Varieties of Product-type Method for Solving Large Non-Hermitian Linear Systems

Bo ZHENG[*,†] and Shao-Liang ZHANG[††]

† Department of Applied Physics, The University of Tokyo
†† Department of Computational Science and Engineering, Nagoya University

Abstract *For solving non-Hermitian linear systems, a famous method is Bi-Conjugate Gradient method (Bi-CG), but its performance is not usually satisfactory because of the unstable convergent behavior and the expensive computational cost. Recently a family of methods named Product-type methods were developed to enhance the Bi-CG method by redefining the residual vector as* $\boldsymbol{r}_n := H_n(A)\boldsymbol{r}_n^{\mathrm{B}}$*, where* $\boldsymbol{r}_n^{\mathrm{B}}$ *is the residual vector of the Bi-CG method and* $H_n(A)$ *is an* n*-degree polynomial named accelerating polynomial. In this research, we present the computational varieties based on the deriving processes of the Product-type methods, and discuss the construction of various algorithms depending on these varieties. Finally, among these algorithms we try to find some variations that are expected to be more numerically stable and have higher convergent speed than the present Product-type methods.*

Key words Bi-CG, Product-type methods

1. Introduction

Nowadays scientific computation become to play an important role in various fields. It is not an exaggeration to say that numerical computations with high speed and high accuracy are occupying the most important part in the work that computers bear. Following the high-speed development of the computer science and technology, some nature phenomena or engineering phenomena can be forecasted in high-speed and high accuracy through computer simulations. These numerical simulations are often done through the formulation by the partial differential equations (PDEs). Many discretizations, such as the finite differential method and the finite element method, are applied to the boundary value problems of PDEs before calculating an approximate solution. As results of the discretizations, usually we have to deal with the large, sparse and non-Hermitian linear systems.

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

where nonsingular coefficient matrix $A \in \mathcal{C}^{N\times N}$ and right hand vector $\boldsymbol{b} \in \mathcal{C}^N$ are given, $\boldsymbol{x} \in \mathcal{C}^N$ is the unknown solution vector. Since A arises from the discretization or finite element approximations with large-scale, sparse and non-Hermitian nonzero entries, for solving these systems it might be followed by vast computational cost and the inaccuracy arising from rounding errors.

For solving non-Hermitian linear systems (1), a famous method is Bi-Conjugate Gradient method (Bi-CG) [1], but it performs not usually satisfactory because of its unstable convergent behavior and expensive computational cost. To improve the Bi-CG method, a family of methods named Product-type methods were developed , where many efforts have been devoted to deriving more efficient methods from restructuring Bi-CG to avoid calculating the transpose matrix-vector multiplications and to accelerate the convergence rate in Bi-CG. A common technique is to redefine the residual as $\boldsymbol{r}_n := H_n(A)\boldsymbol{r}_n^{\mathrm{B}}$: the product between the residual vector of Bi-CG and an undetermined polynomial of degree n that will be referred to hereafter as the accelerating polynomial. CGS [2], Bi-CGSTAB [3] and GPBi-CG [4] were derived from Bi-CG by this common technique. The CGS method by Sonneveld has been proposed earlier, which computes the square of the Bi-CG polynomials without requiring A^{H}. In Bi-CGSTAB, Van der Vorst defined the acceleration polynomial by using two-term recurrence relations to design the residual polynomial by using two-term recurrence relations to design the residual polynomial of Bi-CGSTAB. In GPBi-CG, Zhang proposed a different generalization of Bi-CGSTAB that uses a three-term recurrent poly-

*: The writer has graduated from the University of Tokyo.
Present address: NO.1105, Yanaka 2-5-3, Adachi-ku, Tokyo

nomial at all iterations. He constructed the polynomial from a pair of coupled two-term recurrent polynomials. Last year, in [5] Zheng pointed out that there may exist more variations of Product-type methods by using some symbols summarized from [6] and redefining the vector table in [7].

In this research, we focus on indicating the computational varieties on the deriving processes of the Product-type methods, and discuss the construction of various algorithms depending on these varieties. Finally, among these algorithms we try to find some variations expected to be more numerically stable and have higher convergent speed than the present Product-type methods.

2. Product-type Methods

Before we give the definition of Product-type Methods, let us take a glance at the Bi-CG method which is given as Algorithm 1.

Algorithm 1: Bi-CG method

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0^{\mathrm{B}} = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$, $\boldsymbol{r}_0^*$ is an arbitrary vector, such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0^{\mathrm{B}}) \neq 0$, *e.g.*, $\boldsymbol{r}_0^* = \boldsymbol{r}_0^{\mathrm{B}}$,
for $n = 0, 1, \cdots$, until $\| \boldsymbol{r}_n^{\mathrm{B}} \| \leqq \epsilon \| \boldsymbol{r}_0^{\mathrm{B}} \|$ do:

begin

$$\boldsymbol{p}_n^{\mathrm{B}} = \boldsymbol{r}_n^{\mathrm{B}} + \beta_{n-1}\boldsymbol{p}_{n-1}^{\mathrm{B}}, \tag{2}$$

$$\boldsymbol{p}_n^* = \boldsymbol{r}_n^* + \bar{\beta}_{n-1}\boldsymbol{p}_{n-1}^*,$$

$$\alpha_n = \frac{(\boldsymbol{r}_n^*, \boldsymbol{r}_n^{\mathrm{B}})}{(\boldsymbol{p}_n^*, A\boldsymbol{p}_n^{\mathrm{B}})},$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n^{\mathrm{B}},$$

$$\boldsymbol{r}_{n+1}^{\mathrm{B}} = \boldsymbol{r}_n^{\mathrm{B}} - \alpha_n A\boldsymbol{p}_n^{\mathrm{B}}, \tag{3}$$

$$\boldsymbol{r}_{n+1}^* = \boldsymbol{r}_n^* - \bar{\alpha}_n A^{\mathrm{H}}\boldsymbol{p}_n^*,$$

$$\beta_n = \frac{(\boldsymbol{r}_{n+1}^*, \boldsymbol{r}_{n+1}^{\mathrm{B}})}{(\boldsymbol{r}_n^*, \boldsymbol{r}_n^{\mathrm{B}})},$$

end

In the Bi-CG algorithm, a Krylov subspace [8]

$$K_n(A^{\mathrm{H}}; \boldsymbol{r}_0^*) = \mathrm{Span}\{\boldsymbol{r}_0^*, (A^{\mathrm{H}})\boldsymbol{r}_0^*, (A^{\mathrm{H}})^2\boldsymbol{r}_0^*, \cdots, (A^{\mathrm{H}})^{n-1}\boldsymbol{r}_0^*\},$$

is generated, and the approximate solution $\boldsymbol{x}_n$ has an associated residual $\boldsymbol{r}_n^{\mathrm{B}} (:= \boldsymbol{b} - A\boldsymbol{x}_n)$ which satisfy the orthogonality $\boldsymbol{r}_n^{\mathrm{B}} \perp K_n(A^{\mathrm{H}}; \boldsymbol{r}_0^*)$ [1]. And combining with the Lanczos process [9], the $n+1$th residual vector $\boldsymbol{r}_{n+1}^{\mathrm{B}}$ can be worked out by (3) and the auxiliary vector $\boldsymbol{p}_n^{\mathrm{B}}$ named direction vector can be worked out by (2).

Athough Bi-CG theoretically constructed a effective loop which would be helpful to make the residual convergent to zero, numerically, it performs not so well because of its unstable convergent behavior and expensive computational cost. To enhance the Bi-CG method, the basic idea of Product-type methods is by redefining the residual vector as

$$\boldsymbol{r}_n := H_n(A)\boldsymbol{r}_n^{\mathrm{B}}, \tag{4}$$

where $H_n(A)$ is an n-degree polynomial which contribute to accelerating the convergence of $\boldsymbol{r}_n^{\mathrm{B}}$, and it was designed as follows: [4]

$$H_0(\lambda) = 1, G_0(\lambda) = \zeta_0,$$
$$H_n(\lambda) = H_{n-1}(\lambda) - \lambda G_{n-1}(\lambda), \tag{5}$$
$$G_n(\lambda) = \zeta_n H_n(\lambda) + \eta_n G_{n-1}(\lambda). \tag{6}$$

The well-known Product-type methods such as CGS, Bi-CGSTAB and GPBi-CG can be derived from the computations when they choose the different parameters ζ_n and η_n in (6). According to these different choices of parameters ζ and η, they perform very differently on convergence behaviors. In fact, CGS usually give an unexpected performance, because it magnifies local error peaks in the convergence history of Bi-CG so that cannot to be convergent. Although the convergence of Bi-CGSTAB is not so slow as Bi-CG, it does not perform as well as GPBi-CG in most cases. Let us show this in a common example. We take a Toeplitz matrix A of order 200, and take the right-hand side $\boldsymbol{b}$ that the all elements of $\boldsymbol{b}$ are 1. In Figure 1 we plot the residuals for Bi-CG, CGS, Bi-CGSTAB and GPBi-CG.

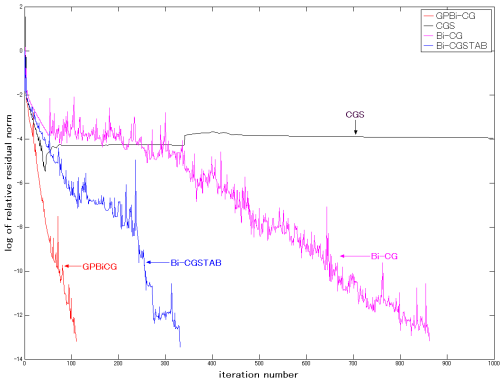

Figure 1: Result of Toeplitz matrix with $\gamma = 1.5$

GPBi-CG is looked as the robustest one in many cases, however, in fact sometimes GPBi-CG even may lead to rather irregular convergence behaviors. The reason can be assumed as the recurrences employed in the algorithms might be not numerically stable enough so that the iterations are lead to a slow convergence effected by the rounding errors. Therefore we put our efforts to find more stable recurrences in this research.

3. Varieties of Product-type method

For simplifying the complexity on expression, first we summarize the symbols used in [6] as follows:

$$\begin{aligned} \boldsymbol{r}_n^{H_m} &:= H_m(A)\boldsymbol{r}_n^{\mathrm{B}},\ \boldsymbol{r}_n^{G_m} := G_m(A)\boldsymbol{r}_n^{\mathrm{B}}, \\ \boldsymbol{p}_n^{H_m} &:= H_m(A)\boldsymbol{p}_n^{\mathrm{B}},\ \boldsymbol{p}_n^{G_m} := G_m(A)\boldsymbol{p}_n^{\mathrm{B}}, \end{aligned} \tag{7}$$

where $m, n = 0, 1, \cdots$. Each of them represents a vector that arise from the product of the polynomial $H_m(A)$ or $G_m(A)$ and the residual vector $\boldsymbol{r}_n^{\mathrm{B}}$ or the direction vector $\boldsymbol{p}_n^{\mathrm{B}}$. And according to (4), $\boldsymbol{r}_n^{H_n}$ represents the residual.

In fact, to derive the algorithm of Product-type method we usually concern the relation between nth residual $\boldsymbol{r}_n^{H_n}$ and $(n+1)$th residual $\boldsymbol{r}_{n+1}^{H_{n+1}}$. To clear this relation, we substitute the recurrences (5)-(6) and (3)-(2) for (4) alternately, and list all the two-term recurrence as follows:

$$\boldsymbol{r}_{n+1}^{H_{n+1}} = \boldsymbol{r}_{n+1}^{H_n} - A\boldsymbol{r}_{n+1}^{G_n}, \tag{8}$$
$$\boldsymbol{r}_{n+1}^{H_{n+1}} = \boldsymbol{r}_n^{H_{n+1}} - \alpha_n A\boldsymbol{p}_n^{H_{n+1}}, \tag{9}$$
$$\boldsymbol{r}_{n+1}^{G_n} = \zeta_n\boldsymbol{r}_{n+1}^{H_n} + \eta_n\boldsymbol{r}_{n+1}^{G_{n-1}}, \tag{10}$$
$$\boldsymbol{r}_{n+1}^{G_n} = \boldsymbol{r}_n^{G_n} - \alpha_n A\boldsymbol{p}_n^{G_n}, \tag{11}$$
$$\boldsymbol{p}_n^{H_{n+1}} = \boldsymbol{p}_n^{H_n} - A\boldsymbol{p}_n^{G_n}, \tag{12}$$
$$\boldsymbol{p}_n^{H_{n+1}} = \boldsymbol{r}_n^{H_{n+1}} + \beta_{n-1}\boldsymbol{p}_{n-1}^{H_{n+1}}, \tag{13}$$
$$\boldsymbol{p}_n^{G_n} = \zeta_n\boldsymbol{p}_n^{H_n} + \eta_n\boldsymbol{p}_n^{G_{n-1}}, \tag{14}$$
$$\boldsymbol{p}_n^{G_n} = \boldsymbol{r}_n^{G_n} + \beta_{n-1}\boldsymbol{p}_{n-1}^{G_n}, \tag{15}$$
$$\boldsymbol{r}_{n+1}^{H_n} = \boldsymbol{r}_n^{H_n} - \alpha_n A\boldsymbol{p}_n^{H_n}, \tag{16}$$
$$\boldsymbol{r}_{n+1}^{G_{n-1}} = \boldsymbol{r}_n^{G_{n-1}} - \alpha_n A\boldsymbol{p}_n^{G_{n-1}}, \tag{17}$$
$$\boldsymbol{p}_n^{H_n} = \boldsymbol{r}_n^{H_n} + \beta_{n-1}\boldsymbol{p}_{n-1}^{H_n}, \tag{18}$$
$$\boldsymbol{p}_n^{G_{n-1}} = \boldsymbol{r}_n^{G_{n-1}} + \beta_{n-1}\boldsymbol{p}_{n-1}^{G_{n-1}}, \tag{19}$$
$$\boldsymbol{r}_n^{H_{n+1}} = \boldsymbol{r}_n^{H_n} - A\boldsymbol{r}_n^{G_n}, \tag{20}$$
$$\boldsymbol{r}_n^{G_n} = \zeta_n\boldsymbol{r}_n^{H_n} + \eta_n\boldsymbol{r}_n^{G_{n-1}}, \tag{21}$$
$$\boldsymbol{p}_{n-1}^{H_{n+1}} = \boldsymbol{p}_{n-1}^{H_n} - A\boldsymbol{p}_{n-1}^{G_n}, \tag{22}$$
$$\boldsymbol{p}_{n-1}^{G_n} = \zeta_n\boldsymbol{p}_{n-1}^{H_n} + \eta_n\boldsymbol{p}_{n-1}^{G_{n-1}}, \tag{23}$$

And all the vectors in these recurrent relations can be described with one table (Figure 2). In this table we arrange 16 vectors into a 2×2-block table, and each block has 4 vectors. Let us assume that let the upper-left block be a known block, and let the lower-right block be a unknown block. Then how to compute the 4 vectors from the known block become concerned. According to the formula list (8)-(23), for computing $\boldsymbol{r}_{n+1}^{H_{n+1}}$, there are two ways can be obtained, (8) or (9). In the case we use the former, the computing route in the vector table (Figure 2) must cross through the lower-left block, since in (8) $\boldsymbol{r}_{n+1}^{H_n}$ is needed and $\boldsymbol{r}_{n+1}^{H_n}$ is in the lower-left block. On the other hand, if we use (9), the route cross through the upper-right block. The same as that, we can also observe that for computing the other vectors $\boldsymbol{p}_n^{G_n}$, $\boldsymbol{p}_n^{H_{n+1}}$ and $\boldsymbol{r}_{n+1}^{G_n}$ there are two ways respectively. Thus, if we denote the route through the upper-right block with $\uparrow$ and the route through the lower-left block with $\leftarrow$, there exist 16 kinds of routes from the known block to the unknown block, since there are 4 vectors in the unknown block and each vector has two kinds: $\uparrow$ and $\leftarrow$. It also means that there are 2^4 ways for computing unknown block from the known block. We denote the 16 routes in the order of $(\boldsymbol{p}_n^{G_n},\ \boldsymbol{p}_n^{H_{n+1}},\ \boldsymbol{r}_{n+1}^{G_n},\ \boldsymbol{r}_{n+1}^{H_{n+1}})$, and then the notations are $(\leftarrow, \leftarrow, \leftarrow, \leftarrow)$, $\cdots$, $(\leftarrow, \uparrow, \leftarrow, \uparrow)$, $\cdots$, $(\uparrow, \uparrow, \uparrow, \uparrow)$. Namely we can obtain 16 sets of vector recurrences, which indicate the variety based on the computations of vectors.

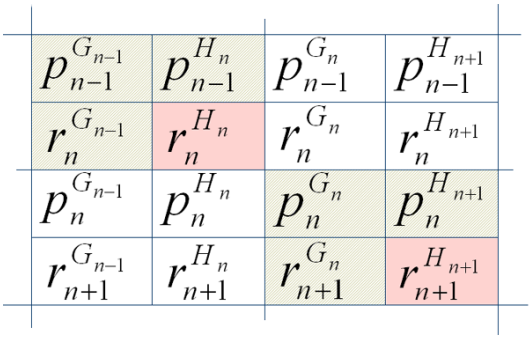

Figure 2: Partial vector table for showing the relations between $\boldsymbol{r}_n^{H_n}$ and $\boldsymbol{r}_{n+1}^{H_{n+1}}$

4. Implementation Details

There also are some varieties on the computations of parameter α, β, ζ and η discussed in my master thesis. And we can combine the varieties to construct many variations of Product-type methods. Among these variations, we propose a method named GPBi-CG_V which is combined as follows:

- routes $(\leftarrow, \leftarrow, \leftarrow, \leftarrow)$;
- computation of α_n and β_{n-1}:

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

- the function for computing ζ_n and η_n:

$$\begin{aligned} &[\zeta_n,\ \eta_n] \overset{\text{return}}{\longleftarrow} f_{\zeta,\eta}(\boldsymbol{r}_{n+1}^{H_n},\ A\boldsymbol{r}_{n+1}^{H_n},\ A\boldsymbol{r}_{n+1}^{G_{n-1}}) \\ &:= \arg\min_{\zeta_n,\eta_n \in \mathcal{C}} \| \boldsymbol{r}_{n+1}^{H_n} - \zeta_n A\boldsymbol{r}_{n+1}^{H_n} - \eta_n A\boldsymbol{r}_{n+1}^{G_{n-1}} \|_2 \end{aligned}$$

And then, we use some auxiliary vectors to substitute for the algorithm for reducing the computational cost. We named it GPBi-CG_V, as a variant of GPBi-CG.

Algorithm 2: GPBi-CG_V

$\boldsymbol{x}_0$ is an initial guess, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $\beta_{-1} = 0$, $\boldsymbol{r}_0^{H_{-1}} = A\boldsymbol{p}_0^{H_{-1}} = 0$. $\boldsymbol{r}_0^*$ is an arbitrary vector, such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0^{H_0}) \neq$

0, e.g., set $\boldsymbol{r}_0^* = \boldsymbol{r}_0^{H_0}$
for $n = 0, 1, \cdots$, until $\| \boldsymbol{r}_n \| \leqq \epsilon \| \boldsymbol{r}_0 \|$ do:
begin

$$\boldsymbol{p}_n^{H_n} = \boldsymbol{r}_n^{H_n} + \beta_{n-1}\boldsymbol{p}_{n-1}^{H_n},$$
$$\alpha_n = (\boldsymbol{r}_0^*, \boldsymbol{r}_n^{H_n})/(\boldsymbol{r}_0^*, A\boldsymbol{p}_n^{H_n}),$$
$$\boldsymbol{r}_{n+1}^{H_n} = \boldsymbol{r}_n^{H_n} - \alpha_n A\boldsymbol{p}_n^{H_n},$$
$$\boldsymbol{r}_{n+1}^{H_{n-1}} = \boldsymbol{r}_n^{H_{n-1}} - \alpha_n(A\boldsymbol{r}_n^{H_{n-1}} + \beta_{n-1}A\boldsymbol{p}_{n-1}^{H_{n-1}}),$$
$$A\_\boldsymbol{p}_n^{G_{n-1}} = A\_\boldsymbol{r}_n^{G_{n-1}} + \beta_{n-1}A\_\boldsymbol{p}_{n-1}^{G_{n-1}},$$
$$A\_\boldsymbol{r}_{n+1}^{G_{n-1}} = \boldsymbol{r}_{n+1}^{H_{n-1}} - \boldsymbol{r}_{n+1}^{H_n},$$
$$[\zeta_n, \eta_n] = f_{\zeta,\eta}(\boldsymbol{r}_{n+1}^{H_n},\ A\boldsymbol{r}_{n+1}^{H_n},\ A\_\boldsymbol{r}_{n+1}^{G_{n-1}}),$$
$$A\_\boldsymbol{p}_n^{G_n} = \zeta_n A\boldsymbol{p}_n^{H_n} + \eta_n A\_\boldsymbol{p}_n^{G_{n-1}},$$
$$A\_\boldsymbol{r}_{n+1}^{G_n} = \zeta_n A\boldsymbol{r}_{n+1}^{H_n} + \eta_n A\_\boldsymbol{r}_{n+1}^{G_{n-1}},$$
$$\boldsymbol{p}_n^{H_{n+1}} = \boldsymbol{p}_n^{H_n} - A\_\boldsymbol{p}_n^{G_n},$$
$$\boldsymbol{r}_{n+1}^{H_{n+1}} = \boldsymbol{r}_{n+1}^{H_n} - A\_\boldsymbol{r}_{n+1}^{G_n},$$
$$\boldsymbol{r}_{n+1}^{G_n} = \zeta_n \boldsymbol{r}_{n+1}^{H_n} + \eta_n(\boldsymbol{r}_n^{G_{n-1}} - \alpha_n A\_\boldsymbol{p}_n^{G_{n-1}}),$$
$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n^{H_n} + \boldsymbol{r}_{n+1}^{G_n},$$
$$\beta_n = (\alpha_n/\zeta_n)(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1}^{H_{n+1}})/(\boldsymbol{r}_0^*, \boldsymbol{r}_n^{H_n}),$$

end

Let us give a brief comparison on computational cost showed in Table 1 between GPBi-CG_V and GPBi-CG.

Table 1: Summary of Operations in per Iteration.

Method	Inner Product	AXPY	MV
GPBi-CG	7	15	2
GPBi-CG_V	7	14	2

where AXPY is short for operations + vector scaling and MV means Matrix-vector product. Thus, we can observe that the proposed algorithm have the almost same cost of GPBi-CG.

5. Numerical Experiments

We present the results of numerical experiments on the range of matrices collections in various fields provided by Matrix Market. We take right-hand vector that all entries are 1 in each linear systems. Also we report the numbers of iterations and $\log_{10}$ of true relative residual 2-norm in Table 2. The stopping criterion used was $\| \boldsymbol{r}_n^{H_n} \| / \| \boldsymbol{b} \| \leqq 10^{-12}$

Table 2. Test problems (N: order of matrix, NNZ: nonzeros in matrix) and the convergence results on number of iterations and true relative residual(TRR). Bold font characters mean the relatively better results compared with GPBi-CG

Matrix	N	NNZ	Iteration Number		TRR	
			GPBiCG_V	GPBiCG	GPBiCG_V	GPBiCG
add20	2395	13151	540	580	-11.83	-11.69
bwm200	200	796	134	163	-10.60	-10.67
cavity05	1182	32632	**758**	905	-9.41	-10.25
cavity10	2597	76171	1424	1291	**-8.86**	-7.75
cavity16	4562	137887	**2237**	2970	-8.54	-8.37
cavity17	4562	131735	3548	3641	**-8.12**	-7.11
cdde3	961	4681	195	217	-11.99	-12.06
e05r0000	236	5846	435	465	-11.58	-11.52
rdb1250l	1250	7300	**204**	723	-11.70	-11.79
rdb800l	800	4640	**162**	334	-11.98	-11.99
sherman5	3312	20793	2377	2497	**-9.71**	-10.01
watt_1	1856	11360	**383**	977	-4.60	-4.61

6. Conlusions

In this research we gave a new point of view around the varieties on the Product-type methods and reanalyzed their deriving processes, through which we constructed many variations. And among these variations we proposed a method named GPBi-CG_V. By our numerical experiments GPBi-CG_V looked having good potentials to solve the non-Hermitian systems. With a more stable and more efficient convergent behavior, it is expected to substitute for the present methods in the world.

References

[1] R. Fletcher. Conjugate Gradient Methods for Indefinite Systems. *Lecture Notes in Mathematics*, 506:73–89, 1976.

[2] P. Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

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

[4] S.-L. Zhang. GPBi-CG: Generalized Product-type Methods Based on Bi-CG for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Comput.*, 18:537–551, 1997.

[5] B. Zheng and S.-L. Zhang. Diversities of Product-type Methods for Solving Linear Systems. *The 2004 Annual Conference of JSIAM*, pages 248–249, 2004.

[6] S.-L. Zhang. A class of Product-type Kryolv-subspace methods for solving non-symmetric linear systems. *J. Comput.Appl. Math.*, 149:297–305, 2002.

[7] S.Röllin, M.H. Gutknecht. Variations of Zhang's Lanczos-type product method. *Appl. Numer. Math.*, 41:119–133, 2002.

[8] F. Chatelin. *Valeurs Propres de Matrices*, 1988.

[9] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards*, 49:33–53, 1952.