Numerical Comparison of Accelerating Polynomials in Product-type Iterative Methods

Seventh SIAM Conference on
Applied Linear Algebra 2000

H. Hasegawa, K. Abe, and S.-L. Zhang

Outline

- Product-type methods
- Accelerating polynomials
- Picking up Bi-CG part
- Reconstructing from a common Bi-CG
- Conclusion

  (about accelerating polynomials and Quadruple arithmetic)

Iterative methods for Nonsymmetric Matrix

- Product-type methods
  - Bi-CG (1976)
  - CGS (1984, Sonneveld)
  - Bi-CGSTAB (1989, van der Vorst)
  - GPBi-CG(1992, Zhang)
  - Bi-CGSTAB(l) (1993)
- Others
  - GCR (1982)
  - GMRES (1983)

Structure of Product-type methods

- Series of residual vectors in Bi-CG method

  $r_0, r_1, r_2, \ldots , r_k, \ldots$

- Series of residual vectors in Product-type method

  $H_0(A)r_0, H_1(A)r_1, \ldots, H_k(A)r_k, \ldots$

accelerated and stabilized by k-th polynomial $H_k(A)$

- Bi-CG part in these methods must be same in Mathematics!

Residual Vectors

- CGS: $r_k^{\,CGS} = R_k(A) r_k$

  $R_k(A)$ is Residual polynomial of Bi-CG method
- Bi-CGSTAB: $r_k^{\,STA} = Q_k(A)\, r_k$

  $Q_0(\lambda) = 1;\ Q_{k+1}(\lambda) = (1-\omega_k \lambda)\, Q_k(\lambda)$

  ω_k minimizes $r_{k+1}^{\,STA} = t_k - \omega_k A t_k$
- GPBi-CG : $r_k^{\,GP} = H_k(A)\, r_k$

  $H_0(\lambda) = 1;\ H_1(\lambda) = 1-\xi_0 \lambda;$

  $H_{k+1}(\lambda) = (1+\eta_k - \xi_k \lambda)\ H_k(\lambda) - \eta_k H_{k-1}(\lambda)$

  η_k and ξ_k minimize $r_{k+1}^{\,GP} = t_k - \eta_k y_k - \xi_k A t_k$

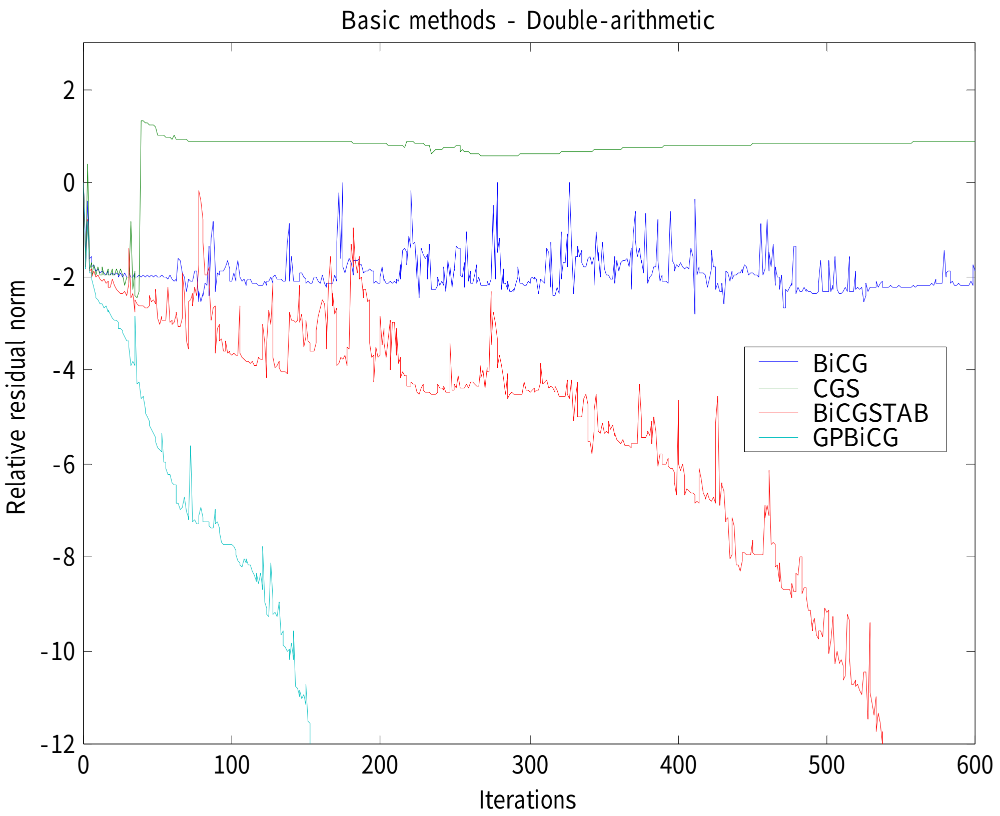
Test problem

Toeplitz Matrix, $\mathrm{N} = 200$, $\gamma = 1.7$

$$A := \begin{bmatrix} 2 & 1 & & & \\ 0 & 2 & 1 & & \\ \gamma & 0 & 2 & 1 & \\ & \gamma & 0 & 2 & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix}$$

Right-hand side $b = (1, 1, \cdots, 1)^T$

Convergence history



Why?

- We wish that Product-type methods show good convergence history, but some of them could not.
- We try to compare accelerating polynomials $R_k(A)$, $Q_k(A)$, and $H_k(A)$.
  - Reconstructing Bi-CG from EACH methods
  - Reconstructing EACH methods from one Bi-CG

  (Bi-CG part in each methods should be same in Mathematics)
- We compute them in Quadruple-arithmetic.

Picking up Bi-CG part

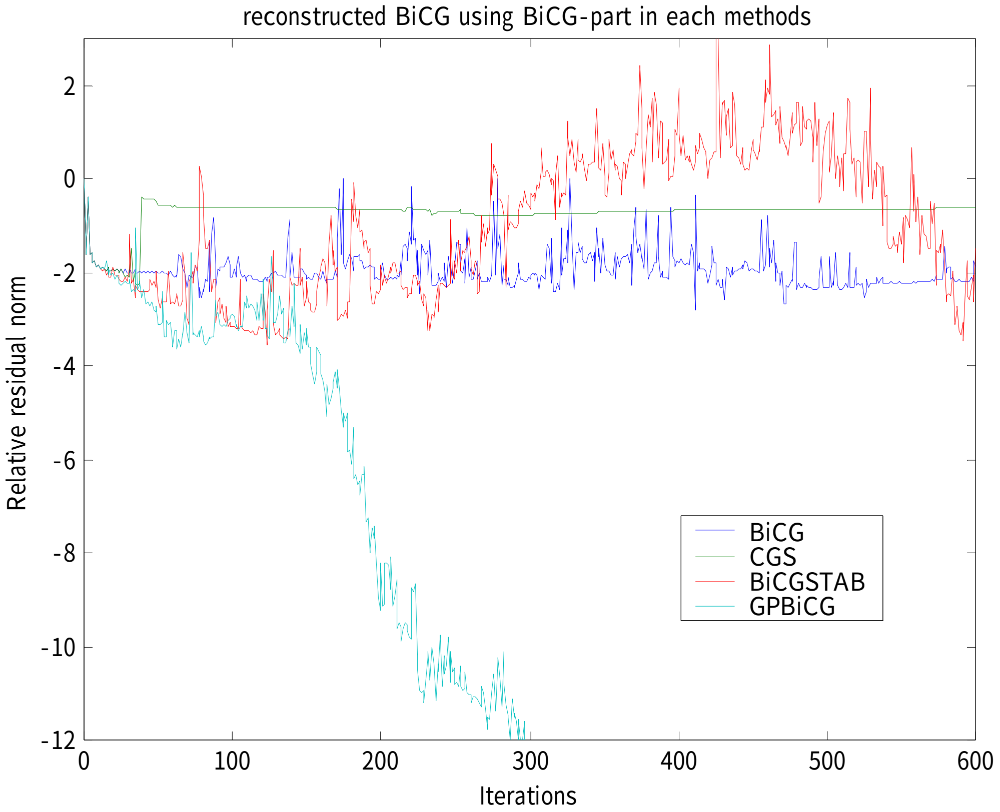
- All methods have Bi-CG part in their process
- We reconstruct Bi-CG process by using alpha and beta of the CGS, Bi-CGSTAB, and GPBi-CG:

  CGS: $r_k^{\ CGS} = R_k(A)\underline{r}_{\underline{k}}$

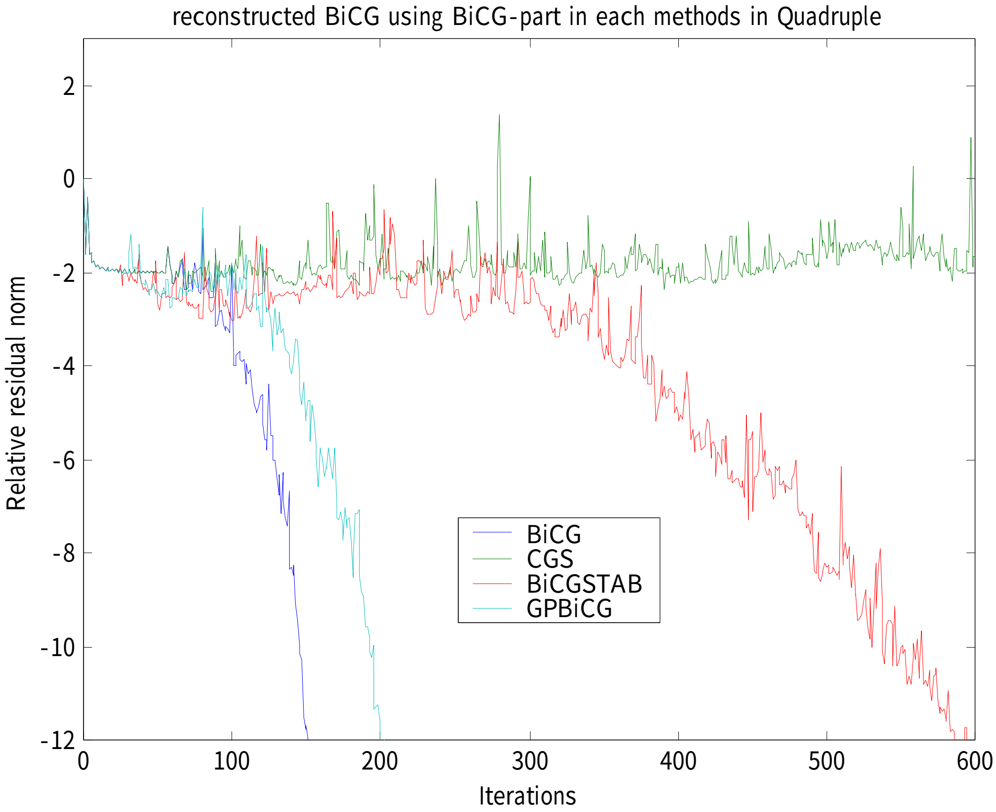
  Bi-CGSTAB: $r_k^{\ STA} = Q_k(A)\ \underline{r}_{\underline{k}}$

  GPBi-CG : $r_k^{\ GP} = H_k(A)\ \underline{r}_{\underline{k}}$
- Bi-CG part must be same in Mathematics, effect of some errors will be shown.

Convergence history of Bi-CG part
(reconstruct Bi-CG using alpha and beta in each methods)



Convergence of Bi-CG part: Quadruple
(reconstruct Bi-CG using alpha and beta in each methods)



How Bi-CG part works?

- Bi-CGSTAB converges by an effect of MR part (Bi-CG part is still unstable)
- GPBi-CG makes Bi-CG part stable
- CGS did not converge in Quadruple arithmetic
- In Quadruple arithmetic, simple Bi-CG is the best (Bi-CG is much affected by Rounding errors)
- In Quadruple arithmetic, Bi-CG part in Bi-CGSTAB is bad convergence even if Bi-CG converges.

Reconstructing from one Bi-CG part

- Bi-CG part must be same in Mathematics, so we force being same Numerically.
- We reconstruct each methods based on the same alpha and beta of the original Bi-CG:

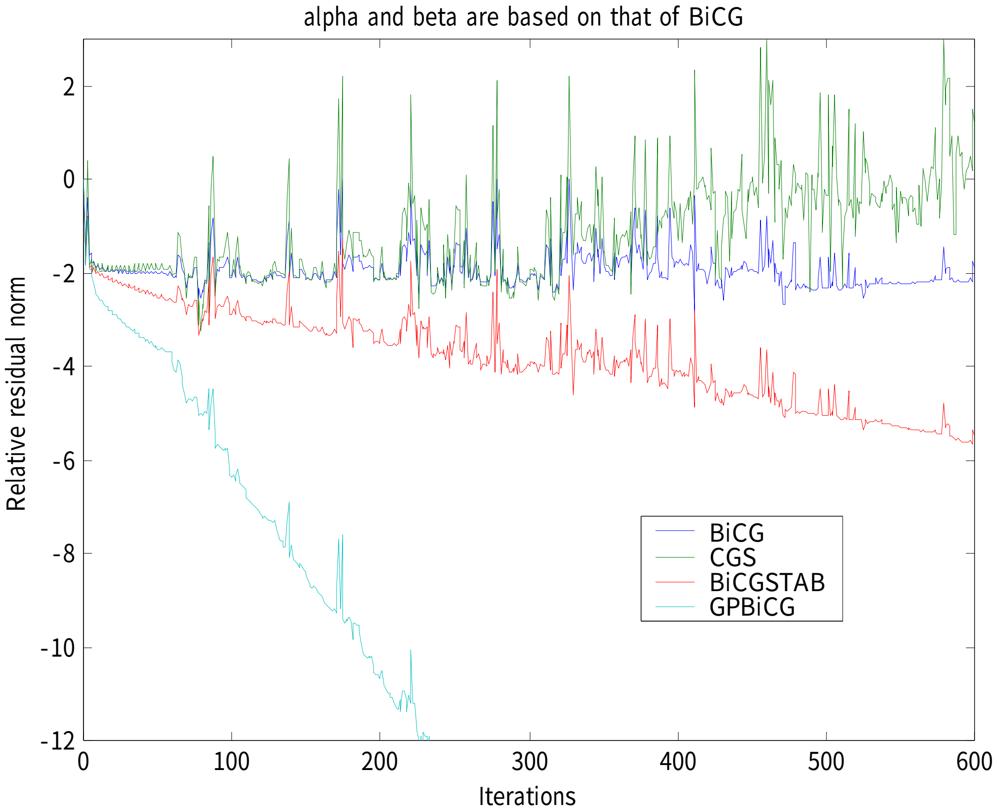
  CGS: $\underline{r_k}^{CGS} = R_k(A) r_k$

  Bi-CGSTAB: $\underline{r_k}^{STA} = Q_k(A)\, r_k$

  GPBi-CG : $\underline{r_k}^{GP} = H_k(A)\, r_k$

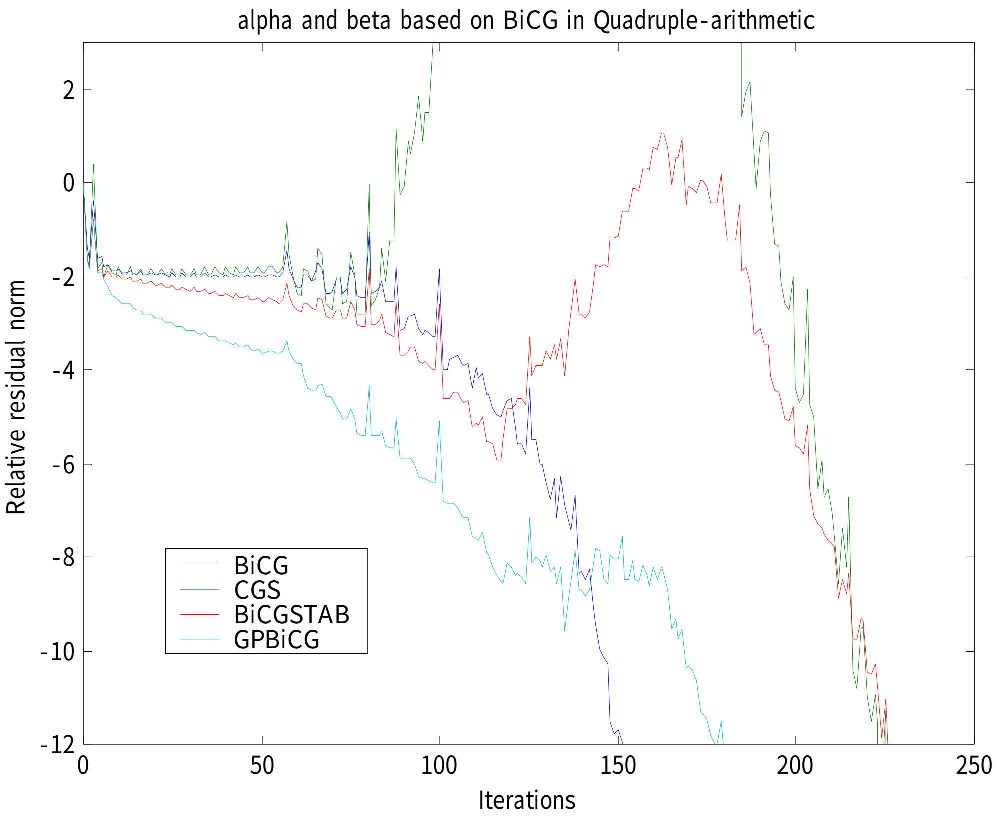
- The effect of accelerating polynomials will be shown.

Convergence history based on one Bi-CG
(alpha and beta in Bi-CG are used in all methods)

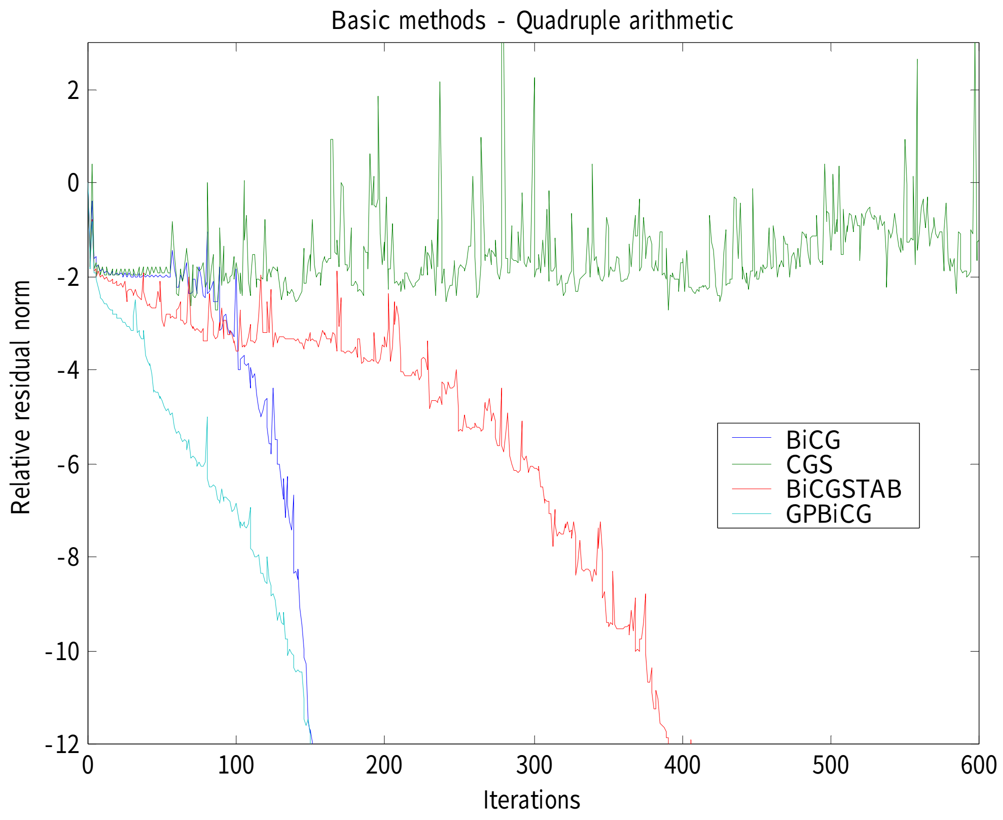


H. Hasegawa, K. Abe, and S.-L. Zhang

Convergence history based on one Bi-CG
(Quadruple arithmetic is used for Bi-CG)



Convergence history based on one Bi-CG
(Quadruple arithmetic is used for ALL)



How accelerating polynomial works

- Qudaruple arithmetic works very well.
- If enough accuracy was provided, Bi-CG was the best.
- Bi-CGSTAB and GPBi-CG work well.
- In Quadruple arithmetic, sometimes it works as braking not as accelerating.
- GPBi-CG is robust in both two conditions.
- CGS does not work in both conditions because of “squared”.

Conclusion

1. Quadruple arithmetic is very powerful tool for accelerating and stabilizing, also easy and simple.
2. Effects of accelerating polynomials are not same. It depends on Computing Accuracy.
3. GPBi-CG converges well, and is robust.
4. Bi-CGSTAB converges well, but is not robust.
5. Bi-CG is the best in more accurate environment.
6. CGS should be out of consideration.

Appendix

- We believe that high performance should be used not only for "Speeding" but also the "Quality of Computation".

- Effectiveness of Iterative algorithms strongly depends on the Computing Accuracy.

- Quadruple arithmetic operation is not expensive in HighPerformance Computers and classic machines.

H. Hasegawa, K. Abe, and S.-L. Zhang