Journal of Computational and Applied Mathematics 235 (2011) 4095–4106

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

# A block IDR(*s*) method for nonsymmetric linear systems with multiple right-hand sides

L. Du [a,*], T. Sogabe [b], B. Yu [c], Y. Yamamoto [d], S.-L. Zhang [a]

[a] *Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan*

[b] *Graduate School of Information Science and Technology, Aichi Prefectural University, Nagakute-cho, Aichi-gun, Aichi, 480-1198, Japan*

[c] *School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning, 116024, PR China*

[d] *Department of Computer Science and Systems Engineering, Kobe University, Rokkodai-cho, Nada-ku, Kobe, 657-8501, Japan*

ARTICLE INFO

*Article history:*
Received 23 March 2010
Received in revised form 18 February 2011

*Keywords:*
Block method
Multiple right-hand sides
Induced dimension reduction
IDR(*s*)
Block IDR(*s*)

ABSTRACT

The IDR(*s*) based on the induced dimension reduction (IDR) theorem, is a new class of efficient algorithms for large nonsymmetric linear systems. IDR(1) is mathematically equivalent to BiCGStab at the even IDR(1) residuals, and IDR(*s*) with $s > 1$ is competitive with most Bi-CG based methods. For these reasons, we extend the IDR(*s*) to solve large nonsymmetric linear systems with multiple right-hand sides. In this paper, a variant of the IDR theorem is given at first, then the block IDR(*s*), an extension of IDR(*s*) based on the variant IDR(*s*) theorem, is proposed. By analysis, the upper bound on the number of matrix-vector products of block IDR(*s*) is the same as that of the IDR(*s*) for a single right-hand side in generic case, i.e., the total number of matrix-vector products of IDR(*s*) may be $m$ times that of of block IDR(*s*), where $m$ is the number of right-hand sides. Numerical experiments are presented to show the effectiveness of our proposed method.



## 1. Introduction

We consider the solution of large and sparse linear systems with multiple right-hand sides of the form

$$AX = B, \tag{1}$$

where the coefficient matrix $A$ is a nonsingular real matrix of order $n$, $X = [x_1, x_2, \ldots, x_m]$ and $B = [b_1, b_2, \ldots, b_m] \in \mathbb{R}^{n\times m}$ with usually $m \ll n$.

It is obvious that we can get the solution of (1) by solving each of the $m$ linear systems independently. In this case, direct methods based on sparse LU factorizations may be a good choice on the premise of low-cost decomposition, because the LU decomposition needs to be performed only once and then solving linear systems with upper (lower) triangular matrices that the cost of which is low. But, for the most part, especially when the matrix is not explicitly available, iterative methods are the only choice. In recent years, many iterative methods have been proposed based on the Krylov subspace, and generalization of these methods are used to solve the multiple right-hand sides problem by taking the advantage of the fact that the linear systems share the same coefficient matrix.

One class of solvers for solving the problem (1) is the seed methods, which consist of selecting a single system as the seed system and generating the corresponding Krylov subspace and then projecting all the residuals of the other linear

* Corresponding author.
*E-mail addresses:* lei-du@na.cse.nagoya-u.ac.jp, du3stone@gmail.com (L. Du), sogabe@ist.aichi-pu.ac.jp (T. Sogabe), yubo@dlut.edu.cn (B. Yu), yamamoto@cs.kobe-u.ac.jp (Y. Yamamoto), zhang@na.cse.nagoya-u.ac.jp (S.-L. Zhang).

0377-0427/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2011.02.035

systems onto the same Krylov subspace to find new approximate solutions as initial approximations. References on this class include [1–3].

Another class is the global methods, which are based on the use of a global projection process onto a matrix Krylov subspace, including global FOM and GMRES methods [4], global BCG and BiCGStab methods [5,6], global Hessenberg and CMRH methods [7] and weighted global methods [8]. Global methods are more suitable for sparse linear systems [8].

The other class is the block solvers which are much more efficient when the matrix $A$ is relatively dense and preconditioners are used. The first block solvers are block conjugate gradient (Bl-CG) algorithm and block biconjugate gradient (Bl-BCG) algorithm proposed in [9]. Variable Bl-CG algorithms for symmetric positive definite problems are implemented on parallel computers [10]. For nonsymmetric problems, the Bl-BCG algorithm [9,11], the block generalized minimal residual (Bl-GMRES) algorithm [12–14], the block quasi minimum residual (Bl-QMR) algorithm [15], the block BiCGStab (Bl-BICGSTAB) algorithm [16], the block Lanczos method [17] and the block least squares (Bl-LSQR) algorithm [18] have been developed.

There are some other methods for the multiple right-hand sides problems. Based on the block and global Arnoldi algorithm, Skew-symmetric methods were proposed in [19]. Single-seed and block-seed projection approaches which are based on the QMR and block QMR algorithms were proposed for non-hermitian systems with multiple right-hand sides in [20]. The hybrid algorithm MHGMRES was presented in [21]. Related work such as Lanczos method with multiple starting vectors for Padé approximation, (see [22] and references therein), has been done.

The IDR($s$) was recently developed in [23], which is based on the induced dimension reduction (IDR) method proposed in [24]. It was claimed in [25,23,26] that IDR(1) $\approx$ IDR is mathematically equivalent to BiCGStab at the even IDR residuals, and IDR($s$) with $s > 1$ related to ML(s)BiCGStab [27] is competitive with most Bi-CG based methods. A new IDR($s$) variant by imposing bi-orthogonalization conditions was developed in [28] which was named IDR($s$)Bio in [29]. But, for strongly nonsymmetric problems, especially for skew-symmetric or nearly skew-symmetric matrices, IDR($s$) also meets the breakdown problem like BiCGStab, exploiting the merit of BiCGStab($\ell$) [30], IDRStab [31] and GBi-CGStab($s$, L) [32] were proposed with higher order stabilization polynomials.

In this paper, we generalize the IDR($s$) to solve linear systems with multiple right-hand sides. The proposed method is referred to as block IDR($s$) (Bl-IDR($s$)).

The remainder of the paper is organized as follows. In the next section, we review the block Krylov subspace and IDR($s$). Then, we will give a variant of the IDR theorem, which is a generalization of the IDR theorem and a theoretical basis of our block IDR($s$) algorithm, and analyze the block method in Section 3. In Section 4, some numerical results are presented to show the effectiveness of the proposed method. Finally, we make some concluding remarks in Section 5.

Throughout this paper the following notation is used. Uppercase letters $M_{n\times n}$, $N_{n\times m}$ denote matrices. If the dimension of a matrix is apparent from the context and there is no confusion, we will drop the index and denote $M_{n\times n}$ by $M$. $I$ represents the identify matrix, $M^H$ is the hermitian transpose of $M$. $\|M\|_F = \sqrt{\mathrm{Tr}(M^H M)}$ where $\|\cdot\|_F$ and $\mathrm{Tr}(\cdot)$ denote the Frobenius norm and trace of the square matrix, respectively. $\|\cdot\|$ is the Euclidean norm throughout the paper unless otherwise stated. $\mathcal{N}(M)$ indicates the nullspace of matrix $M$. The notations of MATLAB style are used, element of $M$ at the $i$th row and $j$th column is $M(i, j)$, the $k$th column of $M$ is specified by a "colon" notation $M(:, k)$ and $M(:, i : j) = [M(:, i), M(:, i + 1), \ldots, M(:, j)]$ is a submatrix of $M$ formed by $j - i + 1$ columns.

## 2. The block Krylov subspace and IDR($s$)

In this section, we recall some fundamental properties of block Krylov subspace from [33,34] and review the IDR($s$) from [23].

### 2.1. Block Krylov subspace

Let $X_0 \in \mathbb{R}^{n\times m}$ be an initial guess and $R_0 = B - AX_0$ be the corresponding block residual matrix.

**Definition 2.1** ([34]). Subspace $\mathcal{K}_k(A, R_0)$ generated by $A$ and increasing powers of $A$ applied to $R_0$

$$\mathcal{K}_k(A, R_0) := \left\{ \sum_{i=0}^{k-1} A^i R_0 \gamma_i;\ \gamma_i \in \mathbb{R}^{m\times m} \right\} \tag{2}$$

is called the block Krylov subspace.

We should not confuse the block Krylov subspace with the matrix Krylov subspace, which is defined as follows.

**Definition 2.2** ([4]). Subspace $\mathbb{K}_k(A, R_0)$ generated by $A$ and increasing powers of $A$ applied to $R_0$

$$\mathbb{K}_k(A, R_0) := \left\{ \sum_{i=0}^{k-1} \alpha_i A^i R_0;\ \alpha_i \in \mathbb{R} \right\}$$

is called the matrix Krylov subspace.

Methods that are used to find approximate solutions $X_k \in X_0 + \mathcal{K}_k(A, R_0)$ are called block methods, the choice $X_k \in X_0 + \mathbb{K}_k(A, R_0)$ leads to the so-called global methods. If we choose $\gamma_i = \alpha_i I_{m\times m}$ $(i = 0, 1, \ldots, k-1)$, $\mathbb{K}_k$ and $\mathcal{K}_k$ can be the same subspace. From this point of view, the matrix Krylov subspace $\mathbb{K}_k(A, R_0)$ can be considered as a subspace of the block Krylov subspace $\mathcal{K}_k(A, R_0)$, i.e. $\mathbb{K}_k(A, R_0) \subset \mathcal{K}_k(A, R_0)$.

For block solvers, let $Z = [z_1, z_2, \ldots, z_m] \in \mathcal{K}_k$, where $z_i \in \mathbb{R}^n (i = 1, \ldots, m)$. From the definition of (2), there are $\gamma_j$'s $\in \mathbb{R}^{m\times m} (j = 0, \ldots, k-1)$ that satisfy

$$Z = \sum_{j=0}^{k-1} A^j R_0 \gamma_j,$$

which implies that

$$z_i = \sum_{l=1}^{m} \sum_{j=0}^{k-1} \gamma_j(l, i) A^j R_0(:, l) \in \mathcal{B}_k(A, R_0),$$

where $\mathcal{B}_k(A, R_0) := \mathcal{K}_k(A, R_0(:, 1)) + \cdots + \mathcal{K}_k(A, R_0(:, m))$.

So, the columns of $X_k = X_0 + Z \in \mathcal{K}_k$ correspond to the approximate solutions of the $m$ single right-hand linear systems. But, unlike the standard Krylov solvers, the search space of block Krylov solvers for each right-hand side is much larger, i.e., approximate solutions $X_k(:, l) = X_0(:, l) + \mathcal{B}_k(A, R_0)$ are updated instead of $X_k(:, l) = X_0(:, l) + \mathcal{K}_k(A, R_0(:, l))$. This is the main reason for using block methods.

Similar to the Krylov subspace, a generalization of the grade was introduced in [35].

**Definition 2.3** ([33,35]). The positive integer $v := v(R_0, A)$ defined by

$$\begin{aligned} v(R_0, A) &:= \min\{k | \dim\ \mathcal{B}_k(A, R_0) = \dim\ \mathcal{B}_{k+1}(A, R_0)\} \\ &= \min\{k | \mathcal{B}_k(A, R_0) = \mathcal{B}_{k+1}(A, R_0)\} \end{aligned}$$

is called block grade of $R_0$ with respect to $A$.

**Corollary 2.4** ([33]). *Let $X_*$ be the exact block solution of $AX = B$, then*

$$X_* \in X_0 + \mathcal{K}_{v(R_0,A)}(A, R_0).$$

### 2.2. IDR(s)

Since the IDR($s$) is mainly based on the Induced Dimension Reduction theorem [23], which is a generalization of the original IDR theorem [24] to complex case, we first review the theorem.

**Theorem 2.5** (*IDR*). *Let $A \in \mathbb{C}^{n\times n}$, $v_0 \in \mathbb{C}^n$, and $\mathcal{G}_0 = \mathcal{K}_v(A, v_0)$. Let $S \subset \mathbb{C}^n$ and define the recursive subspace $\mathcal{G}_j$ as*

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap S), \quad (\omega_j \neq 0) \in \mathbb{C},\ j = 1, 2, \ldots.$$

*If $\mathcal{G}_0 \cap S$ does not contain any eigenvector of $A$, then the following hold:*

(a) $\mathcal{G}_j \subset \mathcal{G}_{j-1}, \forall j > 0$.
(b) $\mathcal{G}_j = \{0\}$ *for some* $j \leq n$.

**Proof.** See [23]. □

From the theorem, we know that the dimension of the series nested subspaces $\mathcal{G}_j$ diminishes with the shrinking of the sequence subspaces $\mathcal{G}_j$. If all the residual $r_i$s can be constructed in the nested subspaces $\mathcal{G}_j$, we may get the approximate solution in finite steps. At most $n + \frac{n}{s}$ matrix-vector products will be needed in the generic case for the IDR($s$) method [23].

The $s + 2$ term IDR($s$) algorithm can be derived as a translation of the IDR theorem as follows.

In order to construct $r_{i+1} \in \mathcal{G}_{j+1}$, let

$$r_{i+1} = (I - \omega_{j+1} A) v_i \quad v_i \in \mathcal{G}_j \cap S \tag{3}$$

and we can choose

$$v_i = r_i - \sum_{j=1}^{s} \gamma_j \Delta r_{i-j}, \quad \text{where } \Delta r_k := r_{k+1} - r_k. \tag{4}$$

To initialize $r_1, \ldots, r_s$, any fairly iterative methods can be used, e.g., BiCGStab.

Then, from Eqs. (3) (4), approximate solution $x_{i+1}$ can be updated as

$$x_{i+1} = x_i + \omega_{j+1} v_i - \sum_{j=1}^{s} \gamma_j \Delta x_{i-j}, \quad \text{where } \Delta x_k := x_{k+1} - x_k. \tag{5}$$

In order to determine the $s$ variables $\gamma_j$, the space $S$ can be chosen to be the left Null space of some $n \times s$ matrix $P = [p_1, \ldots, p_s]$, i.e., $S = \mathcal{N}(P^H)$, which can be generated randomly, since the probability that space $S \cap \mathcal{G}_0$ contains some eigenvector(s) of $A$ is zero [23]. Then $\gamma_j$ can be solved from equation

$$P^H v_i = 0.$$

There is one more parameter $\omega_{j+1}$ that should be computed prior to forming the whole algorithm, and it was suggested in [23] to choose the $\omega$ by minimizing the norm of residual $r_{i+1}$ every $s+1$ steps.

Putting all the relations together, the IDR($s$) algorithm [23] can be summarized in Algorithm 1.

**Algorithm 1** IDR($s$) algorithm

$r_0 \Leftarrow b - Ax_0$; $P \in \mathbb{C}^{n\times s}$;
(Initial $r_0, \ldots, r_s \in \mathcal{G}_0$)
**for** $i = 1$ to $s$ **do**
  $v \Leftarrow Ar_{i-1}$; $\omega \Leftarrow \frac{v^H r_{i-1}}{v^H v}$;
  $\Delta X(:, i) \Leftarrow \omega r_{i-1}$; $\Delta R(:, i) \Leftarrow -\omega v$;
  $x_i \Leftarrow x_{i-1} + \Delta X(:, i)$; $r_i \Leftarrow r_{i-1} + \Delta R(:, i)$;
**end for**
$j \Leftarrow 1$; $i \Leftarrow s$;
$M \Leftarrow P^H \Delta R$; $h \Leftarrow P^H r_i$;
**while** $\frac{\|r_i\|}{\|b\|} > \epsilon$ **do**
  **for** $k = 0$ to $s$ **do**
    Solve $c$ from $Mc = h$;
    $q \Leftarrow -\Delta R c$;
    $v \Leftarrow r_i + q$;
    (Enter the next subspace $\mathcal{G}_l$)
    **if** $k == 0$ **then**
      $t \Leftarrow Av$; $\omega \Leftarrow \frac{t^H v}{t^H t}$;
      $\Delta R(:, j) \Leftarrow q - \omega t$;
      $\Delta X(:, j) \Leftarrow -\Delta X c + \omega v$;
    **else**
      $\Delta X(:, j) \Leftarrow -\Delta X c + \omega v$;
      $\Delta R(:, j) \Leftarrow -A\Delta X(:, j)$;
    **end if**
    (Update approximate solutions $x_i$)
    $r_{i+1} \Leftarrow r_i + \Delta R(:, j)$;
    $x_{i+1} \Leftarrow x_i + \Delta X(:, j)$;
    $\delta m \Leftarrow P^H \Delta R(:, j)$;
    $M(:, j) \Leftarrow \delta m$;
    $h \Leftarrow h + \delta m$;
    $i \Leftarrow i + 1$; $j \Leftarrow j + 1$;
    $j \Leftarrow (j-1)\%s + 1$;(% is the Modulo operation)
  **end for**
**end while**

Two kinds of breakdown may occur during the course of algorithm's execution. The first breakdown happens when the smaller $s \times s$ linear system $Mc = h$ is (nearly) inconsistent. The second appears if the parameter $\omega$ is equal to (or close to) zero, which leads to stagnation of the algorithm.

## 3. The block IDR($s$)

In this section we consider the nonsymmetric linear systems with multiple right-hand sides. In order to propose the block version of IDR($s$), we first give a variant of the IDR theorem, which is an extension of IDR theorem in Section 3.1. For giving an executable procedure from the variant IDR theorem, we discuss the implementation details and formulate the proposed algorithm in Sections 3.2 and 3.3, respectively. Then, in Section 3.4 we compare the computational cost and memory requirements between IDR($s$) and block IDR($s$), and analyze the convergence counted by matrix-vector operations. Finally, a preconditioned version of block IDR($s$) algorithm is presented in Section 3.5.

### 3.1. A variant of the IDR theorem

From the block initial residual $R_0$ and coefficient matrix $A$, the block Krylov subspace can be generated. A variant of the IDR theorem can be naturally obtained based on the block Krylov subspace. In this section, we give the variant IDR theorem at first, which is the theoretical basis of our proposed method.

**Theorem 3.1.** *Let $A \in \mathbb{C}^{n\times n}$, $R_0 \in \mathbb{C}^{n\times m}$, and $\mathcal{G}_0 = \mathcal{B}_{v(R_0,A)}(A, R_0)$. Let $\mathcal{S}$ be any proper subspace of $\mathbb{C}^n$ and define the recursive subspace $\mathcal{G}_j$ as*

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{S}), \quad (\omega_j \neq 0) \in \mathbb{C},\ j = 1, 2, \ldots.$$

*If $\mathcal{G}_0$ and $\mathcal{S}$ do not share a nontrivial invariant subspace of $A$, then the following hold:*

(a) $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, *for all* $j > 0$.
(b) $\mathcal{G}_j = \{0\}$ *for some* $j \leq n$.

**Proof.** In the original IDR theorem [23], $\mathcal{G}_0$ is a complete Krylov subspace $\mathcal{K}_n(A, v)$ for a single vector $v \in \mathbb{C}^n$. However, the only properties of $\mathcal{G}_0$ that are used in the original proof are that $A\mathcal{G}_0 \subset \mathcal{G}_0$ and $\dim(\mathcal{G}_0) \leq n$. These properties are shared by $\mathcal{B}_{v(R_0,A)}(A, R_0)$. See [23] for details. □

### 3.2. Implementation details

Because the block IDR($s$) is a natural extension of the IDR($s$), The $s+2$ term block IDR($s$) method can be derived analogous to Eqs. (3)–(5) as a translation of the variant IDR theorem.

Assume that all of the column vectors of $R_{i-s}, \ldots, R_i$ belong to $\mathcal{G}_j$. Then we can construct block residual $R_{i+1}$ whose column vectors belong to $\mathcal{G}_{j+1}$, by setting

$$R_{i+1} = (I - \omega_{j+1}A)V_i,$$

where $V_i$ is an $n \times m$ matrix whose column vectors belong to $\mathcal{G}_j \cap \mathcal{S}$. To obtain such $V_i$, assume that the subspace $\mathcal{S}$ can be written as $\mathcal{S} = \mathcal{N}(P^H)$ for some $n \times sm$ matrix $P$. Let

$$V_i = R_i - \sum_{l=1}^{s} \Delta R_{i-l}\gamma_l, \quad \text{where } \Delta R_k := R_{k+1} - R_k.$$

Then, the condition for $V_i$ can be written as

$$P^H V_i = 0.$$

The $m \times m$ matrices $\gamma_1, \ldots, \gamma_s$ can be obtained by solving this equation, and the approximate solution can be updated as follows

$$X_{i+1} = X_i + \omega_{j+1}V_i - \sum_{l=1}^{s} \Delta X_{i-l}\gamma_l, \quad \text{where } \Delta X_k := X_{k+1} - X_k.$$

We compute the scalar parameter $\omega_{j+1}$ by minimizing the Frobenius norm of the block residual $R_{i+1} = (I - \omega_{j+1}A)V_i$, i.e., $\omega_{j+1} = \arg\min_\omega \|R_{i+1}\|_F$ which implies that

$$\omega_{j+1} = \frac{\mathrm{Tr}(T^H V_i)}{\mathrm{Tr}(T^H T)}, \quad \text{where } T = AV_i.$$

Finally, we can obtain the following block IDR($s$) algorithm.

### 3.3. The block IDR(s) algorithm

Here, we give the unpreconditioned block IDR($s$) algorithm in Algorithm 2.

In the extreme case that the number of right-hand sides $m = 1$, Algorithms 1 and 2 would perform identically. Meanwhile, Algorithm 2 can be implemented easily if the codes of IDR($s$) are in one's hands.

Like IDR($s$), Algorithm 2 may also meet the two kinds of breakdown. Meanwhile, like other block methods, the deflation procedure can be used for block IDR($s$) to detect and delete (almost) linearly dependent vectors in the block Krylov subspaces. Because the approximate solutions for the multiple right-hand sides may converge at different rates of the block iteration, it is also necessary to detect and then deflate the converged systems. We will not consider deflations in this paper. More details can be found in [15,36].

**Algorithm 2** Unpreconditioned block IDR($s$) algorithm

$R_0 \Leftarrow B - AX_0$; $P \in \mathbb{C}^{n\times sm}$;
**for** $i = 0$ to $s-1$ **do**
  $V \Leftarrow AR_i$; $\omega \Leftarrow \frac{\mathrm{Tr}(V^H R_i)}{\mathrm{Tr}(V^H V)}$;
  $\Delta X(:, im+1:(i+1)m) \Leftarrow \omega R_i$; $\Delta R(:, im+1:(i+1)m) \Leftarrow -\omega V$;
  $X_{i+1} \Leftarrow X_i + \Delta X(:, im+1:(i+1)m)$; $R_{i+1} \Leftarrow R_i + \Delta R(:, im+1:(i+1)m)$;
**end for**
$j \Leftarrow 1$; $i \Leftarrow s$;
$M \Leftarrow P^H \Delta R$; $h \Leftarrow P^H R_i$;
**while** $\max_{j=1:m} \frac{\|R_i(:,j)\|}{\|B_0(:,j)\|} > \epsilon$ **do**
  **for** $k = 0$ to $s$ **do**
    Solve $C$ from $MC = h$;
    $Q \Leftarrow -\Delta R C$;
    $V \Leftarrow R_i + Q$;
    **if** $k == 0$ **then**
      $T \Leftarrow AV$; $\omega \Leftarrow \frac{\mathrm{Tr}(T^H V)}{\mathrm{Tr}(T^H T)}$;
      $\Delta R(:, (j-1)m+1:jm) \Leftarrow Q - \omega T$;
      $\Delta X(:, (j-1)m+1:jm) \Leftarrow -\Delta X C + \omega V$;
    **else**
      $\Delta X(:, (j-1)m+1:jm) \Leftarrow -\Delta X C + \omega V$;
      $\Delta R(:, (j-1)m+1:jm) \Leftarrow -A\Delta X(:, (j-1)m+1:jm)$;
    **end if**
    $R_{i+1} \Leftarrow R_i + \Delta R(:, (j-1)m+1:jm)$;
    $X_{i+1} \Leftarrow X_i + \Delta X(:, (j-1)m+1:jm)$;
    $\Delta m \Leftarrow P^H \Delta R(:, (j-1)m+1:jm)$;
    $M(:, (j-1)m+1:jm) \Leftarrow \Delta m$;
    $h \Leftarrow h + \Delta m$;
    $i \Leftarrow i+1$; $j \Leftarrow j+1$;
    $j \Leftarrow (j-1)\%s + 1$; (% is the Modulo operation)
  **end for**
**end while**

**Table 1**
Computational cost and memory requirement for every $s+1$ block IDR($s$) and IDR($s$) steps.

| Operations | Bl-IDR($s$) | $m \times$ IDR($s$) | Ratio |
|---|---|---|---|
| MVs | $(s+1)m$ | $(s+1)m$ | 1 |
| DOTs | $s^2m^2 + sm^2 + 2m$ | $(s^2+s+2)m$ | $m + \frac{2-2m}{s^2+s+2}$ |
| AXPYs | $2s^2m^2 + 2sm^2 + \frac{3}{2}sm + \frac{5}{2}m$ | $(2s^2 + \frac{7}{2}s + \frac{5}{2})m$ | $m + \frac{(1-m)(3s+5)}{4s^2+7s+5}$ |
| Memory | $(3s+5)m$ | $3s+5$ | $m$ |

### 3.4. Computational cost and memory requirement

We compare the computational cost and memory requirement for every $s+1$ steps of block IDR($s$) and IDR($s$) for all right-hand sides in Table 1. Here, MVs denotes the matrix-vector products, AXPYs are the vector updates as $y = \alpha x + y$, a scalar operation of a vector and an addition of two vectors are counted as a half update, DOTs means the inner product. The memory requirements include storage for the right-hand sides and the solutions, but excluding storage for the coefficient matrix and preconditioner.

From Table 1, it is clear that the cost of block IDR($s$) is almost $m$ times higher than $m\times$ IDR($s$), but the number of matrix-vector products is the same. This means that block IDR($s$) may work better than IDR($s$) under the conditions of the relative smaller iterative steps to IDR($s$) for all the right-hand sides, and expensive matrix-vector products which implies that the coefficient matrix should be relative dense. From the expansion process of the block Krylov subspace, we know that the searching subspace for approximate solutions is as big as the sum of all the single Krylov subspaces. It may be much bigger than every single one, and this is also one of the potential advantages of block Krylov solvers.

By the extended IDR theorem [23], we know that IDR($s$) will reach the exact solution at most $n + \frac{n}{s}$ matrix-vector products in exact arithmetic and the generic case which means the dimension reduction of $\mathcal{G}_j$ per $s+1$ matrix-vector products is $s$ throughout the process. An analogous theorem for the block Algorithm 2 is given next.

**Theorem 3.2.** *Let $A$ be any matrix in $\mathbb{C}^{n\times n}$, let $P$ be an $n \times sm$ matrix whose columns are linearly independent, let $\mathcal{G}_0 = \mathcal{B}_{v(R_0,A)}(A, R_0)$, and let the sequence of spaces $\{\mathcal{G}_j,\ j = 1, 2, \ldots\}$ be defined by*

$$\mathcal{G}_j = (I - \omega_j A)(\mathcal{G}_{j-1} \cap \mathcal{N}(P^H)), \tag{6}$$

*where $\omega_j$ are nonzero numbers such that $I - \omega_j A$ is nonsingular. Let $d_j = \dim(\mathcal{G}_j)$, then the sequence $\{d_j,\ j = 0, 1, \ldots\}$ is monotonically non-increasing and satisfies*

$$0 \le d_j - d_{j+1} \le d_{j-1} - d_j \le sm. \tag{7}$$

**Proof.** This theorem is a slightly modified version of Theorem 2 in [23], where the definition of $\mathcal{G}_0$ is changed and $s$ is replaced with $sm$. By examining the proof of Theorem 2 in [23], we know that the only property of the sequence of subspaces $\{\mathcal{G}_j, j = 1, 2, \ldots\}$ that is used in the proof is $\mathcal{G}_j \subset \mathcal{G}_{j-1}$. As can be seen easily from the proof of Theorem 1 in [23], this holds if $\mathcal{G}_j$ is generated by Eq. (6) and $A\mathcal{G}_0 \subset \mathcal{G}_0$. But the latter condition is clearly satisfied by $\mathcal{G}_0 = \mathcal{B}_{v(R_0,A)}(A, R_0)$. Hence the proof of Theorem 2 in [23] is valid in our case and the Eq. (7) holds. □

By Theorem 3.2, we know that the dimension reduction every $s + 1$ steps is at most $s \times m$. Although this cannot always hold, we still give an estimation: assuming the dimension reduction of $\mathcal{G}_j$ is in the generic case, i.e., the dimension reduction is $sm$ at every $s + 1$ steps, and thus the number of matrix-vector products to compute the solution is at most

$$m(s + 1) \cdot \frac{n}{sm} = n + \frac{n}{s}$$

which is the same as the IDR($s$) for linear systems with a single right-hand side. From this point of view, the block IDR($s$) will reach the exact solution in less steps, and may be more efficient than IDR($s$).

As we know, the special case IDR(1) is mathematically equivalent to BiCGStab at the even IDR residuals [25,23,26]. For the block methods, we can draw a conclusion that block IDR(1) can be also mathematically equivalent to the block BiCGStab.

### 3.5. Preconditioning

It is known to all that preconditioning is essential for the successful use of iterative methods. A suitable preconditioner can make the iteration converge rapidly. We consider the split preconditioned system

$$\tilde{A}\tilde{X} = \tilde{B} \tag{8}$$

with a preconditioner $K = K_1K_2 \approx A$. where $\tilde{A} = K_1^{-1}AK_2^{-1}$, $\tilde{X} = K_2X$ and $\tilde{B} = K_1^{-1}B$. Both the left and right preconditioning can be got in (8) by setting $K_2 = I$ or $K_1 = I$ respectively. If we use the block IDR($s$) Algorithm 2 for (8) with the changes of variables:

$\tilde{R}_k = K_1^{-1}R_k, \tilde{X}_k = K_2X_k$ etc., we can obtain the following preconditioned block IDR($s$) algorithm in Algorithm 3.

A remarkable observation is that we do not need to sedulously deal with the matrix $P$ in Algorithm 3, if it was generated randomly in the unpreconditioned Algorithm 2.

## 4. Numerical experiments

In this section, we give some numerical results concerning the IDR($s$), the block BiCGStab, and the block IDR($s$). Experiment 1 shows the cost of number of matrix-vector multiplications, CPU time and histories of residual corresponding to different number of multiple right-hand sides. The results of other problems with a constant number of multiple right-hand sides are presented in Experiment 2.

All the numerical experiments were performed on a Mac OS X (2.4 GHz) with double precision arithmetic. Codes were written in standard C++ and were compiled using GCC 4.2.1. BLAS 1-3 were used for our implementation [37]. The right-hand sides $B = rand(n, m)$ and matrix $P = rand(n, sm)$, where the *rand* function generated two random matrices with the size of $n \times m$ and $n \times sm$, respectively. Elements of the random matrices were distributed in (0, 1). For all the examples, the initial guess $X_0 = 0$ was taken to be the zero matrix and the stopping criterion was used as $\max_{i=1:m} \frac{\|R_k(:,i)\|}{\|B_0(:,i)\|} < 10^{-8}$. Parameter $s$ in IDR($s$) and block IDR($s$) was set as 4. The preconditioner ILU(0), incomplete LU factorization of the coefficient matrix, was used as left preconditioning in our experiments. A dagger † denotes the corresponding method did not converge within $2n$ matrix-vector products.

All test problems are from Matrix Market [38] and the University of Florida Sparse Matrix Collection [39]. Matrices used in the experiments come from Combinatorial problem(CAG_mat1916), Computational fluid dynamics (CDDE1, CDDE3, EX22, EX25, EX28, EX40, POISSON2D, RAEFSKY1, RAEFSKY2), Finite element modeling (FIDAP001, FIDAP022), Astrophysics (MCFE), Oil reservoir simulation (ORSIRR_1, ORSREG_1), Model reduction (PISTON), Circuit simulation (RAJAT12), Oil reservoir modeling (SAYLR4, SHERMAN2, SHERMAN4, SHERMAN5), Nuclear physics (UTM1700A), Semiconductor device problem sequence(WANG1) and Petroleum engineering (WATT_1).

**Algorithm 3** Preconditioned block IDR($s$) algorithm.

$R_0 \Leftarrow B - AX_0$; $X_0 \Leftarrow KX_0$; $P \in \mathbb{C}^{n\times sm}$;
**for** $i = 0$ to $s-1$ **do**
  $W = K^{-1}R_i$
  $V \Leftarrow AW$; $\omega \Leftarrow \frac{\mathrm{Tr}((K_1^{-1}V)^H K_1^{-1}R_i)}{\mathrm{Tr}((K_1^{-1}V)^H K_1^{-1}V)}$;
  $\Delta X(:, im+1:(i+1)m) \Leftarrow \omega R_i$; $\Delta R(:, im+1:(i+1)m) \Leftarrow -\omega V$;
  $X_{i+1} \Leftarrow X_i + \Delta X(:, im+1:(i+1)m)$; $R_{i+1} \Leftarrow R_i + \Delta R(:, im+1:(i+1)m)$;
**end for**
$j \Leftarrow 1$; $i \Leftarrow s$;
$M \Leftarrow P^H \Delta R$; $h \Leftarrow P^H R_i$;
**while** $\max_{j=1:m} \frac{\|R_i(:,j)\|}{\|B_0(:,j)\|} > \epsilon$ **do**
  **for** $k = 0$ to $s$ **do**
    Solve $C$ from $MC = h$;
    $Q \Leftarrow -\Delta RC$;
    $V \Leftarrow R_i + Q$;
    **if** $k == 0$ **then**
      $T \Leftarrow AK^{-1}V$; $\omega \Leftarrow \frac{\mathrm{Tr}((K_1^{-1}T)^H K_1^{-1}V)}{\mathrm{Tr}((K_1^{-1}T)^H K_1^{-1}T)}$;
      $\Delta R(:, (j-1)m+1:jm) \Leftarrow Q - \omega T$;
      $\Delta X(:, (j-1)m+1:jm) \Leftarrow -\Delta XC + \omega V$;
    **else**
      $\Delta X(:, (j-1)m+1:jm) \Leftarrow -\Delta XC + \omega V$;
      $\Delta R(:, (j-1)m+1:jm) \Leftarrow -AK^{-1}\Delta X(:, (j-1)m+1:jm)$;
    **end if**
    $R_{i+1} \Leftarrow R_i + \Delta R(:, (j-1)m+1:jm)$;
    $X_{i+1} \Leftarrow X_i + \Delta X(:, (j-1)m+1:jm)$;
    $\Delta m \Leftarrow P^H \Delta R(:, (j-1)m+1:jm)$;
    $M(:, (j-1)m+1:jm) \Leftarrow \Delta m$;
    $h \Leftarrow h + \Delta m$;
    $i \Leftarrow i+1$; $j \Leftarrow j+1$;
    $j \Leftarrow (j-1)\%s + 1$; (% is the Modulo operation)
  **end for**
**end while**
$X_i \Leftarrow K^{-1}X_i$;

### 4.1. Experiment 1

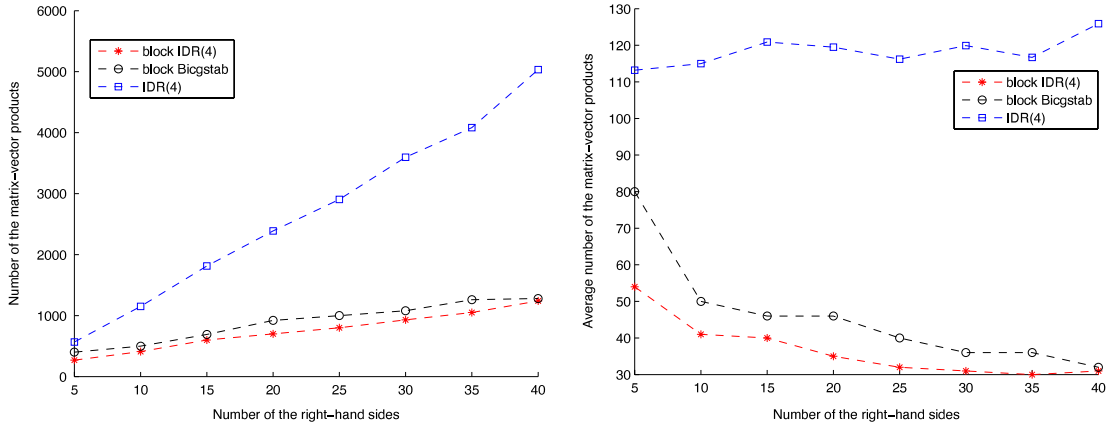
In this example, we illustrated the behavior of IDR($s$), block IDR($s$), and block BiCGStab on different right-hand sides using the test matrix UTM1700A with preconditioning.

Fig. 1 shows the total and average number of matrix-vector products of each method to solve the linear systems with corresponding multiple right-hand sides. For example, in Fig. 1(b), the average number of matrix-vector products for the block BiCGStab method at $m = 5$ is 80, which means the block BiCGStab method required a total of 400 matrix-vector products to obtain the satisfied solutions of the linear systems corresponding to Fig. 1(a).

From Fig. 1, we have the following observations: first, the block methods required fewer matrix-vector products than the IDR($s$) to solve the $m$ linear systems one after another, which implies that block methods will need fewer iterations; second, with the increasing number of multiple right-hand sides, the average numbers of matrix-vector products for both block IDR($s$) and block BiCGStab methods are monotonic decreasing, and less than half of IDR($s$); third, the block IDR($s$) needs fewer matrix-vector products than the block BiCGStab method.

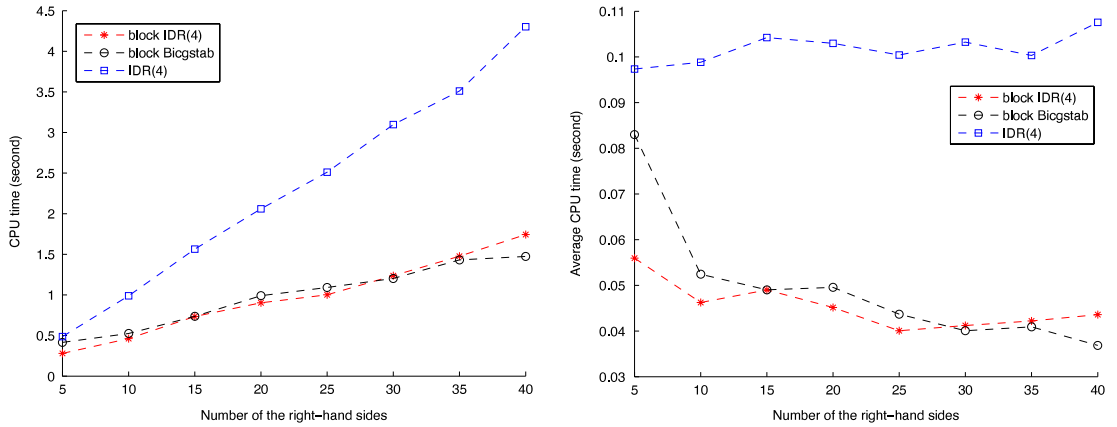
It is not objective if only the matrix-vector products are used as the criterion to measure the three methods. As we know, under the condition of the same matrix-vector products, the DOTs (inner operations) and AXPYs (vector updates) operations of block methods are more costly than the non-block methods. So, we also report the corresponding CPU time ($s$) in Fig. 2.

We can make the following observations from Fig. 2: first, the block methods can solve all the linear systems more effectively than the IDR($s$) to solve the $m$ linear systems one by one, block methods save almost half of the CPU time; second, with the increasing number of multiple right-hand sides, the average CPU time of block IDR($s$) is less first and then more than block BiCGStab method. From Table 1, we have known that operations except matrix-vector products will need more computation with a larger number of multiple right-hand sides, which will reduce the computing proportion of matrix-vector products. This is the reason why the average CPU time does not decrease, even though the average number of matrix-vector products is monotonic decreasing.




(a) Total number. (b) Average number.

**Fig. 1.** Number of the matrix-vector products versus the number of multiple right-hand sides.




(a) Total CPU time. (b) Average CPU time.

**Fig. 2.** CPU time versus the number of multiple right-hand sides.

The histories of the maximum relative residual 2-norm for two specified linear systems: $m = 10$ and $m = 20$ are reported in Fig. 3.

### 4.2. Experiment 2

Next, we compare the block IDR($s$) with the standard IDR($s$) and block BiCGStab on many other test matrices. For all these problems the number of right-hand sides was $m = 10$. Table 2 reports on results obtained for standard IDR($s$), block IDR($s$) and block BiCGStab without preconditioning. Comparisons between standard IDR($s$) and block IDR($s$), block IDR($s$) and block BiCGStab are reported in Tables 3 and 4, respectively.

Without using preconditioning technique, as we can see from Table 2 that none of the three methods can solve all the problems in the given steps. In this case, standard IDR($s$) applied to each single right-hand side linear system is more effective than block methods. but for block methods, the block IDR($s$) seems much more effective than block BiCGStab.

We use the ratio indicator $T(\mathrm{M1})/T(\mathrm{M2})$ to compare the effectiveness between methods M1 and M2, where $T$ can be the total number of matrix-vector products or total CPU time. By definition, we know that method M1 is more effective than M2 when the ratio of CPU time is less then one. Comparisons of number of matrix-vector products and CPU time between standard IDR($s$) and block IDR($s$) are given in Table 3. When a preconditioner was used, the IDR($s$) and block IDR($s$) can reach the approximate solutions for all problems. From Table 3 we see that, block IDR($s$) is less expensive than standard IDR($s$) to solve each single right-hand side linear system, both the iterations and computational time are saved. For example, in order to solve our test problem corresponding to matrix CAG_mat1916, the cost of matrix-vector products and CPU time for block IDR($s$) takes 50% and 51.1% of the standard IDR($s$), respectively.

In Table 4, we evaluate the performance of the two block solvers: block IDR($s$) and block BiCGStab with preconditioning. With respect to the number of matrix-vector products, block IDR($s$) required fewer matrix-vector products than block

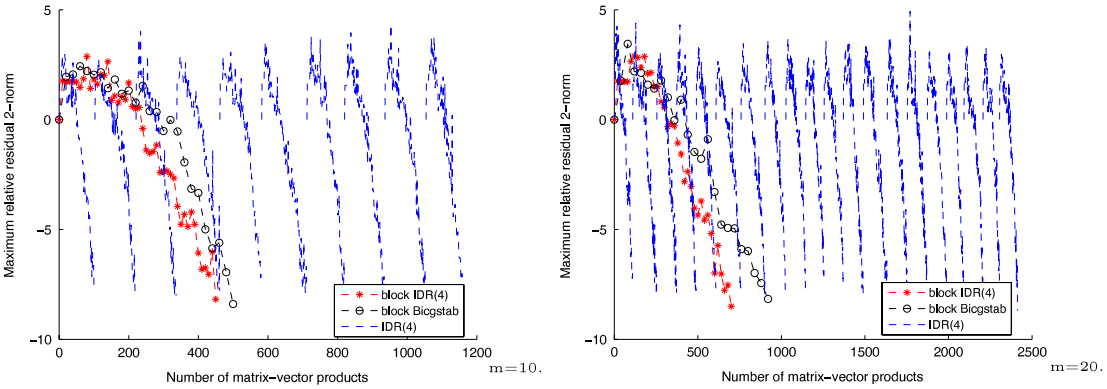



**Fig. 3.** $\mathrm{Log}_{10}$ of the maximum relative residual norms versus the number of matrix-vector products.

**Table 2**
Properties of matrix, number of matrix-vector products without preconditioning.

| Matrix | $N$ | NNZ | Density | #Matrix-vector | | |
|---|---|---|---|---|---|---|
| | | | | IDR($s$) | Bl-IDR($s$) | Bl-BICGSTAB |
| CAG_mat1916 | 1916 | 195,985 | 0.053387 | 28,494 | † | † |
| CDDE1 | 961 | 4681 | 0.005069 | 1582 | 1110 | † |
| CDDE3 | 961 | 4681 | 0.005069 | 2067 | 1850 | † |
| EX22 | 839 | 22,715 | 0.032269 | † | † | † |
| EX25 | 848 | 24,612 | 0.034226 | † | † | † |
| EX28 | 2607 | 77,781 | 0.011480 | † | † | † |
| EX40 | 7740 | 458,012 | 0.007645 | † | † | † |
| FIDAP001 | 216 | 4374 | 0.093750 | † | † | † |
| FIDAP022 | 839 | 22,613 | 0.032124 | † | † | † |
| MCFE | 765 | 24,382 | 0.041663 | † | † | † |
| ORSIRR_1 | 1030 | 6858 | 0.006464 | 16,467 | † | † |
| ORSREG_1 | 2205 | 14,133 | 0.002907 | 5974 | † | † |
| PISTON | 2205 | 100,015 | 0.024390 | † | † | † |
| POISSON2D | 367 | 2417 | 0.017945 | 737 | 300 | 380 |
| RAEFSKY1 | 3242 | 294,276 | 0.027998 | 3304 | 11,800 | † |
| RAEFSKY2 | 3242 | 294,276 | 0.027998 | 4855 | 1350 | † |
| RAJAT12 | 1879 | 12,926 | 0.003661 | † | † | † |
| SAYLR4 | 3564 | 22,316 | 0.001757 | † | † | † |
| SHERMAN2 | 1080 | 23,094 | 0.019799 | † | † | † |
| SHERMAN4 | 1104 | 3786 | 0.003106 | 1185 | 630 | † |
| SHERMAN5 | 3312 | 20,793 | 0.001896 | † | † | † |
| UTM1700A | 1700 | 21,313 | 0.007375 | 19,206 | † | † |
| WANG1 | 2903 | 19,093 | 0.002266 | 4357 | 40,830 | † |
| WATT_1 | 1856 | 11,360 | 0.003298 | † | † | † |

BiCGStab except SHERMAN2. In terms of the CPU time, block IDR($s$) were faster than block BiCGStab in most cases. We find that those matrices that block BiCGStab and cost less CPU time were more relatively sparse than others.

## 5. Conclusions

In this paper, we studied and extended the IDR($s$) into a block version for nonsymmetric linear systems with multiple right-hand sides. In order to define the block algorithm, we have generalized the IDR theorem to the block case and know that the upper bound on the number of matrix-vector products of block IDR($s$) to reach the solution in generic case is $n + \frac{n}{s}$, which is the same as in IDR($s$) for a single right-hand side. Numerical results also show that when a preconditioner is used the proposed method is more effective and less expensive than the IDR($s$) method applied to each right-hand side, and can be competitive with block BiCGStab, especially for the relatively dense matrices. Therefore, we conclude that block IDR($s$) may be a competitive algorithm for solving the linear systems with multiple right-hand sides.

## Acknowledgements

The authors are grateful to the anonymous referees for their valuable and helpful comments that greatly improved the original manuscript of this paper. This research was partially supported by the China Scholarship Council and Grant-in-Aid for Scientific Research (Grant No. 21760058, 21560065, 19560065, 22104004).

**Table 3**
Number of MVs, CPU time and ratio of Bl-IDR($s$) to IDR($s$) using ILU(0) preconditioner.

| Matrix | #MVs | | CPU time (s) | | Ratio | |
|---|---|---|---|---|---|---|
| | Bl-IDR($s$) | IDR($s$) | Bl-IDR($s$) | IDR($s$) | #MVs | Time |
| CAG_mat1916 | 650 | 1301 | 4.333441 | 8.481029 | 0.500 | 0.511 |
| CDDE1 | 250 | 511 | 0.069589 | 0.111104 | 0.489 | 0.626 |
| CDDE3 | 250 | 653 | 0.076209 | 0.142206 | 0.383 | 0.536 |
| EX22 | 280 | 538 | 0.257537 | 0.439266 | 0.520 | 0.586 |
| EX25 | 280 | 576 | 0.269682 | 0.502559 | 0.486 | 0.537 |
| EX28 | 460 | 1105 | 1.40465 | 3.010716 | 0.416 | 0.467 |
| EX40 | 910 | 2329 | 14.86735 | 36.250463 | 0.391 | 0.575 |
| FIDAP001 | 150 | 286 | 0.031254 | 0.049185 | 0.524 | 0.635 |
| FIDAP022 | 290 | 540 | 0.260132 | 0.438105 | 0.537 | 0.594 |
| MCFE | 110 | 133 | 0.118562 | 0.128964 | 0.827 | 0.919 |
| ORSIRR_1 | 280 | 604 | 0.111356 | 0.185055 | 0.464 | 0.602 |
| ORSREG_1 | 450 | 703 | 0.382122 | 0.426071 | 0.640 | 0.897 |
| PISTON | 150 | 272 | 0.596125 | 0.980356 | 0.551 | 0.608 |
| POISSON2D | 170 | 334 | 0.025160 | 0.039475 | 0.509 | 0.637 |
| RAEFSKY1 | 170 | 396 | 1.890616 | 4.055191 | 0.429 | 0.466 |
| RAEFSKY2 | 190 | 499 | 2.076492 | 5.063461 | 0.381 | 0.410 |
| RAJAT12 | 1440 | 2714 | 1.07093 | 1.530555 | 0.531 | 0.700 |
| SAYLR4 | 330 | 669 | 0.458474 | 0.647416 | 0.493 | 0.708 |
| SHERMAN2 | 160 | 237 | 0.165618 | 0.213920 | 0.675 | 0.774 |
| SHERMAN4 | 180 | 402 | 0.052886 | 0.083019 | 0.448 | 0.637 |
| SHERMAN5 | 180 | 399 | 0.237985 | 0.373910 | 0.451 | 0.636 |
| UTM1700A | 450 | 1266 | 0.452041 | 1.057513 | 0.355 | 0.427 |
| WANG1 | 290 | 494 | 0.33163 | 0.404255 | 0.587 | 0.820 |
| WATT_1 | 290 | 520 | 0.196807 | 0.261651 | 0.558 | 0.752 |

**Table 4**
Number of MVs, CPU time and ratio of Bl-IDR($s$) to Bl-BiCGStab using ILU(0) preconditioner.

| Matrix | #MVs | | CPU time (s) | | Ratio | |
|---|---|---|---|---|---|---|
| | Bl-IDR($s$) | Bl-BICGSTAB | Bl-IDR($s$) | Bl-BICGSTAB | #MVs | Time |
| CAG_mat1916 | 650 | 740 | 4.333441 | 5.374543 | 0.878 | 0.806 |
| CDDE1 | 250 | 280 | 0.069589 | 0.065430 | 0.893 | 1.064 |
| CDDE3 | 250 | 300 | 0.076209 | 0.070833 | 0.833 | 1.076 |
| EX22 | 280 | 340 | 0.257537 | 0.305315 | 0.824 | 0.844 |
| EX25 | 280 | 360 | 0.269682 | 0.355676 | 0.778 | 0.758 |
| EX28 | 460 | 760 | 1.40465 | 2.394004 | 0.605 | 0.587 |
| EX40 | 910 | 1340 | 14.86735 | 23.367967 | 0.679 | 0.636 |
| FIDAP001 | 150 | 180 | 0.031254 | 0.032957 | 0.833 | 0.948 |
| FIDAP022 | 290 | 340 | 0.260132 | 0.308387 | 0.853 | 0.844 |
| MCFE | 110 | 120 | 0.118562 | 0.119136 | 0.917 | 0.995 |
| ORSIRR_1 | 280 | 380 | 0.111356 | 0.126720 | 0.737 | 0.879 |
| ORSREG_1 | 450 | 600 | 0.382122 | 0.433426 | 0.750 | 0.882 |
| PISTON | 150 | 200 | 0.596125 | 0.770532 | 0.750 | 0.774 |
| POISSON2D | 170 | 180 | 0.025160 | 0.022323 | 0.944 | 1.127 |
| RAEFSKY1 | 170 | 220 | 1.890616 | 2.476699 | 0.773 | 0.763 |
| RAEFSKY2 | 190 | 220 | 2.076492 | 2.482186 | 0.864 | 0.837 |
| RAJAT12 | 1440 | † | 1.07093 | † | † | † |
| SAYLR4 | 330 | 360 | 0.458474 | 0.438055 | 0.917 | 1.047 |
| SHERMAN2 | 160 | 160 | 0.165618 | 0.155005 | 1.00 | 1.068 |
| SHERMAN4 | 180 | 200 | 0.052886 | 0.044775 | 0.900 | 1.181 |
| SHERMAN5 | 180 | 200 | 0.237985 | 0.229632 | 0.900 | 1.036 |
| UTM1700A | 450 | 500 | 0.452041 | 0.469164 | 0.900 | 0.964 |
| WANG1 | 290 | 320 | 0.33163 | 0.322058 | 0.906 | 1.030 |
| WATT_1 | 290 | 320 | 0.196807 | 0.186647 | 0.906 | 1.054 |

## References


[1] A.M. Abdel-Rehim, R.B. Morgan, W. Wilcox, Improved seed methods for symmetric positive definite linear equations with multiple right-hand sides, 2008, Arxiv preprint arXiv:0810.0330.
[2] T.F. Chan, W. Wang, Analysis of projection methods for solving linear systems with multiple right-hand sides, SIAM J. Sci. Comput. 18 (1997) 1698–1721.
[3] C. Smith, A. Peterson, R. Mittra, A conjugate gradient algorithm for treatment of multiple incident electromagnetic fields, IEEE Trans. Antennas Propagation 37 (1989) 1490–1493.
[4] K. Jbilou, A. Messaoudi, H. Sadok, Global FOM and GMRES algorithms for matrix equations, Appl. Numer. Math. 31 (1999) 49–63.
[5] K. Jbilou, H. Sadok, Global Lanczos-based methods with applications, Technical Report LMA 42, Université du Littoral, Calais, France, 1997.
[6] K. Jbilou, H. Sadok, A. Tinzefte, Oblique projection methods for linear systems with multiple right-hand sides, Elec. Trans. Numer. Anal. 20 (2005) 119–138.

[7] M. Heyouni, The global Hessenberg and global CMRH methods for linear systems with multiple right-hand sides, Numer. Algorithms 26 (2001) 317–332.
[8] M. Heyouni, A. Essai, Matrix Krylov subspace methods for linear systems with multiple right-hand sides, Numer. Algorithms 40 (2005) 137–156.
[9] D. O'Leary, The block conjugate gradient algorithm and related methods, Linear Algebra Appl. 29 (1980) 293–322.
[10] A. Nikishin, A. Yeremin, Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers I: general iterative scheme, SIAM J. Matrix Anal. 16 (1995) 1135–1153.
[11] V. Simoncini, A stabilized QMR version of block BICG, SIAM J. Matrix Anal. Appl. 18 (1997) 419–434.
[12] B. Vital, Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur, Ph.D. Thesis, Université de Rennes, 1990.
[13] V. Simoncini, E. Gallopoulos, Convergence properties of block GMRES and matrix polynomials, Linear Algebra Appl. 247 (1996) 97–119.
[14] H.-L. Liu, B.-J. Zhong, Simpler block GMRES for nonsymmetric systems with multiple right-hand sides, Elec. Trans. Numer. Anal. 30 (2008) 1–9.
[15] R. Freund, M. Malhotra, A Block-QMR algorithm for non-hermitian linear systems with multiple right-hand sides, Linear Algebra Appl. 254 (1997) 119–157.
[16] A. El Guennouni, K. Jbilou, H. Sadok, A block version of BICGSTAB for linear systems with multiple right-hand sides, Elec. Trans. Numer. Anal. 16 (2003) 129–142.
[17] A. El Guennouni, K. Jbilou, H. Sadok, The block Lanczos method for linear systems with multiple right-hand sides, Appl. Numer. Math. 51 (2004) 243–256.
[18] S. Karimi, F. Toutounian, The block least squares method for solving nonsymmetric linear systems with multiple right-hand sides, Appl. Math. Comput. 177 (2006) 852–862.
[19] C.-Q. Gu, H.-J. Qian, Skew-symmetric methods for solving nonsymmetric linear systems with multiple right-hand sides, J. Comput. Appl. Math. 223 (2009) 567–577.
[20] M. Kilmer, E. Miller, C. Rappaport, QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides, SIAM J. Sci. Comput. 23 (2001) 761–780.
[21] V. Simoncini, E. Gallopoulos, An iterative method for nonsymmetric systems with multiple right-hand sides, SIAM J. Sci. Comput. 16 (1995) 917–933.
[22] M.-C. Yeung, D. Boley, Transpose-free multiple Lanczos and its application in Padé approximation, J. Comput. Appl. Math. 117 (2005) 101–127.
[23] P. Sonneveld, M.B. van Gijzen, IDR($s$): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, SIAM J. Sci. Comput. 31 (2008) 1035–1062.
[24] P. Wesseling, P. Sonneveld, Numerical experiments with a multiple grid and a preconditioned lanczos type method, in: Lecture Notes in Mathematics, vol. 771, Springer Verlag, Berlin, Heidelberg, New York, 1980, pp. 543–562.
[25] G.L.G. Sleijpen, P. Sonneveld, M.B. van Gijzen, Bi-CGSTAB as an induced dimension reduction method, Delft University of Technology, Reports of the Department of Applied Mathematical Analysis, Report 08-07, 2008.
[26] 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. Statist. Comput. 13 (1992) 631–644.
[27] M.-C. Yeung, T.F. Chan, ML(K)BiCGSTAB: a BiCGSTAB variant based on multiple Lanczos starting vectors, SIAM J. Sci. Comput. 21 (1999) 1263–1290.
[28] M.B. van Gijzen, P. Sonneveld, An elegant IDR($s$) variant that efficiently exploits bi-orthogonality properties, Delft University of Technology, Reports of the Department of Applied Mathematical Analysis, Report 08-21, 2008.
[29] M.H. Gutknecht, IDR explained, Preprint 13-2009, Institute of Mathematics, Technische Universität Berlin, 2009.
[30] G.L.G. Sleijpen, D.R. Fokkema, BiCGstab($\ell$) for linear equations involving unsymmetric matrices with complex spectrum, Elec. Trans. Numer. Anal. 1 (1993) 11–32.
[31] G.L.G. Sleijpen, M.B. van Gijzen, Exploting BiCGSTAB($\ell$) strategies to induce dimension reduction. Delft University of Technology, Reports of the Department of Applied Mathematical Analysis, Report 09-02, 2009.
[32] M. Tanio, M. Sugihara, GBi-CGSTAB(s,L): IDR($s$) with Higher-Order Stabilization Polynomials, The University of Tokyo, Mathematical Engineering Technical Reports, METR 2009-16, 2009.
[33] M.H. Gutknecht, Block Krylov space methods for linear systems with multiple right-hand sides: an introduction, in: Modern Mathematical Models, Methods and Algorithms for Real World Systems, Anamaya Publishers, New Delhi, India, 2006.
[34] M.H. Gutknecht, T. Schmelzer, The block grade of a block Krylov space, Linear Algebra Appl. 430 (2009) 174–185.
[35] T. Schmelzer, Block Krylov methods for Hermitian linear systems, Diploma Thesis, Department of Mathematics, University of Kaiserslautern, Germany, 2004.
[36] R.B. Morgan, W. Wilcox, Deflated iterative methods for linear equations with multiple right-hand sides, 2004, Arxiv preprint math-ph/0405053.
[37] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der vorst, Numerical Linear Algebra for High Performance Computers, SIAM, Philadelphia, PA, USA, 1998.
[38] Matrix Market, Available online at http://math.nist.gov/MatrixMarket/.
[39] T.A. Davis, University of Florida Sparse Matrix Collection, Available online at http://www.cise.ufl.edu/research/sparse/matrices/.