Development of evaluation system for numerical algorithms to solve linear equations

Shoji Itoh,[1] Hisashi Kotakemori,[2] and Hidehiko Hasegawa[1]

[1] University of Tsukuba, Japan
[2] JST CREST and the University of Tokyo, Japan

Abstract: A system for retrieving performance information on algorithms for solving linear equations is proposed and a method of evaluating numerical algorithms using data analysis is described. The system is web-based with an interface composed of selection menus. Users are able to easily compare the performance of numerical algorithms by selecting a coefficient matrix and candidate algorithms.

Keywords: Numerical algorithm, Performance evaluation, Information retrieval system

1. Introduction

In the field of scientific and technical computation, various equations which describe realistic problems like natural phenomena or engineering problems must be solved numerically. These problems can often be reduced to solving linear equations of the form

$$Ax = b \tag{1}$$

where the coefficient matrix A is typically a large matrix of size n × n. It is important to solve this equation rapidly and accurately.

There are various algorithms for solving such linear equations. Depending on the nature of the problem to be solved, these algorithms may not demonstrate sufficient performance or they may not even be able to find a numeric solution [2]. In practice, therefore, it is necessary to provide some guidelines for selecting the most suitable algorithm to employ.

In general, the performances of candidate algorithms can only be evaluated by implementing the algorithm with the actual simulation code and observing its computational performance. This approach takes some time, however, a situation which simulation researchers generally wish to avoid as much as possible. On the other hand, researchers developing algorithms often do not have sufficient information to select the most suitable numerical algorithm.

In this study, we collected and analyzed data obtained from actual calculations in order to improve this situation. We have been developing a system which allows the performance and characteristics of solution algorithms to be evaluated [6,7].

Data on the performance of iterative methods, with preconditioning, for solving linear equations was collected. A database was constructed from the data obtained, and an information retrieval system facilitating performance comparison and showing a graphical representation of the iterative methods was developed. Using this system, it is possible to instantly compare the performance of candidate algorithms in general numerical experiments.

In Section 2, we describe the conventional problems leading to this research, and we outline the concepts and goals of our work. In Section 3, we describe the system for evaluating the performance of algorithms for solving linear equations, and we describe the design of the system and the computer environment for the data registered in the performance information database. In Section 4, we briefly describe the numerical calculation library Lis used in our performance evaluation system. In Section 5, we describe a system for retrieving performance information that we developed as one way of evaluating the performance of the candidate algorithms.

2. Motivation, concept and aims of this study

There are many algorithms for solving linear equation (1), for example, the direct method based on Gaussian elimination, stationary iterative methods such as the successive over relaxation (SOR) method, and non-stationary iterative methods as represented by the conjugate gradient (CG) method and the biconjugate gradient (BiCG) method. Furthermore, these methods are often used in combination with techniques such as preconditioning to improve their solving efficiency.

Thus, there are many choices for solving linear equations, but there is little information indicating an appropriate match between the linear equations and the solution algorithms to employ (in this paper, we use the term "solution algorithm" to mean the combination of the actual solver and the techniques used in combination with the solver, like preconditioning). In particular, there is often a confusing choice of algorithms for solving linear equations with large, nonsymmetric coefficient matrices.

One algorithm that has been used is to consider, from a theoretical viewpoint, the convergence of an iterative method in the solution algorithm, but in carrying out actual numerical calculations, often the solution does not convergence as predicted by theory. Possible reasons for this are rounding errors involved in floating point arithmetic and the weakness of the theory in predicting the convergence of each algorithm. Regarding

the Krylov subspace method, one report states that, "There is no clear best Krylov subspace method at this time, and there will never be a best overall Krylov subspace method." [2].

Even if there is no concrete resolution of this situation, there is a desire to at least grasp only the correlation between the algorithms and the problem to be solved. The purpose of our work is therefore to develop computer systems and evaluation systems, and the main theme of our research is to construct "a survey and evaluate system for numerical algorithms" in various numerical calculation fields [6,7].

Our evaluation system currently handles solution algorithms for linear equations, and qualitative and quantitative analyses are carried out on the data obtained.

3. Development of performance evaluation system

In order to solve linear equations with high performance and accuracy, various solvers and preconditioning techniques have been proposed and many libraries are available. In this study, we used the sequential version of the Lis library [9,10,11]. The details of Lis are described in Section 4.

In the development of algorithms for solving linear equations, typical test problem used are Matrix Market [12], UF Sparse Matrix Collection [13], and so on. So far in this study, the linear equations of about fifty kinds of test matrices from the Matrix Market problem have been solved using Lis. The vector b on the right-hand side of eq. (1) was generated by substituting a solution vector with all elements set to 1.0. The initial vector was generated using a zero vector. Iterations were stopped when $||r_k||_2/||b||_2 \leq 1.0 \times 10^{-12}$, where r_k is a residual vector in the algorithm and k is the iteration number. The maximum iteration number was set to the same value as the size of the matrix.

Specifications of the computing environment for executing jobs and collecting data are shown in Table 1. Here, to acquire basic information on the algorithm, the job execution was performed in a single CPU, that is, sequential execution.

Table 1. The computing environment for executing jobs and collecting data.

Machine	Sun Fire V880
CPU	UltraSPARC III (900 MHz)
Memory size	16 GB
Operating System	Solaris 9
Compiler	Sun Workshop 6 (cc, f90)

Figure 1 shows a conceptual diagram of this system used for performance data collection, database construction, and data retrieval of the solution algorithms. The system is indicated by "Calculation Server" in Fig. 1; the block labeled "Matrices" uses the Matrix Market and the block labeled "Algorithms" uses the Lis library.

At present, an independent Data Analysis Server (see Fig. 1) is not prepared; instead the data obtained in the Calculation Server are transferred directly to the Web server. In research planned in the future, however, an independent server will be prepared to analyze the information accumulated in the database. The details of the Web Server are described in Section 5.

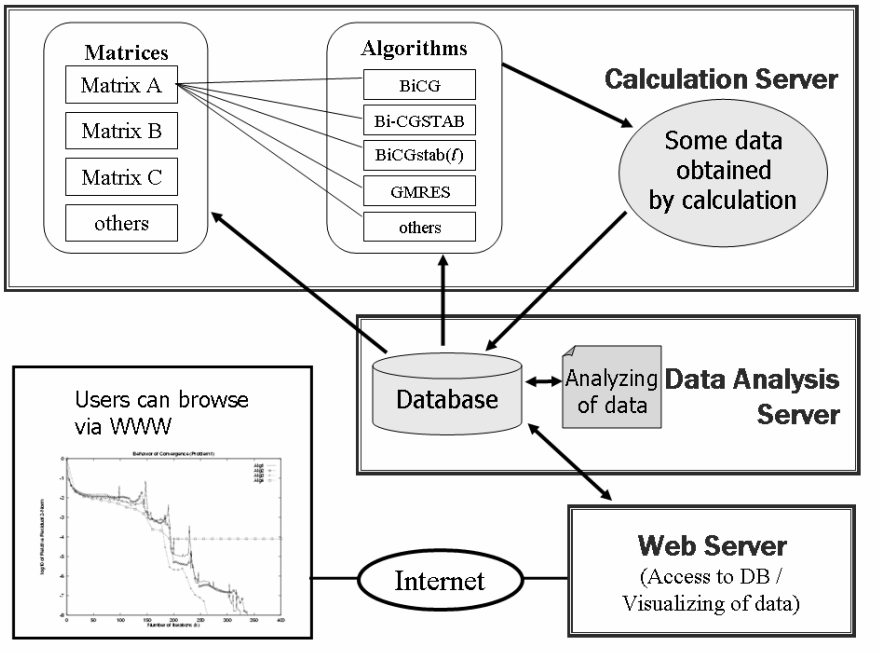


Figure 1 Computational evaluation system (conceptual diagram).

4. Outline of Lis Library

4.1 Concepts

Lis (a Library of Iterative Solvers for linear systems) is a numerical calculation

library for solving linear equations by iterative methods [9,10,11]. Most code in the Lis library is written in the C language, but in Lis-AMG, which includes AMG preconditioning routines, the AMG routines are written in Fortran 90. In the present version, however, calling programs must be written only in the C language (modified version of lis-AMG-1.0.1, 6 November, 2005).

The Lis library includes sequential and parallel versions of the algorithms, and the parallel versions use the message passing interface (MPI) library with distributed memory and OpenMP with shared memory. When compiling, the sequential or parallel version is specified by an argument in the make command (for example, to make the sequential version, the command is "**`make mode=seq`**"). Lis has the following features:

- Processing with a combination of arbitrary iterative methods and preconditioning is supported.
- A variety of sparse matrix storage forms are supported.
- In the call from main program, it is possible to use a common interface for both sequential and parallel versions.

4.2 Components of Lis

Twelve kinds of iterative methods for large real matrices, such as stationary iterative methods (Jacobi, SOR, and so on) and nonstationary iterative methods (CG, BiCG, and so on), were adopted, as shown in Table 2.

Table 2　Iterative methods of Lis

nonstationary	CG
iterative	BiCG
methods	CGS
	BiCGStab
	BiCGStab(ℓ)
	GPBiCG
	TFQMR
	Orthomin(m)
	GMRES(m)
stationary	Jacobi
iterative	Gauss-Seidel
methods	SOR

For the preconditioning, the methods shown in Table 3 were adopted.

Table 3 Preconditioning of Lis

Jacobi
ILU(k)
SSOR
Hybrid type
I+S type
SAINV
SAAMG

These preconditionings include Point Jacobi, Incomplete LU decomposition, etc., which are well-known; the I+S type, which is an effective stationary iterative method [8]; SA-AMG (Algebraic Multi Grid based on Smoothed Aggregation) [4]; the Hybrid type, which uses an iterative method itself, like SOR etc., as preconditioning [1]; and SAINV, which approximates the inverse matrix A^{-1} based on A-orthogonalization [3]. However, of the three stationery iterative methods, only the I+S type is available, and the results of the other preconditionings are all the same as the result obtained without preconditioning. This is specificied by Lis.

A scaling function is also supported in addition to these.

Eleven different storage formats were available, as shown in Table 4.

Table 4 Available storage formats of Lis

Compressed Row Storage	(CRS)
Compressed Column Storage	(CCS)
Modified Compressed Sparse Row	(MSR)
Diagonal	(DIA)
Ellpack-Itpack generalized diagonal	(ELL)
Jagged Diagonal	(JDS)
Block Sparse Row	(BSR)
Block Sparse Column	(BSC)
Variable Block Row	(VBR)
Dense	(DNS)
Coordinate	(COO)

Various iterative methods, preconditioning, and storage formats can be easily combined.

4.3 Function of Lis

The user's program for solving linear equations (1) using Lis can be described by the following process:

- Initialization
- Generation of matrix and vector, and substitution of values
- Solving
- Finalization

5. Information retrieval system for performance information database

5.1 Information stored in database

The following information is stored in the database, for all combinations of matrices and algorithms.

- Residual history with norm of the residual vector used in the algorithm of the iterative method
- The number of iterations until the numerical solution ($\hat{x}$) was obtained
- Information on CPU time and norm of the true residual evaluated by the numerical solution:

$$r = b - A\hat{x} \quad . \tag{2}$$

5.2 Retrieving the solution performance of the algorithms

An image of the screen used in retrieving the necessary information is shown in Figure 2. Here, combinations of problems whose performance is to be retrieved are selected from the menu. In this example, the tentative URL is:

`http://mma.cs.tsukuba.ac.jp/%7Eitosho/sesna/`

Information retrieval is carried out by the following procedure.

a) Setting of display items

In the present retrieval system, three kinds of information can be displayed, namely, a convergence graph of the iterative algorithm, information about the CPU time and the iteration number until the numerical solution was obtained, and detailed information of the calculation results obtained using Lis.

b) Selection of coefficient matrix

At present, only Matrix Market matrices are used. The names of the matrices are displayed in a pull-down menu and one is selected.

c) Selection of solution algorithm (solver and preconditioning)

Two types of input columns are displayed over several lines, and various

combinations of the solver and the preconditioning are selected.

d) Retrieval

After setting the items mentioned above, the Submit button is pressed.

In the example in Figure 2, a matrix called “add32” was selected as the coefficient matrix of the linear equations, None (no preconditioning) and ILU preconditioning were selected for the BiCGStab method, and SAINV and SAAMG preconditionings were selected for the GMRES method.

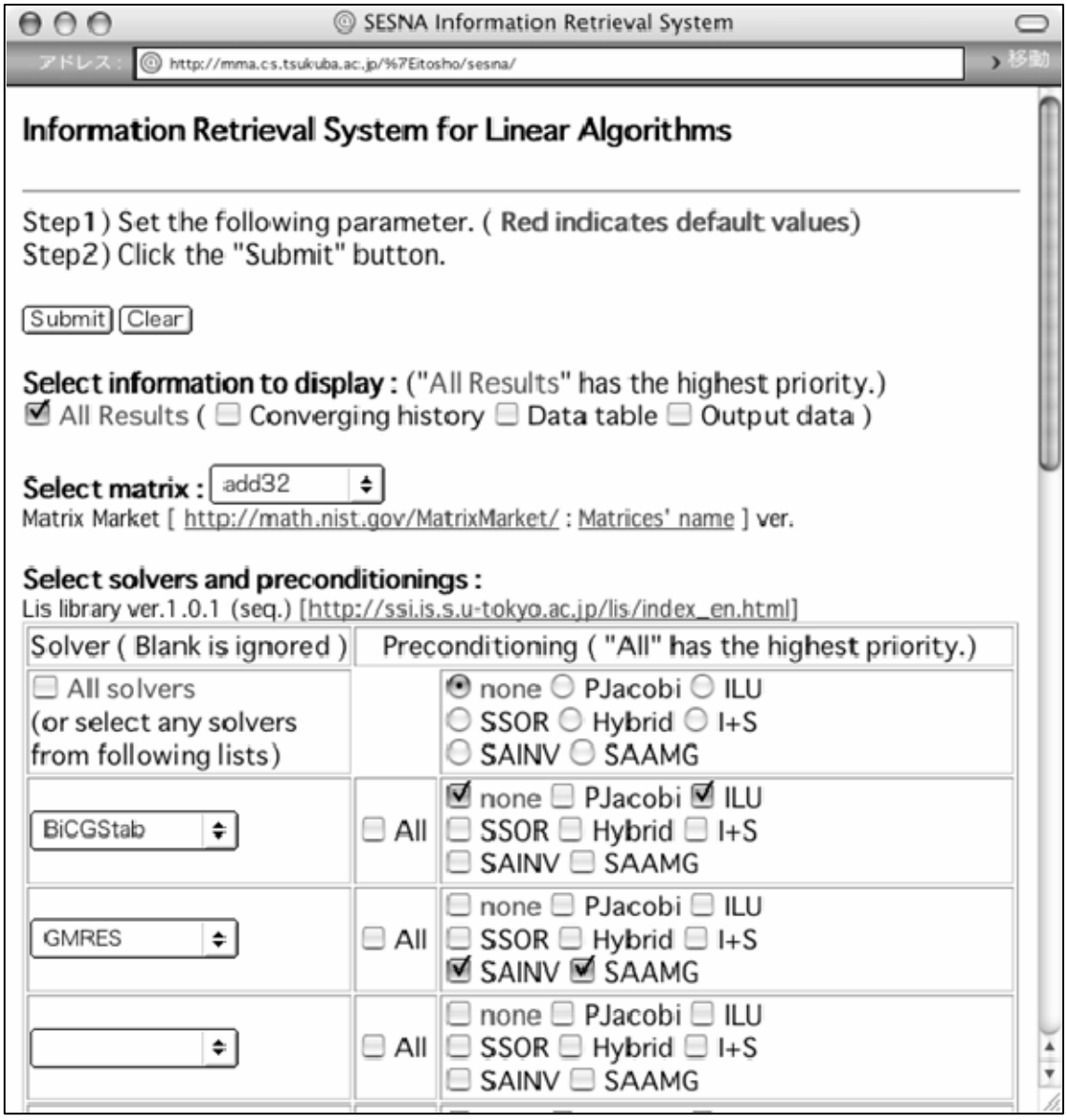


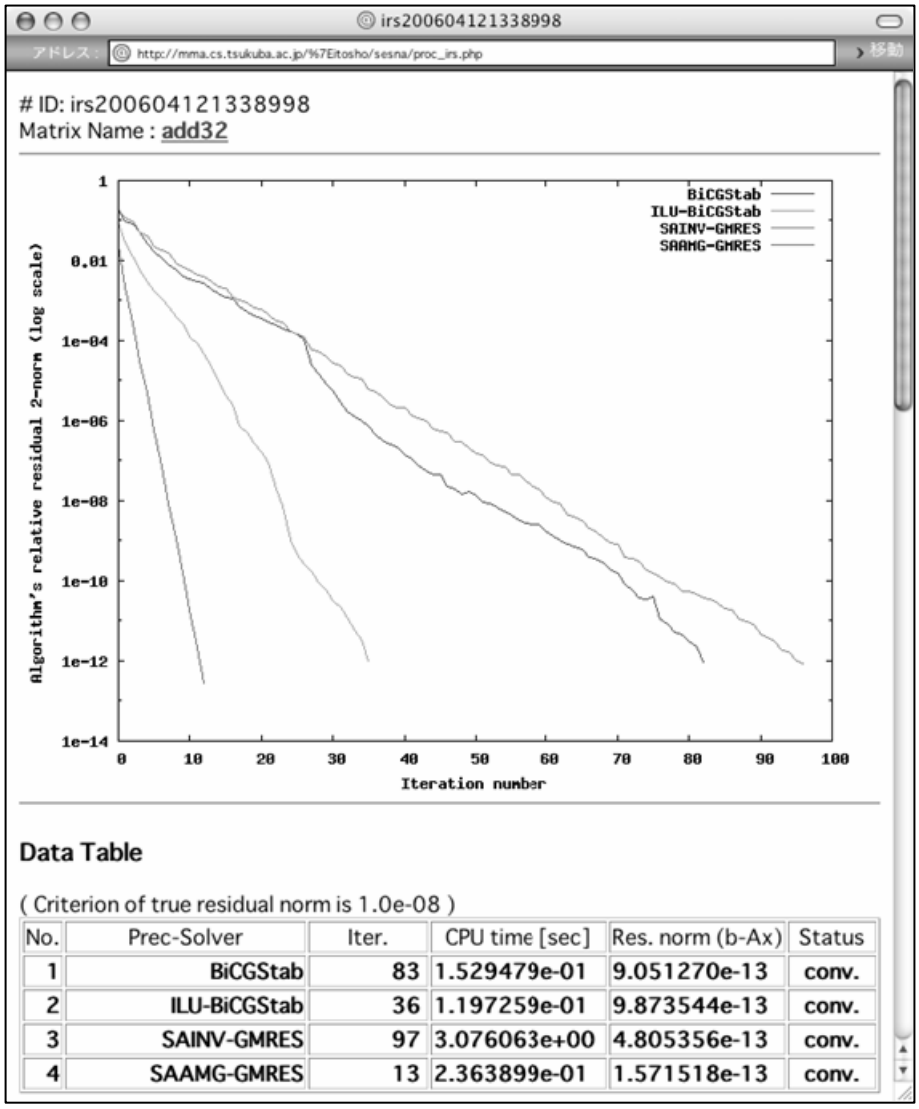
Figure 2　Interface for setting parameters for information retrieval.

5.3 Display of retrieval result

With these settings, the results of the information retrieval are shown in Figs. 3 and 4. Fig. 4 shows the screen that continues from Fig. 3 when the window is scrolled.

Figure 3 shows a graph of the convergence of the relative residual norm obtained using the algorithms selected in Fig. 2. The horizontal axis indicates the number of iterations and the vertical logarithmic axis indicates the relative residual norm. In the actual system, the graph is displayed in color and each algorithm is distinguished by a different color.

Items in the "Data Table" shown at the bottom of Fig. 3 and the top of Fig.4 are shown in Table 5.



No.	Prec-Solver	Iter.	CPU time [sec]	Res. norm (b-Ax)	Status
1	BiCGStab	83	1.529479e-01	9.051270e-13	conv.
2	ILU-BiCGStab	36	1.197259e-01	9.873544e-13	conv.
3	SAINV-GMRES	97	3.076063e+00	4.805356e-13	conv.
4	SAAMG-GMRES	13	2.363899e-01	1.571518e-13	conv.

Figure 3　Display of retrieval results (converging graph).

Table 5　Explanation of items in Data Table

No.	ID number assigned to the selected algorithm
Prec-Solver	Algorithm name (Preconditioning-Solver)
Iter.	The required number of iterations to convergence
CPU time [sec]	CPU time to convergence [second]
Res. norm	Norm of the true residual, given by eq. (2)
Status	Solution status conv.: convergence of true residual no conv.: apparent convergence of only algorithm's residual max. itr.: upper limit of iteration number reached brk. dwn.: breakdown

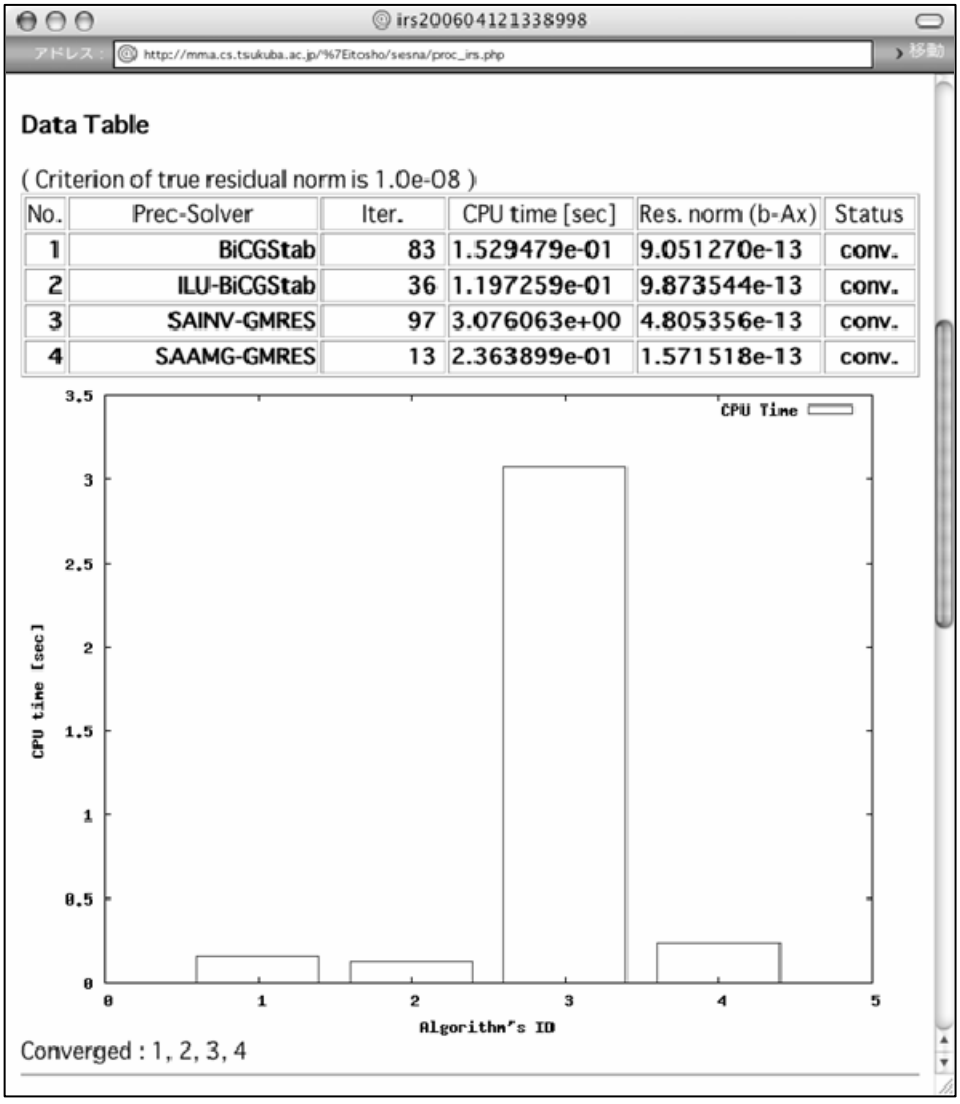


Figure 4　Display of retrieval results, continued (performance information)

The bar graph in Fig. 4 indicates the CPU time. The selected algorithm's ID number is indicated on the horizontal axis, to facilitate visual comparison of each CPU time. The text "Converged:1,2,3,4" at the bottom of this figure shows the ID numbers of converged algorithms.

5.4 Web-based algorithm performance evaluation system

The time required to obtain a solution to a test problem ranges from several seconds to several hours. Furthermore, there are many combinations of the solution problem and algorithm that reach the maximum number of iterations. However, the risk of this happening can be avoided using our system. The web server in our system accesses the required data, draws the graph, processes markup tags for the browser display, and so on in several seconds. This allows substantial labor and time savings without the user having to repeatedly execute jobs using different algorithms, even if the problem is evaluated only to understand the performance and properties of the algorithms.

There are other systems for presenting the efficiency of solution algorithms of linear equations via web, such as TIS of the ITBL portal [5]. In that system, some algorithms are selected to solve linear equations, and the user's problem (matrix, etc.) to be solved is sent to the server. TIS has an advantage in that it gives information on the user's problem itself, but on the other hand, it takes some time to provide a result and repeats several practice requests to compare the performance among algorithms.

Users should therefore use our system or TIS according to the purpose of comparing the performance of the solution algorithms.

6. Summary and prospects

Among the data that our system displays, the iteration number and the convergence history graph show the general performance and characteristics of the candidate solution algorithms, and the CPU time shows the actual performance when implemented as a method in the Lis library.

The database contains accumulated data provided from the performance evaluation of algorithms, and because of its usefulness, this database is made public; such useful information has not been provided in the past. Information on any bugs in the Lis library noticed in this system is fed back to the Lis developers, and the database continues to be updated. The data currently being made public seems to be enough to evaluate the performance of algorithms.

At present, calculation results obtained using only eight kinds of preconditioning

and twelve kinds of solver are provided. In future, the system will also display a rank of the CPU time for all combinations of algorithms.

In future development of this work, we plan to further analyze the relationship between solution algorithms and linear equations using the acquired data. Moreover, we also plan to perform a comparison using other libraries and matrices other than those of the Matrix Market.

Acknowledgments

The first author (SI) wishes to express his gratitude to professor Michael Ng of Hong Kong Baptist University, who invited the author to be one of the organizers of the Hong Kong - Japan Scientific Computing Minisymposium at The 2nd International Conference on Scientific Computing and Partial Differential Equations (SCPDE05) & The First East Asia SIAM Symposium, HKBU, Hong Kong, Dec., 2005, at the design stage of this research, and who gave him an opportunity to present his work.

References

[1] Kuniyoshi Abe, Shao-Liang Zhang, Hidehiko Hasegawa and Ryutaro Himeno, A SOR-base variable preconditioned GCR method, Trans. Japan Soc. Indust. Appl. Math., 11(4), pp. 157-170, 2001 (in Japanese).

[2] Richard Barrett, et. al., Templates for the solution of linear systems: Building Blocks for Iterative Methods, SIAM, 1994.

[3] Robert Bridson and Wei-Pai Tang, Refining an approximate inverse. J. Comput. Appl. Math., Vol. 123, pp. 293-306, 2000.

[4] Akihiro Fujii, Akira Nishida and Yoshio Oyanagi, A parallel AMG algorithm based on domain decomposition, IPSJ trans. ACS, 44 SIG6(ACS1), pp. 1-8, 2003 (in Japanese).

[5] Yoshinari Fukui and Hidehiko Hasegawa, Test of Iterative Solvers on ITBL, In proceedings of the 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2005), pp. 422-425, 2005.

[6] Shoji Itoh and Hidehiko Hasegawa, A Plan to Develop an Evaluating System for Numerical Algorithms, The 2nd International Conference on Scientific Computing and Partial Differential Equations & The First East Asia SIAM Symposium, HKBU, Hong-Kong, Dec., 2005.

[7] Shoji Itoh, A plan to develop an evaluating system on numerical algorithms' properties, High Performance Computing Symposium 2006, Tokyo, Jan., 2006 (in Japanese).

[8] Toshiyuki Kohno, Hisashi Kotakemori and Hiroshi Niki, Improving the Modified

Gauss-Seidel Method for Z-matrices. Linear Algebra and its Applications, Vol. 267, pp. 113-123, 1997.
[9] Hisashi Kotakemori, Hidehiko Hasegawa, Tamito Kajiyama, Akira Nukada, Reiji Suda, and Akira Nishida, Performance evaluation of parallel sparse matrix-vector products on SGI Altix3700, Proc. of the first international workshop on OpenMP (IWOMP2005), June 2005, to appear.
[10] Hisashi Kotakemori, Hidehiko Hasegawa and Akira Nishida, Performance Evaluation of a Parallel Iterative Method Library using OpenMP, In proceedings of the 8th International Conference on High Performance Computing in Asia Pacific Region (HPC Asia 2005), pp.432-436, 2005.
[11] Lis: Iterative method library, http://ssi.is.s.u-tokyo.ac.jp/index_en.html
[12] Matrix Market, http://math.nist.gov/MatrixMarket/
[13] University of Florida Sparse Matrix Collection, http://www.cise.ufl.edu/research/sparse/matrices/