XIII International Conference on Computational Plasticity. Fundamentals and Applications
COMPLAS XIII
E. Oñate, D.R.J. Owen, D. Peric & M. Chiumenti (Eds)

PARALLELIZATION OF BICGMISR METHOD WITH CACHE-CACHE ELEMENTS PRECONDITIONING

K. Iwasato* **and S. Fujino†**

* Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1, Hakozaki, Higashi-ku, Fukuoka, 812-8581 Japan
e-mail: onigili9@gmail.com

†Research Institute for Information Technology, Kyushu University
6-10-1, Hakozaki, Higashi-ku, Fukuoka, 812-8581 Japan
e-mail: fujino@cc.kyushu-u.ac.jp

Key words: Parallel Computing, Synchronization, BiCGMisR method

Abstract. We consider Krylov subspace methods for solving a linear system of equations on parallel computer with distributed memory. For speed-up of parallel computation, it is necessary to shorten the communication time among processors. However, in parallelized Krylov subspace methods, global synchronization points for inner products cause increment of communication time. Thus, we created the strategy for reduction of synchronization points of parallel Krylov subspace methods. We transform the computation of parameter β_k to reduce the number of synchronization points of various Krylov subspace methods per one iteration. In this paper, we apply this strategy to three-term recurrence and propose parallel BiCGMisR method as the effective solver suited to parallel computer with distributed memory. Furthermore, through several numerical experiments, we make clear that parallel BiCGMisR method outperforms other methods from the viewpoints of both elapsed time and speed-up on parallel computer with distributed memory.

1 Introduction

We consider Krylov subspace methods for solving a linear system of equations $A\boldsymbol{x} = \boldsymbol{b}$ where $A \in R^{N\times N}$ is a given non-symmetric matrix. Vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ are a solution vector and a right-hand side vector, respectively. For speed-up of parallel computation, it is necessary to shorten the communication time between processors. However, global synchronization for inner products causes great increment of communication time in parallelized Krylov subspace methods.

Among many Krylov subspace methods, product-type of Krylov subspace methods, e.g., BiCGStab and GPBiCG[5] methods are often used for the purpose of solution for

realistic problems. GPBiCG method, however, needs three times for synchronizations per one iteration.

BiCGSafe (with safety convergence) method[2] using the strategy of associate residual was proposed in 2005. This strategy leads to reduce the number of synchronization from three to two times per one iteration. Furthermore, the variants of GPBiCG method[1] improved GPBiCG itself by using the three-term recurrence which is similar to the one for the Lanczos polynomials. These variants of GPBiCG method reduced the number of synchronization points to two times from three times per one iteration.

Then, we propose BiCGMisR(**Mi**nimization with **s**afety **R**esidual) method as the effective solver suited to parallel computer with distributed memory. BiCGMisR method applies the strategy of minimization using safety residual and three-term recurrence as stabilized polynomial. BiCGMisR method reduced the number of synchronization points to single one from two times per one iteration. Therefore, we can expect that BiCGMisR method outperforms other methods in parallel computing.

In this paper, we evaluate performance of parallel computing of BiCGMisR method. Through several numerical experiments, we make clear that BiCGMisR method outperforms other methods from the viewpoints of both elapsed time and speed-up on parallel computer with distributed memory.

This paper is organized as follows: In section 2, we describe a basic idea of GPBiCG method and its variants. In section 3,4, we describe also a basic idea of BiCGSafe method, and present an algorithm of BiCGMisR method. In section 5, we show diagram of Flat MPI parallelization. In section 6, results of several parallelized Krylov subspace methods will be shown, and it will be made clear that the parallelized BiCGMisR method outperforms other methods on parallel computer with distributed memory. Finally, in section 7, we have concluding remarks.

2 GPBiCG method and its variants

2.1 Basic idea of GPBiCG

The Lanczos polynomial $R_k(\lambda)$ and the auxiliary polynomial $P_k(\lambda)$ satisfy the following two-term recurrence relation as

$$R_0(\lambda) = 1, \quad P_0(\lambda) = 1, \tag{1}$$
$$R_k(\lambda) = R_{k-1}(\lambda) - \alpha_{k-1}\lambda P_{k-1}(\lambda), \tag{2}$$
$$P_k(\lambda) = R_k(\lambda) + \beta_{k-1}P_{k-1}(\lambda), \quad k = 1, 2, \cdots, \tag{3}$$

according to the notation used in Ref.[1]. Here, λ means an eigenvalue of a matrix. We introduce the two parameters ζ_k and η_k. The stabilized polynomial $H_k(\lambda)$ satisfies the three-term recurrence relation as

$$H_0(\lambda) = 1, \quad H_1(\lambda) = (1 - \zeta_0\lambda)H_0(\lambda), \tag{4}$$
$$H_{k+1}(\lambda) = (1 + \eta_k - \zeta_k\lambda)H_k(\lambda) - \eta_k H_{k-1}(\lambda), \quad k = 1, 2, \cdots. \tag{5}$$

The polynomial $H_{k+1}(\lambda)$ which is produced by eqns.(4)-(5) satisfies $H_{k+1}(0) = 1$ and the relation as $H_{k+1}(0) - H_k(0) = 0$ for all k. We set an auxiliary polynomial $G_k(\lambda)$ as

$$G_k(\lambda) \quad = \quad -(H_{k+1}(\lambda) - H_k(\lambda))/\lambda. \tag{6}$$

As a result, we have the following coupled two-term recurrence of the form as

$$H_0(\lambda) = 1, \quad G_0(\lambda) = \zeta_0, \tag{7}$$
$$H_k(\lambda) = H_{k-1}(\lambda) - \lambda G_{k-1}(\lambda), \tag{8}$$
$$G_k(\lambda) = \zeta_k H_k(\lambda) + \eta_k G_{k-1}(\lambda), \quad k = 1, 2, \cdots. \tag{9}$$

We introduce the residual vector $\boldsymbol{r}_k$ as $\boldsymbol{r}_k := H_k(\lambda)R_k(\lambda)\boldsymbol{r}_0$. Here, the vector $\boldsymbol{r}_0$ is the initial residual vector. The recurrence parameters ζ_k, η_k are decided from local minimization of the residual vector of 2-norm $||\boldsymbol{r}_{k+1}||_2$. The Lanczos parameters α_k, β_k are decided from the following equations.

$$\alpha_k = (\tilde{\boldsymbol{r}}_0,\ \boldsymbol{r}_k)/(\tilde{\boldsymbol{r}}_0,\ A\boldsymbol{p}_k), \tag{10}$$
$$\beta_k = -\frac{(\tilde{\boldsymbol{r}}_0,\ A\boldsymbol{t}_k)}{(\tilde{\boldsymbol{r}}_0,\ A\boldsymbol{p}_k)} = \frac{\alpha_k(\tilde{\boldsymbol{r}}_0,\ \boldsymbol{r}_{k+1})}{\zeta_k(\tilde{\boldsymbol{r}}_0,\ \boldsymbol{r}_k)}. \tag{11}$$

2.2 Variants of GPBiCG method

We indicate the variant[1] of GPBiCG method. We apply eqns.(4)-(5) as stabilized polynomials. Here, we can compute the Lanczos coefficients α_k and β_k according to the same computation shown as eqns.(10)-(11). The recurrence parameters ζ_k and η_k are decided from local minimization of the residual vector of 2-norm.

3 Basic idea of BiCGMisR method

We apply eqns.(1)-(3), (7)-(9) to devise of BiCGMisR method. The coefficients α_k, β_k can be gained according to the same computation shown as eqns.(10)-(11). It is known that two parameters ζ_k and η_k are determined by solving the two-dimensional local minimization of the norm of product-type residual $\boldsymbol{r}_{k+1}$ in GPBiCG method. The residual vector $\boldsymbol{r}_{k+1}$, however, does not involve both parameters ζ_k, η_k in the update of residual vector. Appearance of another idea needs for overcoming this issue. Therefore, the key idea is to focus on a safety residual vector defined by the next recurrence. The safety residual vector $\boldsymbol{s\_r}_k$ is defined as below.

$$\boldsymbol{s\_r}_k \quad := \quad \boldsymbol{r}_k - \zeta_k A\boldsymbol{r}_k - \eta_k \boldsymbol{y}_k. \tag{12}$$

Note that the recurrence (12) is not computed in the iterative loop as it is. We utilize the recurrence (12) only for the recurrence parameters ζ_k and η_k. We call this idea strategy of minimization of a safety residual.

4 Algorithm of BiCGMisR method

We derive BiCGMisR method from eqns. (1)-(3) and (4)-(5). The coefficient β_k can be gained as

$$\beta_k = \frac{(A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)} = \frac{(A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k) - \alpha_k (A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}. \tag{13}$$

The coefficient α_k can be computed as well as eqn.(10). Then, we show the algorithm of BiCGMisR method as below.

Algorithm 1: BiCGMisR method

Let $\boldsymbol{x}_0$ be an initial guess, Compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
Choose $\tilde{\boldsymbol{r}}_0$ such that $(\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_0) \neq 0$
Compute $A\boldsymbol{r}_0,\ A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0,\ \boldsymbol{p}_0 = \boldsymbol{r}_0,\ A\boldsymbol{p}_0 = A\boldsymbol{r}_0$,
$\boldsymbol{u}_{-1} = \boldsymbol{q}_{-1} = \boldsymbol{v}_{-1} = \boldsymbol{0}$
for $k = 0, 1, \ldots$ **do**
  $\boldsymbol{y}_k = \boldsymbol{u}_{k-1} - \boldsymbol{r}_k$
  if $||\boldsymbol{r}_k||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**
  $\alpha_k = (\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k)/(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)$
  $\beta_k = \dfrac{(A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0, \boldsymbol{r}_k) - \alpha_k (A^{\mathrm{T}}\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}{(\tilde{\boldsymbol{r}}_0, A\boldsymbol{p}_k)}$
  $\zeta_k = \dfrac{(\boldsymbol{y}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$
  $\eta_k = \dfrac{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{r}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(A\boldsymbol{r}_k, \boldsymbol{r}_k)}{(A\boldsymbol{r}_k, A\boldsymbol{r}_k)(\boldsymbol{y}_k, \boldsymbol{y}_k) - (A\boldsymbol{r}_k, \boldsymbol{y}_k)(\boldsymbol{y}_k, A\boldsymbol{r}_k)}$
  (**if** $k = 0$ **then** $\zeta_k = (A\boldsymbol{r}_k, \boldsymbol{r}_k)/(A\boldsymbol{r}_k, A\boldsymbol{r}_k),\ \eta_k = 0$)
  $\boldsymbol{s}\_\boldsymbol{r}_k = (1 + \eta_k)\boldsymbol{r}_k - \zeta_k A\boldsymbol{r}_k - \eta_k \boldsymbol{u}_{k-1}$
  $\boldsymbol{t}_k = (1 + \eta_k)\boldsymbol{x}_k + \zeta_k \boldsymbol{r}_k - \eta_k \boldsymbol{v}_{k-1}$
  $\boldsymbol{w}_k = (1 + \eta_k)\boldsymbol{p}_k - \zeta_k A\boldsymbol{p}_k - \eta_k \boldsymbol{q}_{k-1}$
  Compute $A\boldsymbol{w}_k$
  $\boldsymbol{u}_k = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$
  $\boldsymbol{v}_k = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{s}\_\boldsymbol{r}_k - \alpha_k A\boldsymbol{w}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{t}_k + \alpha_k \boldsymbol{w}_k$
  Compute $A\boldsymbol{r}_{k+1}$
  $\boldsymbol{q}_k = \boldsymbol{u}_k - \beta_k \boldsymbol{p}_k$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \beta_k \boldsymbol{w}_k$
  $A\boldsymbol{p}_{k+1} = A\boldsymbol{r}_{k+1} - \beta_k A\boldsymbol{w}_k$
end do

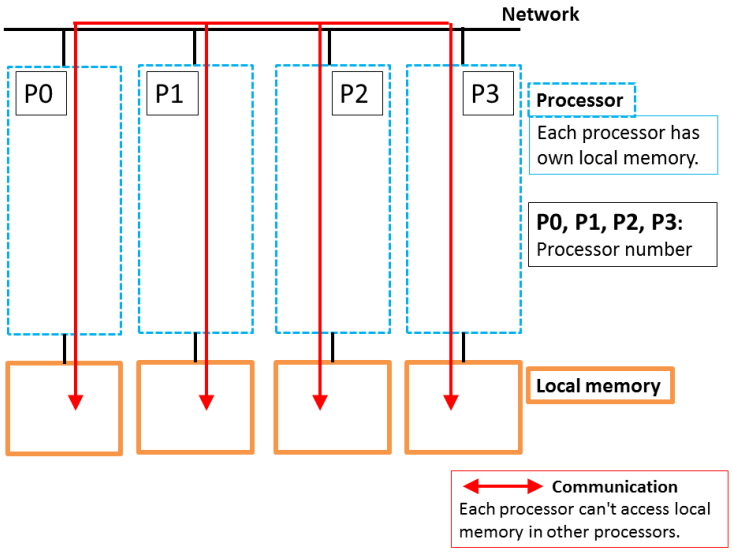

Figure 1: Diagram of Flat MPI parallelization

5 Flat MPI parallelization

In this section, we describe briefly Flat MPI (Message Passing Interface) parallelization. Fig.1 shows diagram of Flat MPI parallelization. The blue blocks in dotted line mean processors of P0, P1, P2 and P3. The orange blocks in thick line mean local memory of processors. Flat MPI parallelization is well known as parallelization on distributed memory architectures. Each processor has own local memory and CPU. However, each processor can't access local memory in other processors. Therefore, it needs communication to refer data on local memory in other processors.

6 Numerical Examinations

6.1 Parallel computational environment and conditions

All computations were done in double precision floating point arithmetic of Fortran90, and performed on Fujitsu PRIMERGY CX400 (CPU: Intel Xeon E5-2680, main memory: 128Gbytes, OS: Red Hat Linux Enterprise, total nodes: 1476 nodes, cores: 16cores/1node). Intel compiler optimum option "-O3" were used. Process parallelization was done by MPI library. Stopping criterion of iterative methods is less than 10^{-8} of the relative residual 2-norm $||\boldsymbol{r}_{k+1}||_2/||\boldsymbol{b} - A\boldsymbol{x}_0||_2$. In all cases the iteration was started with the initial guess solution $\boldsymbol{x}_0 = (0, 0, \ldots, 0)^{\mathrm{T}}$. We adopted uniform random numbers as the initial shadow residual $\tilde{\boldsymbol{r}}_0$. Measurement of the elapsed time was done by system function of gettimeofday. All test matrices were normalized with diagonal scaling. Maximum iteration was

fixed as 100,000. Number of process varied as 1, 16, 32, 64, 128 and 256. Measurements of the elapsed time per each matrix were done at five times.

6.2 Numerical results

Test matrices are derived from Florida Sparse Matrix Collection[4]. Characteristics of test matrices for parallelization are exhibited in Table 1. In Table 1, "nnz" means number of nonzero entries, and "ave. nnz" means average number of nonzero entries per one row.

Table 1: Characteristics of test matrices.

matrix	dimension	nnz	ave. nnz
Freescale1	3,428,755	18,920,347	5.5
air-cfl5	1,536,000	19,435,428	12.7
atmosmodd	1,270,432	8,814,880	6.9
tmt_unsym	917,825	4,584,801	5.0
epb3	84,617	463,625	5.5

We present parallel performance of Hybrid-version GPBiCG, GPBiCG_v1, GPBiCG_v2, BiCGSafe and BiCGMisR method for matrix atmosmodd in Table 2. "TRR(True Relative Residual)" for the approximated solutions $\boldsymbol{x}_{k+1}$ means $\log_{10}(||\boldsymbol{b} - A\boldsymbol{x}_{k+1}||/||\boldsymbol{b} - A\boldsymbol{x}_0||)$. The bold figures means the least time.

- "np", "Mv", "Mv-t." and "itr-t." mean the number of processors, the number of Matrix-vector multiplications, elapsed time of Matrix-vector multiplication. and elapsed time of iteration part except for Matrix-vector multiplication, respectively.
- "ratio", "sp-up ratio1" and "sp-up ratio2" indicate the ratio of "itr-t." to that of GPBiCG, speed-up ratio based on total elapsed time and speed-up ratio based on average elapsed time per one iteration, respectively.
- "percentage" indicates percentage [%] of "itr-t." time in total elapsed time.

From Table 2, we see that the speed-up based on average elapsed time of BiCGMisR method shows around 124. times. Furthermore, BiCGMisR method converged the fastest among several methods. BiCGMisR method outperforms other iterative methods from the viewpoints of both the speed-up and average elapsed time.

In Fig.2, we present comparison of speed-up ratio of several methods for matrix atmosmodd in view of "sp-up ratio2" as shown in Table 2. From Fig.2, we see that BiCGMisR method has the highest speed-up ratio at 256 processors for matrix atmosmodd as well as the highest at single processor.

Table 2: Parallel performance of Flat-version of several Krylov subspace methods for matrix atmosmodd.

method	np	Mv	Mv-t. [s.]	itr-t. [s.]	ratio	tot-t. [s.]	sp-up ratio1	ave-t. [ms.]	sp-up ratio2	percentage[%]	TRR
GPBiCG	1	506	-	-	-	17.915	1.00	35.405	1.00	-	-8.11
	8	470	1.225	0.581	1.00	1.807	9.91	3.845	9.21	32.17	-8.15
	16	500	0.675	0.327	1.00	1.002	17.88	2.004	17.67	32.66	-8.10
	32	480	0.381	0.149	1.00	0.529	33.87	1.102	32.13	28.09	-8.15
	64	494	0.288	0.207	1.00	0.495	36.19	1.002	35.33	41.78	-8.05
	128	502	0.148	0.120	1.00	0.267	67.10	0.532	66.57	44.69	-8.18
	256	496	0.072	0.076	1.00	0.148	121.05	0.298	118.66	51.34	-8.01
GPBiCG_v1	1	506	-	-	-	16.894	1.00	33.387	1.00	-	-8.08
	8	500	1.249	0.748	1.29	1.997	8.46	3.994	8.36	37.48	-8.34
	16	496	0.653	0.375	1.15	1.029	16.42	2.075	16.09	36.49	-8.15
	32	478	0.371	0.172	1.15	0.543	31.11	1.136	29.39	31.61	-8.02
	64	494	0.283	0.220	1.06	0.502	33.65	1.016	32.86	43.71	-8.03
	128	496	0.167	0.100	0.83	0.266	63.51	0.536	62.26	37.40	-8.04
	256	500	0.142	0.052	0.68	0.194	87.08	0.388	86.05	26.80	-8.35
GPBiCG_v2	1	492	-	-	-	16.844	1.00	34.236	1.00	-	-8.02
	8	502	1.218	0.773	1.33	1.991	8.46	3.966	8.63	38.80	-8.06
	16	524	0.682	0.393	1.20	1.076	15.65	2.053	16.67	36.57	-8.16
	32	500	0.381	0.176	1.18	0.556	30.29	1.112	30.79	31.61	-8.18
	64	504	0.289	0.216	1.04	0.505	33.35	1.002	34.17	42.74	-8.32
	128	470	0.155	0.099	0.83	0.255	66.05	0.543	63.10	38.87	-8.13
	256	504	0.140	0.054	0.71	0.194	86.82	0.385	88.94	27.95	-8.29
BiCGSafe	1	500	-	-	-	14.588	1.00	29.176	1.00	-	-8.37
	8	476	1.245	0.411	0.71	1.656	8.81	3.479	8.39	24.84	-8.05
	16	500	0.691	0.241	0.74	0.932	15.65	1.864	15.65	25.85	-8.48
	32	528	0.422	0.114	0.77	0.537	27.17	1.017	28.69	21.30	-8.28
	64	498	0.290	0.179	0.86	0.470	31.04	0.944	30.91	38.18	-8.07
	128	494	0.143	0.096	0.80	0.240	60.78	0.486	60.05	40.14	-8.05
	256	504	0.074	0.054	0.71	0.128	113.97	0.254	114.88	42.38	-8.08
BiCGMisR	1	478	-	-	-	**14.241**	1.00	29.793	1.00	-	-8.05
	8	480	1.571	0.323	0.56	1.895	7.52	3.948	7.55	17.07	-8.05
	16	494	0.804	0.178	0.54	0.982	14.50	1.988	14.99	18.11	-8.01
	32	518	0.479	0.084	0.56	0.564	25.25	1.089	27.36	15.00	-8.05
	64	506	0.337	0.136	0.66	0.473	30.11	0.935	31.87	28.74	-8.18
	128	496	0.167	0.068	0.57	0.235	60.60	0.474	62.88	29.06	-8.07
	256	500	0.078	0.041	0.54	**0.120**	118.68	0.240	**124.14**	34.63	-8.22

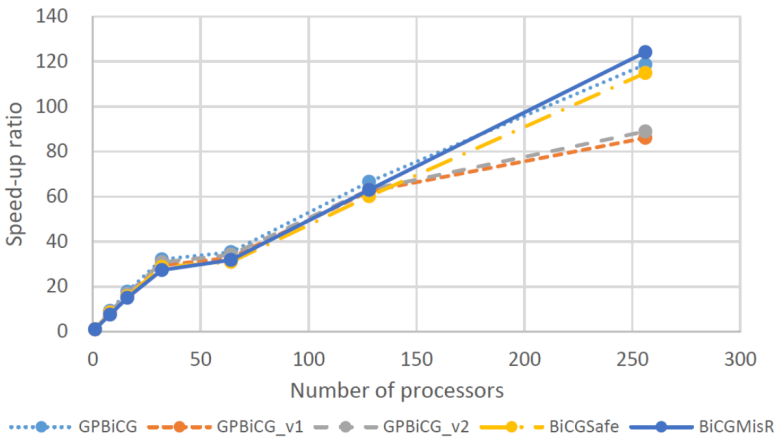

Figure 2: Comparison of speed-up ratio of several methods for matrix atmosmodd.

7 Conclusions and Future work

In this paper, we proposed parallel BiCGMisR method based on two strategies. One of them adopts three-term recurrences as stabilized polynomials, and another of them adopts a minimization technique of a safety residual vector of 2-norm for computing parameters ζ_k and η_k. Through numerical experiments, we demonstrated that BiCGMisR method outperforms other methods in view of computational times in parallel computing. Near future work is that we will derive parallel BiCGMisR method with the CCE (Cache-Cache Elements)[3] preconditioning and examine performance of parallel BiCGMisR with the CCE preconditioning.

REFERENCES

[1] Abe, K., Sleijpen, G.L.G.: Solving linear equations with a stabilized GPBiCG mehtod, Appl. Numer. Math., doi:10.1016/j.apnum.2011.06.010, 2011.

[2] Fujino, S., Fujiwara, M., Yoshida, M.: A proposal of preconditioned BiCGSafe method with safe convergence, Proc. of The 17th IMACSWorld Congress on Scientific Computation, Appl. Math. Simul., CD-ROM, Paris, France, 2005.

[3] Fujino, S., Itou, C., Iwasato, K., Cache-Cache balancing technique for Eisenstat type of preconditioning for parallelism, PMAA14, Universita della Svizzera italiana, Lugano, Switzerland, July 2-4, 2014.

[4] Spares Matrix Collection: http://www.cise.ufl.edu/research/sparse/matrices/index.html

[5] Zhang, S.: GPBiCG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput., Vol.18, pp.537-551, 1997.