Design and analysis of high performance conjugate gradient and Lanczos type algorithms

Tyler Chen

February 21, 2020

Follow along

slides

chen.pw/d/f5b

code

chen.pw/d/i7g

proposal

chen.pw/d/2mn

Acknowledgements

We are gathered on the unceded land of the Coast Salish people, and in particular of the Duwamish Tribe.

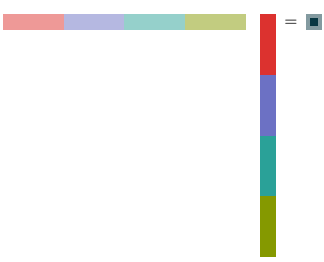
Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

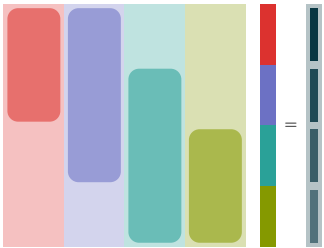
Introduction

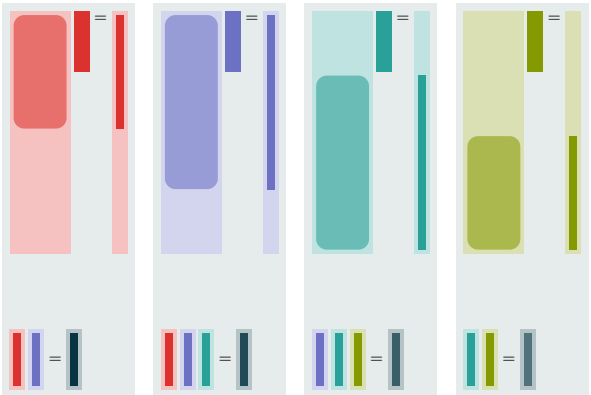
- Solving linear systems and computing eigenvalues (and matrix functions) are among the most common tasks in computational science
  - CG and Lanczos are popular when the matrix is symmetric (postive definite)
- Modern supercomputers have reached exascale (10^{18} flops)
  - on large machines the cost of reading and moving data dominates the cost of floating point operations
    - inner products and dense matrix vector products require global communication
    - sparse matrix vector products require local communication
    - vector updates require no communication
  - Krylov subspace methods can only reach a fraction theoretical flop rate because of communication costs
- Need to address communication costs, while considering numerical properties!

Distributed inner product



Distributed (sparse) matrix product





Conjugate gradient

- Conjugate gradient (CG) is used to solve a linear system $\mathbf{Ax} = \mathbf{b}$ when $\mathbf{A}$ is symmetric positive definite
- CG has low storage and floating point operation costs
  - one matrix vector product, two inner products, and a few vector updates each iteration
- There is some theory about the convergence of CG in finite precision, but many open questions remain

Conjugate gradient

- At each step, CG generates iterate $\mathbf{x}_k$ which minimizes the $\mathbf{A}$-norm of the error over,

$$\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\}$$

- Builds up an $\mathbf{A}$-orthogonal basis of "search directions" $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots$ and minimizes independently along each

$$\langle \mathbf{e}_k, \mathbf{p}_j \rangle_{\mathbf{A}} = \langle \mathbf{A}^{-1}\mathbf{b} - \mathbf{x}_k, \mathbf{p}_j \rangle_{\mathbf{A}} = \langle \mathbf{b} - \mathbf{A}\mathbf{x}_k, \mathbf{p}_j \rangle = \langle \mathbf{r}_k, \mathbf{p}_j \rangle$$

- We don't need to store all these search vectors because of three term Lanczos recurrence!

Conjugate gradient

Algorithm 1 Hestenes and Stiefel Conjugate Gradient

1: **procedure** HS-CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$)
2: initialize()
3: **for** $k = 1, 2, \ldots$ **do**
4: $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
5: $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$
6: $\nu_k = \langle \mathbf{r}_k, \mathbf{r}_k \rangle$
7: $\beta_k = \nu_k / \nu_{k-1}$
8: $\mathbf{p}_k = \mathbf{r}_k + \beta_k \mathbf{p}_{k-1}$
9: $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
10: $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle$
11: $\alpha_k = \nu_k / \mu_k$
12: **end for**
13: **end procedure**

Finite precision conjugate gradient

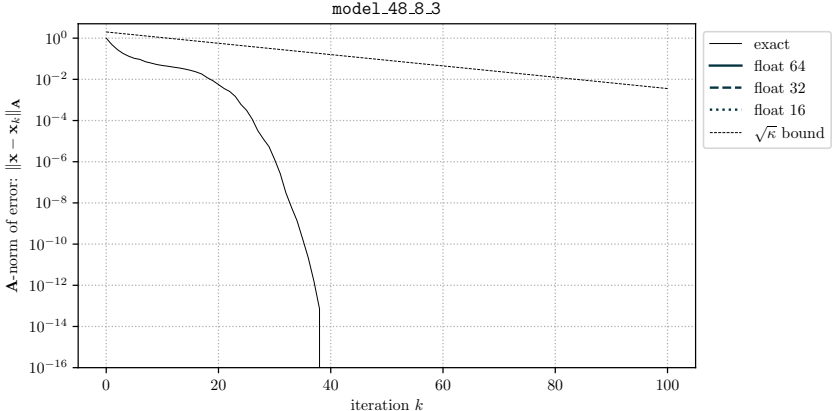


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

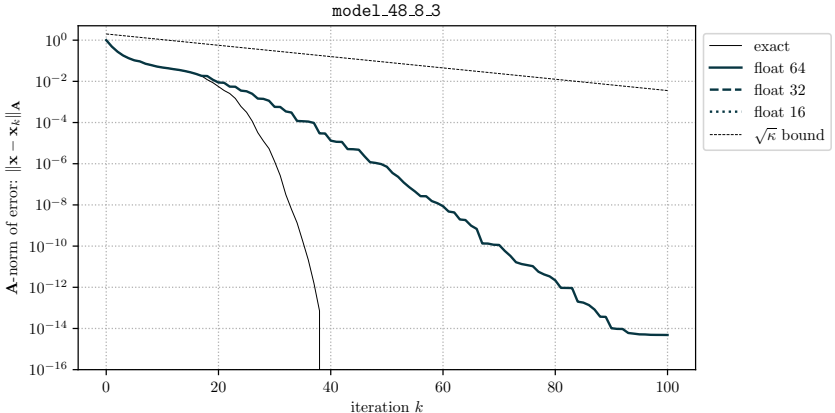


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

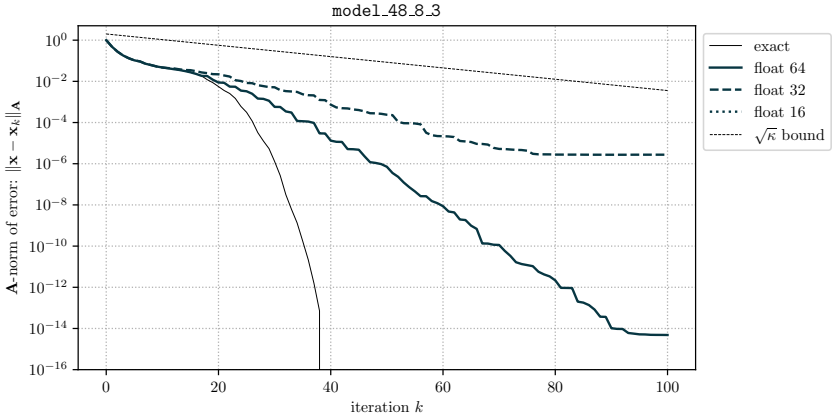


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

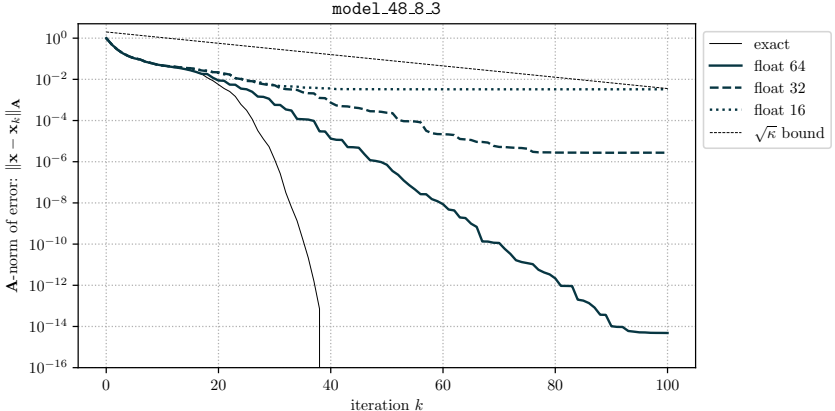


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

- In finite precision orthogonality is lost, so induction based arguments for optimality of iterates no longer hold
- The primary effects are:
  - delay of convergence
  - loss of ultimately attainable accuracy

Lanczos

- Lanczos algorithm used to compute orthogonal basis $\mathbf{q}_0, \mathbf{q}_1, \ldots$for Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{q}_0)$ when $\mathbf{A}$ is symmetric
- Lanczos has similar computational advantages and disadvantages as CG
- Compute using a three term recurrence

$$\mathbf{A}\mathbf{q}_j = \gamma_{j-1}\mathbf{q}_{j-1} + \delta_j\mathbf{q}_j + \gamma_j\mathbf{q}_{j+1}.$$

We can write this recurrence in matrix form as

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{T}_k + \gamma_k\mathbf{q}_{k+1}\mathbf{e}_k^\top.$$

Lanczos

- The Lanczos decomposition gives information about the spectrum of $\mathbf{A}$
- Let $\mathbf{T}_k = \mathbf{S}_k \Theta_k \mathbf{S}_k^\top$. Then Ritz vectors and values $\mathbf{y}_i := \mathbf{Q}_k \mathbf{s}_i, \theta_i$ approximate eigenvectors and values of $\mathbf{A}$ in the following sense,

$$\mathbf{A}\mathbf{y}_i - \theta_i \mathbf{y}_i = \gamma_k \mathbf{q}_{k+1} \mathbf{e}_k^\top \mathbf{s}_i$$

- If $\deg p < k$ then $p(\mathbf{A})\mathbf{b} = \mathbf{Q}_k p(\mathbf{T}_k)\mathbf{Q}_k^\top \mathbf{b}$, so may hope to approximate matrix functions[1] by,

$$f(\mathbf{A})\mathbf{b} \approx \mathbf{Q}_k f(\mathbf{T}_k)\mathbf{Q}_k^\top \mathbf{b}$$

[1] $f(\mathbf{A}) = \mathbf{U} f(\mathbf{\Lambda}) \mathbf{U}^\top$ where $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and $f(\mathbf{\Lambda}) = \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_n))$

Relationship between CG and Lanczos

- CG and Lanczos are equivalent; i.e. can be derived from one another
- CG from Lanczos: iterates given by $\mathbf{x}_k = \mathbf{Q}_k \mathbf{T}_k^{-1} \mathbf{Q}_k^\top \mathbf{b}$
- Lanczos from CG: the columns of $\mathbf{Q}_k$ are $\mathbf{q}_{j+1} = (-1)^j \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}$ and the diagonal and the off-diagonal entries of $\mathbf{T}_k$ are respectively given by

$$\delta_j = \left(\frac{1}{\alpha_{j-1}} - \frac{\beta_{j-1}}{\alpha_{j-2}} \right), \qquad \gamma_j = \frac{1}{\alpha_{j-1}} \frac{\|\mathbf{r}_j\|}{\|\mathbf{r}_{j-1}\|}.$$

Hiding communication

- In each iteration we would like to hide communication by computing all inner products and matrix vector products simultaneously
- Do this by finding mathematically equivalent expressions for an inner product using recurrences
  - The new expressions are not equivalent in finite precision; i.e. the order of things has changed
- Various approaches have been studied[2]
  - Typically maintain two inner products and one matrix vector product per iteration

[2] Saad 1985; Meurant 1987; Saad 1989; Chronopoulos and Gear 1989; Ghysels and Vanroose 2014; Cornelis, Cools, and Vanroose 2019.

Hiding communication

- CG is particularly sensitive to rounding errors
- Some communication hiding variants have very different behavior in finite precision[3]
  - Various approaches to try to deal with this; e.g. residual replacement[4], basis vector shifts[5], etc.

[3] Carson et al. 2018.

[4] Cools et al. 2018.

[5] Cornelis, Cools, and Vanroose 2019.

Hiding communication

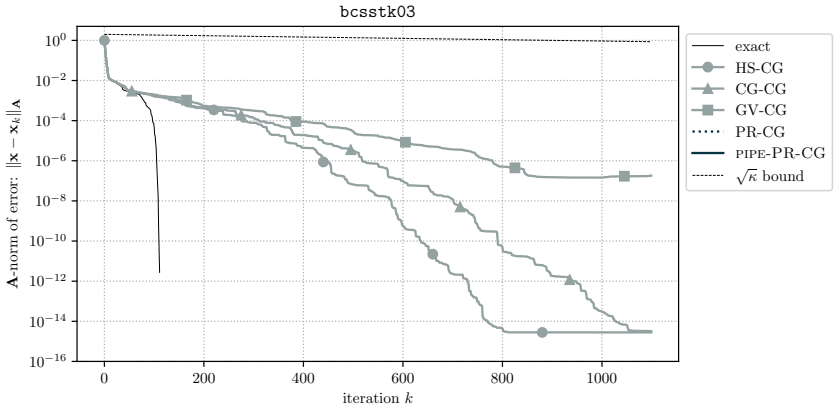
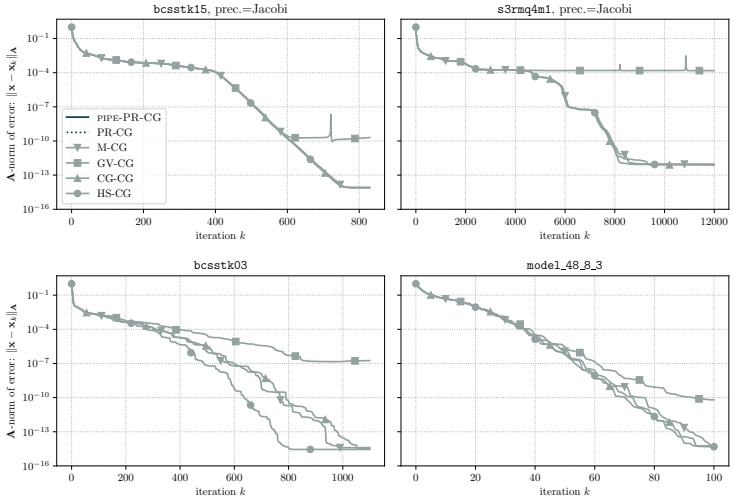


Figure: Convergence of finite precision conjugate gradient

Hiding communication



Finite precision computations

We assume the standard model of floating point arithmetic,

$$|\operatorname{fp}(\alpha \circ \beta) - \alpha \circ \beta| \leq \epsilon |\alpha \circ \beta| \tag{1}$$

for floating point numbers α and β and standard operations $\circ \in \{+, -, \times, \div\}$, where ϵ is the unit roundoff of the machine. From this, to first order,

$$\|\operatorname{fp}(\mathbf{x} + \alpha\mathbf{y}) - (\mathbf{x} + \alpha\mathbf{y})\| \leq \epsilon \left(\|\mathbf{x}\| + 2|\alpha|\|\mathbf{y}\|\right) \tag{2}$$

$$\|\operatorname{fp}(\mathbf{A}\mathbf{x}) - \mathbf{A}\mathbf{x}\| \leq \epsilon\, c\, \|\mathbf{A}\|\|\mathbf{x}\| \tag{3}$$

$$\|\operatorname{fp}(\langle \mathbf{x}, \mathbf{y} \rangle) - \langle \mathbf{x}, \mathbf{y} \rangle\| \leq \epsilon\, n\, \|\mathbf{x}\|\|\mathbf{y}\| \tag{4}$$

where c is a dimensional constant depending on the method of matrix multiplication and n is the length of the vectors $\mathbf{x}$ and $\mathbf{y}$.

Loss of Accuracy

- Most analysis uses residual gap: $\Delta_{\mathbf{r}_k} := (\mathbf{b} - \mathbf{A}\mathbf{x}_k) - \mathbf{r}_k$
  - if $\mathbf{r}_k \to 0$, then $\|\Delta_{\mathbf{r}_k}\| \to \|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$
- Expression for HS-CG involves simple sum of local errors[6]

$$\Delta_{\mathbf{r}_k} = \Delta_{\mathbf{r}_0} + \sum_{j=1}^{k} (\delta_{\mathbf{r}_j} + \mathbf{A}\delta_{\mathbf{x}_k})$$

  where $\delta_{\mathbf{r}_j}$ and $\delta_{\mathbf{x}_j}$ are roundoff terms from computing $\mathbf{r}_j$ and $\mathbf{x}_j$ in finite precision.
- Similar analyses for CG-CG, GV-CG and other pipelined conjugate gradient variants[7]
  - CG-CG similar to HS-CG but GV-CG amplfies rounding errors

[6] Greenbaum 1989; Greenbaum 1997.

[7] Cools et al. 2018; Carson et al. 2018.

Perturbed Lanczos (Greenbaum 1989)

- In finite precision we can still define $\mathbf{q}_{j+1} = (-1)^j \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}$. Then we will have perturbed three term recurrence,

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{T}_k + \gamma_k\mathbf{q}_{k+1}\mathbf{e}_k^\top + \mathbf{F}_k.$$

- If $\langle \mathbf{q}_j, \mathbf{q}_{j-1} \rangle \approx 0$ and $\|\mathbf{f}_j\| \approx 0$, then there exists a larger matrix $\bar{\mathbf{A}}$ so that
  - exact Lanczos applied to $\bar{\mathbf{A}}$ for k steps will produce $\mathbf{T}_k$
  - each eigenvalue of $\bar{\mathbf{A}}$ is (very) near to some eigenvalue of $\mathbf{A}$
- This provides a "backwards" type analysis for Lanczos and CG

Delay of convergence

- A “good” finite precision CG implementation converges like exact CG on a larger matrix whose eigenvalues are clustered near the eigenvalues of $\mathbf{A}$
  - this provides a means by which exact arithmetic theory can be applied to finite precision computations
- e.g. a “good” variant will satisfy a minimax bound of the form,

$$\frac{\|\mathbf{e}_k\|_{\mathbf{A}}}{\|\mathbf{e}_0\|_{\mathbf{A}}} \leq \min_{p \in \mathcal{P}_k} \left[\max_{z \in \mathcal{L}(\mathbf{A})} |p(z)| \right], \qquad \mathcal{L}(\mathbf{A}) = \bigcup_{i=1}^{n} [\lambda_i - \delta, \lambda_i + \delta].$$

- Caveat: it remains to be proved that any commonly used CG implementations are “good”

Delay of convergence

- Analysis in Greenbaum, Liu, and Chen 2019 of three term recurrence error for different variants provides some insight into why different rates of convergence are observed on some problems but not others.
- Derive expressions for $\mathbf{f}_k$ in terms of roundoff errors
  - HS-CG and CG-CG depend only on local errors, but GV-CG depends on errors made at all previous steps!
- If the previous minimax bound is particularly sensitive to the interval size (e.g. `bcsstk03`) then you see different rates of convergence

Predict-and-recompute variants

- **Idea**: use recursively updated quantities as a predictor for their true values to allow iteration to continue, then recompute them directly at a later point in the iteration
- If this is done in a smart way, it won't affect the communication structure of the algorithm
- e.g. may try to use $\mathbf{r}_k \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_k$ to keep residual gap small, but this might
  - introduce large (relative) errors: $\|(\mathbf{b} - \mathbf{A}\mathbf{x}_k) - \mathrm{fp}(\mathbf{b} - \mathbf{A}\mathbf{x}_k)\| \leq \epsilon(\|\mathbf{b}\| - 2c\|\mathbf{A}\|\|\mathbf{x}_k\|)$
  - cause error three term Lanczos recurrence leading to delay of convergence

Predict-and-recompute variants

Algorithm 2 Hestenes and Stiefel Conjugate Gradient

1: **procedure** HS-CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$)
2: initialize()
3: **for** $k = 1, 2, \ldots$ **do**
4: $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
5: $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$
6: $\nu_k = \langle \mathbf{r}_k, \mathbf{r}_k \rangle$
7: $\beta_k = \nu_k / \nu_{k-1}$
8: $\mathbf{p}_k = \mathbf{r}_k + \beta_k \mathbf{p}_{k-1}$
9: $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
10: $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle$
11: $\alpha_k = \nu_k / \mu_k$
12: **end for**
13: **end procedure**

Predict-and-recompute variants

Algorithm 3 Predict-and-recompute conjugate gradient

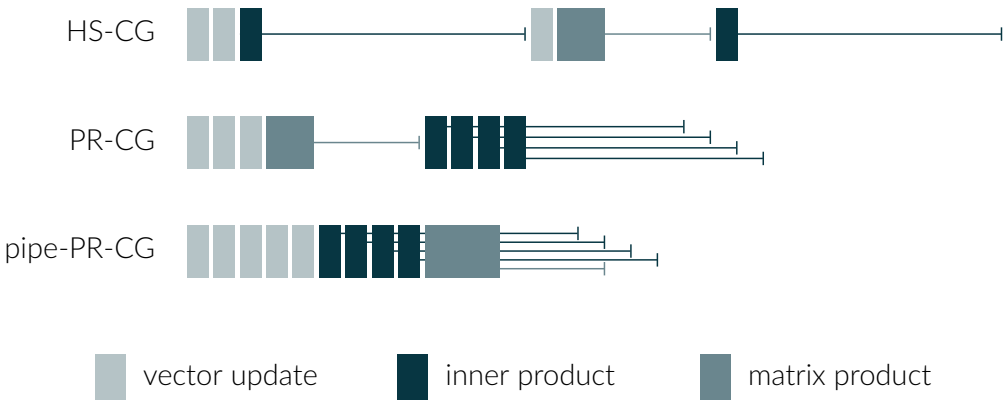
1: **procedure** PR-CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$)
2: initialize()
3: **for** $k = 1, 2, \ldots$ **do**
4: $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
5: $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$
6: $\nu'_k = \nu_{k-1} - 2\alpha_{k-1}\delta_{k-1} + \alpha^2_{k-1}\gamma_{k-1}$
7: $\beta_k = \nu'_k/\nu_{k-1}$
8: $\mathbf{p}_k = \mathbf{r}_k + \beta_k\mathbf{p}_{k-1}$
9: $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
10: $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle,\ \delta_k = \langle \mathbf{r}_k, \mathbf{s}_k \rangle,\ \gamma_k = \langle \mathbf{s}_k, \mathbf{s}_\mathbf{k} \rangle,\ \nu_k = \langle \mathbf{r}_k, \mathbf{r}_k \rangle$
11: $\alpha_k = \nu_k/\mu_k$
12: **end for**
13: **end procedure**

Predict-and-recompute variants

Algorithm 4 Pipelined predict-and-recompute conjugate gradient

1: **procedure** pipe-PR-CG($\mathbf{A}$, $\mathbf{b}$, $\mathbf{x}_0$)
2: initialize()
3: **for** $k = 1, 2, \ldots$ **do**
4: $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
5: $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$
6: $\mathbf{w}'_k = \mathbf{w}_{k-1} - \alpha_{k-1}\mathbf{u}_{k-1}$
7: $\nu'_k = \nu_{k-1} - 2\alpha_{k-1}\delta_{k-1} + \alpha^2_{k-1}\gamma_{k-1}$
8: $\beta_k = \nu'_k/\nu_{k-1}$
9: $\mathbf{p}_k = \mathbf{r}_k + \beta_k\mathbf{p}_{k-1}$
10: $\mathbf{s}_k = \mathbf{w}'_k + \beta_k\mathbf{s}_{k-1}$
11: $\mathbf{u}_k = \mathbf{A}\mathbf{s}_k$
12: $\mathbf{w}_k = \mathbf{A}\mathbf{r}_k$
13: $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle,\ \delta_k = \langle \mathbf{r}_k, \mathbf{s}_k \rangle,\ \gamma_k = \langle \mathbf{s}_k, \mathbf{s}_k \rangle,\ \nu_k = \langle \mathbf{r}_k, \mathbf{r}_k \rangle$
14: $\alpha_k = \nu_k/\mu_k$
15: **end for**
16: **end procedure**

Predict-and-recompute variants

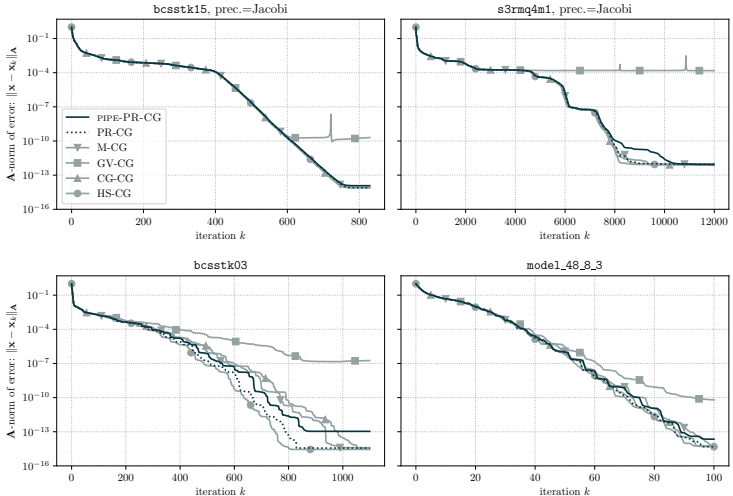


Predict-and-recompute variants

variant	mem.	vec.	scal.	time
HS-CG	4 (+1)	3 (+0)	2	$2\,\mathrm{C_{gr}} + \mathrm{T_{mv}} + \mathrm{C_{mv}}$
CG-CG	5 (+1)	4 (+0)	2	$\mathrm{C_{gr}} + \mathrm{T_{mv}} + \mathrm{C_{mv}}$
M-CG	4 (+2)	3 (+1)	3	$\mathrm{C_{gr}} + \mathrm{T_{mv}} + \mathrm{C_{mv}}$
PR-CG	4 (+2)	3 (+1)	4	$\mathrm{C_{gr}} + \mathrm{T_{mv}} + \mathrm{C_{mv}}$
GV-CG	7 (+3)	6 (+2)	2	$\max(\mathrm{C_{gr}}, \mathrm{T_{mv}} + \mathrm{C_{mv}})$
pipe-PR-M-CG	6 (+4)	5 (+3)	3	$\max(\mathrm{C_{gr}}, \mathrm{T_{2mv}} + \mathrm{C_{mv}})$
pipe-PR-CG	6 (+4)	5 (+3)	4	$\max(\mathrm{C_{gr}}, \mathrm{T_{2mv}} + \mathrm{C_{mv}})$

Table: Summary of costs for various conjugate gradient variants. Values in parenthesis are the additional costs for the preconditioned variants.

Predict-and-recompute variants



Predict-and-recompute variants

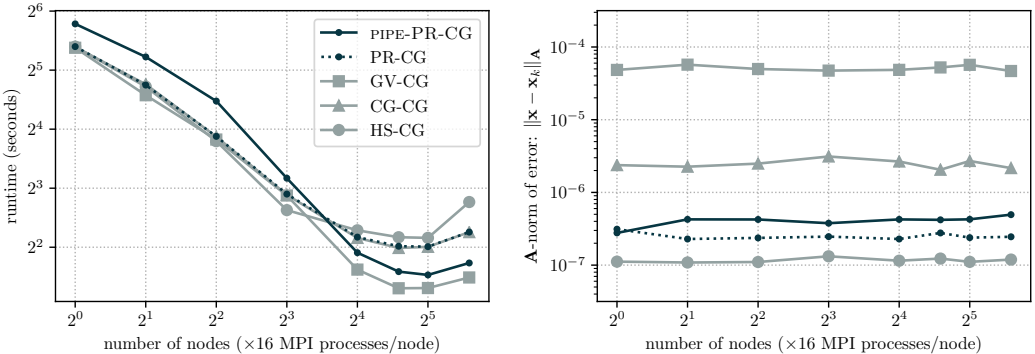


Figure: Convergence of conjugate gradient variants

Predict-and-recompute variants

- Analysis in Chen and Carson 2020 of the residual gap and three term Lanczos recurrence for PR-CG and pipe-PR-CG provides insight into improved convergence
- Practical use remains to be determined
- Included in PETSc v3.13: `-ksp_type pipeprcg`

Predict-and-recompute variants

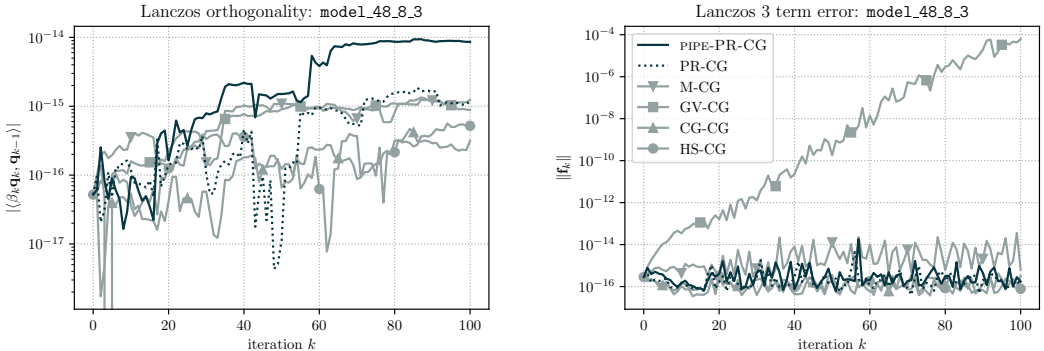


Figure: Error measures for perturbed Lanczos recurrences

Future work

- Try to incorporate predict-and-recompute idea into s-step methods
- Selective re-orthogonalization in low precision or high performance contexts
- Further numerical analysis of CG
  - when does $\mathbf{r}_k \to 0$?
  - when is $\langle \mathbf{r}_k, \mathbf{r}_{k-1} \rangle \approx 0$?
  - can we determine which problems will be “hard” ahead of time?
- Stronger error bounds for Lanczos used to compute matrix functions (for random or deterministic matrices)
- Better algorithms for computing matrix functions

Other projects

- What effect do rounding errors have on algorithms involving randomness?
- Can finite precision be viewed as an exact computation in some space with an algebraic structure?
- Can we develop algorithms meant to perform well in finite precision without starting with an algorithm which is meant to work well in exact precision?

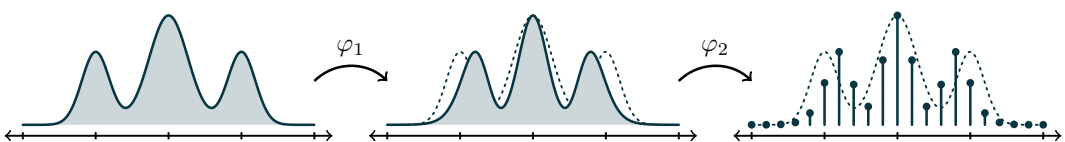


Figure: Discretizing random variable

References

Carson, Erin C. et al. (2018). "The Numerical Stability Analysis of Pipelined Conjugate Gradient Methods: Historical Context and Methodology". In: *SIAM Journal on Scientific Computing* 40.5, A3549–A3580.

Chen, Tyler and Erin C. Carson (2020). "Predict-and-recompute conjugate gradient variants". In:

Chronopoulos, A.T. and Charles William Gear (1989). "s-step iterative methods for symmetric linear systems". In: *Journal of Computational and Applied Mathematics* 25.2, pp. 153–168.

Cools, Siegfried et al. (Mar. 2018). "Analyzing the Effect of Local Rounding Error Propagation on the Maximal Attainable Accuracy of the Pipelined Conjugate Gradient Method". In: *SIAM Journal on Matrix Analysis and Applications* 39, pp. 426–450.

Cornelis, Jeffrey, Siegfried Cools, and Wim Vanroose (2019). *The Communication-Hiding Conjugate Gradient Method with Deep Pipelines*.

Ghysels, Pieter and Wim Vanroose (2014). "Hiding global synchronization latency in the preconditioned Conjugate Gradient algorithm". In: *Parallel Computing* 40.7, pp. 224–238.

Greenbaum, Anne (1989). "Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences". In: *Linear Algebra and its Applications* 113, pp. 7–63.

— (1997). "Estimating the Attainable Accuracy of Recursively Computed Residual Methods". In: *SIAM Journal on Matrix Analysis and Applications* 18.3, pp. 535–551.

Greenbaum, Anne, Hexuan Liu, and Tyler Chen (2019). *On the Convergence Rate of Variants of the Conjugate Gradient Algorithm in Finite Precision Arithmetic*.

Meurant, Gérard (1987). "Multitasking the conjugate gradient method on the CRAY X-MP/48". In: *Parallel Computing* 5.3, pp. 267–280.

Saad, Yousef (1985). "Practical Use of Polynomial Preconditionings for the Conjugate Gradient Method". In: *SIAM Journal on Scientific and Statistical Computing* 6.4, pp. 865–881.

— (1989). "Krylov Subspace Methods on Supercomputers". In: *SIAM Journal on Scientific and Statistical Computing* 10.6, pp. 1200–1232.

Follow along

slides

chen.pw/d/f5b

code

chen.pw/d/i7g

proposal

chen.pw/d/2mn