Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol II
IMECS 2009, March 18 - 20, 2009, Hong Kong

An Overview of A Family of New Iterative Methods Based on IDR Theorem And Its Estimation

Yusuke Onoue * Seiji Fujino † Norimasa Nakashima ‡

Abstract—**After birth of IDR(s) method based on IDR Theorem, two variants of MR_IDR(s) and Bi_IDR(s) methods were proposed one after another. The former method gained stability by adoptation of strategy of minimizing intermediate residual norm with extra computational cost. The latter method became sophiscated and elegant variant with stability by means of adoptation of bi-orthogonalization conditions. In this article, we overview a family of IDR(s) methods, and evaluate performance of these variants through several numerical experiments.**

Keywords: IDR Theorem, IDR(s) method, MR_IDR(s) method, Bi_IDR(s) method

1 Introduction

We consider to solve a unsymmetric linear system of equations,

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

where A is a given unsymmetric coefficient matrix in $R^{N\times N}$, and $\boldsymbol{x}$ is a solution vector in R^N, and $\boldsymbol{b}$ is a right-hand side vector in R^N. Krylov subspace methods are effective for solving linear systems of equations [2]. Krylov subspace is defined as follows:

$$K_n(A;\boldsymbol{r}_0) \quad := \quad \mathrm{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{n-1}\boldsymbol{r}_0\}. \tag{2}$$

Here, $\boldsymbol{r}_0 := \boldsymbol{b} - A\boldsymbol{x}_0$ is an initial residual vector. The members of Krylov subspace methods, product-type Bi-Conjugate Gradient (BiCG) methods are often used for solving nonsymmetric linear systems of equations. BiCG stabilized (BiCGStab) method [2] is one of versions of product-type Bi-Conjugate Gradient (BiCG) methods.

In 2007, one of Krylov subspace method, IDR(s) method is proposed by P. Sonneveld and M. B. van Gijzen [5]. IDR(s) method is based on the IDR Theorem. IDR(s) method is competitive with or superior to most product-type BiCG methods, and outperforms BiCGStab method when $s > 1$.

Furthermore, Minimum Residual IDR(s) (MR_IDR(s)) and Bi-orthogonalized IDR(s) (Bi_IDR(s)) methods are proposed as variants of IDR(s) method one after another by P. Sonneveld and M. B. van Gijzen[6][7].

In this paper, we overview MR_IDR(s) and Bi_IDR(s) methods, and determine effectiveness of these two iterative methods through numerical experiments.

This paper is organized as follows. In section 2, we note outline and algorithm of IDR(s) method. In section 3, we describe outline and algorithm of MR_IDR(s) method. In section 4, we describe outline and algorithm of Bi_IDR(s) method. In section 5, we examine effectiveness of MR_IDR(s) and Bi_IDR(s) methods through numerical experiments. Finally, in section 6, we draw concluding remarks.

2 IDR(s) method

In this section, characteristics of IDR(s) method can be mentioned as follows[5]:

IDR Theorem

Let A be any matrix in $R^{N\times N}$, and $\boldsymbol{v}_0$ be any vector in R^N, and $\mathcal{G}_0$ be the complete Krylov space $K_N(A, \boldsymbol{v}_0)$. Let S denote any space in R^N, and define the sequence spaces $\mathcal{G}_j (j = 1, 2, \ldots)$ as

$$\mathcal{G}_j := (I - \omega_j A)(\mathcal{G}_{j-1} \cap S). \tag{3}$$

Here ω_j's are non-zero scalars. Then, the next two Theorems hold.

(i) $\mathcal{G}_j \subseteq \mathcal{G}_{j-1}$ for all $j > 0$,

(ii) $\mathcal{G}_j = \{\mathbf{0}\}$ for some $j \leq N$.

Computing the first residual $\boldsymbol{r}_{n+1}$ in $\mathcal{G}_{j+1}$

*Graduate School of Information Science and Electrical Engineering, Kyushu University, Email: onoue@zeal.cc.kyushu-u.ac.jp
†Research Institute for Information Technology, Kyushu University, Email: fujino@cc.kyushu-u.ac.jp
‡Graduate School and Faculty of Information Science and Electrical Engineering, Kyushu University, Email: norimasa@csce.kyushu-u.ac.jp

IDR(s) method is built such that $s+1$ residual vectors are forced to be in $\mathcal{G}_j$. The reisdual $\boldsymbol{r}_{n+1}$ is in $\mathcal{G}_{j+1}$ if

$$\boldsymbol{r}_{n+1} = (I - \omega_j A)\boldsymbol{v}_n, \ \boldsymbol{v}_n \in \mathcal{G}_j \cap \mathrm{Null}(P^T). \qquad (4)$$

Here, parameter ω_j is determined by solving minimization of the residual norm $||\boldsymbol{r}_{n+1}||_2$.

Computing a combination of the residual vectors, a vector $\boldsymbol{v}_n$

Vector $\boldsymbol{v}_n$ can be written as a combination of the residual vectors in $\mathcal{G}_j$ because $\boldsymbol{v}_n \in \mathcal{G}_j \cap \mathrm{Null}(P^T)$. We define the forward difference residual vector $\boldsymbol{e}_n := \boldsymbol{r}_{n+1} - \boldsymbol{r}_n$. If $\boldsymbol{r}_{n-i}(i = 0, \dots, s) \in \mathcal{G}_j$ then $\boldsymbol{e}_{n-i}(i = 1, \dots, s) \in \mathcal{G}_j$. Thus, vector $\boldsymbol{v}_n$ can be written as

$$\boldsymbol{v}_n = \boldsymbol{r}_n - \sum_{i=1}^{s} c_i \boldsymbol{e}_{n-i}. \qquad (5)$$

Since $\boldsymbol{v}_n \in \mathrm{Null}(P^T)$, it satisfies $P^T \boldsymbol{v}_n = 0$. We can solve the coefficients c_i by solving an $s \times s$ linear system $P^T \boldsymbol{v}_n = 0$. Besides, we can compute the vector $\boldsymbol{v}_n$.

Computing the kth residual $\boldsymbol{r}_{n+k}$ in subspace $\mathcal{G}_{j+1}$

The kth residual $\boldsymbol{r}_{n+k}(2 \le k \le s+1)$ can be computed by consecutive computations as follows:

Computing the kth residual $\boldsymbol{r}_{n+k}$ in subspace $\mathcal{G}_{j+1}$

1. Solve c_i from $P^T \sum_{i=1}^{s} c_i \boldsymbol{e}_{n+k-1-i} = P^T \boldsymbol{r}_{n+k-1}$
2. $\boldsymbol{v}_{n+k-1} = \boldsymbol{r}_{n+k-1} - \sum_{i=1}^{s} c_i \boldsymbol{e}_{n+k-1-i}$
3. $\boldsymbol{r}_{n+k} = (I - \omega_j A)\boldsymbol{v}_{n+k-1}$

Here, $\boldsymbol{r}_{n+k-1-i} \in \mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ when $1 \le i < k$, and $\boldsymbol{r}_{n+k-1-i} \in \mathcal{G}_j$ when $k \le i \le s$. Therefore, $\boldsymbol{e}_{n+k-1-i} \in \mathcal{G}_j$ and $\boldsymbol{r}_{n+k} \in \mathcal{G}_{j+1}$.

We present the algorithm of IDR(s) method as follows:

Algorithm 1: IDR(s) method

1. Let $\boldsymbol{x}_0$ be a random vector, and put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
2. For $n = 0, \dots, s-1$ Do
3. $\quad \boldsymbol{v}_n = A\boldsymbol{r}_n$
4. $\quad \omega = \dfrac{(\boldsymbol{v}_n, \boldsymbol{r}_n)}{(\boldsymbol{v}_n, \boldsymbol{v}_n)}$
   $\quad \left(\begin{array}{l} \rho = \frac{|(\boldsymbol{v}_n, \boldsymbol{r}_n)|}{||\boldsymbol{v}_n||_2 * ||\boldsymbol{r}_n||_2} \\ \text{If } \rho < \kappa \text{ then } \omega = \frac{\kappa}{\rho}\omega \end{array} \right)$
5. $\quad \boldsymbol{q}_n = \omega \boldsymbol{r}_n, \quad \boldsymbol{e}_n = -\omega \boldsymbol{v}_n$
6. $\quad \boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \boldsymbol{e}_n, \ \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{q}_n$
7. End Do
8. $E_s = (\boldsymbol{e}_{s-1}, \dots, \boldsymbol{e}_0), \quad Q_s = (\boldsymbol{q}_{s-1}, \dots, \boldsymbol{q}_0)$
9. Do $n = s, s+1, \dots$
10. $\quad$ Solve $\boldsymbol{c}_n$ from $P^T E_n \boldsymbol{c}_n = P^T \boldsymbol{r}_n$
11. $\quad \boldsymbol{v}_n = \boldsymbol{r}_n - E_n \boldsymbol{c}_n$
12. $\quad$ If $\mathrm{mod}(n, s+1) = s$ then
13. $\quad\quad \boldsymbol{t}_n = A\boldsymbol{v}_n$
14. $\quad\quad \omega = \dfrac{(\boldsymbol{t}_n, \boldsymbol{v}_n)}{(\boldsymbol{t}_n, \boldsymbol{t}_n)}$
    $\quad\quad \left(\begin{array}{l} \rho = \frac{|(\boldsymbol{t}_n, \boldsymbol{v}_n)|}{||\boldsymbol{t}_n||_2 * ||\boldsymbol{v}_n||_2} \\ \text{If } \rho < \kappa \text{ then } \omega = \frac{\kappa}{\rho}\omega \end{array} \right)$
15. $\quad\quad \boldsymbol{e}_n = -E_n \boldsymbol{c}_n - \omega \boldsymbol{t}_n$
16. $\quad\quad \boldsymbol{q}_n = -Q_n \boldsymbol{c}_n + \omega \boldsymbol{v}_n$
17. $\quad$ Else
18. $\quad\quad \boldsymbol{q}_n = -Q_n \boldsymbol{c}_n + \omega \boldsymbol{v}_n$
19. $\quad\quad \boldsymbol{e}_n = -A\boldsymbol{q}_n$
20. $\quad$ End If
21. $\quad \boldsymbol{r}_{n+1} = \boldsymbol{r}_n + \boldsymbol{e}_n, \ \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \boldsymbol{q}_n$
22. $\quad$ if $||\boldsymbol{r}_{n+1}||_2 / ||\boldsymbol{r}_0||_2 \le \epsilon$ then stop
23. $\quad E_{n+1} = (\boldsymbol{e}_n, \dots, \boldsymbol{e}_{n+1-s}), \ Q_{n+1} = (\boldsymbol{q}_n, \dots, \boldsymbol{q}_{n+1-s})$
24. End Do

In steps 4th and 14th, we note the additional computation for ω. The computation improves the accuracy of IDR(s) method.

3 MR_IDR(s) method

In this section, characteristics of MR_IDR(s) method can be mentioned as follows[7]:

Computing the kth residual $\boldsymbol{r}_{n+k}$ in subspace $\mathcal{G}_{j+1}$

The kth residual $\boldsymbol{r}_{n+k}(1 \le k \le s+1)$ can be computed by consecutive computations as follows:

Computing the kth residual $\boldsymbol{r}_{n+k}$ in subspace $\mathcal{G}_{j+1}$

1. Solve c_i from $P^T \sum_{i=1}^{s} c_i \boldsymbol{g}_{n+k-1-i} = P^T \boldsymbol{r}_{n+k-1}$
2. $\boldsymbol{v}_{n+k-1} = \boldsymbol{r}_{n+k-1} - \sum_{i=1}^{s} c_i \boldsymbol{g}_{n+k-1-i}$
3. $\boldsymbol{r}_{n+k} = (I - \omega_j A)\boldsymbol{v}_{n+k-1}$

Here, the vector $\boldsymbol{g}_n$ is defined as

$$\boldsymbol{g}_n := (-1) * \boldsymbol{e}_n = \boldsymbol{r}_n - \boldsymbol{r}_{n+1}. \qquad (6)$$

$\boldsymbol{r}_{n+k-1-i} \in \mathcal{G}_{j+1} \subseteq \mathcal{G}_j$ when $1 \le i < k$, and $\boldsymbol{r}_{n+k-1-i} \in \mathcal{G}_j$ when $k \le i \le s$. Accordingly, $\boldsymbol{g}_{n+k-1-i} \in \mathcal{G}_j$ and $\boldsymbol{r}_{n+k} \in \mathcal{G}_{j+1}$.

Minimization of the intermediate residual norms

Let matrix $G_{n+k} = (\boldsymbol{g}_{n+k-1}, \ldots, \boldsymbol{n+k-s})$. Having computed k orthogonal columns of G_{n+k}, the kth intermediate residual $\boldsymbol{r}_{n+k}$ can be minimized over the vectors in $\mathcal{G}_{j+1}$ by making $\boldsymbol{r}_{n+k}$ orthogonal to the first k columns of G_{n+k}.

We present the algorithm of MR_IDR(s) method as follows:

Algorithm 2: MR_IDR(s) method

1. Let $\boldsymbol{x}_0$ be a random vector, and put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
2. $G_{-1}, U_{-1} = O \in R^{N\times s}, M_{-1} = I, \omega_0 = 1$
3. $n = 0, j = 0$
4. While $||\boldsymbol{r}_n||_2/||\boldsymbol{r}_0||_2 > \epsilon$ Do
5. Do $k = 1, \ldots, s$
6. $\boldsymbol{m} = P^T\boldsymbol{r}_n$
7. Solve $\boldsymbol{c}$ from $M_{j-1}\boldsymbol{c} = \boldsymbol{m}$
8. $\boldsymbol{v} = \boldsymbol{r}_n - G_{j-1}\boldsymbol{c}, \bar{\boldsymbol{u}} = U_{j-1}\boldsymbol{c} + \omega_j\boldsymbol{v}$
9. $\bar{\boldsymbol{g}} = A\bar{\boldsymbol{u}}$
10. Do $i = 1, \ldots, k-1$
11. $\alpha = (\boldsymbol{g}_{n-i}, \bar{\boldsymbol{g}})$
12. $\bar{\boldsymbol{g}} = \bar{\boldsymbol{g}} - \alpha\boldsymbol{g}_{n-i}, \bar{\boldsymbol{u}} = \bar{\boldsymbol{u}} - \alpha\boldsymbol{u}_{n-i}$
13. End Do
14. $\alpha = \sqrt{(\bar{\boldsymbol{g}}, \bar{\boldsymbol{g}})}$
15. $\boldsymbol{g}_n = \frac{1}{\alpha}\bar{\boldsymbol{g}}, \boldsymbol{u}_n = \frac{1}{\alpha}\bar{\boldsymbol{u}}$
16. $\beta_n = (\boldsymbol{r}_n, \boldsymbol{g}_n)$
17. $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \beta_n\boldsymbol{g}_n, \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \beta_n\boldsymbol{u}_n$
18. $n = n+1$
19. End Do
20. $G_j = (\boldsymbol{g}_{n-1}, \ldots, \boldsymbol{g}_{n-s}), U_j = (\boldsymbol{u}_{n-1}, \ldots, \boldsymbol{u}_{n-s})$
21. $M_j = P^TG_j, \boldsymbol{m} = P^T\boldsymbol{r}_n$
22. Solve $\boldsymbol{c}$ from $M_j\boldsymbol{c} = \boldsymbol{m}$
23. $\boldsymbol{v} = \boldsymbol{r}_n - G_j\boldsymbol{c}, \boldsymbol{t} = A\boldsymbol{v}$
24. $\omega_{j+1} = \frac{(\boldsymbol{t}, \boldsymbol{v})}{(\boldsymbol{t}, \boldsymbol{t})}$
25. $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + U_j\boldsymbol{c} + \omega_{j+1}\boldsymbol{v}$
26. $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - G_j\boldsymbol{c} - \omega_{j+1}\boldsymbol{t}$
27. $n = n+1, j = j+1$
28. End While

In 16th step of the above algorithm, $M_j = P^TG_j$ can be computed cheaply using

$$P^T\boldsymbol{g}_{n-i} = P^T(\boldsymbol{r}_{n-i} - \boldsymbol{r}_{n-i+1})/\beta_{n-i}.$$

4 Bi_IDR(s) method

In this section, characteristics of Bi_IDR(s) method can be mentioned as follows[6]:

Strategies of Bi_IDR(s) method

We assume that $\boldsymbol{r}_{n+1}$ is the first residual in $\mathcal{G}_{j+1}$. Bi_IDR(s) method is built by constructing vectors that satisfy the following two orthogonaly conditions:

$$\boldsymbol{g}_{n+i} \perp \boldsymbol{p}_j \quad (i = 2, \ldots, s, j = 1, \ldots, i), \tag{7}$$

$$\boldsymbol{r}_{n+i+1} \perp \boldsymbol{p}_j \quad (i = 1, \ldots, s, j = 1, \ldots, i). \tag{8}$$

Here, the vector $\boldsymbol{p}_i$ are a column vector of matrix P. The above two orthogonal conditions lead to computational cost reduction and stabilization of convergence property.

Computing the first residual $\boldsymbol{r}_{n+1}$ in $\mathcal{G}_{j+1}$

The orthogonal condition (8) means that the first intermediate residual is orthogonal to $\boldsymbol{p}_1$. The last intermediate residual is orthogonal to $\boldsymbol{p}_1 \sim \boldsymbol{p}_s$. Hence, the last intermediate residual $\boldsymbol{r}_n$ in G_j is orthogonal to $\boldsymbol{p}_1 \sim \boldsymbol{p}_s$. Consequently,

$$\boldsymbol{r}_n \in \mathcal{G}_j \cap \mathrm{Null}(P^T). \tag{9}$$

The first residual $\boldsymbol{r}_{n+1}$ in $\mathcal{G}_{j+1}$ can be computed as

$$\boldsymbol{r}_{n+1} = (I - \omega_jA)\boldsymbol{r}_n. \tag{10}$$

Computing the kth residual $\boldsymbol{r}_{n+k}$ in $\mathcal{G}_{j+1}$

The kth residual $\boldsymbol{r}_{n+k}$ in $\mathcal{G}_{j+1}$ is computed similarly to MR_IDR(s) method. In order to compute $\boldsymbol{r}_{n+k}$, you should solve the following linear systems:

$$P^T\sum_{i=1}^{s} c_i\boldsymbol{g}_{n+k-1-i} = P^T\boldsymbol{r}_{n+k-1},$$

$$\sum_{i=1}^{s} c_i\boldsymbol{p}_j^T\boldsymbol{g}_{n+k-1-i} = \boldsymbol{p}_j^T\boldsymbol{r}_{n+k-1} \ (j = 1, \ldots, s). \tag{11}$$

Here, the orthogonal condition (7) leads to $\boldsymbol{p}_j^T\boldsymbol{g}_{n+k-1-i} = 0$ when $j < k-1-i$. Additionally, the orthogonal condition (8) leads to $\boldsymbol{p}_j^T\boldsymbol{r}_{n+k-1} = 0$ when $j < k-1$. Hence, you don't have to compute $\boldsymbol{p}_j^T\boldsymbol{g}_{n+k-1-i}$ when $j < k-1-i$ and $\boldsymbol{p}_j^T\boldsymbol{r}_{n+k-1}$ when $j < k-1$. In consequence, computational cost of Bi_IDR(s) method is lower than that of IDR(s) and MR_IDR(s) methods.

We present the algorithm of Bi_IDR(s) method as follows:

Algorithm 3: Bi_IDR(s) method

Let $\boldsymbol{x}_0$ be a random vector, and put $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$,
$\boldsymbol{g}_i = \boldsymbol{u}_i = \boldsymbol{0}, i = 1, \ldots, s, M = I, \omega = 1$
$n = 0$
While $||\boldsymbol{r}_n||_2/||\boldsymbol{r}_0||_2 > \epsilon$ Do
$\boldsymbol{f} = P^T\boldsymbol{r}_n, \boldsymbol{f} = (\phi_1, \ldots, \phi_s)$

Do $k = 1, \ldots, s$
  Solve $\boldsymbol{c}$ from $M\boldsymbol{c} = \boldsymbol{f}, \boldsymbol{c} = (\gamma_1, \ldots, \gamma_s)$
  $\boldsymbol{v} = \boldsymbol{r}_n - \sum_{i=k}^{s} \gamma_i \boldsymbol{g}_i, \boldsymbol{u}_k = \omega \boldsymbol{v} + \sum_{i=k}^{s} \gamma_i \boldsymbol{u}_i$
  $\boldsymbol{g}_k = A\boldsymbol{u}_k$
  Do $i = 1, \ldots, k-1$
    $\alpha = \frac{(\boldsymbol{p}_i, \boldsymbol{g}_k)}{\mu_{i,i}}$
    $\boldsymbol{g}_k = \boldsymbol{g}_k - \alpha \boldsymbol{g}_i, \boldsymbol{u}_k = \boldsymbol{u}_k - \alpha \boldsymbol{u}_i$
  End Do
  $\mu_{i,k} = (\boldsymbol{p}_i, \boldsymbol{g}_k), i = k, \ldots, s, M_{i,k} = \mu_{i,k}$
  $\beta = \frac{\phi_k}{\mu_{k,k}}$
  $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \beta \boldsymbol{g}_k, \boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \beta \boldsymbol{u}_k$
  If $k < s$ then
    $\phi_i = 0, i = 1, \ldots, k, \phi_i = \phi_i - \beta \mu_{i,k},$
    $i = k+1, \ldots, s$
    $\boldsymbol{f} = (\phi_1, \ldots, \phi_s)$
  End If
  $n = n + 1$
End Do
$\boldsymbol{t} = A\boldsymbol{r}_n$
$\omega = \frac{(\boldsymbol{t}, \boldsymbol{r})}{(\boldsymbol{t}, \boldsymbol{t})}$
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \omega \boldsymbol{r}_n, \boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \omega \boldsymbol{t}$
$n = n + 1$
End While

5 Numerical Experiments

In this section we discuss numerical experiments of IDR(s) method and MR_IDR(s) method, Bi_IDR(s) method. All computations are carried out in double precision floating-point arithmetic on a PC with a POWER5 processor (1.9GHz). Intel Fortran Compiler90 ver 7.1 and compile option -O3 -qtune=power5 -qarch=pw5 -qhot was used. In all cases the iteration was started with the initial guess solution $\boldsymbol{x}_0 = \boldsymbol{0}$. The maximum iterations was fixed as 10000. The value of s varies at the interval of 1 from 1 to 10. Twelve test matrices are from University of Florida Sparse Matrix Collection[1][3]. Description of test matrices is shown in Table 1. In this Table, "nnz" means number of nonzero entries, and "ave_nnz" means number of nonzero entries per single row.

5.1 Numerical Results

Table 2 shows iterations and CPU time in seconds of three iterative methods. In Table 2, "s_{opt}" means optimum parameter s. CPU time is minimum at optimum parameter s. "itr." means number of iterations. "ratio" means ratio of CPU time of each method to CPU time of IDR(s) method. The figure in bold means minimum CPU time of three iterative methods. From Table 2, the following observations can be made.

Table 1: Description of test matrices.

matrix	dimension	nnz	ave_nnz
big	13,209	91,465	6.92
epb1	14,734	95,053	6.45
epb2	25,228	175,027	6.94
garon2	13,535	373,235	27.58
memplus	17,758	126,150	7.10
poisson3da	13,514	352,762	26.10
poisson3db	85,623	2,374,949	27.74
raefsky2	3,242	293,551	90.55
sme3da	12,504	874,887	69.97
sme3db	29,067	2,081,063	71.60
xenon1	48,600	1,181,120	24.30
xenon2	157,464	3,866,688	24.56

1. Bi_IDR(s) method converges fastest for 10 matrices.

2. CPU time of all methods are fastest at $s = 3$ for matrix epb1.

3. Iterations of MR_IDR(s) method is minimum and that of Bi_IDR(s) method is maximum for matrix epb1.

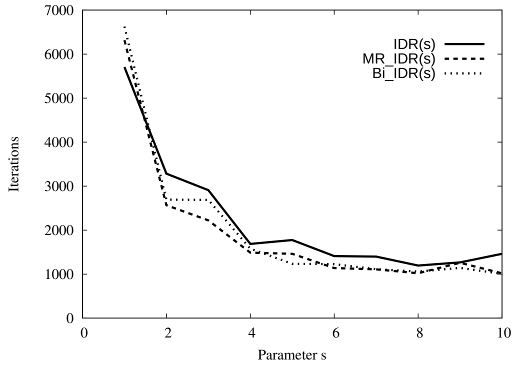
4. CPU time of Bi_IDR(s) method is minimum and that of MR_IDR(s) method is maximum for matrix epb1.

From the second and third, fourth observations you can see that computational cost of Bi_IDR(s) method is minimum and that of MR_IDR(s) method is maximum.

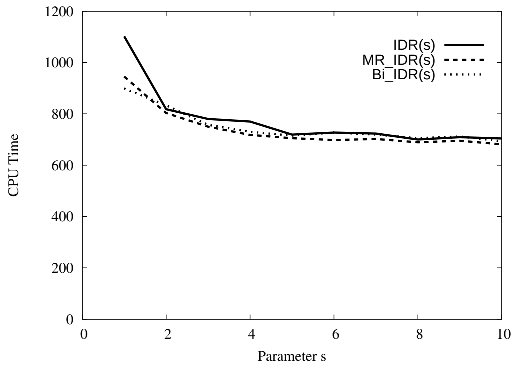
Fig. 1 displays variation of iterations of three iterative methods for matrices big and epb1. In Fig. 1, we show variation of iterations of IDR(s) method in solid line and MR_IDR(s) method in dashed line and Bi_IDR(s) method in dotted line. From Fig. 1 you can see that iterations of MR_IDR(s) method is minimum and that of IDR(s) method is maximum for almost cases.

Fig. 2 shows variation of CPU time of three iterative methods for matrices big and epb1. From Fig. 2 you can see that CPU time of Bi_IDR(s) method is minimum and that of MR_IDR(s) method is maximum for almost cases.

Fig. 3 plots relative residual of three iterative methods for matrices big and epb1. From Fig. 3 you can see that Bi_IDR(s) method converges fastest and MR_IDR(s) method converges slowest, and oscillation of relative residual norms of MR_IDR(s) and Bi_IDR(s) methods is more gentle than that of IDR(s) method.

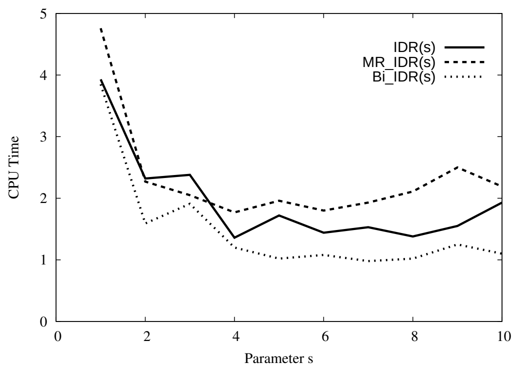


(a) matrix big

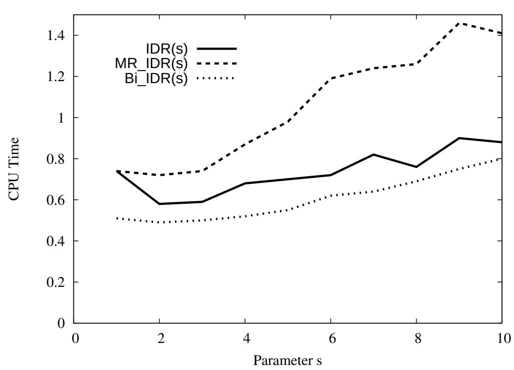


(b) matrix epb1

Figure 1: Variation of iterations of three iterative methods.



(a) matrix big



(b) matrix epb1

Figure 2: Variation of CPU Time of three iterative methods.

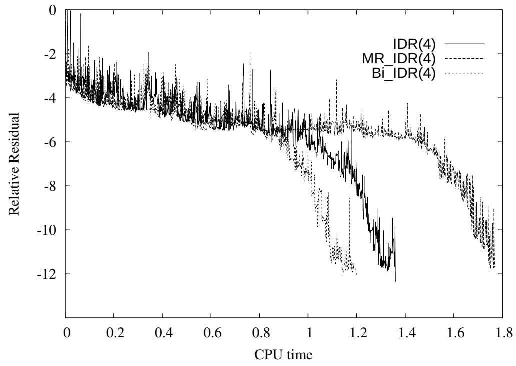
Table 2: Iterations and CPU time in seconds of three iterative methods.

matrix	method	$s_{\mathrm{opt.}}$	itr.	time	ratio [sec.]	mem. [MB]
big	IDR(s)	4	1687	1.36	1.00	2.91
	MR_IDR(s)	4	1489	1.77	1.30	3.82
	Bi_IDR(s)	7	1111	**0.98**	0.72	3.92
epb1	IDR(s)	2	818	0.58	1.00	2.49
	MR_IDR(s)	2	803	0.72	1.24	3.06
	Bi_IDR(s)	2	833	**0.49**	0.84	2.61
epb2	IDR(s)	3	450	0.68	1.00	4.99
	MR_IDR(s)	2	473	0.79	1.16	5.37
	Bi_IDR(s)	2	481	**0.54**	0.79	4.60
garon2	IDR(s)	2	777	1.26	1.00	5.76
	MR_IDR(s)	3	722	1.31	1.04	6.07
	Bi_IDR(s)	2	758	**1.12**	0.89	5.86
memplus	IDR(s)	5	574	0.67	1.00	4.36
	MR_IDR(s)	3	751	0.95	1.42	4.49
	Bi_IDR(s)	2	782	**0.62**	0.93	3.27
poisson-3da	IDR(s)	2	263	0.52	1.00	5.33
	MR_IDR(s)	4	232	0.54	1.04	6.87
	Bi_IDR(s)	4	238	**0.46**	0.88	6.05
poisson-3db	IDR(s)	5	528	15.31	1.00	41.22
	MR_IDR(s)	3	551	16.00	1.05	41.88
	Bi_IDR(s)	5	518	**14.77**	0.96	41.88
raefsky2	IDR(s)	7	420	0.40	1.00	4.05
	MR_IDR(s)	3	491	0.44	1.10	3.92
	Bi_IDR(s)	7	431	**0.38**	0.95	4.07
sme3da	IDR(s)	7	2272	**9.30**	1.00	12.64
	MR_IDR(s)	9	2088	10.01	1.08	15.02
	Bi_IDR(s)	7	2415	9.59	1.03	12.73
sme3db	IDR(s)	9	2668	45.47	1.00	31.25
	MR_IDR(s)	6	3441	56.62	1.25	32.13
	Bi_IDR(s)	4	3546	**44.53**	0.98	28.14
xenon1	IDR(s)	2	2240	12.12	1.00	18.15
	MR_IDR(s)	3	2015	12.98	1.07	21.86
	Bi_IDR(s)	4	1952	**10.74**	0.89	20.75
xenon2	IDR(s)	1	2725	**77.89**	1.00	55.66
	MR_IDR(s)	1	2847	84.88	1.09	59.27
	Bi_IDR(s)	1	2911	78.01	1.00	56.87

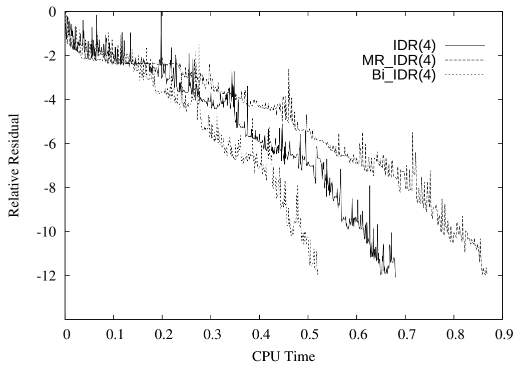
5.2 Verification of the solution vector with degraded accuracy

The solution vector of IDR(s) method sometime isn't accurate when value of s is large. Thereby, we inspect the accuracy of the solution vector of MR_IDR(s) and Bi_IDR(s) method. Test matrix is real unsymmetric Toeplitz matrix. Number of columns of Toeplitz matrix is 2000, and parameter γ is 1.5.

Fig. 4 draws variation of common logarithm of TRR(true relative residual) 2-norm of four iterative methods for matrix Toeplitz. In Fig. 4, TRR 2-norm is defined by $||\boldsymbol{b} - A\boldsymbol{x}_n||_2/||\boldsymbol{b} - A\boldsymbol{x}_0||_2$. We show variation of TRR of IDR(s) method in solid line, and IDR(s) method with the additional operation for parameter ω in chained line , MR_IDR(s) method in dashed line , Bi_IDR(s) method in



(a) matrix big



(b) matrix epb1

Figure 3: Relative residual history of three iterative methods.

dotted line. We set parameter κ for the additional computation as $\kappa = 0.7$, because the $\kappa = 0.7$ is recommended by Sleijpen and van der Vorst[4].

From Fig. 4, you can see that the accuracy of IDR(s) improves if the additional operation for parameter ω is adopted, and the solution vector of MR_IDR(s) and Bi_IDR(s) methods is more accurate than that of IDR(s) method.

6 Concluding Remarks

We overviewed MR_IDR(s) and Bi_IDR(s) methods based on IDR Theorem. Next, we evaluated performance of these methods through numerical experiments. As a result, we concluded that MR_IDR(s) method converges slower than original IDR(s) method because of high cost of evaluating intermediate residua norm. On the other hand, Bi_IDR(s) method converges faster than original IDR(s) method because of low computational cost. Furthermore, MR_IDR(s) and Bi_IDR(s) methods clearly improve the accuracy of original IDR(s) method.

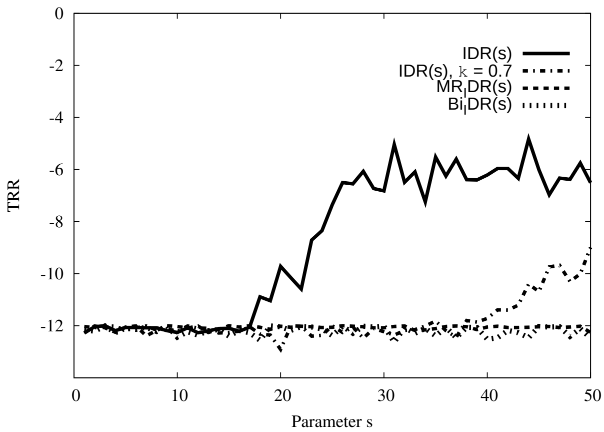


Figure 4: Variation of True Relative Residual 2-norm of four iterative methods for matrix Toeplitz.

References

[1] Davis, T. : Univ. of Florida sparse matrix collection, http://www.cise.ufl.edu/research/sparse/matrices /index.html

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

[3] Webpage of Matrix Market: http://math.nist.gov/ MatrixMarket/matrices.html

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

[5] Sonneveld, P., van Gijzen, M.B.: IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems, TR 07-07, Delft Univ. of Tech., 2007.

[6] van Gijzen, M.B., Sonneveld, P.: An elegant IDR(s) variant that efficiently exploits bi-orthogonality properties, TR 08-21, Delft Univ. of Tech., 2008.

[7] van Gijzen, M.B., Sonneveld, P.: An IDR(s) variant with minimal intermediate residual norms, The Proc. of Int. Kyoto-Forum on Krylov Subspace method, pp.85-92, 2008.

[8] Wesseling, P., Sonneveld, P.: Numerical Experiments with a Multiple Grid- and a Preconditioned Lanczos Type Methods, Lecture Notes in Math., Springer, No.771, pp.543-562, 1980.