ICMASE 2026 14 September 2026

GPBiCG(m, ℓ, k): An Extension of GPBiCG(m, ℓ) and Preliminary Adaptive Parameter Strategies

Presented by Dr. **Ronan Dupont**

Under the supervision of
Tomohiro Sogabe (曽我部知広) and Shao-Liang Zhang (張紹良)

名古屋大学大学院工学研究科張・曽我部研究室
Graduate School of Engineering, Nagoya University, Japan
Zhang–Sogabe Laboratory

名古屋大学 Nagoya University
ICMASE2026
MOONSHOT RESEARCH & DEVELOPMENT PROGRAM

Introduction

- GPBiCG(m, ℓ) alternates m BiCGSTAB-type updates and ℓ GPBiCG-type updates, both driven by local residual minimization.
- **Why add k?** GPBiCG(m, ℓ, k) inserts k LOPBiCG-type updates based on an orthogonality condition. The third phase changes the residual polynomial and can complement the two minimization phases on matrix-dependent convergence trajectories.
- We therefore ask, in this order:
  1. Does the additional k-phase improve convergence and CPU time?
  2. Do the gains remain visible with conventional ILU(0)?
  3. Can a live controller reduce the sensitivity to (m, ℓ, k)?

Narrative of the study

Static extension first; adaptive parameter mitigation second. The adaptive study follows from the observed matrix dependence of the best triplet, rather than replacing the static contribution.

BiCGSTAB H. A. van der Vorst, *SIAM J. Sci. Statist. Comput.*, 13(2), 631–644, 1992.

GPBiCG S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

GPBiCG(m, ℓ) S. Fujino, *Appl. Numer. Math.*, 41(1), 107–117, 2002.

LOPBiCG(ℓ) R.-J. Zhao, T. Sogabe, T. Kemmochi, and S.-L. Zhang, *Numer. Linear Algebra Appl.*, 32(5), e70033, 2025.

Presentation roadmap

Static extension first; adaptive parameter mitigation second

I **Krylov reminder**

II **GPBiCG**(m, ℓ, k)

III **Numerical study**

IV **Adaptive strategy**

V **Conclusions and perspectives**

Krylov reminder

Reminder: Krylov Product-Type Methods

Linear system and Krylov space

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad A \in \mathbb{R}^{n\times n},$$

$$\boldsymbol{x}_n \in \boldsymbol{x}_0 + \mathcal{K}_n(A, \boldsymbol{r}_0),$$

$$\mathcal{K}_n(A, \boldsymbol{r}_0) = \mathrm{span}\{\boldsymbol{r}_0, A\boldsymbol{r}_0, \ldots, A^{n-1}\boldsymbol{r}_0\}.$$

Product-type idea

BiCGSTAB and GPBiCG retain the Bi-Lanczos recurrence while applying a polynomial action to improve the residual trajectory:

$$\boldsymbol{r}_{n+1} = \phi_n(A)\boldsymbol{r}_n.$$

They seek smoother or faster convergence than the underlying BiCG process.

What changes in this work?

The BiCG recurrence is shared. GPBiCG(m, ℓ, k) changes the sequence of product-type updates applied within each cycle.

BiCGSTAB H. A. van der Vorst, *SIAM J. Sci. Statist. Comput.*, 13(2), 631–644, 1992.

GPBiCG S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

GPBiCG in One Iteration

- GPBiCG keeps the BiCG search direction, but adds a two-parameter polynomial smoothing step.
- At iteration n, it first computes a BiCG intermediate residual:

$$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n.$$

- It then chooses two scalars (ζ_n, η_n) to reduce the next residual:

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

1: Build the BiCG direction $\boldsymbol{p}_n$ and scalar α_n.
2: Form $\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n$.
3: Compute the auxiliary vector $\boldsymbol{y}_n$ from the previous recurrence.
4: Choose (ζ_n, η_n) by a local residual-minimization rule.
5: Update $\boldsymbol{x}_{n+1}$, $\boldsymbol{r}_{n+1}$, and the recurrence vectors.

Why this matters for GPBiCG(m, ℓ, k)

The new method changes the frequency of the BiCGSTAB, GPBiCG and LOPBiCG choices for (ζ_n, η_n) inside one cycle.

 **GPBiCG** S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

II

GPBiCG(m, ℓ, k)

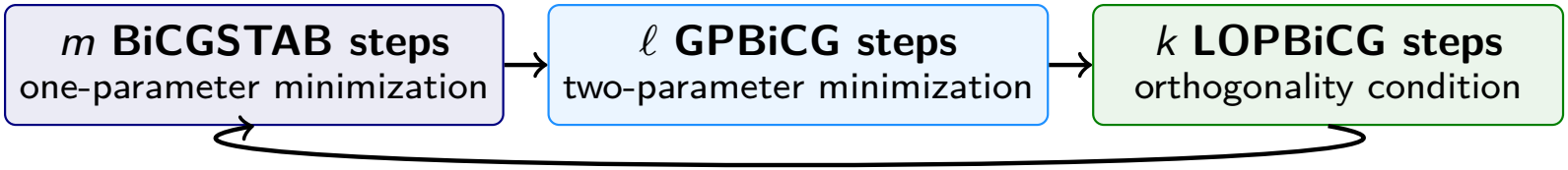
From GPBiCG(m, ℓ) to a Three-Phase Cycle

All three phases use $\boldsymbol{r}_{n+1} = \boldsymbol{t}_n - \eta_n \boldsymbol{y}_n - \zeta_n A\boldsymbol{t}_n$, where $\boldsymbol{y}_n$ is the auxiliary vector of the GPBiCG recurrence.

Phase	Selection rule	ζ_n	η_n
BiCGSTAB	$\min_\zeta \|\boldsymbol{r}_{n+1}\|_2$	$\dfrac{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)}$	0
GPBiCG	$\min_{\zeta,\eta} \|\boldsymbol{r}_{n+1}\|_2$	$\dfrac{(\boldsymbol{y}_n, \boldsymbol{y}_n)(A\boldsymbol{t}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{y}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (A\boldsymbol{t}_n, \boldsymbol{y}_n)(\boldsymbol{y}_n, A\boldsymbol{t}_n)}$	$\dfrac{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, A\boldsymbol{t}_n)(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (A\boldsymbol{t}_n, \boldsymbol{y}_n)(\boldsymbol{y}_n, A\boldsymbol{t}_n)}$
LOPBiCG	$(\boldsymbol{t}_n, \boldsymbol{r}_{n+1}) = 0$	$\dfrac{(\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}$	0

Table 1: Exact choices of (ζ_n, η_n) in the three phases.

Why k? The orthogonality-based LOPBiCG phase supplies a polynomial action different from the two local residual-minimization phases.



Repeat the $m + \ell + k$ update cycle with the common BiCG recurrence.

BiCGSTAB H. A. van der Vorst, *SIAM J. Sci. Statist. Comput.*, 13(2), 631–644, 1992.

GPBiCG S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

GPBiCG(m, ℓ) S. Fujino, *Appl. Numer. Math.*, 41(1), 107–117, 2002.

LOPBiCG(ℓ) R.-J. Zhao, T. Sogabe, T. Kemmochi, and S.-L. Zhang, *Numer. Linear Algebra Appl.*, 32(5), e70033, 2025.

GPBiCG(m, ℓ, k) Cycle Pseudo-Code

1: Initialize $\boldsymbol{x}_0$, $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, and a shadow residual $\boldsymbol{r}_0^*$.
2: **while** $\|\boldsymbol{r}_n\|_2 / \|\boldsymbol{b}\|_2 > \varepsilon$ **do**
3: **for** $j = 1, \ldots, m$ **do**
4: Perform one BiCGSTAB-like update.
5: **end for**
6: **for** $j = 1, \ldots, \ell$ **do**
7: Perform one GPBiCG residual-minimizing update.
8: **end for**
9: **for** $j = 1, \ldots, k$ **do**
10: Perform one LOPBiCG orthogonality update.
11: **end for**
12: Evaluate the true relative residual.
13: **end while**

At each update

- build the BiCG direction;
- form the intermediate residual $\boldsymbol{t}_n$;
- select (ζ_n, η_n) from the active phase;
- update $\boldsymbol{x}_{n+1}$ and $\boldsymbol{r}_{n+1}$.

Stopping rule

The experiments report the true residual $\|\boldsymbol{b} - A\boldsymbol{x}_n\|_2 / \|\boldsymbol{b}\|_2$, not only the recursively updated residual norm.

III

Numerical study

Static Numerical Comparison

Eight nonsymmetric matrices from the SuiteSparse Matrix Collection

Matrix	N	nnz	$\kappa_2(A)$	Matrix	N	nnz	$\kappa_2(A)$
utm300	300	3,155	1.50×10^6	memplus	17,758	99,147	1.29×10^5
add20	2,395	13,151	1.20×10^4	epb2	25,228	175,027	2.62×10^3
Goodwin_017	3,317	97,773	1.30×10^5	wang4	26,068	177,196	4.03×10^5
poisson3Da	13,514	352,762	1.12×10^3	crashbasis	160,000	1,750,416	–

N: dimension; nnz: nonzero entries; $\kappa_2(A)$: 2-norm condition number; –: unavailable.

Comparators

- BiCGSTAB and BiCGSTAB2;
- GPBiCG and GPBiCG(m, ℓ).

All solvers use the same matrix, right-hand side, initial guess and stopping rule.

Grid and stopping rule

$(m, \ell, k) \in \{1, \ldots, 7\}^3$ (343 triplets).

Initial guess: $\boldsymbol{x}_0 = \boldsymbol{0}$.

Right-hand side: $\boldsymbol{b} = A\boldsymbol{1}$, with $\boldsymbol{1} = (1, \ldots, 1)^\mathsf{T}$ the all-ones vector.

Stopping rule: $\|\boldsymbol{b} - A\boldsymbol{x}_n\|_2 / \|\boldsymbol{b}\|_2 \leq 10^{-12}$.

Settings: none and conventional ILU(0).

Without Preconditioning: Residual Histories

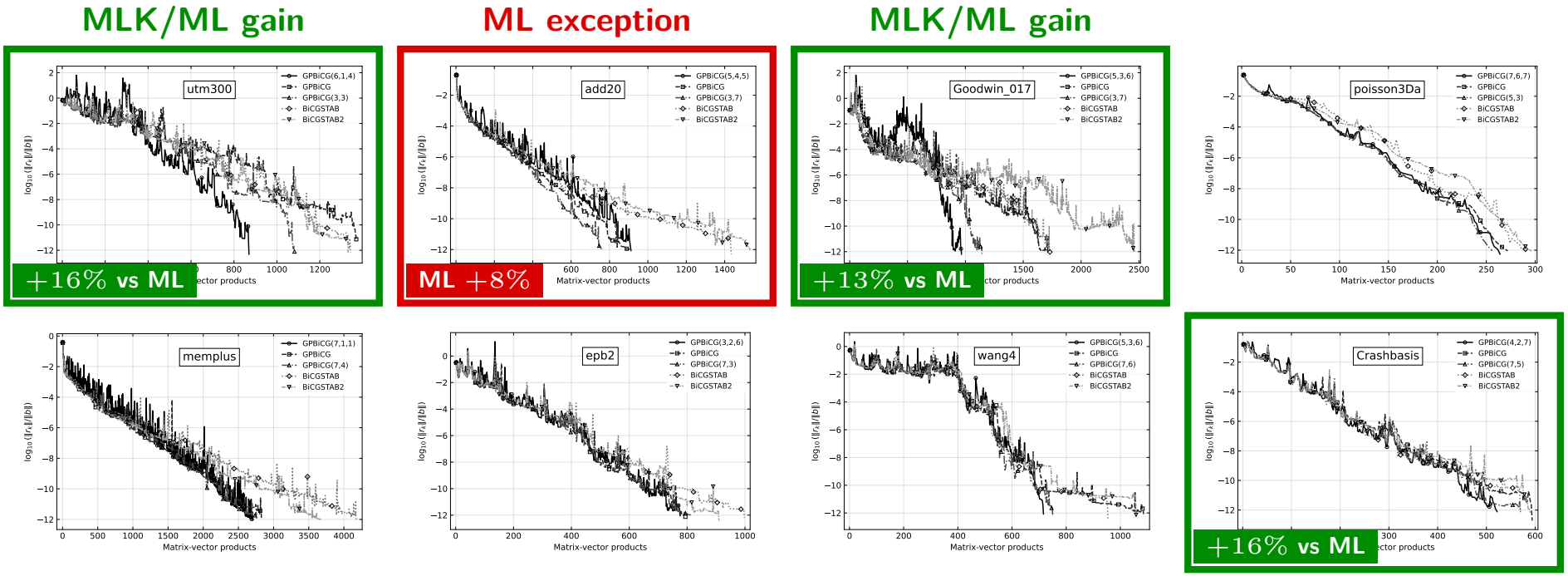


Figure 1: Residual histories without preconditioning. The frames identify the trajectories discussed below; all eight cases remain visible.

- At 10^{-12}, GPBiCG(m, ℓ, k) is faster than GPBiCG(m, ℓ) on 7/8 matrices; only `add20` favours the two-parameter family.
- The largest CPU gains over GPBiCG(m, ℓ) are about 16% on `utm300`, 13% on `Goodwin_017`, and 16% on `Crashbasis`.
- The residual paths are not uniformly ordered: the useful triplet is still matrix-dependent, which motivates the adaptive study.

With ILU(0): Residual Histories

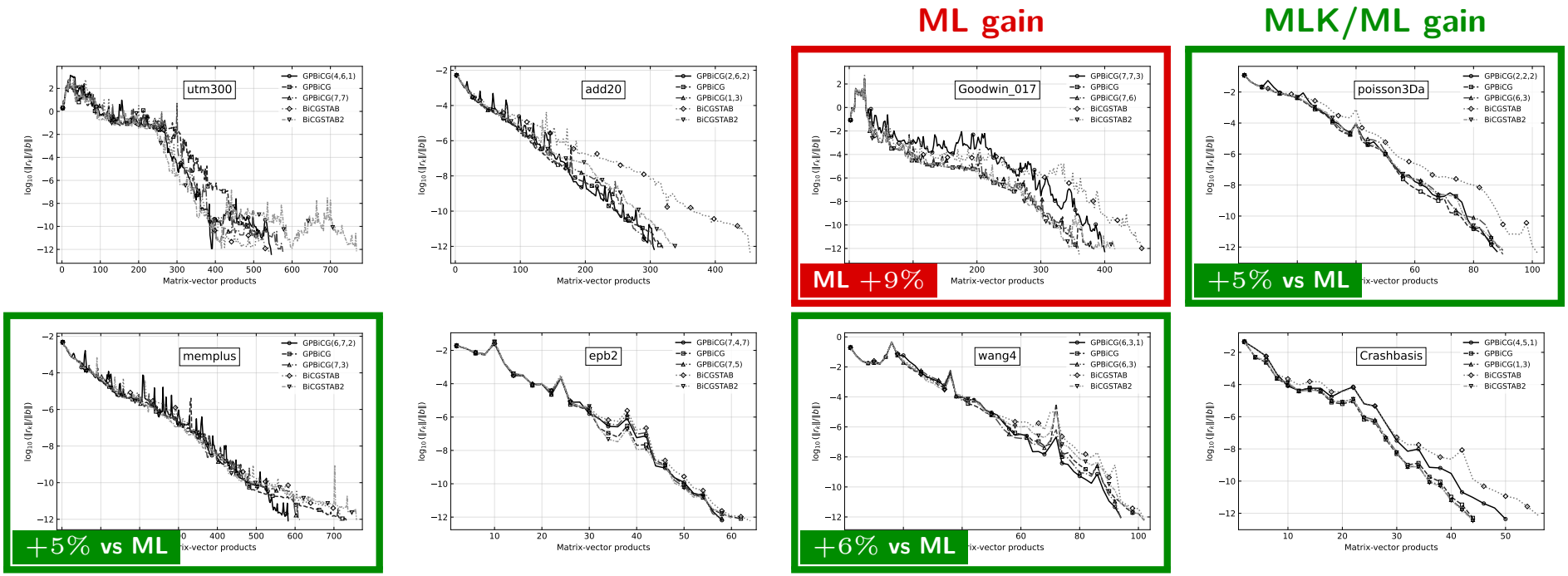


Figure 2: Residual histories with ILU(0). Frames direct attention to the mixed MLK/ML outcome quantified in the CPU table.

- With ILU(0), GPBiCG(m, ℓ, k) is faster than GPBiCG on all six comparable converged rows, with speedups from 1.13 to 1.37.
- Against GPBiCG(m, ℓ) the conclusion is deliberately mixed: 3 wins, 1 near tie, and 3 losses on the converged comparable rows.
- ILU(0) therefore supports the method, but also shows that preconditioning reduces the room for the extra k-phase to help.

Static Results: CPU Synthesis

Without preconditioning

- GPBiCG(m, ℓ, k) is the fastest solver on 6/8 matrices.
- It is faster than GPBiCG(m, ℓ) on 7/8 matrices.
- Largest CPU reductions vs GPBiCG(m, ℓ): about 16% on `utm300` and `Crashbasis`, and 13% on `Goodwin_017`.

With ILU(0)

- GPBiCG(m, ℓ, k) is the fastest converged method on 4/7 comparable rows.
- It is faster than plain GPBiCG on all comparable rows, with speedups from 1.13 to 1.37.
- Against GPBiCG(m, ℓ), the result is mixed: 3 wins, 1 near tie, 3 losses.

Interpretation

The k-phase is useful, but not universally. The supported conclusion is not domination; it is parameter sensitivity plus recoverable performance.

IV

Adaptive strategy

Why Adaptation Is Needed

- The static grid establishes useful GPBiCG(m, ℓ, k) gains, but the best triplet remains matrix-dependent.
- Exhaustive selection among 343 triplets is suitable for an systematic benchmark, not for routine use.
- The live controller therefore asks two practical questions:
  - can residual information reduce the penalty of a poor triplet?
  - can the solver move toward a no-manual-triplet interface?
- CPU time includes monitoring and transitions, because adaptation must repay its own cost.

Adaptive Strategy: One Cycle, One Decision

The admissible grid is $\Theta = \{1, \dots, 7\}^3$. At a cycle boundary, one step may be transferred between phases while $m + \ell + k$ remains fixed.

Cycle-boundary decision

Observe. Run cycle q with fixed $\theta_q = (m_q, \ell_q, k_q) \in \Theta$. For $p \in \{\mathrm{B}, \mathrm{G}, \mathrm{L}\}$, compute

$$\rho_{q,p} = \left(\frac{\|\boldsymbol{r}_{q,p}^{\mathrm{out}}\|_2}{\|\boldsymbol{r}_{q,p}^{\mathrm{in}}\|_2} \right)^{1/n_{q,p}}, \qquad g_q = \frac{\|\boldsymbol{r}_q^{\mathrm{out}}\|_2}{\|\boldsymbol{r}_q^{\mathrm{in}}\|_2}.$$

Propose. $p_+ = \arg\min_p \rho_{q,p}$, $p_- = \arg\max_p \rho_{q,p}$, and $\theta' = \theta_q + e_{p_+} - e_{p_-}$.

Accept or keep. If $g_q < 1$, $\max_p \rho_{q,p} < 1$, and $\theta' \in \Theta$, set $\theta_{q+1} = \theta'$. Otherwise keep $\theta_{q+1} = \theta_q$.

Accepted example

$$\theta_q = (2, 4, 3), \qquad g_q = 0.84,$$
$$(\rho_{\mathrm{B}}, \rho_{\mathrm{G}}, \rho_{\mathrm{L}}) = (0.91, 0.97, 0.88),$$
$$p_+ = \mathrm{L}, \qquad p_- = \mathrm{G},$$
$$\theta' = (2, 4 - 1, 3 + 1),$$
$$\theta' = (2, 3, 4) \in \Theta.$$

All three conditions hold: $0.84 < 1$, $0.97 < 1$, and $\theta' \in \Theta$.

$$\boxed{\theta_{q+1} = (2, 3, 4)}.$$

Next decision: after one complete cycle with θ_{q+1}.

Random-start safeguard. Experiment II also requires predicted saved work to exceed confidence-weighted overhead; the full gate is in backup.

Adaptive GPBiCG(m, ℓ, k): Full Pseudo-Code

Complete controller

Require: A, $\boldsymbol{b}$, $\boldsymbol{x}_0$, tolerance ε, $\theta_0 \in \Theta = \{1, \ldots, 7\}^3$

$q \leftarrow 0$; compute $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
while $\|\boldsymbol{r}_q\|_2 / \|\boldsymbol{b}\|_2 > \varepsilon$ and budget remains **do**
  Run one complete cycle $C(\theta_q)$; keep θ_q fixed
  Store the residuals at each phase entry and exit
  Compute $\rho_{q,p}$ for $p \in \{\mathrm{B}, \mathrm{G}, \mathrm{L}\}$ and g_q
  $p_+ \leftarrow \arg\min_p \rho_{q,p}$; $p_- \leftarrow \arg\max_p \rho_{q,p}$
  $\theta' \leftarrow \theta_q + e_{p_+} - e_{p_-}$
  if $g_q < 1$ and $\max_p \rho_{q,p} < 1$ and $\theta' \in \Theta$ **then**
    $\theta_{q+1} \leftarrow \theta'$
  else
    $\theta_{q+1} \leftarrow \theta_q$
  end if
  $q \leftarrow q + 1$
end while
return the latest iterate $\boldsymbol{x}$

Definitions and timing

Coordinate vectors

$$e_{\mathrm{B}} = (1, 0, 0),$$
$$e_{\mathrm{G}} = (0, 1, 0),$$
$$e_{\mathrm{L}} = (0, 0, 1).$$

Phase score

$$\rho_{q,p} = \left(\frac{\|\boldsymbol{r}_{q,p}^{\mathrm{out}}\|_2}{\|\boldsymbol{r}_{q,p}^{\mathrm{in}}\|_2} \right)^{1/n_{q,p}}.$$

Here $n_{q,p}$ is the number of iterations in phase p. The complete-cycle contraction is $g_q = \|\boldsymbol{r}_q^{\mathrm{out}}\|_2 / \|\boldsymbol{r}_q^{\mathrm{in}}\|_2$.

Timing rule Cycle q selects θ_{q+1}; no switch occurs inside a cycle. Experiment II adds the confidence gate given in backup.

Two Tests Separate Rescue from Parameter Freedom

- Same numerical base in both tests: the eight article matrices and the same admissible grid

$$(m, \ell, k) \in \{1, \ldots, 7\}^3.$$

- **Experiment I asks a rescue question.**
  - all 343 initial triplets are prescribed,
  - static and live runs start from the same triplet,
  - stopping rule: true relative residual 10^{-10}.
- **Experiment II asks a no-manual-triplet question.**
  - 100 seeded random starts per matrix,
  - stopping rule: true relative residual 10^{-12},
  - CPU time includes all starts, including budget-limited failures.
- In both tests, reported CPU time includes monitoring and accepted or rejected parameter changes.

Experiment I Shows Selective Rescue, Not Free Speedup

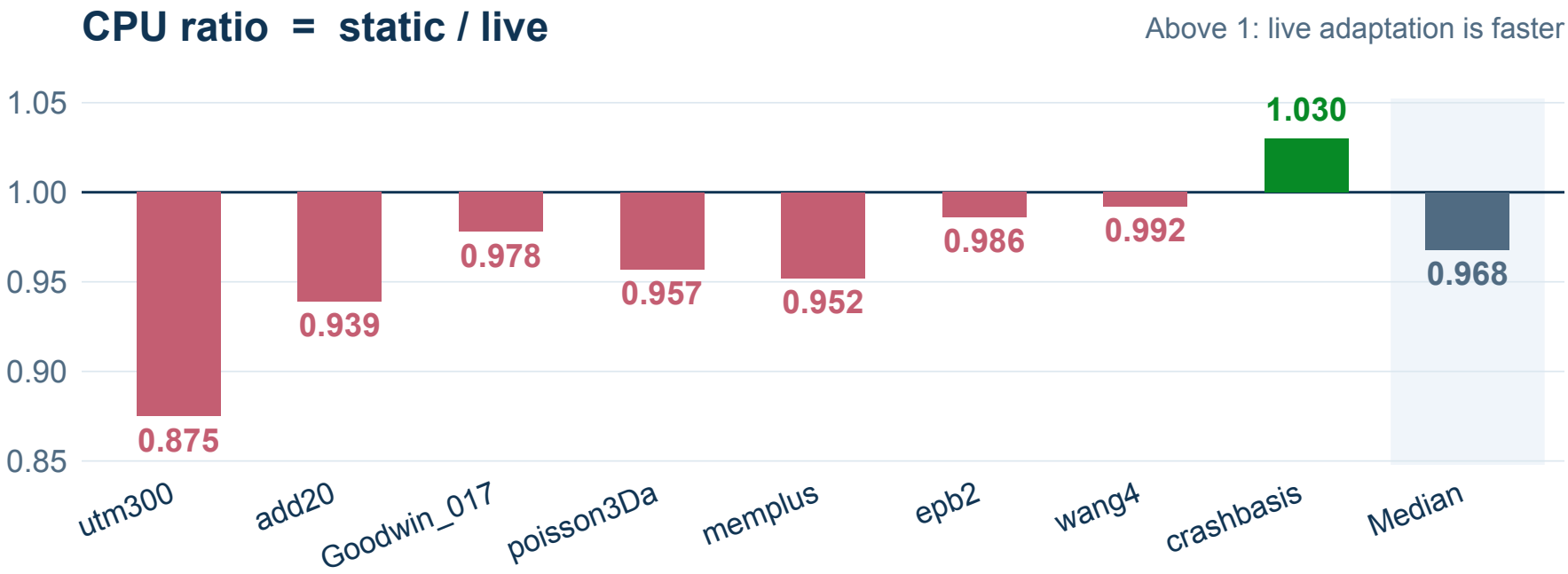


Figure 3: Prescribed-start rescue experiment. Ratios above one favour the live run over the static run from the same triplet.

- 2744 paired starts; transitions occur in about 25% of runs.
- Median static/live $= 0.968$: overhead is visible when the initial triplet is already acceptable.
- Clear rescue on `crashbasis`: static/live $= 1.030$, with live runs faster in 69.1% of prescribed starts.
- Interpretation: the controller is a safeguard. It is not expected to win when the prescribed triplet is already good.

Experiment I Is a Paired CPU Test

Matrix	Starts	Transitions	Static/live	Live wins
utm300	343	1.7%	0.875	3.8%
add20	343	30.9%	0.939	8.7%
Goodwin_017	343	5.2%	0.978	15.5%
poisson3Da	343	48.7%	0.957	2.9%
memplus	343	63.6%	0.952	19.5%
epb2	343	4.7%	0.986	28.3%
wang4	343	19.5%	0.992	35.9%
crashbasis	343	25.9%	**1.030**	**69.1%**
Eight-matrix median	343	22.7%	0.968	17.5%

Table 2: All eight matrices are tested from the same 343 prescribed triplets. Static/live > 1 means that adaptation is faster; all CPU costs and the 10^{-10} stopping rule are included.

- The median ratio is 0.968: monitoring overhead dominates on many acceptable starts.
- `crashbasis` is the clear matrix-level rescue case (1.030, with 69.1% live wins); isolated starts on other matrices can gain more.
- The defensible claim is reduced sensitivity to poor triplets, not uniform acceleration over the corresponding static runs.

Experiment II Tests the Harder Parameter-Free Interface

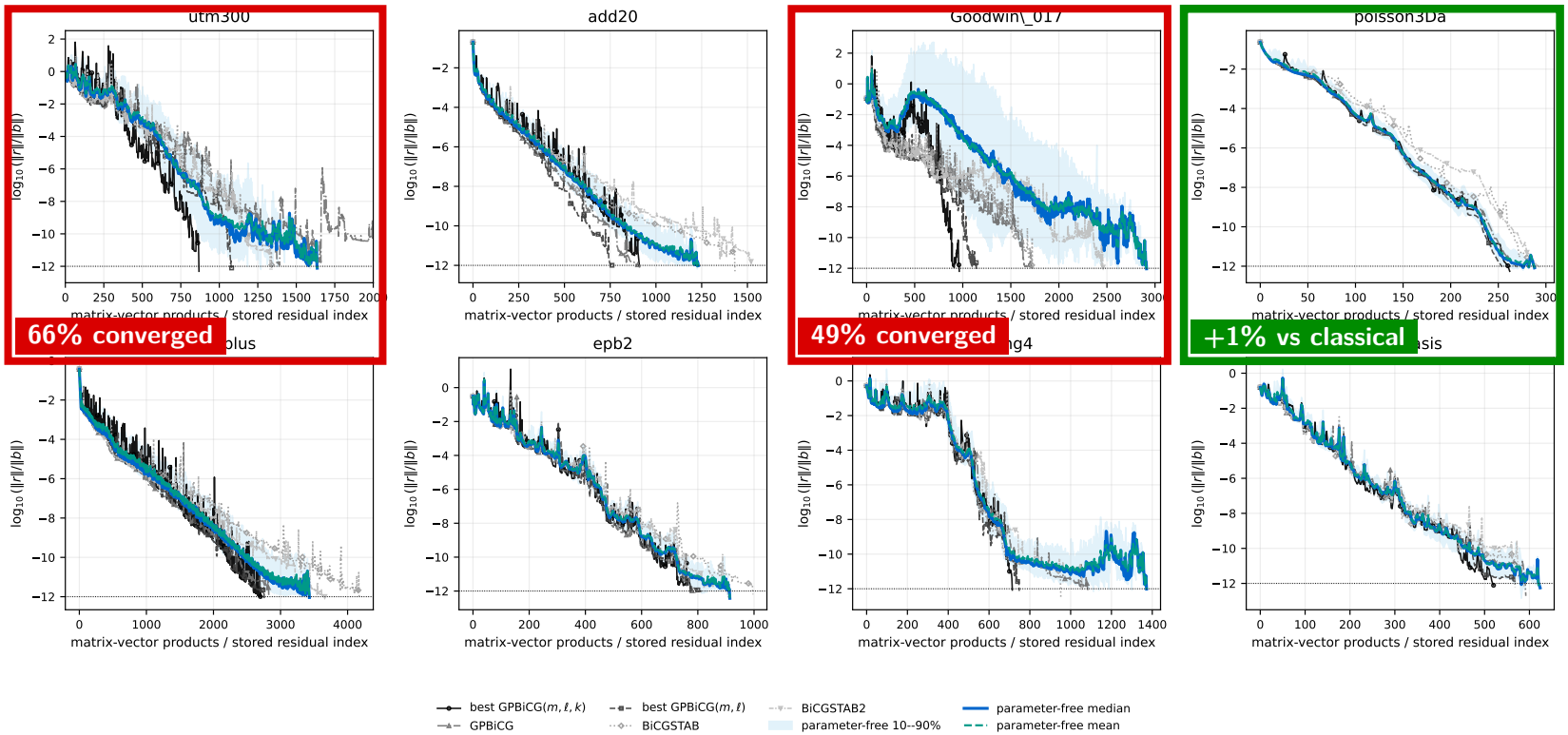


Figure 4: Fully parameter-free experiment: median over 100 random starts and 10–90% residual envelope.

- Over 800 random-start runs, the parameter-free variant converges in 88% of cases.
- It beats the best no-tuning classical baseline only on `poisson3Da`; on six matrices the mean CPU time stays within roughly 16%.
- Wide envelopes on difficult cases show that pure random initialization does not fully remove parameter sensitivity.
- This motivates a structured warm-up portfolio rather than a single blind initial triplet.

Experiment II Quantifies the Price of Blind Starts

Matrix	Conv.	PF mean (s)	Best MLK/PF	Best classical/PF	Classical winner
utm300	66%	0.0051	0.63	0.95	BiCGSTAB
add20	100%	0.0315	0.76	0.87	GPBiCG
Goodwin_017	49%	0.1605	0.53	0.85	BiCGSTAB
poisson3Da	100%	0.0885	0.93	**1.01**	BiCGSTAB
memplus	97%	0.6646	0.83	0.96	GPBiCG
epb2	99%	0.2277	0.88	0.90	BiCGSTAB
wang4	93%	0.3000	0.67	0.67	BiCGSTAB
crashbasis	100%	1.1870	0.80	0.84	BiCGSTAB

Table 3: Fully parameter-free (PF) experiment. Best MLK is the offline oracle from the completed 7^3 grid; the classical reference is the fastest of GPBiCG, BiCGSTAB and BiCGSTAB2. Each ratio is comparator CPU divided by PF mean CPU, so values above one favour PF.

- Best MLK/PF measures the price of removing triplet tuning: its median ratio is 0.78, so the tuned offline oracle remains faster on all matrices.
- Best classical/PF is the strict no-tuning comparison, with the classical baseline selected independently on every matrix.
- PF wins that test on `poisson3Da`; its median ratio is $\simeq 0.885$, corresponding to a median CPU penalty of about 13–14%.
- Thus the result is reduced triplet sensitivity at a measurable cost, not superiority over the best possible tuned triplet.

What Is Faster, and Under Which Test?

Experiment I: rescue, 10^{-10}

- 8 matrices and all $7^3 = 343$ prescribed starts.
- Static and live runs begin from the same triplet.
- Median static/live $= 0.968$, so monitoring overhead remains visible.
- On `crashbasis`, static/live $= 1.030$ and live wins 69.1% of starts.

Experiment II: no triplet, 10^{-12}

- 100 seeded random starts per matrix.
- 88% convergence over 800 runs.
- Faster than GPBiCG on $5/8$ matrices and BiCGSTAB2 on $7/8$.
- Beats the best no-tuning baseline only on `poisson3Da`; median CPU penalty $\simeq 13$–14%.

Interpretation

Rescue from a prescribed poor start is demonstrated; full removal of the initial triplet still carries a measurable CPU cost.

Adaptive Results and Remaining Limitations

- GPBiCG(m, ℓ, k) is a useful extension because it gives a richer residual trajectory.
- The price is parameter sensitivity:
  - the best triplet depends on the matrix,
  - poor triplets can be inefficient.
- Live adaptation can rescue some poor starts and reduce sensitivity to the initial triplet.
- Full parameter freedom is promising but not solved:
  - the random-start model does not uniformly dominate classical baselines,
  - a structured warm-up portfolio is the next natural step.

V

Conclusions and perspectives

Conclusions and Perspectives

- **Method contribution:** GPBiCG(m, ℓ, k) adds a controlled LOPBiCG phase to the GPBiCG(m, ℓ) cycle.
- **Numerical contribution:** the extra phase can improve difficult unpreconditioned cases, but the best triplet remains matrix-dependent.
- **Adaptive contribution:** live information can rescue some poor prescribed starts; fully parameter-free random starts are promising but not yet uniformly superior.
- **Perspective from MoonSolve/Kakeshi:** replace blind tuning by a small portfolio, short observation, select/continue/eliminate/abstain decisions, and end-to-end CPU accounting.

Backup: CPU Speedups without Preconditioning

Matrix	CPU MLK	CPU ML	CPU GPBiCG	ML/MLK	GPBiCG/MLK
utm300	0.0032	0.0038	0.0403†	1.19	–
add20	0.0240	0.0222	0.0273	0.93	1.14
Goodwin_017	0.0846	0.0974	0.1534	1.15	1.81
poisson3Da	0.0819	0.0843	0.0964	1.03	1.18
memplus	0.5491	0.5769	0.6410	1.05	1.17
epb2	0.2009	0.2124	0.2322	1.06	1.16
wang4	0.2009	0.2112	0.3285	1.05	1.64
Crashbasis	0.9497	1.1351	1.4206	1.20	1.50

Times are CPU seconds at true relative residual tolerance 10^{-12}. Ratios larger than one favor GPBiCG(m, ℓ, k). The dagger marks a non-converged diagnostic GPBiCG row and is excluded from its speedup. The blue-framed trajectories in the main talk correspond to the largest MLK-over-ML gains in this table.

Backup: CPU Speedups with ILU(0)

Matrix	Best MLK	Best ML	CPU MLK	CPU ML	vs GPBiCG	vs ML
utm300	$(7, 5, 2)^{\dagger}$	$(3, 4)^{\dagger}$	–	–	–	–
memplus	$(3, 5, 4)$	$(7, 3)$	0.188	0.197	1.37	1.05
add20	$(2, 3, 1)$	$(2, 4)$	0.0136	0.0133	1.13	0.982
epb2	$(3, 3, 3)$	$(7, 4)$	0.0304	0.0305	1.20	1.00
Goodwin_017	$(7, 7, 3)$	$(2, 5)$	0.0647	0.0587	–	0.907
wang4	$(4, 5, 6)$	$(6, 3)$	0.0469	0.0497	1.23	1.06
poisson3Da	$(6, 5, 5)$	$(4, 3)$	0.0673	0.0710	1.21	1.05
Crashbasis	$(7, 7, 2)$	$(7, 7)$	0.195	0.181	1.21	0.931

Values larger than one favor GPBiCG(m, ℓ, k). Dagger rows are non-converged diagnostics and are excluded. The comparison against GPBiCG(m, ℓ) is mixed after ILU(0): three wins, one near tie, and three losses on converged comparable rows.

Backup: Mathematical Definitions of the Switch Rule

For $p \in \{\mathrm{B}, \mathrm{G}, \mathrm{L}\}$, let $n_{q,p}$ be the number of iterations in phase p during cycle q. Denote its entry and exit residuals by $\boldsymbol{r}_{q,p}^{\mathrm{in}}$ and $\boldsymbol{r}_{q,p}^{\mathrm{out}}$. The phase score is

$$\rho_{q,p} = \left(\frac{\|\boldsymbol{r}_{q,p}^{\mathrm{out}}\|_2}{\|\boldsymbol{r}_{q,p}^{\mathrm{in}}\|_2} \right)^{1/n_{q,p}}.$$

This equals the geometric mean of the consecutive residual ratios, because all intermediate norms cancel. Smaller is better. Define

$$p_b = \arg\min_p \rho_{q,p}, \qquad p_w = \arg\max_p \rho_{q,p}, \qquad \theta' = \theta_q + e_{p_b} - e_{p_w}.$$

The complete-cycle ratio is

$$g_q = \frac{\|\boldsymbol{r}_q^{\mathrm{out}}\|_2}{\|\boldsymbol{r}_q^{\mathrm{in}}\|_2}.$$

The rescue rule accepts only if $g_q < 1$, $\max_p \rho_{q,p} < 1$, $p_b \neq p_w$, and $\theta' \in \Theta$. After a switch, adaptation reopens only after two productive cycles. For the random-start experiment, the same θ' is further filtered by

$$\Delta W_q(\theta') > C_q H_q(\theta'),$$

where ΔW_q is predicted saved work, H_q is transition overhead, and C_q penalizes rough or weakly separated phase signals.

Backup: Confidence Gate Used in Random Starts

Convert phase scores into smoothed log-contractions:

$$\lambda_{q,p} = \max(0, -\log \rho_{q,p}), \qquad \widehat{\lambda}_{q,p} \leftarrow (1-\alpha)\widehat{\lambda}_{q-1,p} + \alpha\lambda_{q,p}, \quad \alpha = 0.30.$$

For $\theta = (m, \ell, k)$, define

$$\mu_q(\theta) = \frac{m\widehat{\lambda}_{q,\mathrm{B}} + \ell\widehat{\lambda}_{q,\mathrm{G}} + k\widehat{\lambda}_{q,\mathrm{L}}}{m + \ell + k}.$$

If $L_q = \max(0, \log(r_q^{\mathrm{rel}}/\varepsilon))$, then

$$\Delta W_q(\theta') = \frac{L_q}{\mu_q(\theta_q)} - \frac{L_q}{\mu_q(\theta')}, \quad H_q = |\operatorname{len}(\theta') - \operatorname{len}(\theta_q)| + 1.$$

The random-start experiment accepts the local move only if

$$\Delta W_q(\theta') > C_q H_q,$$

with

$$C_q = 1 + \min(2, R_q) + \min\left(1, \frac{0.25}{\max(\sigma_q, 0.05)}\right), \quad \sigma_q = \log \frac{\rho_{q,p_w}}{\rho_{q,p_b}}.$$

Here R_q is the smoothed residual-roughness indicator.

Backup: One Adaptive Decision in Pseudo-Code

1. Start cycle q with fixed triplet $\theta_q = (m_q, \ell_q, k_q)$.
2. Run the complete GPBiCG(m_q, ℓ_q, k_q) cycle and store residual ratios for the three phases.
3. Compute $\rho_{q,\mathrm{B}}$, $\rho_{q,\mathrm{G}}$, $\rho_{q,\mathrm{L}}$, and the complete-cycle ratio g_q.
4. If the cycle is not productive, keep $\theta_{q+1} = \theta_q$.
5. Otherwise compute $p_b = \arg\min \rho_{q,p}$ and $p_w = \arg\max \rho_{q,p}$.
6. Form $\theta' = \theta_q + e_{p_b} - e_{p_w}$. If it is outside $\Theta = \{1, \ldots, 7\}^3$, keep θ_q.
7. Experiment I accepts this feasible move and locks the new triplet. Experiment II accepts it only if the confidence break-even gate also predicts a net saved work.

Backup: Reproducibility of the Adaptive Campaign

- Same admissible grid throughout:

$$\Theta = \{1, \ldots, 7\}^3.$$

- **Experiment I: prescribed-start rescue.** All 343 triplets are tested per matrix; static and live runs start from the same triplet; the paired stopping rule is $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2 / \|\boldsymbol{b}\|_2 \leq 10^{-10}$.
- **Experiment II: fully parameter-free interface.** There are 100 fixed uniformly sampled starts per matrix; the stopping rule is 10^{-12}; non-converged starts remain in the convergence rate and CPU averages up to the budget.
- Experiment I uses the same phase-score rescue rule on every matrix: productive-cycle test, local best-minus-worst move, and two-cycle lock after a switch.
- Experiment II uses the same confidence break-even gate for all 100 random starts and all matrices. Nothing is tuned case by case.
- CPU time includes monitoring, score updates, rejected decisions, accepted switches, and the consequences of bad switches.

Complete GPBiCG algorithm (1/2)

Initialization

Choose $\boldsymbol{x}_0$ and set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$.

Choose $\boldsymbol{r}_0^*$ with $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, e.g. $\boldsymbol{r}_0^* = \boldsymbol{r}_0$.

Set $\boldsymbol{t}_{-1} = \boldsymbol{w}_{-1} = 0$ and $\beta_{-1} = 0$.

For $n = 0, 1, \ldots$ **until** $\|\boldsymbol{r}_n\| \leq \varepsilon \|\boldsymbol{b}\|$

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

$$\alpha_n = \frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}{(\boldsymbol{r}_0^*, A\boldsymbol{p}_n)},$$

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

$$\boldsymbol{t}_n = \boldsymbol{r}_n - \alpha_n A\boldsymbol{p}_n.$$

For $n = 0$: $\quad \zeta_n = \dfrac{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)}, \qquad \eta_n = 0.$

S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

Complete GPBiCG algorithm (2/2)

Coefficient choice

For $n > 0$, write $\boldsymbol{v}_n = A\boldsymbol{t}_n$ and

$$\Delta_n = (\boldsymbol{v}_n, \boldsymbol{v}_n)(\boldsymbol{y}_n, \boldsymbol{y}_n) - (\boldsymbol{y}_n, \boldsymbol{v}_n)(\boldsymbol{v}_n, \boldsymbol{y}_n).$$

$$\zeta_n = \frac{(\boldsymbol{y}_n, \boldsymbol{y}_n)(\boldsymbol{v}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{t}_n)(\boldsymbol{v}_n, \boldsymbol{y}_n)}{\Delta_n},$$

$$\eta_n = \frac{(\boldsymbol{v}_n, \boldsymbol{v}_n)(\boldsymbol{y}_n, \boldsymbol{t}_n) - (\boldsymbol{y}_n, \boldsymbol{v}_n)(\boldsymbol{v}_n, \boldsymbol{t}_n)}{\Delta_n}.$$

Update the solution, residual and recurrence

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

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

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n + \boldsymbol{z}_n,$$

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

$$\beta_n = \frac{\alpha_n}{\zeta_n}\frac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{n+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_n)}, \qquad \boldsymbol{w}_n = A\boldsymbol{t}_n + \beta_n A\boldsymbol{p}_n.$$

S.-L. Zhang, *SIAM J. Sci. Comput.*, 18(2), 537–551, 1997.

Complete GPBiCG(m, ℓ, k) algorithm

Keep the initialization and all recurrence updates from Backups 7–8.

At each iteration, use $s_n = n \bmod (m + \ell + k)$ to select (ζ_n, η_n).

$0 \leq s_n < m$: BiCGSTAB phase

$$\zeta_n = \frac{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, A\boldsymbol{t}_n)}, \qquad \eta_n = 0.$$

$m \leq s_n < m + \ell$: GPBiCG phase

Use the two explicit GPBiCG coefficients on Backup 8, with $\boldsymbol{v}_n = A\boldsymbol{t}_n$ and the common denominator Δ_n.

$m + \ell \leq s_n < m + \ell + k$: LOPBiCG phase

$$\zeta_n = \frac{(\boldsymbol{t}_n, \boldsymbol{t}_n)}{(A\boldsymbol{t}_n, \boldsymbol{t}_n)}, \qquad \eta_n = 0.$$

Associated article, Algorithm 1, p. 5. GPBiCG coefficients: Zhang (1997).