Product-type Krylov-Subspace Methods for Solving Nonsymmetric Linear Systems (Domain Decomposition Methods and Related Topics)

ZHANG, SHAO-LIANG; NATORI, MAKOTO; JIN, CHENG-HAI

数理解析研究所講究録 (1997), 989: 92-102

1997-04

http://hdl.handle.net/2433/61069

Departmental Bulletin Paper

Kyoto University
Product-type Krylov-Subspace Methods for Solving Nonsymmetric Linear Systems

SHAO-LIANG ZHANG *, MAKOTO NATORI †, CHENG-HAI JIN ‡

Abstract. Recently S.-L. Zhang has proposed a unification and generalization of results involving product-type Krylov-subspace methods for the iterative solution of nonsymmetric linear systems. A characteristic of this class of methods (that includes CGS and Bi-CGSTAB) is the relationship

\[ r_n = H_n(A)R_n(A)r_0 \]

where \( r_n \) is the residual vector corresponding to the \( n \)-th iterate \( x_n \), and \( R_n \) is the Lanczos polynomial. The polynomial \( H_n \) in the product \( H_n(A)R_n(A) \) is chosen to speed up and/or stabilize convergence, while satisfying a standard three-term recurrence relations. Such product-type methods can be regarded as unification and generalization of Bi-CGSTAB. From the unification and generalization, we can see how CGS and Bi-CGSTAB fit into a more general framework.

Key words. Bi-CG, Bi-CGSTAB, CGS, the Lanczos polynomial, nonsymmetric linear systems, product-type methods, residual polynomial, three-term recurrence relations.

AMS(MOS) subject classification. 65F10

1 Introduction

Operations of transpose matrix-vector multiplications are needed in the Bi-Conjugate Gradient method (Bi-CG hereafter) [2] for solving nonsymmetric linear systems because a Krylov subspace generated from the transpose matrix is used. In order to avoid calculating the transpose matrix-vector multiplications and improve the convergence rate in Bi-CG, recently many efforts have been devoted to deriving more efficient methods from restructuring Bi-CG. A common technique to design a new method by means of restructuring Bi-CG is to define its residual polynomial by product of two polynomial factors where one factor is the Lanczos polynomial from Bi-CG and the other one is an undetermined \( n \) degree polynomial. For example, the Conjugate Gradients-Squared (CGS hereafter) [5], Bi-CGSTAB [7] and GPBi-CG [8] were derived from Bi-CG by this technique. In CGS, P. Sonneveld defined the undetermined polynomial by the same Lanczos polynomial, i. e., defined the residual polynomial of CGS by the square of that of Bi-CG. CGS was recognized as a powerful variant of Bi-CG in a lot of numerical experiments [5]. However, it often observed that CGS has a rather irregular and oscillatory convergence behaviour in many situations because of the presence of the round-off errors [7]. Therefore, in Bi-CGSTAB, H. A. Van der Vorst selected a polynomial with
two-term recurrence relations instead of one factor of CGS to design the residual polynomial of Bi-CGSTAB. Since the undetermined parameters with respect to the residual polynomial of Bi-CGSTAB are chosen at least to minimize the residual 2-norm per iteration, Bi-CGSTAB is a rather stable and more efficient variant of Bi-CG. In fact, many numerical experiments also indicated that Bi-CGSTAB can often run faster and its convergence behaviour is more smooth than CGS [7]. In [8], S.-L. Zhang proposed a unification and generalization of results involving product-type Krylov subspace methods for the iterative solution of nonsymmetric linear systems, implemented several new methods.

This paper is organized as follows: in the next section, we characterize product-type Krylov-subspace methods based on Bi-CG, and derive a set of recurrence formulas among the related iterates. In §3, several implementations of algorithms of the product-type methods are considered, and some well-known variants are recalled. The way in which preconditioning can be incorporated in the algorithms is discussed in §4. In §5, we report some numerical experiments and show that GPBi-CG may be very attractive in comparison with Bi-CGSTAB in many situations. Finally, we make some concluding remarks in §6.

Throughout this paper, superscript BCG is used to distinguish iterates generated in the algorithm of Bi-CG.

## 2 Product-type Krylov-subspace Methods

The algorithm of Bi-CG, for solving linear system $Ax = b$ where $A$ be an $N \times N$ large and sparse nonsymmetric matrix with complex spectrum, given by R. Fletcher [2] reads:

**Algorithm 1** Unpreconditioned Bi-CG

$x_{0}^{\text{BCG}}$ is an initial guess, set $p_{0}^{\text{BCG}} = r_{0}^{\text{BCG}} = p_{0}^{\text{BCG}} = r_{0}^{\text{BCG}} = b - Ax_{0}^{\text{BCG}},$

for $n = 0, 1, \cdots$ until $\| r_{n}^{\text{BCG}} \| \leq \epsilon \| b \|$ do:

\[
\begin{align*}
\alpha_{n} &= \frac{(r_{n}^{\text{BCG}} r_{n}^{\text{BCG}})}{(p_{n}^{\text{BCG}} A p_{n}^{\text{BCG}})}, \\
x_{n+1}^{\text{BCG}} &= x_{n}^{\text{BCG}} + \alpha_{n} p_{n}^{\text{BCG}}, \\
r_{n+1}^{\text{BCG}} &= r_{n}^{\text{BCG}} - \alpha_{n} A p_{n}^{\text{BCG}}, \\
\beta_{n} &= \frac{(r_{n}^{\text{BCG}} r_{n}^{\text{BCG}})}{(r_{n}^{\text{BCG}} r_{n}^{\text{BCG}})}, \\
P_{n+1}^{\text{BCG}} &= r_{n+1}^{\text{BCG}} + \beta_{n} p_{n}^{\text{BCG}}, \\
P_{n+1}^{\text{BCG}} &= r_{n+1}^{\text{BCG}} + \beta_{n} p_{n}^{\text{BCG}},
\end{align*}
\]

Let $r_{0}^{\text{BCG}}$ and $r_{0}^{\text{BCG}}$ be abbreviated as $r_0$ and $r_0^*$. In the algorithm of Bi-CG, two Krylov subspaces $K_n(A; r_0) := \text{span}\{r_0, Ar_0, \cdots, A^{n-1}r_0\}$ and $K_n(A^T; r_0^*) := \text{span}\{r_0^*, A^T r_0^*, \cdots, (A^T)^{n-1} r_0^*\}$ are generated, and the approximative solution $x_n^{\text{BCG}}$ is given in such way that the residual $r_n^{\text{BCG}}(= b - Ax_n^{\text{BCG}}$ theoretically) is made orthogonal with respect to $K_n(A^T; r_0^*)$. Therefore, we have the
following orthogonalties of Bi-CG[2].

\begin{equation}
(2.3) \quad r_{n}^{\text{BG}} \perp K_{n}(A^{T}; r_{n}^{*}), \quad A p_{n}^{\text{BG}} \perp K_{n}(A^{T}; r_{n}^{*}).
\end{equation}

Notice that $r_{n}^{\text{BG}*}$ and $p_{n}^{\text{BG}*}$ can be written as

\begin{equation}
r_{n}^{\text{BG}*} = (-1)^{n} \prod_{i=0}^{n-1} \alpha_{i}(A^{T})^{n} r_{0}^{*} + g_{1}, \quad p_{n}^{\text{BG}*} = (-1)^{n} \prod_{i=0}^{n-1} \alpha_{i}(A^{T})^{n} r_{0}^{*} + g_{2}.
\end{equation}

with $g_{1}$ and $g_{2} \in K_{n}(A^{T}, r_{0}^{*})$. By the orthogonalties (2.3), auxiliary formulas for computing $\alpha_{n}$ and $\beta_{n}$ can be recovered:

\begin{equation}
(2.4) \quad \alpha_{n} = \frac{(r_{n}^{\text{BG}*}, r_{n}^{\text{BG}})}{(p_{n}^{\text{BG}*}, A p_{n}^{\text{BG}})} = \frac{((A^{T})^{n} r_{0}^{*}, r_{n}^{\text{BG}})}{((A^{T})^{n} r_{0}^{*}, A p_{n}^{\text{BG}})}, \quad \beta_{n} = \frac{(r_{n+1}^{\text{BG}*}, r_{n+1}^{\text{BG}})}{(p_{n+1}^{\text{BG}*}, r_{n+1}^{\text{BG}})} = -\frac{((A^{T})^{n+1} r_{0}^{*}, r_{n+1}^{\text{BG}})}{((A^{T})^{n} r_{0}^{*}, r_{n}^{\text{BG}})}.
\end{equation}

Here, we attempt to use an $n$ degree polynomial $H_{n}$ to accelerate $r_{n}^{\text{BG}}$, say, make $H_{n}(A)r_{n}^{\text{BG}}$ as new residual converge towards zero fast. By doing so, we can derive a class of methods which have product-type residual polynomial. We characterize the product-type methods based on Bi-CG by the following process:

- residual polynomial of a product-type method is defined by product of two $n$ degree polynomials

\begin{equation}
(2.5) \quad H_{n}(\lambda) R_{n}(\lambda)
\end{equation}

where $R_{n}$ is the Lanczos polynomial [6] and $H_{n}$ is undetermined.

The design of the polynomial $H_{n}$ in practice is desired as:

1. to make the polynomial $H_{n}$ satisfy short-term recurrence relations so that little computational work and low storage costs are required per iteration;

2. to choose parameters of $H_{n}$ reasonably to get rather fast and stable convergence behaviour.

Next, we will describe the basic idea to establish a standard polynomial $H_{n}$ which leads to generalized product-type methods based on Bi-CG.

We introduce two independent parameters $\zeta_{n}$ and $\eta_{n}$ and define the polynomial $H_{n}$ as follows:

\begin{align}
(2.6) & \quad H_{0}(\lambda) := 1, \quad H_{1}(\lambda) := (1 - \zeta_{0}\lambda)H_{0}(\lambda), \\
(2.7) & \quad H_{n+1}(\lambda) := (1 + \eta_{n} - \zeta_{n}\lambda)H_{n}(\lambda) - \eta_{n} H_{n-1}(\lambda)
\end{align}

where $\zeta_{n}$ and $\eta_{n}$ are undetermined parameters.

Noticing that $H_{n}(0) = 1$ holds for any $n$, we have $H_{n+1}(0) - H_{n}(0) = 0$ for any $n$. Thus, we can find an auxiliary polynomial $G_{n}(\lambda)$ with degree $n$, and obtain a double sets of polynomials $H_{n}(\lambda)$ and $G_{n}(\lambda)$ mutually interlocked by recurrence relations:

\begin{equation}
(2.8) \quad H_{n+1}(\lambda) = H_{n}(\lambda) - \lambda G_{n}(\lambda); \quad G_{n+1}(\lambda) = \zeta_{n+1} H_{n+1}(\lambda) + \eta_{n+1} G_{n}(\lambda).
\end{equation}

Now, let us derive the product-type methods with residual:

\begin{equation}
(2.9) \quad r_{n} := H_{n}(A)r_{n}^{\text{BG}} = b - Ax_{n}
\end{equation}
which can be obtained with the following iterates:

\begin{align}
(2.10) & \quad t_n := H_n(A)r_{n+1}^{\text{BGC}}, \quad y_n := AG_{n-1}(A)r_{n+1}^{\text{BGC}}, \quad p_n := H_n(A)p_n^{\text{BGC}}, \\
(2.11) & \quad w_n := AH_n(A)p_{n+1}^{\text{BGC}}, \quad u_n := AG_n(A)p_n^{\text{BGC}}, \quad z_n := G_n(A)r_{n+1}^{\text{BGC}}.
\end{align}

According to the recurrence relations (2.1) \sim (2.2) and (2.6) \sim (2.8), we have a set of recurrence formulas among the sequences of the iterates \( r_n, p_n, t_n, u_n, w_n, y_n \) and \( z_n \):

\begin{align}
(2.12) & \quad r_{n+1} = t_n - \eta ny_n - \zeta_n At_n \\
(2.13) & \quad = r_n - \alpha_n Ap_n - A\zeta_n, \\
(2.14) & \quad t_n = r_n - \alpha_n Ap_n, \\
(2.15) & \quad y_{n+1} = t_n - r_{n+1} - \alpha_{n+1}w_n + \alpha_{n+1}Ap_{n+1}, \\
(2.16) & \quad p_{n+1} = r_{n+1} + \beta_n(p_n - u_n), \\
(2.17) & \quad w_n = At_n + \beta_n Ap_n, \\
(2.18) & \quad u_n = \zeta_n Ap_n + \eta_n(t_{n-1} - r_n + \beta_{n-1}u_{n-1}), \\
(2.19) & \quad z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n.
\end{align}

From (2.9) and (2.13), we have a formula to update the approximating solution \( x_{n+1} \):

\begin{align}
(2.20) & \quad x_{n+1} = x_n + \alpha_n p_n + z_n.
\end{align}

Since the coefficient of the highest order term of \( H_n \) is \((-1)^n \prod_{i=0}^{n-1} \zeta_i \), we have

\begin{align}
(r_0^*, r_n) &= (-1)^n \prod_{i=0}^{n-1} \zeta_i((A^T)^n r_0^*, r_n^{\text{BGC}}), \quad (r_0^*, Ap_n) = (-1)^n \prod_{i=0}^{n-1} \zeta_i((A^T)^n r_0^*, Ap_n^{\text{BGC}}).
\end{align}

Then, from the formula (2.4), \( \alpha_n \) and \( \beta_n \) can be recovered from the iterates \( r_{n+1}, r_n \) and \( p_n \):

\begin{align}
(2.21) & \quad \alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)}, \quad \beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}.
\end{align}

Due to the lack of a criterion for choices of \( \zeta_n \) and \( \eta_n \), it is very hard in fact to determine the parameters \( \zeta_n \) and \( \eta_n \) that are closely and indissolubly connected with convergence behaviour in practice. Implementations of the parameters \( \zeta_n \) and \( \eta_n \) will be discussed in §3 in detail.

### 3 Details of Implementation

In accordance with the requirement (2) described in §2, we summarize several possibilities to select \( \zeta_n \) and \( \eta_n \) for the actual implementation of the product-type methods based on Bi-CG. As well-known variants, the algorithms of CGS, Bi-CGSTAB and GPBi-CG will be recalled in terms of special choices.
3.1 The Choice for GPBi-CG

It is convenient to determine parameters $\zeta_n$ and $\eta_n$ in terms of minimizing the residual 2-norm as the function of $\zeta$ and $\eta$:

$$f(\zeta, \eta) := \| r_{n+1} \| = \| t_n - \eta y_n - \zeta At_n \|.$$

Thus, we have a variant of the product-type methods, and name it GPBi-CG [8]:

**Algorithm 2** Unpreconditioned GPBi-CG

$x_0$ is an initial guess, $r_0 = b - Ax_0$; set $r_0^* = r_0$, $t_{-1} = w_{-1} = 0$, $\beta_{-1} = 0$;

for $n = 0, 1, \cdots$ until $\| r_n \| \leq \epsilon \| b \|$ do:

$$p_n = r_n + \beta_{n-1}(p_{n-1} - u_{n-1}),$$

$$\alpha_n = \langle r_0^*, Ap_n \rangle / \langle r_0^*, r_n \rangle,$$

$$y_n = t_{n-1} - r_n - \alpha_n w_{n-1} + \alpha_n Ap_n,$$

$$t_n = r_n - \alpha_n A p_n,$$

$$\zeta_n = \frac{(y_n, y_n)(At_n, t_n) - (y_n, At_n)(At_n, y_n)}{(At_n, At_n)(y_n, y_n) - (y_n, At_n)(At_n, y_n)},$$

$$\eta_n = \frac{(At_n, At_n)(y_n, t_n) - (y_n, At_n)(At_n, t_n)}{(At_n, At_n)(y_n, At_n) - (y_n, At_n)(At_n, y_n)},$$

(if $n = 0$, then $\zeta_n = \frac{(At_n, t_n)}{(At_n, At_n)}, \eta_n = 0$)

$$u_n = \zeta_n A p_n + \eta_n (t_{n-1} - r_n + \beta_{n-1} u_{n-1}),$$

$$z_n = \zeta_n r_n + \eta_n z_{n-1} - \alpha_n u_n,$$

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

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

$$\beta_n = \frac{\alpha_n \cdot \langle r_0^*, r_{n+1} \rangle}{\zeta_n \langle r_0^*, r_n \rangle},$$

$$w_n = At_n + \beta_n A p_n;$$

3.2 The Choice for CGS

Suppose that $\zeta_n = \alpha_n$ and $\eta_n = \frac{\beta_n}{\alpha_n - 1} \alpha_n$ in recurrence relations (2.6)$\sim$(2.7), we obtain a significant variant of the product-type methods which only depends on information of Bi-CG. It is easy to see that this variant is mathematically equivalent to CGS.

Notice that $t_{n-1} - r_n = Az_{n-1}$. In this case, we have $H_n = R_n$ and $G_n = P_n$, and use the recurrence formula (2.13) to update $r_{n+1}$. This fact leads to relation $p_n - u_n = z_n/\alpha_n$ for any $n$, then the iterates $t_n$, $y_n$ and $w_n$ can be omitted in the recurrence formulas (2.12)$\sim$(2.19).

Noticing that $(r_0^*, Ap_n) = (r_0^*, At_n)$, and setting new iterates $u_n := A^{-1} u_n/\alpha_n$ and $z_n := z_n/\alpha_n$, then the algorithm of CGS [5] is recalled as follows:

**Algorithm 3** Unpreconditioned CGS

$x_0$ is an initial guess, $r_0 = b - Ax_0$; set $r_0^* = r_0$, $\beta_{-1} = 0$;

for $n = 0, 1, \cdots$ until $\| r_n \| \leq \epsilon \| b \|$ do:

$$p_n = r_n + \beta_{n-1} z_{n-1},$$
$$u_n = p_n + \beta_{n-1}(z_{n-1} + \beta_{n-1}u_{n-1}),$$
$$\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, A u_n)},$$
$$x_{n} = p_{n} - \alpha_{n} A u_{n},$$
$$x_{n+1} = x_{n} + \alpha_{n}(p_{n} + z_{n}),$$
$$r_{n+1} = r_{n} - \alpha_{n} A (p_{n} + z_{n}),$$
$$\beta_{n} = \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)};$$

3.3 The Choice for Bi-CGSTAB

If one attempts to get a variant of the product-type methods with little computational work, one would define all $\eta_n$ by a quantity $\omega$ called the relaxation factor in advance. Here, we suppose that $\eta_n = 0$ for any $n$, and $\zeta_n$ is selected to minimize the residual 2-norm as function of $\zeta$

$$f(\zeta) := \| r_{n+1} \| = \| t_n - \zeta A t_n \|. $$

In this case, $u_n = \zeta_n A p_n$, and then $x_n = \zeta_n t_n$. Notice that the iterates $y_n, u_n, z_n$ and $w_n$ become worthless. In this way an important and economical variant will be obtained again, recalled Bi-CGSTAB[7]:

**Algorithm 4** Unconditioned Bi-CGSTAB

- $x_0$ is an initial guess, $r_0 = b - Ax_0$; set $r_0^* = r_0$, $\beta_{-1} = 0$;
- for $n = 0, 1, \cdots$ until $\| r_n \| \leq \epsilon \| b \|$ do:
  - $p_n = r_n + \beta_{n-1}(p_{n-1} - \zeta_{n-1} A p_{n-1}),$
  - $\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, A p_n)},$
  - $t_n = r_n - \alpha_n A p_n,$
  - $\zeta_n = \frac{(A t_n, t_n)}{(A t_n, A t_n)},$
  - $x_{n+1} = x_n + \alpha_n p_n + \zeta_n t_n,$
  - $r_{n+1} = t_n - \zeta_n A t_n,$
  - $\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)};$

4 Preconditioned Algorithms

For solving realistic problems, any variant will be hardly competitive without preconditioning techniques. All variants can be combined with the efficient preconditioning techniques, such as incomplete $LU$ factorizations [3].

Let $K$ be a suitable preconditioning matrix, i. e., $K \approx A$. We write $K = K_1K_2$, and apply CGS, Bi-CGSTAB, and GPBi-CG to the explicitly preconditioned system

$$\tilde{A}\tilde{x} = \tilde{b},$$

with $\tilde{A} = K_1^{-1} A K_2^{-1}$, $\tilde{x} = K_2 x$, and $\tilde{b} = K_1^{-1} b$. For example, for $K_1 = I$ we have preconditioning from the right; for $K_2 = I$ we have preconditioning from the left, and for $K_1 = L, K_2 = U$ we have the well-known preconditioning from both sides.
Now we write the algorithm of GPBi-CG for (4.1), and denote all the occurring iterates by \( \tilde{\cdot} \), e.g., \( \tilde{r}_n \).

With the change of variables:

\[
\tilde{x}_n \Rightarrow K_2 x_n, \quad \tilde{p}_n \Rightarrow K_2 p_n, \quad \tilde{u}_n \Rightarrow K_2 u_n, \quad \tilde{z}_n \Rightarrow K_2 z_n,
\]
\[
\tilde{r}_n \Rightarrow K_1^{-1} r_n, \quad \tilde{t}_n \Rightarrow K_1^{-1} t_n, \quad \tilde{w}_n \Rightarrow K_1^{-1} w_n, \quad \tilde{y}_n \Rightarrow K_1^{-1} y_n, \quad \tilde{r}_0^* = K^T r_0^*.
\]

then we have the algorithm of preconditioned GPBi-CG. Here, for computing \( \zeta_n \) and \( \eta_n \), we are minimizing the current residual for the original system rather than the preconditioned one.

**Algorithm 5**  Preconditioned GPBi-CG

\[
x_0 \text{ is an initial guess, } r_0 = b - Ax_0; \text{ set } r_0^* = r_0, \quad t_{-1} = w_{-1} = 0, \quad \beta_{-1} = 0;
\]

for \( n = 0, 1, \cdots \) until \( \| r_n \| \leq \varepsilon \| b \| \) do:

\[
p_n = K^{-1} r_n + \beta_{n-1} (p_{n-1} - u_{n-1}),
\]

\[
\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)},
\]

\[
y_n = t_{n-1} - r_n - \alpha_n w_{n-1} + \alpha_n A p_n,
\]

\[
t_n = r_n - \alpha_n A p_n,
\]

\[
K^{-1} t_n = K^{-1} r_n - \alpha_n K^{-1} A p_n,
\]

\[
\zeta_n = \frac{(y_n, y_n) (A K^{-1} t_n, t_n) - (y_n, t_n) (A K^{-1} t_n, y_n)}{(A K^{-1} t_n, t_n) (y_n, y_n) - (y_n, A K^{-1} t_n) (A K^{-1} t_n, y_n)},
\]

\[
\eta_n = \frac{(A K^{-1} t_n, y_n) (y_n, t_n) - (y_n, A K^{-1} t_n) (A K^{-1} t_n, y_n)}{(A K^{-1} t_n, t_n) (y_n, y_n) - (y_n, A K^{-1} t_n) (A K^{-1} t_n, y_n)},
\]

(if \( n = 0 \), then \( \zeta_n = \frac{(A K^{-1} t_n, t_n)}{(A K^{-1} t_n, A K^{-1} t_n)} \), \( \eta_n = 0 \))

\[
u_n = \zeta_n K^{-1} A p_n + \eta_n (K^{-1} t_{n-1} - K^{-1} r_n + \beta_{n-1} u_{n-1}),
\]

\[
x_n = \zeta_n K^{-1} r_n + \eta_n x_{n-1} - \alpha_n u_n,
\]

\[
x_{n+1} = x_n + \alpha_n p_n + z_n,
\]

\[
r_{n+1} = t_n - \nu_n y_n - \zeta_n A K^{-1} t_n,
\]

\[
\beta_n = \frac{\alpha_n}{\zeta_n} \frac{(r_0^*, r_{n+1})}{(r_0^*, A K^{-1} u_n)},
\]

\[
w_n = A K^{-1} t_n + \beta_n A p_n;
\]

When we rewrite the algorithm of CGS for (4.1), then with the change of variables:

\[
\tilde{x}_n \Rightarrow K_2 x_n, \quad \tilde{p}_n \Rightarrow K_1^{-1} p_n, \quad \tilde{u}_n \Rightarrow K_1^{-1} u_n, \quad \tilde{z}_n \Rightarrow K_1^{-1} z_n, \quad \tilde{r}_n \Rightarrow K_1^{-1} r_n, \quad \tilde{r}_0^* = K_1^T r_0^*;
\]

we have the algorithm of preconditioned CGS.

**Algorithm 6**  Preconditioned CGS

\[
x_0 \text{ is an initial guess, } r_0 = b - Ax_0; \text{ set } r_0^* = r_0, \quad \beta_{-1} = 0;
\]

for \( n = 0, 1, \cdots \) until \( \| r_n \| \leq \varepsilon \| b \| \) do:

\[
p_n = r_n + \beta_{n-1} x_{n-1},
\]

\[
u_n = p_n + \beta_{n-1} (z_{n-1} + \beta_{n-1} u_{n-1}),
\]

\[
\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, A K^{-1} u_n)},
\]
$x_n = p_n - \alpha_n A K^{-1} u_n,$
$x_{n+1} = x_n + \alpha_n K^{-1} (p_n + z_n),$
$r_{n+1} = r_n - \alpha_n A K^{-1} (p_n + z_n),$
$\beta_n = \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}; \tag{4.1}$

When we rewrite the algorithm of Bi-CGSTAB for (4.1), then with the change of variables:

$\tilde{x}_n \Rightarrow K_2 x_n, \tilde{p}_n \Rightarrow K_1^{-1} p_n, \tilde{r}_n \Rightarrow K_1^{-1} r_n, \tilde{t}_n \Rightarrow K_1^{-1} t_n, \tilde{r}_{01}^* = K^T r_0^*,$

we have the algorithm of preconditioned Bi-CGSTAB. Here, for computing $\zeta_n$, we are minimizing the current residual for the original system rather than the preconditioned one.

**ALGORITHM 7** Preconditioned Bi-CGSTAB

$x_0$ is an initial guess, $r_0 = b - A x_0$; set $r_0^* = r_0$, $\beta_{-1} = 0$;

for $n = 0, 1, \cdots$ until $\| r_n \| \leq \epsilon \| b \|$ do:

$p_n = r_n + \beta_{n-1} (p_{n-1} - \zeta_{n-1} A p_{n-1}),$

$\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, AK^{-1} p_n)}$, \\
$t_n = r_n - \alpha_n A K^{-1} p_n,$ \\
$\zeta_n = \frac{(AK^{-1} t_n, t_n)}{(AK^{-1} t_n, AK^{-1} t_n)},$ \\
$x_{n+1} = x_n + \alpha_n K^{-1} p_n + \zeta_n K^{-1} t_n,$ \\
$r_{n+1} = t_n - \zeta_n A K^{-1} t_n,$ \\
$\beta_n = \frac{\alpha_n (r_0^*, r_{n+1})}{\zeta_n (r_0^*, r_n)}.$

Finally, we estimate computational work in the above algorithms. Preconditioned GPBi-CG requires evaluation of two matrix vector products with $A$, two solvers for $K$, $32N$ flops for vector updates, and seven inner products. Preconditioned CGS requires evaluation of two matrix vector products with $A$, two solvers for $K$, $13N$ flops for vector updates, and two inner products. Preconditioned Bi-CGSTAB requires evaluation of two matrix vector products with $A$, two solvers for $K$, $12N$ flops for vector updates, and four inner products. In practical situations, however, a few vector updates and inner products lead to only a small increase in computational work per iteration step, especially on vector and parallel computers vector updates and inner products are usually computed much faster rather than matrix vector products with $A$, and solvers for $K$.

5 Numerical Experiments

In this section we consider some numerical experiments to show the characteristic behaviour of GPBi-CG for certain linear systems with complex spectrum. These experiments have been carried out with CGS, Bi-CGSTAB and GPBi-CG applied to the explicitly preconditioned system $L^{-1} A U^{-1} (U x) = L^{-1} b$ in double precision floating point arithmetic on a SUN SPARCstation IPX computer. In all cases the iteration was started with $x_0 = 0$, and the convergence plots show the relative residual 2-norms $\| r_n \| / \| r_0 \|$ (on the vertical axis) versus the iteration number $n$ (on the horizontal axis). The convergence behaviours of CGS were omitted because CGS has shown rather irregular and oscillatory convergence behaviours in all cases.
5.1 Example 1

In the first example, we consider two nonsymmetric linear systems which come from central difference discretization of the following partial differential equation (described in [7])

\[-(Au_x)_x - (Au_y)_y + \gamma \exp(2(x^2 + y^2))u_x = F\]

over the unit square. Along the boundaries we have Dirichlet conditions: \( u = 1 \), for \( y = 0, x = 0 \) and \( x = 1 \), and \( u = 0 \) for \( y = 1 \). The function \( A \) is defined as shown in Fig. 1; \( F = 0 \) everywhere, except for the small subsquare in the center where \( F = 100 \).

\[ u = 0 \]

\[ u = 1 \]

\[ F = 10^2 \]

\[ A = 10^4 \]

\[ A = 10^{-5} \]

\[ A = 10^2 \]

\[ u = 1 \]

**Fig. 1.** The coefficients for example 1.

We consider two cases of \( \gamma = 2 \) and \( \gamma = 0 \), and take a 101 \( \times \) 101 gridmesh which lead to systems with 100\(^2\) unknowns. The linear systems were preconditioned by incomplete \( LU \) factorizations.

**Fig. 2.** The history of the residual 2-norms (i.e., \( \log(||r_n|| / ||r_0||) \)) vs. \( n \) for example 1 (\( \gamma = 2 \))

**Fig. 3.** The history of the residual 2-norms (i.e., \( \log(||r_n|| / ||r_0||) \)) vs. \( n \) for example 1 (\( \gamma = 0 \))

Fig. 2 shows the history of the residual 2-norms when \( \gamma = 2 \). We observe that in this case, though GPBi-CG converges, it does not improve the iteration process with respect to efficiency.
Fig. 3 shows the history of the residual 2-norms when $\gamma = 0$. We observe that in this case, GPBi-CG converges slightly faster.

5.2 Example 2

As the second example, we consider two linear systems with complex spectrum which come from a $100 \times 100$ and a $200 \times 200$ central difference discretization of the Helmholtz equation over $[0, \pi] \times [0, \pi]$ described in [1]

$$u_{xx} + u_{yy} + k^2 u = 0$$

with Dirichlet condition $u = 0$ along $y = \pi$, Neumann conditions $u_x = i\sqrt{k^2 - \frac{1}{4}} \cos(\frac{\pi}{2})$ along $x = 0$ and $u_y = 0$ along $y = 0$, and radiation condition $u_x - i\sqrt{k^2 - \frac{1}{4}} u = 0$ along $x = \pi$. This leads to two systems with unknowns $101 \times 100$, $201 \times 200$. Here we only consider the case $k = 2.27$. The linear systems were preconditioned by incomplete $LU$ factorizations.

Although GPBi-CG is more expensive with respect to the number of inner products and vector updates, we observe in Fig. 4 that GPBi-CG performs much better so that the total CPU-time of GPBi-CG needs only 57% of that of Bi-CGSTAB for this coarser grid.

In Fig. 5, GPBi-CG required 734 iteration steps to get the residual 2-norm below $10^{-12}$, Bi-CGSTAB required 2404 iteration steps. For this finer grid, GPBi-CG needs only 41% of the total CPU-time of Bi-CGSTAB.

6 Concluding Remarks

In view of more stable convergence behaviour and little work and low storage cost, we emphasize that the polynomial $H_n$ generated by the three-term recurrence relation (2.7) is better than the others
which come from such restarted iterative method as GMRES(k) (k > 2) [4] for the requirement (1) described in §2. From our experiments we have learned that GPBi-CG may be an attractive method.

References


