社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report 2008−HPC−118 (7)
2008／12／16

A determination of coefficients of GPBiCG-AR method by reconsideration on inner product

Moethuthu Seiji Fujino‡ Yusuke Onoue
(Graduate School of Information Science and Electrical Engineering, Kyushu University
‡Research Institute for Information Technology, Kyushu University)

Abstract: Product-type iterative methods, e.g., GPBi-CG, GPBiCG_AR methods utilize three-term recurrences in the algorithm. Moreover, coefficients ζ_n, η_n included in the accelerated polynomials $H_n(\lambda), G_n(\lambda)$ are computed for every iteration step by means of local minimization of 2-norm of residual $\boldsymbol{r}_{n+1}$ and associate residual $a\_\boldsymbol{r}_{n+1}$. In this article, we evaluate heuristic variants of GPBiCG_AR method without and with preconditioning by means of improvement of determination of coefficients ζ_n, η_n.

1 Introduction

Generalized Product Bi-Conjugate Gradient (abbreviated as GPBi-CG) method [8] is an attractive iterative method for the solution of a linear system of equations with nonsymmetric coefficient matrix. However, the popularity of GPBi-CG method has diminished over time except for the context of limited field of analysis because of instability of convergence rate. Therefore some versions of GPBi-CG method which have stability of convergence compared with the original GPBi-CG method have been proposed.

We proposed a safety variant (abbreviated as BiCGSafe) of Generalized Product type Bi-CG method from the viewpoint of reconstruction of the residual polynomial and determination of two acceleration parameters ζ_n and η_n [2]. It embodied that a particular strategy for remedying the instability of convergence, acceleration parameters are decided from minimization of the associate residual of 2-norm [6]. However, we could not reveal the origin of instability of GPBi-CG method because of reconstruction of the algorithm. Though both convergence rate and stability of BiCGSafe method were improved, instability itself of GPBi-CG method could not corresponds directly to its algorithm.

As a result, we proposed the original GP-BiCG_AR (GPBiCG with Associate Residual) method [3] in 2007, and could verify robustness of GPBiCG_AR method. Moreover, we proposed simple variant of GPBiCG_AR2 method to improve convergence rate. That is, in simple variant of GPBiCG_AR2 method, three-term recurrences are adopted for odd iteration steps and two-term recurrences are adopted for even iteration steps, one after the other. This remedy makes enhancement of convergence rate of GPBiCG_AR method at fairly satisfied degree. However, sometimes this remedy has a limitation of enhancement to a certain extent.

This paper is organized as follows. In section 2 we briefly review GP-BiCG and GPBiCG_AR methods. In section 3 we consider on coefficients of GPBi-CG like methods. In particular, we propose simple and heuristic variants of GP-BiCG_AR method by means of improvement of determination of coefficients ζ_n, η_n. In section 4 we verify effectiveness of heuristic variant of GPBiCG_AR method without and with preconditioning through numerical experiments. In section 5 we draw some conclusions and remarks.

2 GPBiCG_AR methods

We consider iterative methods for solving a linear system of equations

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

where $A \in R^{N\times N}$ is a given nonsymmetric matrix, and $\boldsymbol{x}$, $\boldsymbol{b}$ is a solution vector and right-hand side vector, respectively. When A is a large, sparse matrix which arises from realistic problems, the efficient solution of (1) is substantially very difficult. This difficulty has led to the development of a rich variety of generalized CG type methods having varying degrees of success (see, e.g., [7]).

The biconjugate gradient (BiCG) method based of the Lanczos algorithm is a crucial example of a generalized CG method. In many cases, the Lanczos algorithm give some of the fastest solution times and stability of convergence among all generalized CG methods. The Lanczos algorithm, however, is known to break down in some cases. In practice, the occurrence of breakdown can cause failure to irregularly converge to the solution of (1). The fact that the Lanczos algorithms perform well in some cases but fail in others heightens the need for further insight and development of the Lanczos type iterative methods. We note that the basic recurrence relations between Lanczos polynomials $R_n(\lambda)$ and $P_n(\lambda)$ hold as follows:

$$R_0(\lambda) = 1, \quad P_0(\lambda) = 1, \tag{2}$$
$$R_{n+1}(\lambda) = R_n(\lambda) - \alpha_n \lambda P_n(\lambda), \tag{3}$$
$$P_{n+1}(\lambda) = R_{n+1}(\lambda) + \beta_n P_n(\lambda), \quad n = 1, 2, .. \tag{4}$$

Then we can introduce the three-term recurrence relations for Lanczos polynomials $R_n(\lambda)$ only by eliminating $P_n(\lambda)$ from (2) and (4) as follows:

$$R_0(\lambda) = 1, \quad R_1(\lambda) = (1 - \alpha_0 \lambda) R_0(\lambda) \tag{5}$$
$$R_{n+1}(\lambda) = (1 + \frac{\beta_{n-1}}{\alpha_{n-1}} \alpha_n - \alpha_n \lambda) R_n(\lambda) \tag{6}$$
$$- \frac{\beta_{n-1}}{\alpha_{n-1}} \alpha_n R_{n-1}(\lambda), \quad n = 1, 2, .. \tag{7}$$

GPBi-CG method[8] was discovered that an often excellent convergence property can be gained by choosing for acceleration polynomials $H_n(\lambda)$ that are built up in the three-term recurrence form as polynomial $R_n(\lambda)$ in (5) and (7) by adding suitable undetermined parameters ζ_n and η_n as follows:

$$H_0(\lambda) = 1, \quad H_1(\lambda) = (1 - \zeta_0 \lambda) H_0(\lambda), \tag{8}$$
$$H_{n+1}(\lambda) = (1 + \eta_n - \zeta_n \lambda) H_n(\lambda) - \eta_n H_{n-1}(\lambda), \quad n = 1, 2, \ldots. \tag{9}$$

The polynomials $H_n(\lambda)$ satisfy $H_n(0) = 1$ and the relation as $H_{n+1}(0) - H_n(0) = 0$ for all n. Here we introduce an auxiliary polynomials $G_n(\lambda)$ as

$$G_n(\lambda) := \frac{H_n(\lambda) - H_{n+1}(\lambda)}{\zeta_n \lambda}. \tag{10}$$

By reconstruction of (7) using the acceleration polynomials $H_n(\lambda)$ and $G_n(\lambda)$, we have the following coupled two-term recursion of the form as

$$H_0(\lambda) = 1, \quad G_0(\lambda) = \zeta_0, \tag{11}$$
$$H_n(\lambda) = H_{n-1}(\lambda) - \lambda G_{n-1}(\lambda), \tag{12}$$
$$G_n(\lambda) = \zeta_n H_n(\lambda) + \eta_n G_{n-1}(\lambda), \tag{13}$$
$$n = 1, 2, \ldots.$$

Using these acceleration polynomials $H_n(\lambda)$ and $G_n(\lambda)$, his discover led to the generalized product-type methods based on Bi-CG method for solving the linear system with nonsymmetric coefficient matrix. He refered as GPBi-CG method [8]. However, the original Lanczos algorithm is also known to break down or nearly break down in some cases. In practice, the occurrence of a break down cause failure to converge to the solution of linear equations, and the increase of the iterations introduce numerical error into the approximate solution. Therefore, convergence of the generalized product-type methods is affected. Comparatively little is known about the theoretical properties of the generalized product-type methods. The fact that the generalized product-type methods perform very well in some cases but fail in others motivates the need for further insight into the construction of polynomials for the product-type residual $H_{n+1}(\lambda)R_{n+1}(\lambda)$.

In a usual approach, acceleration parameters are decided from local minimization of the residual vector of 2-norm $||\boldsymbol{r}_{n+1}(:= H_{n+1}(\lambda)R_{n+1}(\lambda))||_2$, where $R_{n+1}(\lambda)$ denotes the residual polynomial of Lanczos algorithm and $H_{n+1}(\lambda)$ denotes the acceleration polynomial for convergence. Instead, it embodies that a particular strategy for remedying the instability of convergence. That is, the algorithm of GPBiCG_AR method based on local minimization of associate residual $\boldsymbol{a\_r}_n(:= H_{n+1}(\lambda)R_n(\lambda))$ with two parameters ζ_n and η_n is written as follows:

$$\boldsymbol{a\_r}_n = \boldsymbol{r}_n - \eta_n A\boldsymbol{z}_{n-1} - \zeta_n A\boldsymbol{r}_n. \tag{14}$$

Here $\boldsymbol{r}_n$ is the residual vector of the algorithm. Matrix-vector multiplication of $A\boldsymbol{u}_n$ and $A\boldsymbol{r}_{n+1}$ are directly computed according to definition of multiplication of matrix A and vector. On the other hand, $A\boldsymbol{p}_n$ and $A\boldsymbol{z}_n$ are computed using its recurrence. In the algorithm of GPBiCG_AR method, modification parts which differ from the conventional GPBi-CG method are indicated with underlines. The description as $\underline{\text{compute } A\boldsymbol{u}_n}$ means that multiplication of matrix A and vector $\boldsymbol{u}_n$ as $A\boldsymbol{u}_n$ is done according to multiplication's definition. The algorithm of the original GPBiCG_AR method are written as follows:

```
$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$,
set $\beta_{-1} = 0$, $\underline{\text{compute } A\boldsymbol{r}_0}$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \leq \varepsilon \, ||\boldsymbol{r}_0||$ do :
begin
  $\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1})$,
  $A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}(A\boldsymbol{p}_{n-1} - A\boldsymbol{u}_{n-1})$,
  $\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}$,
  $\boldsymbol{a}_n = \boldsymbol{r}_n$, $\boldsymbol{b}_n = A\boldsymbol{z}_{n-1}$, $\boldsymbol{c}_n = A\boldsymbol{r}_n$,
  $\zeta_n = \frac{(\boldsymbol{b}_n, \boldsymbol{b}_n)(\boldsymbol{c}_n, \boldsymbol{a}_n) - (\boldsymbol{b}_n, \boldsymbol{a}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{b}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}$,
  $\eta_n = \frac{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{a}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{a}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{b}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}$,
  (if $n = 0$, then $\zeta_n = \frac{(\boldsymbol{c}_n, \boldsymbol{a}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)}$, $\eta_n = 0$)
```

```
    $u_n = \zeta_n Ap_n + \eta_n(t_{n-1} - r_n + \beta_{n-1}u_{n-1})$,
    $\underline{\text{compute } Au_n}$,
    $t_n = r_n - \alpha_n Ap_n$,
    $z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n$,
    $Az_n = \zeta_n Ar_n + \eta_n Az_{n-1} - \alpha_n Au_n$,
    $x_{n+1} = x_n + \alpha_n p_n + z_n$,
    $r_{n+1} = t_n - Az_n$,
    $\underline{\text{compute } Ar_{n+1}}$,
    $\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}$,
end
```

3 Coefficients determination

3.1 GPBi-CG

Coefficients ζ_n, η_n are computed for **every** iteration step by means of local minimization of 2-norm of residual $r_{n+1} := H_{n+1}(\lambda)R_{n+1}(\lambda)$.

$$a_n = t_n,\ b_n = y_n,\ c_n = At_n, \tag{15}$$

$$\zeta_n = \frac{(b_n, b_n)(c_n, a_n) - (b_n, a_n)(c_n, b_n)}{(c_n, c_n)(b_n, b_n) - (b_n, c_n)(c_n, b_n)}, \tag{16}$$

$$\eta_n = \frac{(c_n, c_n)(b_n, a_n) - (b_n, c_n)(c_n, a_n)}{(c_n, c_n)(b_n, b_n) - (b_n, c_n)(c_n, b_n)} \tag{17}$$

However, residual vector r_{n+1} and solution vector x_{n+1} are derived from reccurences with different root, respectively.

$$t_n = r_n - \alpha_n Ap_n, \tag{18}$$

$$r_{n+1} = t_n - \zeta_n At_n - \eta_n y_n,$$

$$(\neq t_n - Az_n) \tag{19}$$

$$x_{n+1} = x_n + \alpha_n p_n + z_n. \tag{20}$$

Because coefficients ζ_n, η_n are designed so as to be included in the recurrence of residual vector r_{n+1}. Accordingly solution vector x_{n+1} may differ from residual vector r_{n+1} during iteration steps.

3.2 GPBiCG_AR

Coefficients ζ_n, η_n are computed for **every** iteration step by means of local minimization of 2-norm of associate residual $a\_r_{n+1} (= r_n - \eta_n Az_{n-1} - \zeta_n Ar_n)$.

$$a_n = r_n,\ b_n = Az_{n-1},\ c_n = Ar_n, \tag{21}$$

$$\zeta_n = \text{eqn.}(16),\ \eta_n = \text{eqn.}(17) \tag{22}$$

On the contrary, in GPBiCG_AR, both residual vector r_{n+1} and solution vector x_{n+1} are derived from recurrence with the same root as below.

$$t_n = r_n - \alpha_n Ap_n, \tag{23}$$

$$x_{n+1} = x_n + \alpha_n p_n + z_n, \tag{24}$$

$$r_{n+1} = t_n - Az_n. \tag{25}$$

However, coefficients ζ_n, η_n are not included in the recurrence of residual vector r_{n+1} itself. Therefore, we adopt an associate residual vector $a\_r_{n+1} := H_{n+1}(\lambda)R_n(\lambda)$ for determination of ζ_n, η_n.

3.3 Variants of GPBiCG_AR2

Coefficients ζ_n, η_n of simple variant of GPBiCG_AR2 are decided as below.

```
if step is even number, then
    $\zeta_n = \text{eqn.}(16)$, $\eta_n = \text{eqn.}(17)$
else (i.e., step is odd number)
    $\zeta_n = \frac{(c_n, a_n)}{(c_n, c_n)}$, $\eta_n = 0$,
end if
```

Coefficients ζ_n, η_n of heuristic variant of GPBiCG_AR2 are decided as below. κ is given as $0 < \kappa \leq 1.0$ for heuristicly taking account of effect of term of $\eta_n Az_{n-1}$.

```
$\rho = \frac{|(a_n, c_n)|}{||a_n||\ ||c_n||}$,
if $\rho < \kappa$, then
    $\zeta_n = \text{eqn.}(16)$, $\eta_n = \text{eqn.}(17)$
else
    $\zeta_n = \frac{(c_n, a_n)}{(c_n, c_n)}$, $\eta_n = 0$,
end if
```

Criterion as $\frac{|(a_n, c_n)|}{||a_n||\ ||c_n||} = \frac{|(r_n, Ar_n)|}{||r_n||||Ar_n||} < \kappa$ based on Ref.[5], is introduced in place of odd or even number of iteration steps. Parameter κ is a scalar value included in $[0, 1]$. This variant of GPBiCG_AR2 method with alternative recurrences is refered to as a **heuristic variant of GPBiCG_AR2** method with parameter κ. The algorithm of heuristic variant of GPBiCG_AR2 method is listed as follows:

```
$x_0$ is an initial guess, $r_0 = b - Ax_0$,
choose $r_0^*$ such that $(r_0^*, r_0) \neq 0$,
set $\beta_{-1} = 0$, $\kappa$ is given as $0 < \kappa \leq 1$,
compute $Ar_0$,
for $n = 0, 1, \cdots$ until $||\boldsymbol{r}_{n+1}|| \le \varepsilon\ ||\boldsymbol{r}_0||$ do :
begin
  $\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1})$,
  $A\boldsymbol{p}_n = A\boldsymbol{r}_n + \beta_{n-1}(A\boldsymbol{p}_{n-1} - A\boldsymbol{u}_{n-1})$,
  $\alpha_n = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}$,
  $\boldsymbol{a}_n = \boldsymbol{r}_n,\ \boldsymbol{b}_n = A\boldsymbol{z}_{n-1},\ \boldsymbol{c}_n = A\boldsymbol{r}_n$,
  $\rho = \dfrac{|(\boldsymbol{a}_n, \boldsymbol{c}_n)|}{||\boldsymbol{a}_n||\ ||\boldsymbol{c}_n||}$,
  if $\rho < \kappa$, then
    $\zeta_n = \dfrac{(\boldsymbol{b}_n, \boldsymbol{b}_n)(\boldsymbol{c}_n, \boldsymbol{a}_n) - (\boldsymbol{b}_n, \boldsymbol{a}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{b}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}$,
    $\eta_n = \dfrac{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{a}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{a}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)(\boldsymbol{b}_n, \boldsymbol{b}_n) - (\boldsymbol{b}_n, \boldsymbol{c}_n)(\boldsymbol{c}_n, \boldsymbol{b}_n)}$,
    $\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
    compute $A\boldsymbol{u}_n$,
    $\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
    $\boldsymbol{z}_n = \zeta_n \boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n$,
    $A\boldsymbol{z}_n = \zeta_n A\boldsymbol{r}_n + \eta_n A\boldsymbol{z}_{n-1} - \alpha_n A\boldsymbol{u}_n$,
  else
    $\zeta_n = \dfrac{(\boldsymbol{c}_n, \boldsymbol{a}_n)}{(\boldsymbol{c}_n, \boldsymbol{c}_n)},\ \eta_n = 0$,
    $\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n$,
    compute $A\boldsymbol{u}_n$,
    $\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
    $\boldsymbol{z}_n = \zeta_n \boldsymbol{t}_n$,
    $A\boldsymbol{z}_n = \zeta_n A\boldsymbol{r}_n - \alpha_n A\boldsymbol{u}_n$,
  end if
  $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n$,
  $\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - A\boldsymbol{z}_n$,
  compute $A\boldsymbol{r}_{n+1}$,
  $\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
end
```

4 Numerical experiments

In this section numerical experiments will be presented. All computations were done in double precision floating point arithmetics, and performed on HP workstation xw4200 with CPU of Intel(R) Pentium (R) 4, clock of 3.9GHz, main memory of 3GB, OS of Suse Linux version 9.2. Compile option with "-O0" is used for cases of non-preconditionings. On the other hand, compile option with "-O3" is used for cases of preconditionings. The right-hand side $\boldsymbol{b}$ was imposed from the physical load conditions. The stopping criterion for successful convergence of the iterative methods is less than 10^{-7} of the relative residual 2-norm $||\boldsymbol{r}_{n+1}||_2/||\boldsymbol{r}_0||_2$. The maximum number of iterations is fixed as 10^4. The initial shadow residual $\boldsymbol{r}_0^*$ is set as $\boldsymbol{r}_0$. Parameter κ varies from 0.0 to 1.0 at the interval of 0.05.

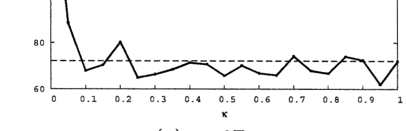
4.1 Preconditioning

In Table 1 we present iterations, computation times in seconds and ratios of heuristic variant of GPBiCG_AR2 with κ to the original GPBiCG_AR method. The bold figures means the least time for each matrix. The ratio shown in the most right column implies computation time of heuristic variant of GPBiCG_AR2 method with κ to the original GPBiCG_AR method. The values of heuristic variant of GPBiCG_AR2 method represents those of the least computation time. From Table 1, we can see that heuristic variant of GPBiCG_AR2 method with κ outperforms well. Test matrices are derived from Florida sparse matrix collection[1].

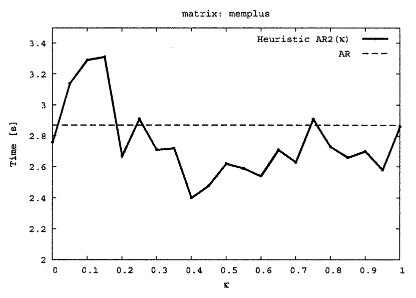
Table 1: Iterations, computation times and ratios of heuristice variant of GPBiCG_AR2 method with κ to the original GPBiCG_AR and GPBi-CG methods without preconditioning.

matrix	GP		AR		heuristic AR2			
	itr.	time [s]	itr.	time [s]	itr.	time [s]	κ	ratio to AR (GP)
wang4	398	2.82	395	2.61	360	**2.09**	.00	.80 (.74)
sme3Db	4612	203.2	4738	211.1	3893	**170.7**	.55	.81 (.84)
memplus	624	3.10	601	2.87	514	**2.40**	.40	.84 (.77)
comsol	261	0.53	255	0.51	215	**0.44**	.30	.86 (.83)
wang3	187	1.32	181	1.20	166	**1.05**	.20	.88 (.80)
poi-3db	161	12.10	162	13.13	154	**11.59**	.60	.88 (.96)
bjtcai	4058	49.65	3818	45.13	3431	**40.11**	.10	.89 (.81)
sme3Da	5410	97.44	4028	72.03	3636	**64.92**	.25	.90 (.67)
epb1	347	1.31	351	1.26	330	**1.15**	.35	.91 (.83)
af23560	1942	24.57	2058	25.35	1882	**23.16**	.60	.91 (.94)
epb3	2866	61.46	2647	53.30	2443	**49.00**	.60	.92 (.80)
xenon1	488	14.86	521	15.44	481	**14.22**	.75	.92 (.96)
ex19	2237	15.15	2218	14.74	2098	13.91	.15	.94 (.92)
epb2	230	1.59	239	1.57	230	**1.45**	.60	.94 (.92)
ns3Da	739	25.51	767	26.41	700	24.79	.50	.94 (.97)
3D_3D	max	–	6344	173.9	6071	**165.6**	.90	.95 (–)
OK01	1839	154.7	1866	155.4	1786	**148.0**	.25	.95 (.96)
0_0_200	207	1.99	204	1.83	197	**1.76**	.90	.96 (.88)
ex11	981	22.67	917	21.00	901	**20.67**	.95	.98 (.91)

Fig.1 presents computation time in seconds of heuristic variant of GPBiCG_AR2 without preconditioning when κ varies from 0 to 1 at the interval of 0.05 for matrices sme3Da and memplus. From Fig.1, it can be seen that GPBiCG_AR2 method works well exception small values of parameter κ.



(a)sme3Da



(b)memplus

Figure 1: Computation time of heuristic variant of AR2 without preconditioning when κ varies from 0 to 1 at the interval of 0.05.

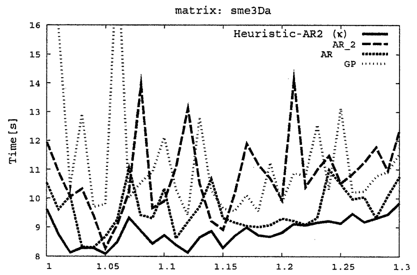
4.2 Non-preconditioning

In Table 2 we present iterations and computation times in seconds of some iterative methods with and without preconditioning. "NaN" means halt of computation because of overflow operations during iteration process. Parameter "γ" of heuristic AR2 method means the least computation time among various γ_s.

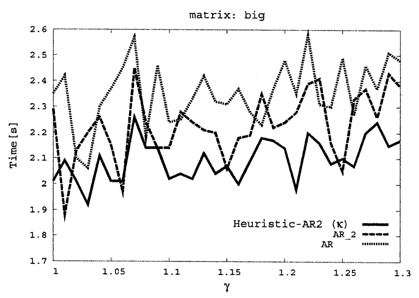
Table 2: Iterations and computation times of iterative methods with and without preconditioning.

matrix	pre-	heuristic AR2				GP		AR		AR2	
	cond.	γ	itr.	time	κ	itr.	time	itr.	time	itr.	time
big	non	-	2235	3.47	1.0	2528	4.17	2235	3.41	3318	4.90
	pre.	1.03	610	1.92	.10	NaN	-	641	2.06	693	2.20
epb1	non	-	346	0.51	.05	347	0.62	351	0.58	353	0.56
	pre.	1.00	68	0.21	.60	NaN	-	67	0.22	67	0.23
epb2	non	-	231	0.73	.70	230	0.79	239	0.77	238	0.73
	pre.	1.02	20	0.13	.50	NaN	-	19	0.13	19	0.13
sme-	non	-	3469	27.7	.95	5410	43.5	4028	32.0	3925	31.0
3da	pre.	1.05	487	8.09	.00	590	9.80	520	8.68	497	8.28

Fig.2 depicts computation times for preconditioned heuristic AR2 and other methods with various γ for matrices sme3da and big. From Fig.2, heuristic AR_2 method outperforms for various accelerated parameter γ_s of ILU decomposition in view of CPU times. Fig.3 shows history of relative residual 2-norm of heuristic AR2 and other iterative methods for sme3da, big and epb2.

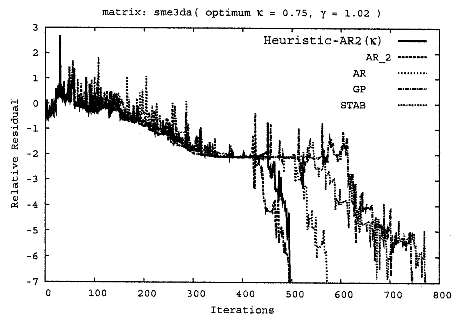


(a)sme3da

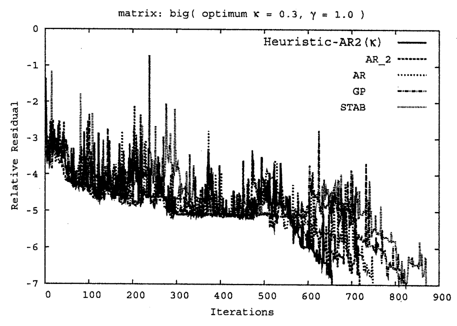


(b)big

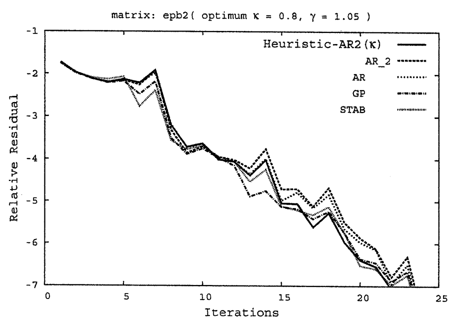
Figure 2: Computation times for preconditioned heuristic AR2 and other methods with various γ_s.



(a)sme3da



(b)big



(c)epb2

Figure 3: History of relative residual 2-norm of heuristic AR2 and other iterative methods.

Table 3 tabulates iterations and computation times of some iterative methods for matrix sme3da when accelerated parameter γ of ILU decomposition for diagonal entries varies. "$\kappa_{\mathrm{opt.}}$" means the least computation time among various κ_s. In Table 3 bold figures means the least time for each matrix. Heuristic AR2 method work well among test iterative methods. Moreover, it is noted that, when $\gamma = 1.06$, iteration counts of GP method needs suddenly 1150 iterations compared with other parameter γ_s. When $\gamma = 1.0$, the same tendency appears as for iteration counts (= 1456 itr.) of GP method. On the other hand, iteration counts of heuristic AR2 method keep almost constant iterations for various parameter γ_s.

Table 4 exhibits iterations and computation times of some iterative methods for matrix big. GP method did not converge because of overflow operations during iteration process. AR2 and heuristic AR2 methods are competitive.

Table 3: Iterations and computation times of some iterative methods for matrix sme3da when accelerated γ of ILU decomposition varies.

γ	heuristic AR2			GP		AR		AR2	
	itr.	time	$\kappa_{\mathrm{opt.}}$	itr.	time	itr.	time	itr.	time
1.00	586	9.63	0.60	<u>1456</u>	23.38	641	10.53	732	11.94
1.01	529	8.76	0.75	978	15.83	582	9.62	666	10.92
1.02	487	8.13	0.75	631	10.44	612	10.09	613	10.06
1.03	497	8.30	1.00	792	12.93	496	8.32	629	10.33
1.04	498	8.29	1.00	583	9.71	497	**8.31**	570	9.40
1.05	487	**8.09**	0.00	590	9.80	520	8.68	497	**8.28**
1.06	513	8.49	0.20	<u>1150</u>	18.56	559	9.27	549	9.09
1.07	566	9.34	0.65	604	10.02	673	11.06	622	10.22
1.08	535	8.88	0.55	637	10.55	570	9.43	864	13.95
1.09	508	8.44	0.60	662	10.92	564	9.32	585	9.68
1.10	527	8.73	0.90	737	12.12	627	10.34	601	9.91
1.15	495	8.30	0.40	580	**9.66**	562	9.33	536	8.92
1.20	532	8.82	0.60	594	9.86	564	9.31	603	9.95
1.25	552	9.14	0.25	804	13.14	637	10.50	641	10.53
1.30	599	9.85	0.65	700	11.55	656	10.76	763	12.39

Table 4: Iterations and computation times of some iterative methods for matrix big.

γ	heuristic AR2			GP		AR		AR2	
	itr.	time	$\kappa_{\mathrm{opt.}}$	itr.	time	itr.	time	itr.	time
1.00	629	2.01	0.30	NaN	-	737	2.35	732	2.29
1.01	652	2.09	0.80	NaN	-	762	2.42	604	**1.88**
1.02	627	2.01	0.40	NaN	-	664	2.10	678	2.13
1.03	610	**1.92**	0.10	NaN	-	641	**2.06**	693	2.20
1.04	666	2.11	0.20	NaN	-	721	2.30	720	2.26
1.05	630	2.01	0.50	NaN	-	744	2.37	681	2.15
1.06	631	2.01	0.25	NaN	-	756	2.45	627	1.97
1.07	711	2.26	0.35	NaN	-	804	2.57	778	2.45
1.08	673	2.14	0.95	NaN	-	684	2.18	708	2.24
1.09	668	2.14	0.55	NaN	-	766	2.46	678	2.14
1.10	635	2.02	0.40	NaN	-	704	2.24	678	2.14
1.15	647	2.07	0.40	NaN	-	720	2.31	649	2.06
1.20	671	2.14	0.60	NaN	-	776	2.48	717	2.24
1.25	655	2.10	0.50	NaN	-	775	2.49	648	2.05
1.30	675	2.17	0.70	NaN	-	782	2.48	748	2.38

5 Concluding remarks

The heuristic variant of GPBiCG_AR2 method outperforms well compared with the conventional GPBi-CG, GPBiCG_AR and simple variant of GPBiCG_AR2 methods.

A future work is to choose an optimum κ for each matrix. Moreover, it is crucial to examine the relationship between κ used in BiCGStab method [5] and that used in GPBiCG_AR method.

References

[1] Florida sparse matrix collection: http:// www.cise.ufl.edu/research/sparse/matrices/

[2] S. Fujino, M. Fujiwara and M. Yoshida: BiCGSafe method based on minimization of associate residual, Transaction of JSCES 2005. (in Japanese)

[3] Moe Thuthu, S. Fujino: Stability of GPBiCG_AR method based on minimization of associate residual, The Abstract of Asian Symposium on Computer Mathematics, December 2007, Singapore.

[4] Moe Thuthu, S. Fujino: Stability of GPBiCG_AR method based on minimization of associate residual, Transaction of ASCM, 5081(2008), 108-120.

[5] G. Sleijpen, H. van der Vorst: Maintaining convergence properties of BiCGstab methods in finite precision arithmetic, Numerical Algorithms, **10**(1995), pp.203-223.

[6] H.A. van der Vorst: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., **13**(1992), 631-644.

[7] H.A. van der Vorst: Iterative Krylov preconditionings for large linear systems, Cambridge University Press, Cambridge, 2003.

[8] S.-L. Zhang: GPBi-CG: Generalized product-type preconditionings based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput., **18**(1997), 537-551.