数理解析研究所講究録
第 [illegible] 巻 1993 年 86-103

ランチョス・プロセスに基づく積型反復解法とその一般化

張紹良　藤野清次 (計算流体力学研究所)

Abstract

Recently Bi-CGSTAB as a variant of Bi-CG has been proposed for solving nonsymmetric linear systems, and its attractive convergence behaviour has been confirmed in many numerical experiments. Bi-CGSTAB can be characterized by its residual polynomial which consists of the product of the residual polynomial of Bi-CG with other polynomial of form generated from two-term recurrence relations. In this paper, we propose an unified way to generalize a class of product-type methods whose residual polynomials can be factored by the residual polynomial of Bi-CG and other polynomial of form with standard three-term recurrence relations. Such product-type methods that base on Bi-CG can be regarded as generalizations of Bi-CGSTAB. From the unified way, the well-known variants of the product-type methods, like CGS, Bi-CGSTAB, Bi-CGSTAB2, are reacquired again.

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

AMS(MOS) subject classification. 65F10

1 Introduction

For solving nonsymmetric linear systems, the Bi-Conjugate Gradient (Bi-CG hereafter)[8, 3] was presented as a generalization of the Conjugate Gradient (CG hereafter)[7] for solving symmetric positive definte systems. However, operations of transpose matrix-vector multications are needed in Bi-CG 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 reconstructing Bi-CG. A common technique to design a new method by means of reconstructing

Bi-CG is to define its residual polynomial by product of two polynomial factors where one factor is the residual polynomial of Bi-CG and the other one is an undetermined polynomial of form. For example, the Conjugate Gradients-Squared (CGS hereafter)[13] and Bi-CGSTAB[16] were derived from Bi-CG by this technique. In CGS, P. Sonneveld defined the undetermined polynomial of form by the same residual polynomial of Bi-CG, 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[13, 11]. In many situations, however, the property of square may cause the convergence behaviour of CGS to be rather irregular and oscillatory because of the presence of the round-off errors[16, 17, 6]. Therefore, in Bi-CGSTAB, H. A. Van der Vorst selected a polynomial of form 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 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[16, 5, 1]. Besides, M. H. Gutknecht replaced the two-term recurrent polynomial in Bi-CGSTAB with a three-term recurrent polynomial only at even iterations to solve linear systems with complex spectrum, and implemented the algorithm of Bi-CGSTAB2[4]. Bi-CGSTAB2 is explained as a generalizaton of Bi-CGSTAB, say, the combination Bi-CG with GMRES(2), in the sense that Bi-CGSTAB is the combination of Bi-CG with GMRES(1)[4, 12].

The purpose of this paper is to propose an unified way to extend Bi-CGSATB. In the sense to define the undetermined polynomial of form by standard three-term recurrence relations modelled after the residual polynomial of Bi-CG, we generalize a class of generalizatons of Bi-CGSTAB by means of reconstruting Bi-CG. The well-known CGS, Bi-CGSTAB and Bi-CGSTAB2 are included in the generalized product-type methods based on Bi-CG without the disadvantage to store extra iterates like that in GMRES[12] and GCR[2]. This paper is organized as follows: in the next section, we introduce the background concerned with reconstructing Bi-CG, and characterize the product-type methods based on Bi-CG. In §3, we propose a standard three-term recurrent polynomial of form to construct the residual polynomial of the product-type

methods, and derive the recurrence formulas among the related iterates. In §4, an equivalent approach to reduce the recurrence formulas is presented. Several implementations of variants of the product-type methods are considered, and some well-known variants are recalled in §5.

Throughout this paper, superscripts BCG, CGS, STA, ST2 are used to distinguish iterates generated in the algorithms of Bi-CG, CGS, Bi-CGSTAB and Bi-CGSTAB2 respectively.

2 Reconstructing Bi-CG

To begin with, let us consider the solution of the following linear system

$$Ax = b \tag{2.1}$$

where A is an $N \times N$ large and sparse nonsymmetric matrix, and always assumed to be nonsingular.

The algorithm of Bi-CG for solving the system (2.1), described first by C. Lanczos[8], was given by R. Fletcher[3] as follows:

ALGORITHM 1 Bi-CG

Choose an initial guess x_0^{BCG},
and 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:

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

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

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

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

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

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

$$K_n(A; r_0) := \mathrm{span}\{r_0, Ar_0, \cdots, A^{n-1}r_0\} \text{ and} \tag{2.2}$$

$$K_n(A^T; r_0^*) := \mathrm{span}\{r_0^*, A^T r_0^*, \cdots, (A^T)^{n-1} r_0^*\} \tag{2.3}$$

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[3].

$$r_n^{\mathrm{BCG}} \perp K_n(A^T; r_0^*), \quad \text{and} \quad Ap_n^{\mathrm{BCG}} \perp K_n(A^T; r_0^*). \tag{2.4}$$

Let R_n and P_n denote the polynomials corresponding to the residual r_n^{BCG} and the direction p_n^{BCG}, then the iterates r_n^{BCG}, p_n^{BCG}, $r_n^{\mathrm{BCG}*}$ and $p_n^{\mathrm{BCG}*}$ can been expressed as follows:

$$r_n^{\mathrm{BCG}} = R_n(A)r_0,\ p_n^{\mathrm{BCG}} = P_n(A)r_0,\ r_n^{\mathrm{BCG}*} = R_n(A^T)r_0^*,\ p_n^{\mathrm{BCG}*} = P_n(A^T)r_0^*. \tag{2.5}$$

R_n is the residual polynomial of Bi-CG, and so-called the Lanczos polynomial. Notice that the basic recurrence relations between R_n and P_n hold as follows:

$$\begin{aligned} &R_0(\lambda) = 1, \quad P_0(\lambda) = 1, \\ &R_{n+1}(\lambda) = R_n(\lambda) - \alpha_n \lambda P_n(\lambda), \quad P_{n+1}(\lambda) = R_{n+1}(\lambda) + \beta_n P_n(\lambda), \quad \text{for } n = 1, 2, \cdots, \end{aligned} \tag{2.6}$$

Throughout this paper, it is always assumed that the residual r_n^{BCG} is convergent towards zero. In the sequel, it is known that two disadvantages appear in the algorithm of Bi-CG:

- to construct the sequence of the residuals r_0^{BCG}, r_1^{BCG}, $\cdots$, r_n^{BCG} in the Krylov subspace $K_n(A, r_0)$, one has to use the Krylov subspace $K_n(A^T, r_0^*)$ so that the operations of transpose matrix-vector multiplications appear per iteration;
- although $r_n^{\mathrm{BCG}*}$ also converges towards zero, this convergence property can not be exploited directly for improving the convergence rate of the residual r_n^{BCG}.

Next, an unified way to remedy above disadvantages is proposed. Here, we attempt to use a polynomial H_n to accelerate r_n^{BCG}, say, make $H_n(A)r_n^{\mathrm{BCG}}(= H_n(A)R_n(A)r_0)$ converge towards zero fast. By doing so, we can derive a class of methods which have product-type residual polynomial from reconstructing Bi-CG. Here, 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 the following two n degree polynomials of forms

$$H_n(\lambda)R_n(\lambda) \tag{2.7}$$

where R_n is the Lanczos polynomial and H_n is undetermined.

In the product-type methods, $r_n^{\mathrm{BCG}*}$ would be never formed, nor the transpose matrix-vector multiplications would be needed. According to the description above, we know that CGS, Bi-CGSTAB and Bi-CGSTAB2 all belong to such kind of product-type methods based on Bi-CG.

In general, when one attempts to derive a new product-type method from Bi-CG, one would encounter the following three problems:

(A) by means of establishing new formulas, to compute the first-kind of parameters α_n and β_n which are used to determine R_n;

(B) to define the construction of H_n;

(C) by analyzing the stability of the methods, to compute the second-kind of parameters which are used to determine H_n. This aspect is the key-point to design the product-type methods.

In fact, the problem (A) can be solved by a technique depended on the orthogonalities (2.4). This technique is due to P. Sonneveld[13], and developed later by H. A. Van der Vorst[16]. Notice that $r_n^{\mathrm{BCG}*}$ and $p_n^{\mathrm{BCG}*}$ can be written as

$$r_n^{\mathrm{BCG}*} = R_n(A^T)r_0^* = \left((-1)^n \prod_{i=0}^{n-1} \alpha_i\right)(A^T)^n r_0^* + z_1;$$

$$p_n^{\mathrm{BCG}*} = P_n(A^T)r_0^* = \left((-1)^n \prod_{i=0}^{n-1} \alpha_i\right)(A^T)^n r_0^* + z_2.$$

with z_1 and $z_2 \in K_n(A^T, r_0^*)$. By the orthogonalities (2.4), auxiliary formulas for computing α_n and β_n can be recovered:

$$\alpha_n = \frac{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})}{(p_n^{\mathrm{BCG}*}, Ap_n^{\mathrm{BCG}})} = \frac{((A^T)^n r_0^*, r_n^{\mathrm{BCG}})}{((A^T)^n r_0^*, Ap_n^{\mathrm{BCG}})}, \tag{2.8}$$

$$\beta_n = \frac{(r_{n+1}^{\mathrm{BCG}*}, r_{n+1}^{\mathrm{BCG}})}{(r_n^{\mathrm{BCG}*}, r_n^{\mathrm{BCG}})} = -\alpha_n \frac{((A^T)^{n+1} r_0^*, r_{n+1}^{\mathrm{BCG}})}{((A^T)^n r_0^*, r_n^{\mathrm{BCG}})}. \tag{2.9}$$

The formulas (2.8)~(2.9) imply that we can directly use the basis $(A^T)^n r_0^*$ of the Krylov subspace $K_{n+1}(A^T; r_0^*)$ instead of vectors $r_n^{\mathrm{BCG}*}$ and $p_n^{\mathrm{BCG}*}$ to determine the first-kind of parameters α_n and β_n in the algorithm of Bi-CG. The formulas are fundamental to establish the product-type methods based on Bi-CG.

In order to deal with the problem (B), i.e. designing the polynomial of form H_n in (2.7), three different polynomials of forms had been presented[13, 16, 4]:

- in CGS, the polynomial of form H_n is chosen to be the Lanczos polynomial R_n;
- in Bi-CGSTAB, the polynomial of form H_n is defined by a factored polynomial of form Q_n determined by a second-kind of parameter ω_{n-1}

$$Q_n = (1 - \omega_{n-1}\lambda)Q_{n-1}; \tag{2.10}$$

- in Bi-CGSTAB2, the polynomial of form H_n is denoted by τ_n and chosen to satisfy the following recurrence relations

$$\tau_{2n+1} = (1 - \chi_n\lambda)\tau_{2n}, \quad \tau_{2n+2} = (\zeta_n + \eta_n\lambda)\tau_{2n+1} + (1 - \zeta_n)\tau_{2n}. \tag{2.11}$$

The operations for the second-kind of parameters can be omitted in CGS. For the problem (C), the second-kind of parameter ω_n (resp. χ_n, ζ_n and η_n) in Bi-CGSTAB (resp. Bi-CGSTAB2) is determined in the such way that the residual norm per iteration can be minimized locally.

The design of the polynomial of form H_n in practice is desired as:

(1) to make the polynomial of form H_n satisfy short-term recurrence relations so that little computational work and low storage costs are required per iteration;

(2) to choose the second-parameter of H_n reasonably to get rather fast and stable convergence behaviour.

3 Generalized product-type methods based on Bi-CG

In this section, we will describe the basic idea to establish a standard polynomial of form H_n which leads to generalized product-type methods based on Bi-CG for solving the linear system (2.1).

3.1 Construction of polynomial of form

By eliminating P_n from (2.6), one would recover the three-term recurrence relations for R_n alone[14]:

$$R_0(\lambda) = 1, \quad R_1(\lambda) = (1 - \alpha_0\lambda)R_0(\lambda), \tag{3.1}$$

$$R_{n+1}(\lambda) = \left(1 + \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n - \alpha_n\lambda\right) R_n(\lambda) - \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n R_{n-1}(\lambda), \quad n = 1, 2, \cdots. \tag{3.2}$$

The three-term recurrence relations are fundamental for the efficiency of all iterative methods based on the Lanczos process. In fact, the three-term recurrence relations of Bi-CG provide a modest hint that motivate our search for developing generalized product-type methods based on Bi-CG. Here, we introduce two independent parameters ζ_n and η_n and let the polynomial of form H_n modell after the three-term recurrence relations (3.1)~(3.2) of R_n as follows:

$$H_0(\lambda) := 1, \quad H_1(\lambda) := (1 - \zeta_0\lambda)H_0(\lambda), \tag{3.3}$$

$$H_{n+1}(\lambda) := (1 + \eta_n - \zeta_n\lambda)H_n(\lambda) - \eta_n H_{n-1}(\lambda). \tag{3.4}$$

where the undetermined parameters ζ_n and η_n belong to the second-kind.

3.2 Recurrence formulas for iterates

Using the basic recurrence relations (2.6), we have the following set of recurrence relations among the products of polynomials H_nR_n, H_nP_n, $H_{n-1}R_n$,$H_{n-2}R_n$ and $H_{n-1}P_n$:

$$H_{n+1}R_{n+1} = (1 + \eta_n)H_nR_{n+1} - \zeta_n\lambda H_nR_{n+1} - \eta_n H_{n-1}R_{n+1}, \tag{3.5}$$

$$H_{n+1}P_{n+1} = H_{n+1}R_{n+1} + \beta_n(1 + \eta_n - \zeta_n\lambda)H_nP_n - \beta_n\eta_n H_{n-1}P_n, \tag{3.6}$$

$$H_nR_{n+1} = H_nR_n - \alpha_n\lambda H_nP_n, \tag{3.7}$$

$$H_{n-1}R_{n+1} = H_{n-1}R_n - \alpha_n\lambda H_{n-1}P_n, \tag{3.8}$$

$$H_nP_{n+1} = H_nR_{n+1} + \beta_n H_nP_n. \tag{3.9}$$

Let us build up the product-type methods with residual:

$$r_n := H_n(A)r_n^{\mathrm{BCG}} = H_nR_n(A)r_0 = b - Ax_n \tag{3.10}$$

which can be obtained with the following iterates:

$$(3.11)\quad p_n := H_n(A)p_n^{\mathrm{BCG}},\ t_n := H_{n-1}(A)r_n^{\mathrm{BCG}},\ s_n := H_{n-2}(A)r_n^{\mathrm{BCG}},\ w_n := H_{n-1}(A)p_n^{\mathrm{BCG}}.$$

According to the recurrence relations (3.5)~(3.9), we have the following recurrence formulas among the sequences of the iterates r_n, p_n, t_n, s_n and w_n:

$$(3.12)\qquad r_{n+1} = -\eta_n s_{n+1} + (1+\eta_n)t_{n+1} - \zeta_n A t_{n+1},$$

$$(3.13)\qquad p_{n+1} = r_{n+1} - \beta_n\eta_n w_n + \beta_n(1+\eta_n)p_n - \beta_n\zeta_n A p_n,$$

$$(3.14)\qquad t_{n+1} = r_n - \alpha_n A p_n,$$

$$(3.15)\qquad s_{n+1} = t_n - \alpha_n A w_n,$$

$$(3.16)\qquad w_{n+1} = t_{n+1} + \beta_n p_n.$$

From the definition (3.10) of the residual r_n and the recurrence formula (3.12) of the residual r_n, we have the formula to update the approximating solution x_{n+1}:

$$(3.17)\quad x_{n+1} = -\eta_n(x_{n-1} + \alpha_{n-1}p_{n-1} + \alpha_n w_n) + (1+\eta_n)(x_n + \alpha_n p_n) + \zeta_n t_{n+1}.$$

3.3 Computations for α_n and β_n

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

$$(r_0^*, r_n) = (r_0^*, H_n(A)r_n^{\mathrm{BCG}}) = (H_n(A^T)r_0^*, r_n^{\mathrm{BCG}}) = \left((-1)^n \prod_{i=0}^{n-1}\zeta_i\right)((A^T)^n r_0^*, r_n^{\mathrm{BCG}}),$$

$$(r_0^*, Ap_n) = (r_0^*, AH_n(A)p_n^{\mathrm{BCG}}) = (H_n(A^T)r_0^*, Ap_n^{\mathrm{BCG}}) = \left((-1)^n \prod_{i=0}^{n-1}\zeta_i\right)((A^T)^n r_0^*, Ap_n^{\mathrm{BCG}}).$$

Then, from the formulas (2.8)~(2.9), α_n and β_n can be recovered from the iterates r_{n+1}, r_n and p_n:

$$(3.18)\qquad \alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)},\quad \beta_n = \frac{\alpha_n}{\zeta_n}\cdot\frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}.$$

Now, we have accomplished a part of the purpose of this paper, i.e. proposed an unified way to derive generalized product-type methods from the reconstruction of Bi-CG. Due to the lack

of a criterion for choices of ζ_n and η_n, it is very hard in fact to determine the second-kind of parameters ζ_n and η_n that are closely and indissolubly connected with convergence behaviour in practice. Implementations of the second-kind of parameters ζ_n and η_n will be discussed in §5 in detail.

4 Equivalent approach

Notice that the n-1th iterates x_{n-1} and p_{n-1} are used to update x_{n+1} in the formula (3.17). It is not appreciated in any ideal iterative method. In this section, we propose another approach to establish polynomial of form $H_n(\lambda)$ in (2.7) which is mathematically equivalent to the polynomial of form (3.4). By doing so, we can present new class of product-type methods in which x_{n+1} can be updated without the n-1th iterates, like x_{n-1} and p_{n-1}.

4.1 Equivalent polynomial of form

Let $H_n(\lambda)$ be generated by the three-term recurrence realtions (3.4), we find an important fact that $H_n(0) = 1$ holds for any n. Then, 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 so that:

$$H_{n+1}(\lambda) - H_n(\lambda) = -\zeta_n \lambda G_n(\lambda). \tag{4.1}$$

Rewrite the three-term recurrence realtions (3.4) as

$$H_{n+1}(\lambda) - H_n(\lambda) = -\zeta_n \lambda H_n(\lambda) + \eta_n (H_n(\lambda) - H_{n-1}(\lambda)), \tag{4.2}$$

and let $\mu_n = \zeta_n$ and $\nu_n = \zeta_n \dfrac{\eta_{n+1}}{\zeta_{n+1}}$, then the double sets of polynomials $H_n(\lambda)$ and $G_n(\lambda)$ mutually interlocked by the following recurrence relations can be obtained:

$$H_{n+1}(\lambda) = H_n(\lambda) - \mu_n \lambda G_n(\lambda); \tag{4.3}$$

$$G_{n+1}(\lambda) = H_{n+1}(\lambda) + \nu_n G_n(\lambda). \tag{4.4}$$

Using the recurrence relations (4.3) and (4.4), we can reduce the formula (3.17).

4.2 Equivalent recurrence formulas for iterates

To compute the product of polynomials H_nR_n, we make use of a new set of recurrence relations among the products of polynomials H_nR_n, H_nP_n, G_nR_n, G_nP_n and $G_{n-1}R_n$ instead of the set of recurrence relations (3.5)~(3.9):

$$H_{n+1}R_{n+1} = H_nR_n - \alpha_n\lambda H_nP_n - \mu_n\lambda G_nR_{n+1}, \tag{4.5}$$

$$H_{n+1}P_{n+1} = H_{n+1}R_{n+1} + \beta_n H_{n+1}P_n, \tag{4.6}$$

$$G_nR_{n+1} = G_nR_n - \alpha_n\lambda G_nP_n, \tag{4.7}$$

$$G_{n+1}R_{n+1} = H_{n+1}R_{n+1} + \nu_n G_nR_{n+1}, \tag{4.8}$$

$$G_{n+1}P_{n+1} = H_{n+1}P_{n+1} + \nu_n G_nR_{n+1} + \beta_n\nu_n G_nP_n. \tag{4.9}$$

To obtain residaul $r_n := H_n(A)r_n^{\mathrm{BCG}}$, let us define auxiliary iterates as follows:

$$p_n := H_n(A)p_n^{\mathrm{BCG}},\ u_n := G_n(A)p_n^{\mathrm{BCG}},\ t_n := G_n(A)r_n^{\mathrm{BCG}},\ s_n := G_{n-1}(A)r_n^{\mathrm{BCG}}. \tag{4.10}$$

Then, we have the following recurrence formulas among the iterates:

$$r_{n+1} = r_n - \alpha_n Ap_n - \mu_n As_{n+1}, \tag{4.11}$$

$$p_{n+1} = r_{n+1} + \beta_n p_n - \beta_n\mu_n Au_n, \tag{4.12}$$

$$s_{n+1} = t_n - \alpha_n Au_n, \tag{4.13}$$

$$t_{n+1} = r_{n+1} + \nu_n s_{n+1}, \tag{4.14}$$

$$u_{n+1} = p_{n+1} + \nu_n s_{n+1} + \beta_n\nu_n u_n. \tag{4.15}$$

The formula to update the approximative solution x_{n+1} is writen as:

$$x_{n+1} = x_n + \alpha_n p_n + \mu_n s_{n+1}. \tag{4.16}$$

From the equivalence between the three-term recurrence relations (3.4) and (4.3)~(4.4), it is easy to see that the approximative solutions x_{n+1} generated by formula (4.16) is mathematically

equivalent to the approximative solutions x_{n+1} generated by formula (3.17). The formula (3.17) is reduced to the formula (4.16).

Clearly, the formulas of (3.18) for computing α_n and β_n are still valid here:

$$\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)}, \quad \beta_n = \frac{\alpha_n}{\mu_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}. \tag{4.17}$$

5 Implementation details

In accordance with the requirement (2) described at the end of §2, we summarize several possibilities to select μ_n and ν_n (resp. 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 Bi-CGSTAB2 will be recalled in terms of special choices.

5.1 The choice for CGS

Suppose that $\mu_n = \alpha_n$ and $\nu_n = \beta_n$ in recurrence relations (4.3)~(4.4), 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.

In this case, we find that $H_n = R_n$ and $G_n = P_n$. This fact leads the relation $p_n = t_n$ in recurrence formulas (4.11)~(4.15) to hold. Letting r_n^{CGS}, p_n^{CGS}, e_n^{CGS} and h_n^{CGS} denote new iterates $R_n(A)r_n^{\mathrm{BCG}}$, $P_n(A)p_n^{\mathrm{BCG}}$, $R_n(A)p_n^{\mathrm{BCG}}$ and $P_{n-1}(A)r_n^{\mathrm{BCG}}$ respectively, and rearranging recurrence formulas (4.11)~(4.15), then the algorithm of CGS[13] is recalled as follows:

ALGORITHM 2 CGS

Choose an initial guess x_0^{CGS},
and set $r_0^{\mathrm{CGS}*} = p_0^{\mathrm{CGS}} = e_0^{\mathrm{CGS}} = r_0^{\mathrm{CGS}} = b - Ax_0^{\mathrm{CGS}}$.
For $n = 0, 1, \cdots$ until $\| r_n^{\mathrm{CGS}} \| \leq \varepsilon \| b \|$ do:

$$\alpha_n = \frac{(r_0^{\mathrm{CGS}*}, r_n^{\mathrm{CGS}})}{(r_0^{\mathrm{CGS}*}, Ap_n^{\mathrm{CGS}})},$$
$$h_{n+1}^{\mathrm{CGS}} = e_n^{\mathrm{CGS}} - \alpha_n Ap_n^{\mathrm{CGS}},$$
$$x_{n+1}^{\mathrm{CGS}} = x_n^{\mathrm{CGS}} + \alpha_n(e_n^{\mathrm{CGS}} + h_{n+1}^{\mathrm{CGS}}),$$
$$r_{n+1}^{\mathrm{CGS}} = r_n^{\mathrm{CGS}} - \alpha_n A(e_n^{\mathrm{CGS}} + h_{n+1}^{\mathrm{CGS}}),$$
$$\beta_n = \frac{(r_0^{\mathrm{CGS}*}, r_{n+1}^{\mathrm{CGS}})}{(r_0^{\mathrm{CGS}*}, r_n^{\mathrm{CGS}})},$$
$$e_{n+1}^{\mathrm{CGS}} = r_{n+1}^{\mathrm{CGS}} + \beta_n h_{n+1}^{\mathrm{CGS}},$$
$$p_{n+1}^{\mathrm{CGS}} = e_{n+1}^{\mathrm{CGS}} + \beta_n(h_{n+1}^{\mathrm{CGS}} + \beta_n p_n^{\mathrm{CGS}});$$

Note that another version of CGS can be derived from recurrence formulas (3.12)~(3.16) under the assumptions $\zeta_n = \alpha_n$ and $\eta_n = \frac{\beta_{n-1}}{\alpha_n}\alpha_n$, but this version seems to be computationally expensive in practice.

5.2 The choice for Bi-CGSTAB

In the subsection, we attempt to get some variants of the product-type methods with little computational work. Here, suppose that parameter η_n (resp. ν_n) in recurrence formulas (3.12)~(3.16) (resp. (4.11)~(4.15)) had been given in advance, the another parameter ζ_n (resp. μ_n) is selected to minimize the residual norm as function of ζ (resp. μ)

$$f(\zeta) := \| r_{n+1} \| = \| t_{n+1} + \eta_n(t_{n+1} - s_{n+1}) - \zeta A t_{n+1} \|$$

$$(\text{resp. } f(\mu) := \| r_{n+1} \| = \| r_n - \alpha_n A p_n - \mu A s_{n+1} \|)$$

which reaches its minimum value at

$$\zeta_n = \frac{(t_{n+1} + \eta_n(t_{n+1} - s_{n+1}), A t_{n+1})}{(A t_{n+1}, A t_{n+1})}. \quad (\text{resp. } \mu_n = \frac{(r_n - \alpha_n A p_n, A s_{n+1})}{(A s_{n+1}, A s_{n+1})}.)$$

Moreover, we consider the following special case that all η_n (resp. ν_n) are defined by a quantity ω. ω is called the relaxation factor. This aspect about how to choose the relaxation factor for solving realistic problems needs further research.

1. According to the recurrence formulas (3.12)~(3.16), we have the following variant of the product-typr methods, and name it GPBi-CG(ω).

ALGORITHM 3 GPBi-CG(ω)

Choose an initial guess x_0,
and set $r_0^* = p_0 = r_0 = b - Ax_0$, $w_0 = t_0 = 0$, $\eta_0 = 0$.
For $n = 0, 1, \cdots$ until $\| r_n \| \leq \varepsilon \| b \|$ do :

$$\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, A p_n)},$$
$$s_{n+1} = t_n - \alpha_n A w_n,$$
$$t_{n+1} = r_n - \alpha_n A p_n,$$
$$\zeta_n = \frac{(t_{n+1} + \omega(t_{n+1} - s_{n+1}), A t_{n+1})}{(A t_{n+1}, A t_{n+1})},$$
$$x_{n+1} = x_n + \alpha_n p_n + \zeta_n t_{n+1} + \omega(x_n - x_{n-1} + \alpha_n p_n - \alpha_{n-1} p_{n-1} - \alpha_n w_n),$$
$$r_{n+1} = t_{n+1} + \omega(t_{n+1} - s_{n+1}) - \zeta_n A t_{n+1},$$

$$\beta_n = \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)},$$
$$w_{n+1} = t_{n+1} + \beta_n p_n,$$
$$p_{n+1} = r_{n+1} + \beta_n(p_n - \zeta_n A p_n + \omega(p_n - w_n));$$

2. According to the recurrence formulas (4.11)$\sim$(4.15), we have the reduced variant of GPBi-CG(ω), and call it RGPBi-CG(ω).

ALGORITHM 4 RGPBi-CG(ω)

Choose an initial guess x_0,
and set $r_0^* = u_0 = t_0 = p_0 = r_0 = b - Ax_0$.
For $n = 0, 1, \cdots$ until $\| r_n \| \leq \varepsilon \| b \|$ do:
$$\alpha_n = \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)},$$
$$s_{n+1} = t_n - \alpha_n A u_n,$$
$$\mu_n = \frac{(r_n - \alpha_n A p_n, As_{n+1})}{(As_{n+1}, As_{n+1})},$$
$$x_{n+1} = x_n + \alpha_n p_n + \mu_n s_{n+1},$$
$$r_{n+1} = r_n - \alpha_n A p_n - \mu_n A s_{n+1},$$
$$\beta_n = \frac{\alpha_n}{\mu_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)},$$
$$p_{n+1} = r_{n+1} + \beta_n(p_n - \mu_n A u_n),$$
$$t_{n+1} = r_{n+1} + \omega s_{n+1},$$
$$u_{n+1} = p_{n+1} + \omega(s_{n+1} + \beta_n u_n);$$

3. It is clear that x_{n-1} and p_{n-1} appear in the formula to update x_{n+1} in GPBi-CG(ω) is very awkward. Fortunately, we can remedy this difficulty by letting $\omega = 0$. Notice that the iterates s_n and w_n become worthless in this case. In this way an important and economical variant will be obtained again, recalled Bi-CGSTAB[16]:

ALGORITHM 5 Bi-CGSTAB

Choose an initial guess x_0^{STA},
and set $r_0^{\mathrm{STA}*} = p_0^{\mathrm{STA}} = r_0^{\mathrm{STA}} = b - Ax_0^{\mathrm{STA}}$.
For $n = 0, 1, \cdots,$ until $\| r_n^{\mathrm{STA}} \| \leq \varepsilon \| b \|$ do:
$$\alpha_n = \frac{(r_0^{\mathrm{STA}*}, r_n^{\mathrm{STA}})}{(r_0^{\mathrm{STA}*}, Ap_n^{\mathrm{STA}})},$$
$$t_{n+1}^{\mathrm{STA}} = r_n^{\mathrm{STA}} - \alpha_n A p_n^{\mathrm{STA}},$$
$$\zeta_n = \frac{(t_{n+1}^{\mathrm{STA}}, At_{n+1}^{\mathrm{STA}})}{(At_{n+1}^{\mathrm{STA}}, At_{n+1}^{\mathrm{STA}})}$$
$$x_{n+1}^{\mathrm{STA}} = x_n^{\mathrm{STA}} + \alpha_n p_n^{\mathrm{STA}} + \zeta_n t_{n+1}^{\mathrm{STA}},$$

$$
\begin{aligned}
r_{n+1}^{\mathrm{STA}} &= t_{n+1}^{\mathrm{STA}} - \zeta_n A t_{n+1}^{\mathrm{STA}}, \\
\beta_n &= \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^{\mathrm{STA}*}, r_{n+1}^{\mathrm{STA}})}{(r_0^{\mathrm{STA}*}, r_n^{\mathrm{STA}})}, \\
p_{n+1}^{\mathrm{STA}} &= r_{n+1}^{\mathrm{STA}} + \beta_n (p_n^{\mathrm{STA}} - \zeta_n A p_n^{\mathrm{STA}});
\end{aligned}
$$

Similarly, Bi-CGSTAB can be also derived from RGPBi-CG(ω) by setting $\omega = 0$ where the iterates t_n and u_n generated in RGPBi-CG(ω) can be omitted.

5.3 The choice for GPBi-CG

It is convenient to determine parameters ζ_n and η_n in terms of minimizing the residual norm as the function of ζ and η[4]:

$$
f(\zeta, \eta) := \| r_{n+1} \| = \| t_{n+1} + \eta(t_{n+1} - s_{n+1}) - \zeta A t_{n+1} \| .
$$

It suffices to solve a 2×2 linear equations.

Thus, we have the following variant of the product-type methods, and name it GPBi-CG:

ALGORITHM 6 GPBi-CG

Choose an initial guess x_0,
and set $r_0^* = p_0 = r_0 = b - Ax_0$, $w_0 = t_0 = 0$, $\eta_0 = 0$.
For $n = 0, 1, \cdots$ until $\| r_n \| \leq \varepsilon \| b \|$ do:

$$
\begin{aligned}
\alpha_n &= \frac{(r_0^*, r_n)}{(r_0^*, Ap_n)}, \\
s_{n+1} &= t_n - \alpha_n A w_n, \\
t_{n+1} &= r_n - \alpha_n A p_n, \\
\zeta_n &= \frac{(s_{n+1} - t_{n+1}, s_{n+1} - t_{n+1})(t_{n+1}, At_{n+1}) - (s_{n+1} - t_{n+1}, At_{n+1})(t_{n+1}, s_{n+1} - t_{n+1})}{(At_{n+1}, At_{n+1})(s_{n+1} - t_{n+1}, s_{n+1} - t_{n+1}) - (s_{n+1} - t_{n+1}, At_{n+1})^2}, \\
\eta_n &= \frac{(At_{n+1}, At_{n+1})(t_{n+1}, s_{n+1} - t_{n+1}) - (s_{n+1} - t_{n+1}, At_{n+1})(t_{n+1}, At_{n+1})}{(At_{n+1}, At_{n+1})(s_{n+1} - t_{n+1}, s_{n+1} - t_{n+1}) - (s_{n+1} - t_{n+1}, At_{n+1})^2}, \\
x_{n+1} &= x_n + \alpha_n p_n + \zeta_n t_{n+1} + \eta_n (x_n - x_{n-1} + \alpha_n p_n - \alpha_{n-1} p_{n-1} - \alpha_n w_n), \\
r_{n+1} &= t_{n+1} + \eta_n (t_{n+1} - s_{n+1}) - \zeta_n A t_{n+1}, \\
\beta_n &= \frac{\alpha_n}{\zeta_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)}, \\
w_{n+1} &= t_{n+1} + \beta_n p_n, \\
p_{n+1} &= r_{n+1} + \beta_n (p_n - \zeta_n A p_n + \eta_n (p_n - w_n));
\end{aligned}
$$

5.4 The chioce for RGPBi-CG

In the recurrence formulas (4.11)~(4.15), we select the parameters μ_n and ν_{n-1} so that the residual norm $\| r_{n+1} \|$ with respect to μ_n and ν_{n-1} reaches its minimum value at nth iteration.

Denote $r_n - \alpha_n A p_n$ by w_n, and $s_n - \alpha_n A(s_n + \beta_{n-1} u_{n-1})$ by q_n. Then, rearranging the recurrence formulas (4.11)~(4.15), we obtain the reduced variant of GPBi-CG, and name it RGPBi-CG.

ALGORITHM 7 RGPBi-CG

Choose an initial guess x_0, $r_0 = b - Ax_0$,

and set $r_0^* = u_0 = p_0 = r_0$, $\alpha_0 = \dfrac{(r_0^*, r_0)}{(r_0^*, Ap_0)}$,

$s_1 = w_0 = r_0 - \alpha_0 A p_0$, $\mu_0 = \dfrac{(s_1, As_1)}{(As_1, As_1)}$.

For $n = 0, 1, \cdots$ until $\| r_n \| \leq \varepsilon \| b \|$ do:

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

$$r_{n+1} = w_n - \mu_n A s_{n+1},$$

$$\beta_n = \frac{\alpha_n}{\mu_n} \cdot \frac{(r_0^*, r_{n+1})}{(r_0^*, r_n)},$$

$$p_{n+1} = r_{n+1} + \beta_n (p_n - \mu_n A u_n),$$

$$\alpha_{n+1} = \frac{(r_0^*, r_{n+1})}{(r_0^*, Ap_{n+1})},$$

$$w_{n+1} = r_{n+1} - \alpha_{n+1} A p_{n+1},$$

$$q_{n+1} = s_{n+1} - \alpha_{n+1} A(s_{n+1} + \beta_n u_n),$$

$$\mu_n = \frac{(Aq_{n+1}, Aq_{n+1})(w_{n+1}, Aw_{n+1}) - (Aw_{n+1}, Aq_{n+1})(w_{n+1}, Aq_{n+1})}{(Aw_{n+1}, Aw_{n+1})(Aq_{n+1}, Aq_{n+1}) - (Aw_{n+1}, Aq_{n+1})^2},$$

$$\nu_n = \frac{(Aw_{n+1}, Aw_{n+1})(w_{n+1}, Aq_{n+1}) - (Aw_{n+1}, Aq_{n+1})(w_{n+1}, Aw_{n+1})}{(Aq_{n+1}, Aq_{n+1})(w_{n+1}, Aw_{n+1}) - (Aw_{n+1}, Aq_{n+1})(w_{n+1}, Aq_{n+1})},$$

$$s_{n+2} = w_{n+1} + \nu_n q_{n+1},$$

$$u_{n+1} = p_{n+1} + \nu_n (s_{n+1} + \beta_n u_n);$$

5.5 Hybrid variants

Another possibility of implementations is to combine one variant of the product-type methods with other one of the product-type methods without restarting. For example, during the first half of algorithm, we use Bi-CGSTAB, and during the latter half, we use CGS.

Acturally, Bi-CGSTAB2 can be explained as a hybrid variant of Bi-CGSTAB and GPBi-CG. In other words, for choice of the second-kind parameters, the minimization of residual norm is defined: as function of two variables ζ and η at even iterations; and as function of one variable ζ under the assumption $\eta_n = 0$ at odd iterations.

If $n = 2k$, then let $\tilde{\chi}_k = \zeta_n$, else let $\tilde{\zeta}_k = 1 + \eta_n$ and $\tilde{\eta}_k = -\zeta_n$ in the GPBi-CG algorithm, Bi-CGSTAB2 reads as follows[4]:

ALGORITHM 8 Bi-CGSTAB2

Choose an initial guess x_0^{ST2},
and set $r_0^{\mathrm{ST2}*} = p_0^{\mathrm{ST2}} = r_0^{\mathrm{ST2}} = b - Ax_0^{\mathrm{ST2}}$,
for $n = 0, 1, \cdots$, until $\| r_n^{\mathrm{ST2}} \| \leq \varepsilon \| b \|$ do:
when $n = 2k$,

$$\alpha_n = \frac{(r_0^{\mathrm{ST2}*}, r_n^{\mathrm{ST2}})}{(r_0^{\mathrm{ST2}*}, Ap_n^{\mathrm{ST2}})},$$

$$t_{n+1}^{\mathrm{ST2}} = r_n^{\mathrm{ST2}} - \alpha_n Ap_n^{\mathrm{ST2}},$$

$$\tilde{\chi}_k = \frac{(t_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})}{(At_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})},$$

$$x_{n+1}^{\mathrm{ST2}} = x_n^{\mathrm{ST2}} + \alpha_n p_n^{\mathrm{ST2}} + \tilde{\chi}_k t_{n+1}^{\mathrm{ST2}},$$

$$r_{n+1}^{\mathrm{ST2}} = t_{n+1}^{\mathrm{ST2}} - \tilde{\chi}_k At_{n+1}^{\mathrm{ST2}},$$

$$\beta_n = \frac{\alpha_n}{\tilde{\chi}_k} \cdot \frac{(r_0^{\mathrm{ST2}*}, r_{n+1}^{\mathrm{ST2}})}{(r_0^{\mathrm{ST2}*}, r_n^{\mathrm{ST2}})},$$

$$w_{n+1}^{\mathrm{ST2}} = t_{n+1}^{\mathrm{ST2}} + \beta_n p_n^{\mathrm{ST2}},$$

$$p_{n+1}^{\mathrm{ST2}} = r_{n+1}^{\mathrm{ST2}} + \beta_n p_n^{\mathrm{ST2}} - \beta_n \tilde{\chi}_k Ap_n^{\mathrm{ST2}},$$

when $n = 2k + 1$,

$$\alpha_n = \frac{(r_0^{\mathrm{ST2}*}, r_{n+1}^{\mathrm{ST2}})}{(r_0^{\mathrm{ST2}}, Ap_n^{\mathrm{ST2}})},$$

$$s_{n+1}^{\mathrm{ST2}} = t_n^{\mathrm{ST2}} - \alpha_n Aw_n^{\mathrm{ST2}},$$

$$t_{n+1}^{\mathrm{ST2}} = r_n^{\mathrm{ST2}} - \alpha_n Ap_n^{\mathrm{ST2}},$$

$$\tilde{\zeta}_k = \frac{(t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})(s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}}) - (At_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})(s_{n+1}^{\mathrm{ST2}}, t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}})}{(t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}})(At_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}}) - (t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})^2},$$

$$\tilde{\eta}_k = \frac{(t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})(s_{n+1}^{\mathrm{ST2}}, t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}) - (t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}})(s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})}{(t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}})(At_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}}) - (t_{n+1}^{\mathrm{ST2}} - s_{n+1}^{\mathrm{ST2}}, At_{n+1}^{\mathrm{ST2}})^2},$$

$$x_{n+1}^{\mathrm{ST2}} = (1 - \tilde{\zeta}_k)(x_{n-1}^{\mathrm{ST2}} + \alpha_{n-1} p_{n-1}^{\mathrm{ST2}} + \alpha_n w_n^{\mathrm{ST2}}) + \tilde{\zeta}_k (x_n^{\mathrm{ST2}} + \alpha_n p_n^{\mathrm{ST2}}) - \tilde{\eta}_k t_{n+1}^{\mathrm{ST2}},$$

$$r_{n+1}^{\mathrm{ST2}} = (1 - \tilde{\zeta}_k) s_{n+1}^{\mathrm{ST2}} + \tilde{\zeta}_k t_{n+1}^{\mathrm{ST2}} + \tilde{\eta}_k At_{n+1}^{\mathrm{ST2}},$$

$$\beta_n = -\frac{\alpha_n}{\tilde{\eta}_k} \cdot \frac{(r_0^{\mathrm{ST2}*}, r_{n+1}^{\mathrm{ST2}})}{(r_0^{\mathrm{ST2}*}, r_n^{\mathrm{ST2}})},$$

$$p_{n+1}^{\mathrm{ST2}} = r_{n+1}^{\mathrm{ST2}} + \beta_n (1 - \tilde{\zeta}_k) w_n^{\mathrm{ST2}} + \beta_n (\tilde{\zeta}_k p_n^{\mathrm{ST2}} + \tilde{\eta}_k Ap_n^{\mathrm{ST2}});$$

6 Summary

In view of more stable convergence behaviour and little work and low storage, we emphasize that the polynomial of form Π_n generated by the three-term recurrence relation (3.4) is better than the others which come from such truncated iterative method as GMRES(k)($k > 2$)[12, 4] for the requirement (1) described at the end of §2,

For solving realistic problems, any variant will be hardly competitive without preconditioning

techniques, however, we did not give any detailed description of preconditioning techniques in this paper. All variants can be combined with the efficient preconditioning techniques, such as imcomplete LU factorizations. For more detailed discussion on such studies see, e.g., [9, 10, 15].

References

[1] M. Driessen and H. A. Van der Vorst, Bi-CGSTAB *in semiconductor modelling*, in: W. Fichtner(ed.), Simulation of semiconductor devices and processes, 4(1991), pp. 45-54.

[2] S. C. Eisenstat, H. C. Elman and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20(1983), pp. 345-357.

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

[4] M. H. Gutknecht, *Variants of* Bi-CGSTAB *for matrices with complex spectrum*, Interdisciplinary Project Center for Supercomputing(IPS) Research Report, 91-14.

[5] S. Fujino, N. Matsumoto and H. Suitou, *Application of* Bi-CGSTAB *method to fluid flow problems*, in: Proceedings of 5th Symposium on Fluid, Japan(1991), pp. 501-504(in Japanese).

[6] S. Fujino and S. L. Zhang, *Analysis on convergence behavior of the* CGS *and* Bi-CGSTAB *method*, To appear on International Association for Mathematics and Computers in Simulation (IMACS).

[7] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49(1952), pp. 409-435.

[8] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49(1952), pp. 33-53.

[9] 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(1981), pp. 148-162.

[10] J. A. Meijerink and H. A. Van der Vorst, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comput. Phys., 44(1981), pp. 134-155.

[11] N. M. Nachtigal, S. C. Reddy and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, Technical Report 90-2, MIT, 1990.

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

[13] P. Sonneveld, CGS, *A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10(1989), pp. 26-52.

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

[15] H. A. Van der Vorst, *Preconditioning by incomplete decompositions*, Ph.D. Thesis, University of Utrecht, The Netherlands, 16 December 1982.

[16] 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.

[17] S. L. Zhang and S. Fujino, *The residual polynomials of* CGS *and* Bi-CGSTAB (in preparation).