数理解析研究所講究録
989巻 1997年 92-102

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

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

Abstarct. 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 multications 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

*Institute of Information Sciences and Electronics, University of Tsukuba. Tennodai 1-1-1, Tsukuba 305, Japan. E-mail:zhang@is.tsukuba.ac.jp.

†Institute of Information Sciences and Electronics, University of Tsukuba. Tennodai 1-1-1, Tsukuba 305, Japan. E-mail:natori@is.tsukuba.ac.jp.

‡Doctoral Program in Engineering, University of Tsukuba. Tennodai 1-1-1, Tsukuba 305, Japan. E-mail:kinsk@kiko.is.tsukuba.ac.jp.

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^{BCG} is an initial guess, set $p_0^{\mathrm{BCG}*} = r_0^{\mathrm{BCG}*} = p_0^{\mathrm{BCG}} = r_0^{\mathrm{BCG}} = b - Ax_0^{\mathrm{BCG}}$;

for $n = 0, 1, \cdots$ until $\| r_n^{\mathrm{BCG}} \| \leq \varepsilon \| b \|$ do :

$$\alpha_n = \frac{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})}{(p_n^{\mathrm{BCG}*}, Ap_n^{\mathrm{BCG}})},$$

$$x_{n+1}^{\mathrm{BCG}} = x_n^{\mathrm{BCG}} + \alpha_n p_n^{\mathrm{BCG}},$$

$$r_{n+1}^{\mathrm{BCG}} = r_n^{\mathrm{BCG}} - \alpha_n A p_n^{\mathrm{BCG}}, \tag{2.1}$$

$$r_{n+1}^{\mathrm{BCG}*} = r_n^{\mathrm{BCG}*} - \alpha_n A^T p_n^{\mathrm{BCG}*},$$

$$\beta_n = \frac{(r_{n+1}^{\mathrm{BCG}*}, r_{n+1}^{\mathrm{BCG}})}{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})},$$

$$p_{n+1}^{\mathrm{BCG}} = r_{n+1}^{\mathrm{BCG}} + \beta_n p_n^{\mathrm{BCG}}, \tag{2.2}$$

$$p_{n+1}^{\mathrm{BCG}*} = r_{n+1}^{\mathrm{BCG}*} + \beta_n p_n^{\mathrm{BCG}*};$$

Let r_0^{BCG} and $r_0^{\mathrm{BCG}*}$ be abbreviated as r_0 and r_0^*. In the algorithm of Bi-CG, two Krylov subspaces

$$K_n(A; r_0) := \mathrm{span}\{r_0, Ar_0, \cdots, A^{n-1}r_0\} \text{ and } K_n(A^T; r_0^*) := \mathrm{span}\{r_0^*, A^T r_0^*, \cdots, (A^T)^{n-1} r_0^*\}$$

are generated, and the approximative solution x_n^{BCG} is given in such way that the residual $r_n^{\mathrm{BCG}} (= b - Ax_n^{\mathrm{BCG}}$ theoretically) is made orthogonal with respect to $K_n(A^T; r_0^*)$. Therefore, we have the

following orthogonalities of Bi-CG[2].

$$\boldsymbol{r}_n^{\mathrm{BCG}} \perp K_n(A^T; \boldsymbol{r}_0^*), \quad A\boldsymbol{p}_n^{\mathrm{BCG}} \perp K_n(A^T; \boldsymbol{r}_0^*). \tag{2.3}$$

Notice that $\boldsymbol{r}_n^{\mathrm{BCG}*}$ and $\boldsymbol{p}_n^{\mathrm{BCG}*}$ can be written as

$$\boldsymbol{r}_n^{\mathrm{BCG}*} = (-1)^n \prod_{i=0}^{n-1} \alpha_i (A^T)^n \boldsymbol{r}_0^* + \boldsymbol{g}_1, \quad \boldsymbol{p}_n^{\mathrm{BCG}*} = (-1)^n \prod_{i=0}^{n-1} \alpha_i (A^T)^n \boldsymbol{r}_0^* + \boldsymbol{g}_2.$$

with $\boldsymbol{g}_1$ and $\boldsymbol{g}_2 \in K_n(A^T, \boldsymbol{r}_0^*)$. By the orthogonalities (2.3), auxiliary formulas for computing α_n and β_n can be recovered:

$$\alpha_n = \frac{(\boldsymbol{r}_n^{\mathrm{BCG}*}, \boldsymbol{r}_n^{\mathrm{BCG}})}{(\boldsymbol{p}_n^{\mathrm{BCG}*}, A\boldsymbol{p}_n^{\mathrm{BCG}})} = \frac{((A^T)^n \boldsymbol{r}_0^*, \boldsymbol{r}_n^{\mathrm{BCG}})}{((A^T)^n \boldsymbol{r}_0^*, A\boldsymbol{p}_n^{\mathrm{BCG}})}, \quad \beta_n = \frac{(\boldsymbol{r}_{n+1}^{\mathrm{BCG}*}, \boldsymbol{r}_{n+1}^{\mathrm{BCG}})}{(\boldsymbol{r}_n^{\mathrm{BCG}*}, \boldsymbol{r}_n^{\mathrm{BCG}})} = -\alpha_n \frac{((A^T)^{n+1} \boldsymbol{r}_0^*, \boldsymbol{r}_{n+1}^{\mathrm{BCG}})}{((A^T)^n \boldsymbol{r}_0^*, \boldsymbol{r}_n^{\mathrm{BCG}})}. \tag{2.4}$$

Here, we attempt to use an n degree polynomial H_n to accelerate $\boldsymbol{r}_n^{\mathrm{BCG}}$, say, make $H_n(A)\boldsymbol{r}_n^{\mathrm{BCG}}$ 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

$$H_n(\lambda) R_n(\lambda) \tag{2.5}$$

  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 ζ_n and η_n and define the polynomial H_n as follows:

$$H_0(\lambda) := 1, \quad H_1(\lambda) := (1 - \zeta_0 \lambda) H_0(\lambda), \tag{2.6}$$

$$H_{n+1}(\lambda) := (1 + \eta_n - \zeta_n \lambda) H_n(\lambda) - \eta_n H_{n-1}(\lambda) \tag{2.7}$$

where ζ_n and η_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:

$$H_{n+1}(\lambda) = H_n(\lambda) - \lambda G_n(\lambda); \quad G_{n+1}(\lambda) = \zeta_{n+1} H_{n+1}(\lambda) + \eta_{n+1} G_n(\lambda). \tag{2.8}$$

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

$$\boldsymbol{r}_n := H_n(A)\boldsymbol{r}_n^{\mathrm{BCG}} = \boldsymbol{b} - A\boldsymbol{x}_n \tag{2.9}$$

which can be obtained with the following iterates:

$$\boldsymbol{t}_n := H_n(A)\boldsymbol{r}_{n+1}^{\mathrm{BCG}}, \quad \boldsymbol{y}_n := AG_{n-1}(A)\boldsymbol{r}_{n+1}^{\mathrm{BCG}}, \quad \boldsymbol{p}_n := H_n(A)\boldsymbol{p}_n^{\mathrm{BCG}}, \tag{2.10}$$

$$\boldsymbol{w}_n := AH_n(A)\boldsymbol{p}_{n+1}^{\mathrm{BCG}}, \quad \boldsymbol{u}_n := AG_n(A)\boldsymbol{p}_n^{\mathrm{BCG}}, \quad \boldsymbol{z}_n := G_n(A)\boldsymbol{r}_{n+1}^{\mathrm{BCG}}. \tag{2.11}$$

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 $\boldsymbol{r}_n$, $\boldsymbol{p}_n$, $\boldsymbol{t}_n$, $\boldsymbol{u}_n$, $\boldsymbol{w}_n$, $\boldsymbol{y}_n$ and $\boldsymbol{z}_n$:

$$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \eta_n \boldsymbol{y}_n - \zeta_n A\boldsymbol{t}_n \tag{2.12}$$

$$= \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n - A\boldsymbol{z}_n, \tag{2.13}$$

$$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n, \tag{2.14}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{t}_n - \boldsymbol{r}_{n+1} - \alpha_{n+1}\boldsymbol{w}_n + \alpha_{n+1}A\boldsymbol{p}_{n+1}, \tag{2.15}$$

$$\boldsymbol{p}_{n+1} = \boldsymbol{r}_{n+1} + \beta_n(\boldsymbol{p}_n - \boldsymbol{u}_n), \tag{2.16}$$

$$\boldsymbol{w}_n = A\boldsymbol{t}_n + \beta_n A\boldsymbol{p}_n, \tag{2.17}$$

$$\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1}), \tag{2.18}$$

$$\boldsymbol{z}_n = \zeta_n \boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n. \tag{2.19}$$

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

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n. \tag{2.20}$$

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

$$(\boldsymbol{r}_0^*, \boldsymbol{r}_n) = (-1)^n \prod_{i=0}^{n-1} \zeta_i((A^T)^n \boldsymbol{r}_0^*, \boldsymbol{r}_n^{\mathrm{BCG}}), \quad (\boldsymbol{r}_0^*, A\boldsymbol{p}_n) = (-1)^n \prod_{i=0}^{n-1} \zeta_i((A^T)^n \boldsymbol{r}_0^*, A\boldsymbol{p}_n^{\mathrm{BCG}}).$$

Then, from the formula (2.4), α_n and β_n can be recovered from the iterates $\boldsymbol{r}_{n+1}$, $\boldsymbol{r}_n$ and $\boldsymbol{p}_n$:

$$\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}, \quad \beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}. \tag{2.21}$$

Due to the lack of a criterion for choices of ζ_n and η_n, it is very hard in fact to determine the parameters ζ_n and η_n that are closely and indissolubly connected with convergence behaviour in practice. Implementations of the parameters ζ_n and η_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 ζ_n and η_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 ζ_n and η_n in terms of minimizing the residual 2-norm as the function of ζ and η:

$$f(\zeta,\eta) := \| \boldsymbol{r}_{n+1} \| = \| \boldsymbol{t}_n - \eta \boldsymbol{y}_n - \zeta A\boldsymbol{t}_n \| .$$

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

ALGORITHM **2** Unpreconditioned GPBi-CG

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = b - A\boldsymbol{x}_0$; set $\boldsymbol{r}_0^* = \boldsymbol{r}_0$, $\boldsymbol{t}_{-1} = \boldsymbol{w}_{-1} = 0$, $\beta_{-1} = 0$;
for $n = 0, 1, \cdots$ until $\| \boldsymbol{r}_n \| \leq \varepsilon \| b \|$ do :
$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \boldsymbol{u}_{n-1})$,
$\alpha_n = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)}$,
$\boldsymbol{y}_n = \boldsymbol{t}_{n-1} - \boldsymbol{r}_n - \alpha_n \boldsymbol{w}_{n-1} + \alpha_n A\boldsymbol{p}_n$,
$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$,
$\zeta_n = \dfrac{(\boldsymbol{y}_n, \boldsymbol{y}_n)(A\boldsymbol{t}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{y}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (\boldsymbol{y}_n, A\boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{y}_n)}$,
$\eta_n = \dfrac{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, A\boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (\boldsymbol{y}_n, A\boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{y}_n)}$,
(if n = 0, then $\zeta_n = \dfrac{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)}$, $\eta_n = 0$)
$\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n + \eta_n(\boldsymbol{t}_{n-1} - \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{u}_{n-1})$,
$\boldsymbol{z}_n = \zeta_n \boldsymbol{r}_n + \eta_n \boldsymbol{z}_{n-1} - \alpha_n \boldsymbol{u}_n$,
$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n$,
$\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \eta_n \boldsymbol{y}_n - \zeta_n A\boldsymbol{t}_n$,
$\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}$,
$\boldsymbol{w}_n = A\boldsymbol{t}_n + \beta_n A\boldsymbol{p}_n$;

3.2 The Choice for CGS

Suppose that $\zeta_n = \alpha_n$ and $\eta_n = \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n$ 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 $\boldsymbol{t}_{n-1} - \boldsymbol{r}_n = A\boldsymbol{z}_{n-1}$. In this case, we have $H_n = R_n$ and $G_n = P_n$, and use the recurrence formula (2.13) to update $\boldsymbol{r}_{n+1}$. This fact leads to relation $\boldsymbol{p}_n - \boldsymbol{u}_n = \boldsymbol{z}_n/\alpha_n$ for any n, then the iterates $\boldsymbol{t}_n$, $\boldsymbol{y}_n$ and $\boldsymbol{w}_n$ can be omitted in the recurrence formulas (2.12)$\sim$(2.19).

Noticing that $(\boldsymbol{r}_0^*, A\boldsymbol{p}_n) = (\boldsymbol{r}_0^*, A\boldsymbol{u}_n)$, and setting new iterates $\boldsymbol{u}_n := A^{-1}\boldsymbol{u}_n/\alpha_n$ and $\boldsymbol{z}_n := \boldsymbol{z}_n/\alpha_n$, then the algorithm of CGS [5] is recalled as follows:

ALGORITHM **3** Unpreconditioned CGS

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = b - A\boldsymbol{x}_0$; set $\boldsymbol{r}_0^* = \boldsymbol{r}_0$, $\beta_{-1} = 0$;
for $n = 0, 1, \cdots$ until $\| \boldsymbol{r}_n \| \leq \varepsilon \| b \|$ do :
$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}\boldsymbol{z}_{n-1}$,

$$
\begin{aligned}
&\boldsymbol{u}_n = \boldsymbol{p}_n + \beta_{n-1}(\boldsymbol{z}_{n-1} + \beta_{n-1}\boldsymbol{u}_{n-1}),\\
&\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{u}_n)},\\
&\boldsymbol{z}_n = \boldsymbol{p}_n - \alpha_n A\boldsymbol{u}_n,\\
&\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n(\boldsymbol{p}_n + \boldsymbol{z}_n),\\
&\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n A(\boldsymbol{p}_n + \boldsymbol{z}_n),\\
&\beta_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)};
\end{aligned}
$$

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 η_n by a quantity ω called the relaxation factor in advance. Here, we suppose that $\eta_n = 0$ for any n, and ζ_n is selected to minimize the residual 2-norm as function of ζ

$$f(\zeta) := \| \boldsymbol{r}_{n+1} \| = \| \boldsymbol{t}_n - \zeta A\boldsymbol{t}_n \| .$$

In this case, $\boldsymbol{u}_n = \zeta_n A\boldsymbol{p}_n$, and then $\boldsymbol{z}_n = \zeta_n \boldsymbol{t}_n$. Notice that the iterates $\boldsymbol{y}_n$, $\boldsymbol{u}_n$, $\boldsymbol{z}_n$ and $\boldsymbol{w}_n$ become worthless. In this way an important and economical variant will be obtained again, recalled Bi-CGSTAB[7]:

ALGORITHM 4 Unconditioned Bi-CGSTAB

$$
\begin{aligned}
&\boldsymbol{x}_0 \text{ is an initial guess, } \boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0; \text{ set } \boldsymbol{r}_0^* = \boldsymbol{r}_0,\ \beta_{-1} = 0;\\
&\text{for } n = 0, 1, \cdots \text{ until } \| \boldsymbol{r}_n \| \le \varepsilon \| \boldsymbol{b} \| \text{ do:}\\
&\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_{n-1}(\boldsymbol{p}_{n-1} - \zeta_{n-1} A\boldsymbol{p}_{n-1}),\\
&\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)},\\
&\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n,\\
&\zeta_n = \frac{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)},\\
&\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \zeta_n \boldsymbol{t}_n,\\
&\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \zeta_n A\boldsymbol{t}_n,\\
&\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)};
\end{aligned}
$$

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 imcomplete LU factorizations [3].

Let K be a suitable preconditioning matrix, i. e., $K \approx A$. We write $K = K_1 K_2$, and apply CGS, Bi-CGSTAB, and GPBi-CG to the explicitly preconditioned system

$$\tilde{A}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}. \tag{4.1}$$

with $\tilde{A} = K_1^{-1} A K_2^{-1}$, $\tilde{\boldsymbol{x}} = K_2 \boldsymbol{x}$, and $\tilde{\boldsymbol{b}} = K_1^{-1}\boldsymbol{b}$. 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 $\tilde{}$, e.g., $\tilde{r}_n$.

With the change of variables:

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

then we have the algorithm of preconditioned GPBi-CG. Here, for computing ζ_n and η_n, we are minimizing the current residual for the original system rather than the preconditioned one.

ALGORITHM 5 Preconditioned 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 $\| r_n \| \leq \varepsilon \| b \|$ do :
$p_n = K^{-1} r_n + \beta_{n-1}(p_{n-1} - u_{n-1})$,
$\alpha_n = \dfrac{(r_0^*, r_n)}{(r_0^*, Ap_n)}$,
$y_n = t_{n-1} - r_n - \alpha_n w_{n-1} + \alpha_n A p_n$,
$t_n = r_n - \alpha_n A p_n$,
$K^{-1} t_n = K^{-1} r_n - \alpha_n K^{-1} A p_n$,
$\zeta_n = \dfrac{(y_n, y_n)(AK^{-1}t_n, t_n) - (y_n, t_n)(AK^{-1}t_n, y_n)}{(AK^{-1}t_n, AK^{-1}t_n)(y_n, y_n) - (y_n, AK^{-1}t_n)(AK^{-1}t_n, y_n)}$,
$\eta_n = \dfrac{(AK^{-1}t_n, AK^{-1}t_n)(y_n, t_n) - (y_n, AK^{-1}t_n)(AK^{-1}t_n, t_n)}{(AK^{-1}t_n, AK^{-1}t_n)(y_n, y_n) - (y_n, AK^{-1}t_n)(AK^{-1}t_n, y_n)}$,
(if n = 0, then $\zeta_n = \dfrac{(AK^{-1}t_n, t_n)}{(AK^{-1}t_n, AK^{-1}t_n)}$, $\eta_n = 0$)
$u_n = \zeta_n K^{-1} A p_n + \eta_n (K^{-1} t_{n-1} - K^{-1} r_n + \beta_{n-1} u_{n-1})$,
$z_n = \zeta_n K^{-1} r_n + \eta_n z_{n-1} - \alpha_n u_n$,
$x_{n+1} = x_n + \alpha_n p_n + z_n$,
$r_{n+1} = t_n - \eta_n y_n - \zeta_n A K^{-1} t_n$,
$\beta_n = \dfrac{\alpha_n}{\zeta_n} \cdot \dfrac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}$,
$w_n = AK^{-1} t_n + \beta_n A p_n$;

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

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

we have the algorithm of preconditioned CGS.

ALGORITHM 6 Preconditioned 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 $\| r_n \| \leq \varepsilon \| b \|$ do :
$p_n = r_n + \beta_{n-1} z_{n-1}$,
$u_n = p_n + \beta_{n-1}(z_{n-1} + \beta_{n-1} u_{n-1})$,
$\alpha_n = \dfrac{(r_0^*, r_n)}{(r_0^*, AK^{-1}u_n)}$,

$$
\begin{aligned}
z_n &= p_n - \alpha_n A K^{-1} u_n, \\
x_{n+1} &= x_n + \alpha_n K^{-1}(p_n + z_n), \\
r_{n+1} &= r_n - \alpha_n A K^{-1}(p_n + z_n), \\
\beta_n &= \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)};
\end{aligned}
$$

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

$$\tilde{x}_n \Rightarrow K_2 x_n,\ \tilde{p}_n \Rightarrow K_1^{-1} p_n,\ \tilde{r}_n \Rightarrow K_1^{-1} r_n,\ \tilde{t}_n \Rightarrow K_1^{-1} t_n,\ \tilde{r}_0^* = K_1^T r_0^*,$$

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

ALGORITHM 7 Preconditioned Bi-CGSTAB

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

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 uaually 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 $L^{-1}AU^{-1}(Ux) = L^{-1}b$ 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 $F = 100$.

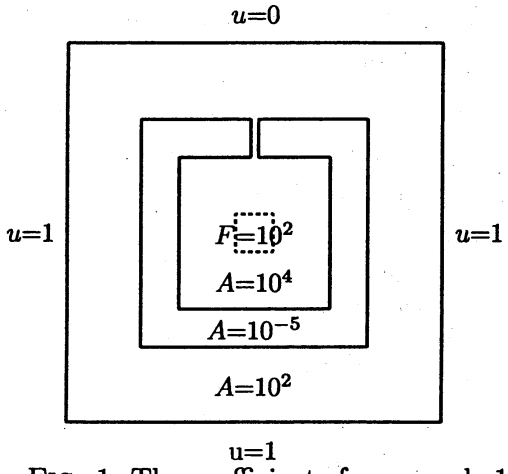

FIG. 1. The coefficients for example 1.

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

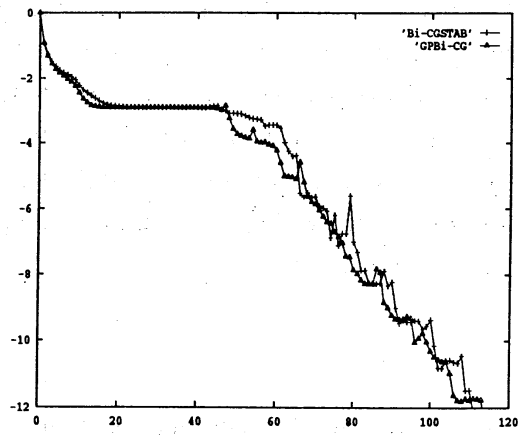

FIG. 2. The history of the residual 2-norms (*i. e.*, $\log(\| r_n \| / \| r_0 \|)$ *vs.* n) for example 1 ($\gamma = 2$)

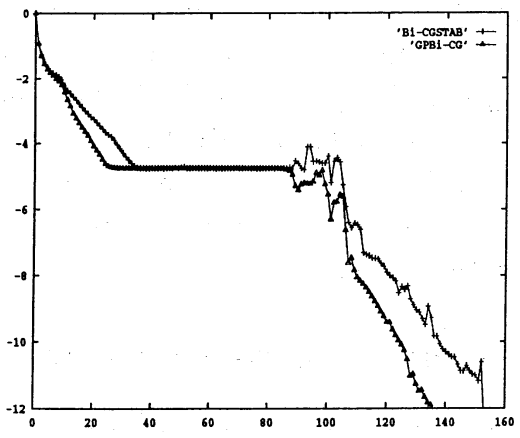

FIG. 3. The history of the residual 2-norms (*i. e.*, $\log(\| r_n \| / \| r_0 \|)$ *vs.* n) for example 1 ($\gamma = 0$)

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 100×100 and a 200×200 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 $u_x = i\sqrt{k^2 - \frac{1}{4}}\cos(\frac{y}{2})$ 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 unkowns 101×100, 201×200. Here we only consider the case $k = 2.27$. The linear systems were preconditioned by incomplete LU factorizations.

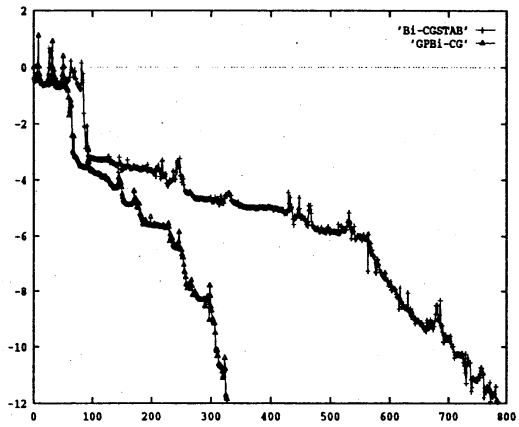

FIG. 4. The history of the residual 2-norms (*i. e.*, $\log(\| r_n \| / \| r_0 \|)$) *vs.* n) for example 2 (101×100)

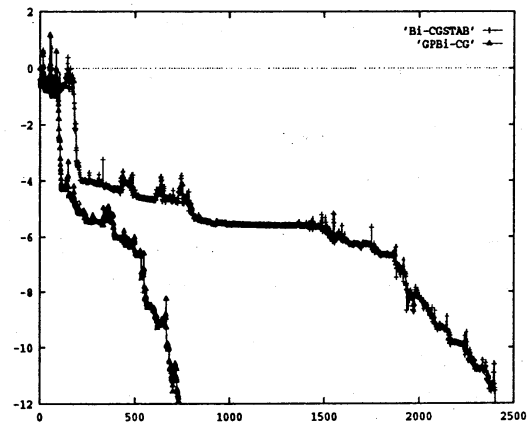

FIG. 5. The history of the residual 2-norms (*i. e.*, $\log(\| r_n \| / \| r_0 \|)$) *vs.* n) for example 2 (201×200)

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

[1] A. Bayliss, C. I. Glodstein and E. Turkel, *An iterative method for the Helmholtz equation*, J. Comput. Phys., 49(1983), pp. 443-457.

[2] R. Fletcher, *Conjugate gradient methods for indefinite systems*, Lecture Notes in Mathematics 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73-89.

[3] J. A. Meijerink and H. A. Van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31(1977), pp. 148-162.

[4] Y. Saad and M. H. Schultz, GMRES: *A generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Stat. Comput., 7(1986), pp. 856-869.

[5] P. Sonneveld, CGS, *A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10(1989), pp. 36-52.

[6] E. L. Stiefel, *Kernel polynomials in linear algebra and their numerical applications*, in: Further contributions to the determination of eigenvalues, NBS Applied Math. Ser., 49(1958), pp. 1-22.

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

[8] S. L. Zhang, GPBi-CG: *Generalized Product-type Methods Based on* Bi-CG *for Solving Nonsymmetric Linear Systems.* Submitted to SIAM J. Sci. Comput., 18(1997), to appear.