On Convergence Speed of Parallel Variants of GPBiCG Method for Solving Linear Equations

Kuniyoshi Abe and Soichiro Ikuno
Faculty of Economics and Information, Gifu Shotoku University, Gifu 5008288, Japan
School of Computer Science, Tokyo University of Technology, Tokyo 1920982, Japan
E-mail: abe@gifu.shotoku.ac.jp, ikuno@cs.teu.ac.jp

Abstract. The hybrid Bi-Conjugate Gradient (Bi-CG) methods such as Bi-CG stabilized (Bi-CGSTAB), Generalized Product-type based Bi-CG (GPBiCG), and BiCGstab($\ell$) are well-known for efficiently solving linear equations. GPBiCG and BiCGstab($\ell$) are more effective and robust than Bi-CGSTAB on problems with strongly nonsymmetric matrices. On present petascale high-performance computing hardware, the scalability of Krylov subspace methods has recently become increasingly prominent. The main bottleneck for efficient parallelization is the inner products which require a global reduction. The parallel variants of Bi-CGSTAB reducing the number of global communication phases and hiding the communication latency have been proposed. However, it has been reported that the convergence of the parallel variants of Bi-CGSTAB is affected by rounding errors than that of the standard Bi-CGSTAB, and is not as robust as the standard. In this paper, therefore, following [1], we design parallel variants of GPBiCG, which converges faster and is more robust than Bi-CGSTAB. Then we compare the convergence speed between the standard GPBiCG and the parallel variants by numerical experiments.

1. Introduction

The Bi-Conjugate Gradient (Bi-CG) method [2] is well-known for solving linear equations

$$Ax = b,$$

for $x$, where $A$ is a given n-by-n matrix, and $b$ is a given n-vector. A number of hybrid Bi-CG methods such as the Conjugate Gradient Squared (CGS) [3], Bi-CG STABilized (Bi-CGSTAB) [4], BiCGstab($\ell$) [5], BiCG$\times$MR2 [6] and Generalized Product-type Bi-CG (GPBiCG) [7] methods have been developed to improve the convergence of Bi-CG. The convergence of GPBiCG/BiCG$\times$MR2 and BiCGstab($\ell$) is faster and more robust than that of Bi-CGSTAB. Four variants of GPBiCG/BiCG$\times$MR2 [8], which are more robust than the original GPBiCG, have been proposed. The two variants of them are derived by combining the recurrence of Bi-CG with the three term recurrence, and the others are done by combining the recurrence of Bi-CG with the coupled two-term recurrences suggested by Rutishauser.

On present petascale high-performance computing hardware, the scalability of Krylov subspace methods on parallel computers has recently become increasingly prominent. The main bottleneck for efficient parallelization is the inner products which require a global reduction, i.e., an accumulation of data from all to one processor. The resulting global synchronization phases cause communication overhead on parallel computers. The reduction of the synchronization...
Standard GPBiCG algorithm

The standard GPBiCG algorithm [7] is described as follows:

2. Parallel variants of GPBiCG

In this paper, therefore, following [1], we design parallel variants of GPBiCG, which are referred to as pipelined GPBiCG. We examine the convergence behavior among the standard GPBiCG, the variants of GPBiCG and their pipelined GPBiCG by numerical experiments. The numerical experiments show that the pipelined variant of GPBiCG using the coupled two-term recurrence is less affected by rounding errors than the other pipelined GPBiCG, and that the pipelined algorithms of the standard one and the variant using the three term recurrence are more affected by rounding errors than the original counterparts.

2. Parallel variants of GPBiCG

The standard GPBiCG algorithm [7] is described as follows:

Standard GPBiCG algorithm

Select $x_0$ and $r_0$. Compute $r_0 = b - Ax_0$, and $p_0 = r_0$. Set $\beta_{-1} = 0$, $t_{-1} = w_{-1} = 0$. While $\|r_k\|_2/\|r_0\|_2 > \text{tol}$ do

- $p_k = r_k + \beta_{k-1}(p_{k-1} - v_{k-1})$
- Computation $p_k = Ap_k$
- Reduction $r_k \leftarrow p_k$
- $\sigma = \mathbf{r}_k^T p_k$, $\alpha_k = \rho_0/\sigma$. $\delta r_k = t_{k-1} - r_k$
- $y_k = \delta r_k - \alpha_k(w_{k-1} + p_k)$, $t_k = r_k - \alpha_k p_k$
- Computation $t_k = At_k$
- Reduction $t_k^* t_k^*$, $t_k^* t_k^* y_k$, $y_k^* t_k$, $y_k^* y_k$
- $[\zeta_k, \eta_k] = \text{PolCoef}(t_k, t_k^* y_k)$
- $v_k = \zeta_k p_k + \eta_k(\delta r_k + \beta_{k-1} v_{k-1})$
- $z_k = \zeta_k r_k + \eta_k z_{k-1} - \alpha_k v_k$
- $r_{k+1} = t_k - \eta_k y_k - \zeta_k t_k$, $x_{k+1} = x_k + \alpha_k p_k + z_k$
- Reduction $t_k^* r_{k+1}$
- $\rho_1 = \mathbf{r}_0^* r_{k+1}$, $\beta_k = (\alpha_k/\zeta_k)(\rho_1/\rho_0)$, $\rho_0 = \rho_1$
- $w_k = t_k^* + \beta_k p_k$

end

The recurrence coefficients $\zeta_k$ and $\eta_k$ are determined so as to minimize the residual norm $\|r_{k+1}\|_2 = \|t_k - \zeta_k t_k - \eta_k y_k\|_2$. The algorithm for the computations of the coefficients $\zeta_k$ and $\eta_k$ can be found below.

Computation of the coefficients $\zeta_k$ and $\eta_k$

\begin{align*}
\text{Function } [\zeta, \eta] & = \text{PolCoef}(t, t^*, y) \\
\mu_1 & = y^* y, \quad \nu = y^* t^*, \quad \mu_2 = t^* t^*, \quad \omega_1 = y^* t, \quad \omega_2 = t^* t \\
\delta & = \mu_1 \mu_2 - |\nu|^2, \quad \zeta = (\mu_1 \omega_2 - \nu \omega_1)/\delta, \quad \eta = (\mu_2 \omega_1 - \nu \omega_2)/\delta \\
(\text{If } k = 0, \text{ then } \zeta = \omega_2/\mu_2, \quad \eta = 0)
\end{align*}
The standard GPBiCG requires two MVs per iteration. The MVs can be considered as semi-local operations on parallel computers. The MVs are therefore regarded as to be primarily compute-bound. The standard GPBiCG additionally has three global reduction steps to compute the inner products required in the computations of $\alpha_k$, $\beta_k$, and $\zeta_k$ and $\eta_k$. On parallel computers the inner products require global communication among all workers to assemble the locally computed inner product fractions and redistribute the final scalar result to all workers.

Following [1], we design a parallel variant of GPBiCG. First, the inner product for the computation of $\alpha_k$ is merged with that of $\beta_k$ to reduce the number of global communication phases. The vector $p'_k := Ap_k$ is computed by an explicit multiplication by $A$ in the standard GPBiCG, but is obtained by using a recurrence formula

$$p'_k := Ap_k = Ar_k + \beta_k^{-1}(Ap_k - Av_k) = Ar_k + \beta_k^{-1}(p'_{k-1} - Av_{k-1}).$$

Then the denominator ($\tilde{r}_0, p'_0$) of $\alpha_k$ is computed by

$$\tilde{r}_0p'_k = \tilde{r}_0(Ar_k) + \beta_k^{-1}(\tilde{r}_0p'_{k-1} - \tilde{r}_0(Av_{k-1})).$$

The inner product for the computation of $\alpha_{k+1}$ can be moved below that of $\beta_k$.

Next since the three explicit MVs $Ar_k$, $t'_k := At_k$ and $v'_k := Av_k$ are required, the vectors $t'_k := At_k$ and $v'_k := Av_k$ are computed by using the recurrence formulas

$$t'_k := At_k = Ar_k - \alpha_k Ap'_k,$$
$$v'_k := Av_k = \zeta_k Ap'_k + \eta_k (t'_{k-1} - Ar_k + \beta_k^{-1}v'_{k-1}).$$

Now the two explicit MVs $Ar_{k+1}$ and $Ap'_k$ are required. The computation of the inner product $\tilde{r}_0(At_{k+1})$ with the global reduction phase needs to be computed after executing the explicit MV $Ar_{k+1}$. Thus the global communication step cannot be overlapped with the computation of the MV. Moreover, the MV $Ap'_k$ cannot also be overlapped with the global reduction for the computations of $\zeta_k$ and $\eta_k$ since $t'_k$ is required for the computations of $\zeta_k$ and $\eta_k$, and the vector $Ap'_k$ is used in the recurrence for updating $t'_k$. The communication latency cannot be hidden behind the computations of the MVs $Ar_{k+1}$ and $Ap'_k$. Therefore, the vectors $r'_{k+1} := Ar_{k+1}$ and $p''_k := Ap'_k$ need to be updated by the recurrence formulas

$$r'_{k+1} := Ar_{k+1} = t'_k - \eta_k Ay_k - \zeta_k At'_k,$$
$$p''_k := Ap'_k = Ar'_{k+1}(p'_{k-1} - Av_{k-1}).$$

The recurrence formulas for keeping two explicit MVs and updating the residual vectors

$$y'_k := Ay_k = t'_k - r'_k - \alpha_k(w'_{k-1} + p''_k),$$
$$t'_k := At'_k = Ar'_k - \alpha_k Ap''_k,$$
$$w'_k := Av'_k = t'_k + \beta_k p''_k,$$
$$v''_k := Av'_k = \zeta_k Ap''_k + \eta_k(t'_{k-1} - Ar'_{k+1} + \beta_k v''_{k-1})$$

are required. Then the explicit MVs $Ar'_{k+1}$ and $Ap''_k$ needs to be computed, and can be overlapped with the global communication phase for the computations of $\zeta_k$ and $\eta_k$, and $\beta_k$ and $\alpha_{k+1}$. A parallel variant of GPBiCG, which is referred to as a pipelined GPBiCG, can be found below.

In the pipelined GPBiCG, the inner product for the computation of $\alpha_k$ is merged with that of $\beta_k$, and thus the number of global communication phases is reduced. In the second global reduction phase, the inner products are communicated simultaneously. This leads to only one global synchronization point for computing the coefficients $\alpha_{k+1}$ and $\beta_k$. The vectors $y_k$, $t_k$ and $t'_k$ for the first global reduction phase are independent of the update $p''_k$, and the vectors
The computation was executed for the second global reduction phase are also independent of the update \( r_k^{p+1} \). Therefore the MVs \( A\bar{p}_k \) and \( A\bar{r}_k \) are moved above the inner products. After the local computation of the inner products has been executed on each worker, each global reduction can be overlapped with the computation of the corresponding MV. This overlap hides the communication latency behind the computation of the explicit MV. On the other hand, the way of the computation for the denominator of \( \alpha_k \) in the pipelined GPBiCG is different from that in the standard GPBiCG. Thus it may lead to different convergence behavior, i.e., the pipelined GPBiCG may be more affected by rounding errors than the standard GPBiCG.

Pipelined GPBiCG algorithm

Select \( x_0 \) and \( r_0 \). Compute \( r_0 = b - Ax_0 \), \( r_0' = Ar_0 \), \( r_0'' = A^2r_0 \), \( \rho_0 = r_0'r_0' \), \( \sigma = r_0'r_0'' \) and \( \alpha_0 = \rho_0/\sigma \).

Set \( \beta_{-1} = 0 \), \( t_{-1} = t'_{-1} = t''_{-1} = w_{-1} = w'_{-1} = 0 \).

While \( ||r_k||/||r_0|| > tol \) do

\[
\begin{align*}
  p_k &= r_k + \beta_{-1}(p_{k-1} - v_{k-1}), \\
  p'_{k} &= r'_{k} + \beta_{k-1}(p'_{k-1} - v'_{k-1}), \\
  p''_{k} &= r''_{k} + \beta_{k-1}(p''_{k-1} - v''_{k-1}), \\
  \delta r_k &= t_{k-1} - r_k, \\
  \delta r'_k &= t'_{k-1} - r'_k, \\
  \delta r''_k &= t''_{k-1} - r''_k, \\
  y_k &= \delta r_k - \alpha_k(w_{k-1} + p_k), \\
  y'_k &= \delta r'_k - \alpha_k(w'_{k-1} + p'_k), \\
  t_k &= r_k - \alpha_k p_k, \\
  t'_k &= r'_k - \alpha_k p'_k.
\end{align*}
\]

Computation \( p''_k = Ap''_k \)

Reduction \( t''_k, t''_k, t''_k, y''_k \), \( y''_k \), \( y''_k \), \( y''_k \)

\[
\begin{align*}
  [\zeta, \eta] &= \text{PolCoef}(t_k, t'_k, y_k) \\
  t''_k &= \delta v''_k - \alpha p''_k, \\
  v_k &= \zeta p_k + \eta(k\delta r_k - \beta_{k-1}v_{k-1}), \\
  v'_k &= \zeta p'_k + \eta(k\delta r'_k - \beta_{k-1}v'_{k-1}), \\
  v''_k &= \zeta p''_k + \eta(k\delta r''_k - \beta_{k-1}v''_{k-1}), \\
  x_k &= \zeta r_k + \eta(\delta r_k - \beta_{k-1}v_{k-1}), \\
  y_k &= \zeta t'_k - \delta v_k, \\
  r_{k+1} &= t_k - \eta y_k + x_k t'_k + r''_k, \\
  r_{k+1} &= t_k - \eta y'_k + x_k t'_k + r''_k, \\
  x_{k+1} &= x_k + \eta y_k + x_k. \\
\end{align*}
\]

Computation \( r''_{k+1} = Ar''_{k+1} \)

Reduction \( r''_{k+1}, r''_{k+1}, r''_{k+1}, r''_{k+1}, r''_{k+1} \)

\[
\begin{align*}
  \rho_1 &= r''_{k+1}, \\
  \beta_k &= (\alpha_k/\zeta)(\rho_1/\rho_0), \\
  \rho_0 &= \rho_1, \\
  \sigma &= r''_{k+1} + \beta_k(r''_{k+1} - r''_{k+1}), \\
  \alpha_{k+1} &= \rho_1/\sigma, \\
  w_k &= t'_k + \beta_k p'_k, \\
  w''_k &= t''_k + \beta_k p''_k.
\end{align*}
\]

end 

In [8], four variants of GPBiCG, of which the convergence behavior is more stable than the standard, have been proposed. The two variants of them are derived by using the three term recurrence, and the others are done by using the coupled two-term recurrence suggested by Rutishauser. Similarly, following [1], we design a pipelined variant using the three term recurrence, which is referred to as a pipelined variant 1, and another pipelined variant using the coupled two-term recurrence, which is referred to as a pipelined variant 2. Note that we skip to display the two pipelined algorithms of GPBiCG.

3. Numerical experiments

In this section, we examine the convergence behavior among the original GPBiCG (indicated by GPBiCG), the two variants of GPBiCG (indicated by variant 1 and variant 2), the pipelined GPBiCG (indicated by GPBiCG (pipelined)), the pipelined variant 1 of GPBiCG (indicated by variant 1 (pipelined)), and the pipelined variant 2 of GPBiCG (indicated by variant 2 (pipelined)). Numerical experiments were carried out in double-precision floating-point arithmetic on a PC.

As shown in [17], applying 5-point central differences to the two-dimensional convection-diffusion equation \(-u_{xx} - u_{yy} + \gamma(xu_x + yu_y)/\beta u = f(x, y)\) over the unit square \( \Omega_u = (0, 1) \times (0, 1) \) with the Dirichlet boundary conditions \( u|_{\partial \Omega_u} = 0 \) yields a linear system (1) with a nonsymmetric matrix. The mesh size is chosen as 101 (= \( M + 1 \)) in both directions of \( \Omega_u \), so that the resulting linear system has an \( M^2 \times M^2 \) coefficient matrix. The right-hand side vector \( \bar{b} \) is determined by substituting a solution vector \( \bar{x} = (1, \ldots, 1)^T \) into the equation \( \bar{b} = A\bar{x} \). The numerical computation was executed for \((\gamma, \beta) = (1000, -100)\). The iteration was started with random
vector $x_0$ (with seed 49). The initial shadow residual was set to $r_0^* = r_0$. The iterations are terminated when the reduction of residual norms (i.e., $\|r_k\|_2/\|r_0\|_2$) becomes $\leq 10^{-12}$ or $\leq 10^{-10}$.

Table 1 shows the number of MVs (indicated by MVs) and the true relative residual 2-norms $\log_{10}(\|b - Ax_k\|_2/\|r_0\|_2)$ (abbreviated as True res.) at termination for tol = $10^{-12}$ or $10^{-10}$. The iterations are stopped when the relative residual norms do not attain $10^{-12}$ or $10^{-10}$ by 10,000 MVs. Note that Bi-CGSTAB does not converge on the problem.

Table 1. Number of MVs and true relative residual norms for GPBiCG, the variant 1, the variant 2, GPBiCG (pipelined), the variant 1 (pipelined), and the variant 2 (pipelined) on the problems with $(\gamma, \beta) = (1000, -100)$ for tol = $10^{-12}$ (on the middle column) or tol = $10^{-10}$ (on the right column).

| $(\gamma, \beta)$ = (1000, -100) | MVs (ratio) | True res. | MVs (ratio) | True res. |
|---------------------------------|-------------|-----------|-------------|-----------|
| tol = $10^{-12}$                |             |           |             |           |
| GPBiCG                          | 1810 (1.00) | -12.0     | 1750 (1.00) | -10.2     |
| variant 1                       | 2026 (1.12) | -11.8     | 1958 (1.11) | -10.3     |
| variant 2                       | 2842 (1.57) | -11.9     | 2576 (1.47) | -10.0     |
| GPBiCG (pipelined)              | --- (---)   | -0.4      | --- (---)   | -0.4      |
| variant 1 (pipelined)           | 3148 (1.73) | 0.7       | 2038 (1.16) | -9.1      |
| variant 2 (pipelined)           | 2144 (1.18) | -11.2     | 2070 (1.18) | -10.0     |
| tol = $10^{-10}$                |             |           |             |           |

4. Conclusion

We have designed three pipelined variants of GPBiCG, and then compared convergence behavior among GPBiCG, the variants 1 and 2, and the implementations of the pipelined GPBiCG by numerical experiments. GPBiCG (pipelined) and the variant 1 (pipelined) using the three term recurrence are more affected by rounding errors than the original counterparts, respectively, but the variant 2 (pipelined) has as almost same convergence as the original one. The variant 2 (pipelined) is useful and more robust than GPBiCG (pipelined) and the variant 1 (pipelined).

This research was partly supported by JSPS KAKENHI Grant Number 19K12008.

References

[1] Cools S and Vanroose W 2017 Parallel Comput. 65 1–20
[2] Fletcher R 1976 Lect. Notes Math. 506 73–89
[3] Sonneveld P 1989 SIAM J. Sci. Comput. 10 36–52
[4] van der Vorst H A 1992 SIAM J. Sci. Stat. Comput. 13 631–644
[5] Sleijpen G L G and Fokkema D R 1993 ETNA 1 11–31
[6] Gutknecht M H 1993 SIAM J. Sci. Comput. 14 1020–1033
[7] Zhang S L 1997 SIAM J. Sci. Comp. 18 537–551
[8] Abe K and Sleijpen G L G 2013 Appl. Numer. Math. 67 4–16
[9] Chronopoulos A T and Gear C W 1989 J. Comput. Math. 25 153–168
[10] Erhel J 1995 ETNA 3 160–176
[11] Sturler E D 1991 Proceedings of the 13th IMACS World Congress on Computational and Applied Mathematics vol 9 IMACS (Criterion Press)
[12] Demmel J W, Heath M T and van der Vorst H A 1993 Acta Numerica 2 111–197
[13] Ghysels P, Ashby T J, Meerenbergen K and Vanroose W 2013 SIAM J. Sci. Comput. 35 C48–C71
[14] Ghysels P and Vanroose W 2014 Parallel Comput. 40 224–238
[15] Sturler E D and van der Vorst H A 1995 Appl. Numer. Math. 18 441–459
[16] Abe K 2018 18th Asia Simulation Conference, AsiaSim 2018 Proceedings (Modeling, Design and Simulation of Systems, Communications in Computer and Information Science) vol 946 (Springer) pp 401–403
[17] Saad Y 1993 SIAM J. Sci. Stat. Comput. 14 461–469