Predict-and-recompute conjugate gradient variants

Tyler Chen and Erin C. Carson

February 12, 2020

Follow along

slides

chen.pw/d/0ak

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.

Collaborators

- Erin C. Carson
- Anne Greenbaum
- Kelly Liu

Introduction

- Conjugate gradient (CG) is used to solve a linear system $\mathbf{Ax} = \mathbf{b}$ when $\mathbf{A} \in \mathbb{R}^{n \times n}$ 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
- Modern supercomputers have reached exascale (10^{18} flops)
  - Krylov subspace methods can only reach a fraction of this rate because of communication costs
- Convergence of CG in finite precision is not very well understood in general
- **Need to address communication costs, while considering numerical properties!**

The conjugate gradient algorithm

Algorithm 1 Hestenes and Stiefel Conjugate Gradient (preconditioned)

procedure HS-CG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{x}_0$)
  initialize()
  for $k = 1, 2, \ldots$ **do**
    $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
    $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1},\ \tilde{\mathbf{r}}_k = \mathbf{M}^{-1}\mathbf{r}_k$
    $\nu_k = \langle \tilde{\mathbf{r}}_k, \mathbf{r}_k \rangle$
    $\beta_k = \nu_k / \nu_{k-1}$
    $\mathbf{p}_k = \tilde{\mathbf{r}}_k + \beta_k \mathbf{p}_{k-1}$
    $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
    $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle$
    $\alpha_k = \nu_k / \mu_k$
  end for
end procedure

Communication costs

- 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

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 ways of doing this have been studied before; see for instance[1]
  - typically maintain two inner products and one matrix vector product per iteration

[1] 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 any rounding errors
- changing the order in which computations are performed can have an impact
- some of the communication hiding variants have very different behavior[2]
  - various approaches to deal with this; e.g. residual replacement[3], basis vector shifts[4], etc.

[2] Carson, Rozložník, Strakoš, Tichý, and Tůma 2018.

[3] Cools, Fatih Yetkin, Agullo, Giraud, and Vanroose 2018.

[4] Cornelis, Cools, and Vanroose 2019.

Hiding communication

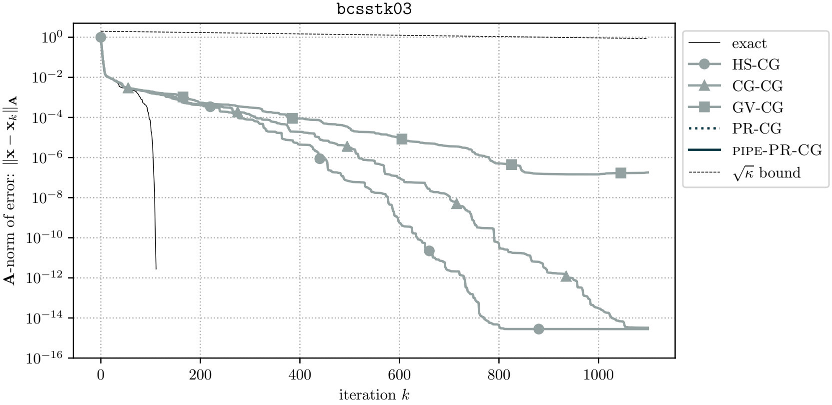
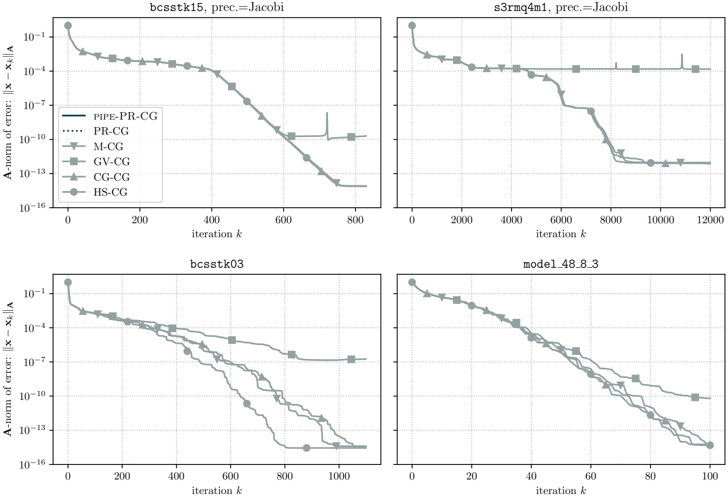


Figure: Convergence of finite precision conjugate gradient

Hiding communication



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
- There is numerical analysis theory for both effects:
  - Delay of convergence: perturbed Lanczos recurrence[5]
    - $\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{T}_k + \gamma_k \mathbf{q}_{k+1}\mathbf{e}_k^\top + \mathbf{F}_k$
  - Loss of accuracy: residual gap[6]
    - $\Delta_{\mathbf{r}_k} := (\mathbf{b} - \mathbf{A}\mathbf{x}_k) - \mathbf{r}_k$

[5] Paige 1976; Paige 1980; Greenbaum 1989.

[6] Greenbaum 1997.

Finite precision conjugate gradient

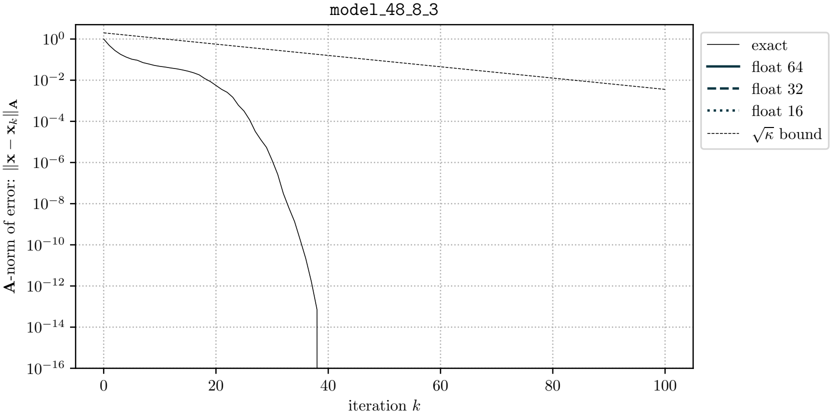


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

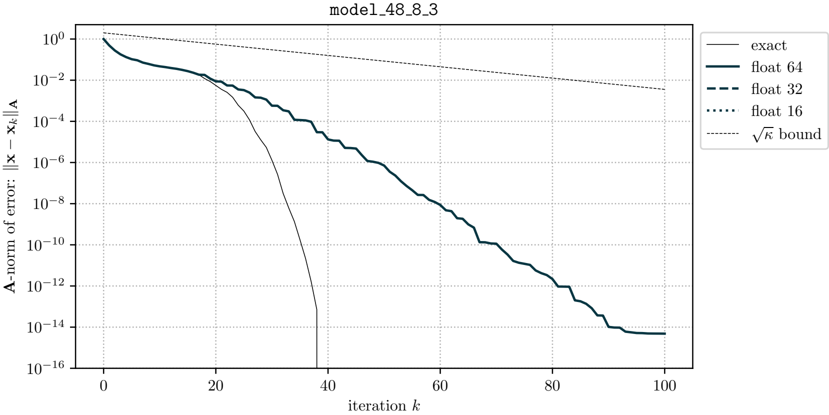


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

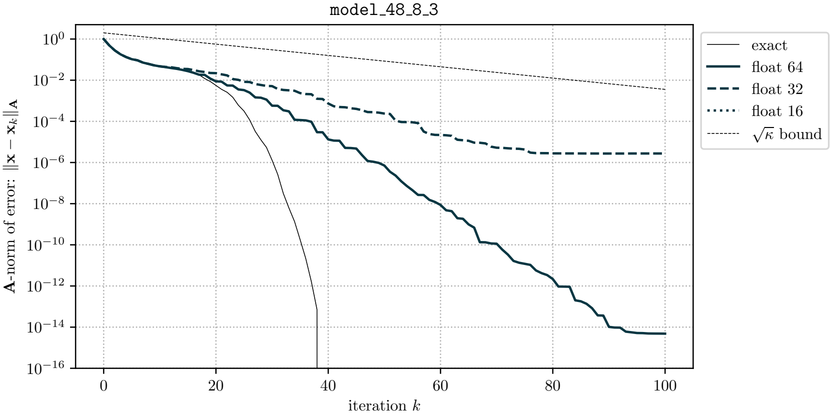


Figure: Convergence of finite precision conjugate gradient

Finite precision conjugate gradient

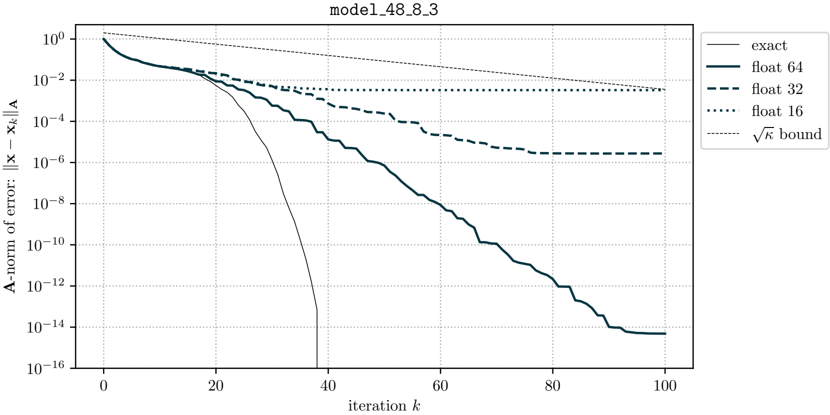


Figure: Convergence of finite precision conjugate gradient

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

Predict-and-recompute variants

Algorithm 2 Hestenes and Stiefel Conjugate Gradient (preconditioned)

procedure HS-CG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{x}_0$)
  initialize()
  for $k = 1, 2, \ldots$ **do**
    $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
    $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}, \;\; \tilde{\mathbf{r}}_k = \mathbf{M}^{-1}\mathbf{r}_k$
    $\nu_k = \langle \tilde{\mathbf{r}}_k, \mathbf{r}_k \rangle$
    $\beta_k = \nu_k / \nu_{k-1}$
    $\mathbf{p}_k = \tilde{\mathbf{r}}_k + \beta_k \mathbf{p}_{k-1}$
    $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k$
    $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k \rangle$
    $\alpha_k = \nu_k / \mu_k$
  end for
end procedure

Predict-and-recompute variants

Algorithm 3 Predict-and-recompute conjugate gradient

procedure PR-CG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{x}_0$)
    initialize()
    for $k = 1, 2, \ldots$ **do**
        $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
        $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1},\ \tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_{k-1} - \alpha_{k-1}\tilde{\mathbf{s}}_{k-1}$
        $\nu'_k = \nu_{k-1} - 2\alpha_{k-1}\delta_{k-1} + \alpha_{k-1}^2\gamma_{k-1}$
        $\beta_k = \nu'_k/\nu_{k-1}$
        $\mathbf{p}_k = \tilde{\mathbf{r}}_k + \beta_k\mathbf{p}_{k-1}$
        $\mathbf{s}_k = \mathbf{A}\mathbf{p}_k,\ \tilde{\mathbf{s}}_k = \mathbf{M}^{-1}\mathbf{s}_k$
        $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k\rangle,\ \delta_k = \langle \tilde{\mathbf{r}}_k, \mathbf{s}_k\rangle,\ \gamma_k = \langle \tilde{\mathbf{s}}_k, \mathbf{s}_k\rangle,\ \nu_k = \langle \tilde{\mathbf{r}}_k, \mathbf{r}_k\rangle$
        $\alpha_k = \nu_k/\mu_k$
    end for
end procedure

Predict-and-recompute variants

Algorithm 4 Pipelined predict-and-recompute conjugate gradient

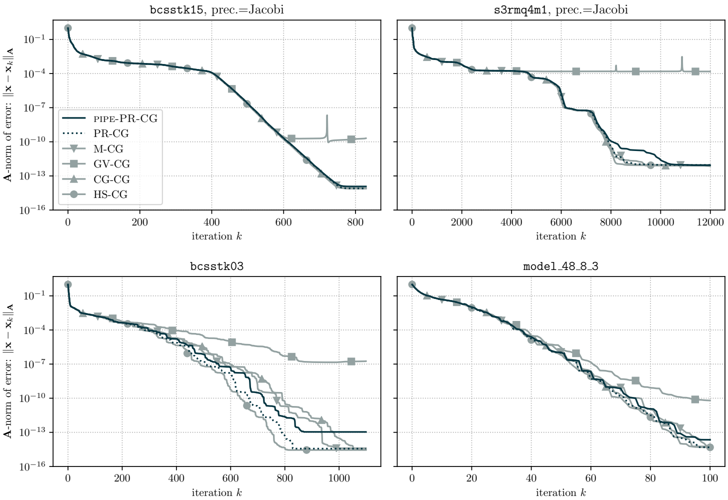
procedure pipe-PR-CG($\mathbf{A}$, $\mathbf{M}$, $\mathbf{b}$, $\mathbf{x}_0$)
  initialize()
  for $k = 1, 2, \ldots$ **do**
    $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$
    $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{s}_{k-1}$, $\tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_{k-1} - \alpha_{k-1}\tilde{\mathbf{s}}_{k-1}$
    $\mathbf{w}'_k = \mathbf{w}_{k-1} - \alpha_{k-1}\mathbf{u}_{k-1},\ \tilde{\mathbf{w}}'_k = \tilde{\mathbf{w}}_{k-1} - \alpha_{k-1}\tilde{\mathbf{u}}_{k-1}$
    $\nu'_k = \nu_{k-1} - 2\alpha_{k-1}\delta_{k-1} + \alpha_{k-1}^2\gamma_{k-1}$
    $\beta_k = \nu'_k/\nu_{k-1}$
    $\mathbf{p}_k = \tilde{\mathbf{r}}_k + \beta_k\mathbf{p}_{k-1}$
    $\mathbf{s}_k = \mathbf{w}'_k + \beta_k\mathbf{s}_{k-1},\ \tilde{\mathbf{s}}_k = \tilde{\mathbf{w}}'_k + \beta_k\tilde{\mathbf{s}}_{k-1}$
    $\mathbf{u}_k = \mathbf{A}\tilde{\mathbf{s}}_k,\ \tilde{\mathbf{u}}_k = \mathbf{M}^{-1}\mathbf{u}_k$
    $\mathbf{w}_k = \mathbf{A}\tilde{\mathbf{r}}_k,\ \tilde{\mathbf{w}}_k = \mathbf{M}^{-1}\mathbf{w}_k$
    $\mu_k = \langle \mathbf{p}_k, \mathbf{s}_k\rangle,\ \delta_k = \langle \tilde{\mathbf{r}}_k, \mathbf{s}_k\rangle,\ \gamma_k = \langle \tilde{\mathbf{s}}_k, \mathbf{s}_k\rangle,\ \nu_k = \langle \tilde{\mathbf{r}}_k, \mathbf{r}_k\rangle$
    $\alpha_k = \nu_k/\mu_k$
  end for
end procedure

Predict-and-recompute variants

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

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

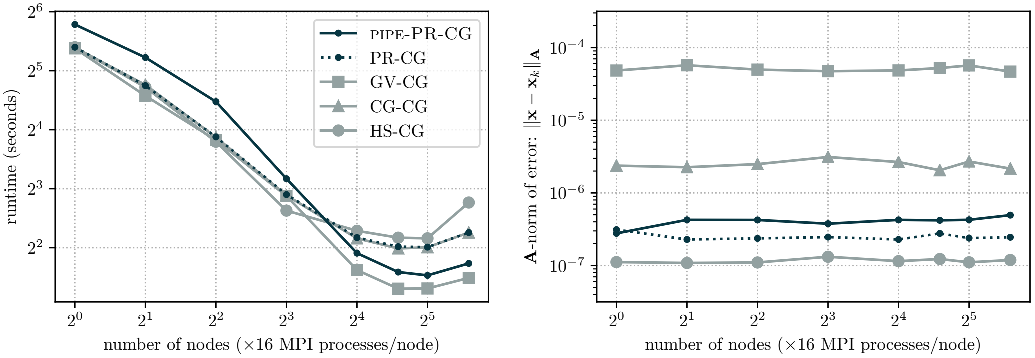


Figure: Convergence of conjugate gradient variants

Predict-and-recompute variants

- expressions for residual gap, and three term Lanczos recurrence, and ν'_k gap for PR-CG and pipe-PR-CG provides insight into improved convergence[7]
- practical use remains to be determined
- but, will be included in PETSc v3.13: `-ksp_type pipeprcg`
  - in this code the two matrix products are not overlapped with one another: is there an easy way to do this in PETSc?

[7] Chen and Carson 2020.

Predict-and-recompute variants

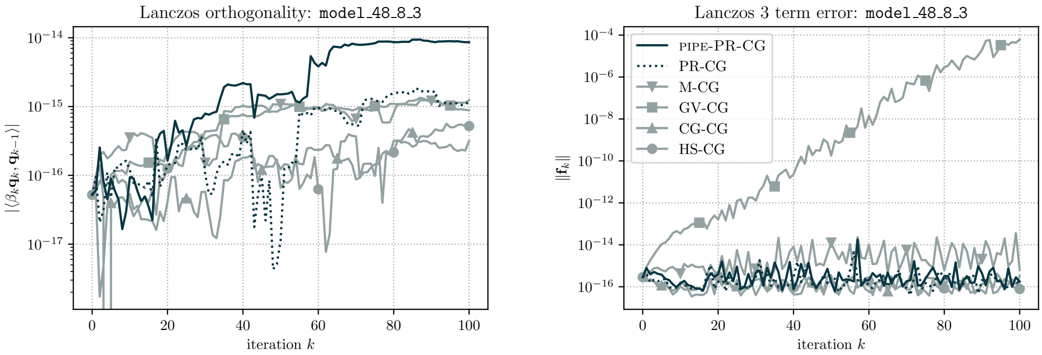


Figure: Perturbed Lanczos recurrence measures

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?

References

Carson, Erin C., Miroslav Rozložník, Zdeněk Strakoš, Peter Tichý, and Miroslav Tůma (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*. arXiv preprint: 1905.01549.

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, Emrullah Fatih Yetkin, Emmanuel Agullo, Luc Giraud, and Wim Vanroose (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). *Iterative Methods for Solving Linear Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics.

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

Paige, Christopher Conway (Dec. 1976). “Error Analysis of the Lanczos Algorithm for Tridiagonalizing a Symmetric Matrix”. In: *IMA Journal of Applied Mathematics* 18.3, pp. 341–349.

— (1980). “Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem”. In: *Linear Algebra and its Applications* 34, pp. 235–258.

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/0ak

code

chen.pw/d/i7g

preprint

chen.pw/d/f91