mehaut, J. F.; Neelov, A.; Goedecker, S. *J. Comput. Phys.* **2009**, *131*, 034103.

(53) Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L.; Shao, Y. H.; Aspuru-Guzik, A. *J. Chem. Theory Comput.* **2010**, *6*, 135.

(54) Asadchev, A.; Allada, V.; Felder, J.; Bode, B. M.; Gordon, M. S.; Windus, T. L. *J. Phys. Chem.* **2010**, *6*, 696.

(55) Vysotskiy, V. P.; Cederbaum, L. S. *J. Chem. Theory Comput.* **2011**, *7*, 320.

(56) DePrince, A. E., III; Hammond, J. R. *J. Chem. Theory Comput.* **2011**, *7*, 1287.

(57) Ma, W.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. *J. Chem. Theory Comput.* **2011**, *7*, 1316.

(58) Janes, P. P.; Rendell, A. P. *J. Chem. Theory Comput.* **2011**, *7*, 1631.

(59) Uejima, Y.; Terashima, T.; Maezono, R. *J. Comput. Chem.* **2011**, *32*, 2264.

(60) Wilkinson, K. A.; Sherwood, P.; Guest, M. F.; Naidoo, K. J. *J. Comput. Chem.* **2011**, *32*, 2313.
(61) Tesla C1060 GPU computing processor. http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C1060_US_Jan10_lores_r1.pdf (accessed June 6, 2012).
(62) Tesla C2050/C2070 GPU computing processor. http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf (accessed June 6, 2012).
(63) Intel Corporation, Intel QuickPath Architecture, White Paper. http://www.intel.com/content/dam/doc/white-paper/performance-quickpath-architecture-paper.pdf (accessed June 6, 2012).
(64) Kovalenko, A.; Ten-No, S.; Hirata, F. *J. Comput. Chem.* **1999**, *20*, 928.
(65) Ooura, T. General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package. http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html (accessed June 13, 2012).
(66) Anderson, D. *J. Assoc. Comput. Mach.* **1965**, *12*, 547.
(67) Kovalenko, A.; Hirata, F. *J. Phys. Chem. B.* **1999**, *103*, 7942.
(68) Kovalenko, A.; Hirata, F. *J. Chem. Phys.* **1999**, *110*, 10095.
(69) Kovalenko, A.; Hirata, F. *J. Chem. Phys.* **2000**, *112*, 10391.
(70) Singer, S. J.; Chandler, D. *Mol. Phys.* **1985**, *55*, 621.
(71) Kovalenko, A.; Hirata, F. *J. Chem. Phys.* **1999**, *110*, 10095.
(72) Inaba, T.; Sato, F. *J. Comput. Chem.* **2007**, *28*, 984.
(73) Cieplak, P.; Caldwell, J.; Kollman, P. *J. Comput. Chem.* **2001**, *22*, 1048.
(74) Berendsen, H. J. C.; Grigera, J. R.; Straatsma, T. P. *J. Chem. Phys.* **1987**, *91*, 6269.
(75) Pettitt, B. M.; Rossky, P. J. *J. Chem. Phys.* **1982**, *77*, 1451.
(76) Nukada, A.; Ogata, Y.; Endo, T.; Matsuoka, S. Bandwidth Intensive 3-D FFT kernel for GPUs using CUDA. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Austin, TX, November 15−21, 2008; Teller, P. J., Ed.; IEEE Press: Piscataway, NJ, 2008.
(77) Nukada, A.; Matsuoka, S. Auto-Tuning 3-D FFT Library for CUDA GPUs. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, Portland, OR, November 14−20, 2009; Pinfold, W., Ed.; ACM New York: New York, 2009.