

Communication-avoiding Krylov subspace methods

Mark Hoemmen

mhoemmen@cs.berkeley.edu

University of California Berkeley EECS

SIAM Parallel Processing for Scientific Computing 2008



Overview

- Current Krylov methods: *communication-limited*
- Can rearrange them to *avoid communication*
- Can do this in a *numerically stable* way
- Requires rethinking *preconditioning*



Motivation

- Two communication-bound kernels
- Can rearrange *each kernel* to avoid communication, but. . .
- Data dependency between the two precludes rearrangement. . .
- *Unless* you rearrange the Krylov method!



Krylov methods: Two communication-bound kernels

- Sparse matrix-vector multiplication (SpMV)
 - Share/communicate source vector w/ neighbors
 - Low computational intensity per processor
- Orthogonalization: $\Theta(1)$ reductions per vector
 - Arnoldi/GMRES:
 - Modified Gram-Schmidt or Householder QR
 - Lanczos/CG:
 - Recurrence orthogonalizes implicitly



Krylov methods: Two communication-bound kernels

- Sparse matrix-vector multiplication (SpMV)
 - Share/communicate source vector w/ neighbors
 - Low computational intensity per processor
- Orthogonalization: $\Theta(1)$ reductions per vector
 - Arnoldi/GMRES:
 - Modified Gram-Schmidt or Householder QR
 - Lanczos/CG:
 - Recurrence orthogonalizes implicitly



Potential to avoid communication

- SpMV: Matrix powers kernel (Marghoob)
 - Compute $[v, Av, A^2v, \dots, A^s v]$
 - Tiling to reuse matrix entries
 - Parallel: same latency cost as one SpMV
 - Sequential: only read matrix $O(1)$ times
- Orthogonalization: TSQR (Julien)
 - Just as stable as Householder QR
 - Parallel: same latency cost as one reduction
 - Sequential: only read vectors once



Potential to avoid communication

- SpMV: Matrix powers kernel (Marghoob)
 - Compute $[v, Av, A^2v, \dots, A^s v]$
 - Tiling to reuse matrix entries
 - Parallel: same latency cost as one SpMV
 - Sequential: only read matrix $O(1)$ times
- Orthogonalization: TSQR (Julien)
 - Just as stable as Householder QR
 - Parallel: same latency cost as one reduction
 - Sequential: only read vectors once



Problem: Data dependencies limit reuse

- Krylov methods advance one vector at a time
- SpMV, then orthogonalize, then SpMV, ...

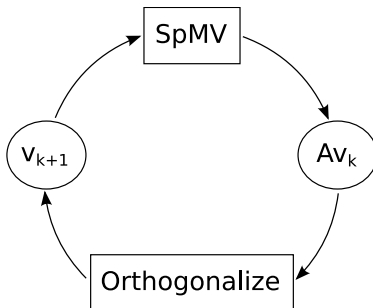


Figure: Data dependencies in Krylov subspace methods.

s-step Krylov methods: break the dependency

- Matrix powers kernel
 - Compute basis of $\text{span}\{v, Av, A^2v, \dots, A^s v\}$
- TSQR
 - Orthogonalize basis
- Use R factor to reconstruct upper Hessenberg H resp. tridiagonal T
- Solve least squares problem or linear system with H resp. T for coefficients of solution update



s-step Krylov methods: break the dependency

- Matrix powers kernel
 - Compute basis of $\text{span}\{v, Av, A^2v, \dots, A^sv\}$
- TSQR
 - Orthogonalize basis
- Use R factor to reconstruct upper Hessenberg H resp. tridiagonal T
- Solve least squares problem or linear system with H resp. T for coefficients of solution update



Example: GMRES



Original GMRES

- 1: **for** $k = 1$ to s **do**
- 2: $w = Av_{k-1}$
- 3: Orthogonalize w against v_0, \dots, v_{k-1} using Modified Gram-Schmidt
- 4: **end for**
- 5: Compute solution using H



Version 2: Matrix powers kernel & TSQR

- 1: $W = [v_0, Av_0, A^2v_0, \dots, A^s v_0]$
- 2: $[Q, R] = TSQR(W)$
- 3: Compute H using R
- 4: Compute solution using H
 - s powers of A for no extra latency cost
 - s steps of QR for one step of latency
 - But...



Basis computation not stable

- v, Av, A^2v, \dots looks familiar...
- It's the power method!
 - Converges to principal eigenvector of A
 - Expect increasing linear dependence...
- Basis condition number *exponential* in s



Basis computation not stable

- v, Av, A^2v, \dots looks familiar...
- It's the power method!
 - *Converges* to principal eigenvector of A
 - Expect increasing linear dependence...
- Basis condition number *exponential* in s



Basis computation not stable

- v, Av, A^2v, \dots looks familiar...
- It's the power method!
 - *Converges* to principal eigenvector of A
 - Expect increasing linear dependence...
- Basis condition number *exponential* in s



Version 3: Different basis

- Just like polynomial interpolation
- Use a different basis, e.g.:
 - Newton basis $W = [v, (A - \theta_1 I)v, (A - \theta_2 I)(A - \theta_1 I)v, \dots]$
 - Get shifts θ_i for free – Ritz values
 - Can change shifts with each group of s
 - Chebyshev basis $W = [v, T_1(v), T_2(v), \dots]$
 - Use condition number bounds to scale $T_k(z)$
 - Uncertain sensitivity of $\kappa_2(W)$ to bounds



Basis condition number

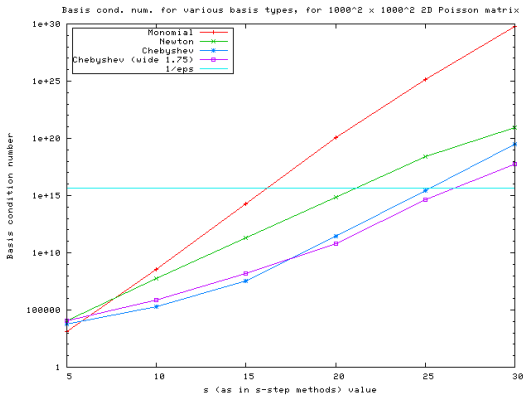


Figure: Condition number of various bases as a function of basis length s . Matrix A is a $10^6 \times 10^6$ 2-D Poisson operator.



Numerical experiments

- Diagonal $10^4 \times 10^4$ matrix, $\kappa_2(A) = 10^8$
- $s = 24$
- Newton: basis condition # about 10^{14}
- Monomial: basis condition # about 10^{16}



Better basis pays off: restarting

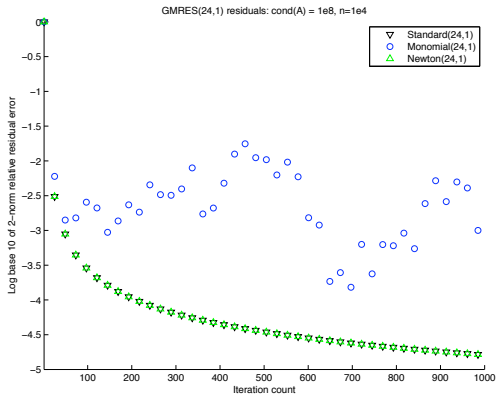


Figure: Restart after every group of s steps



Better basis pays off: less restarting

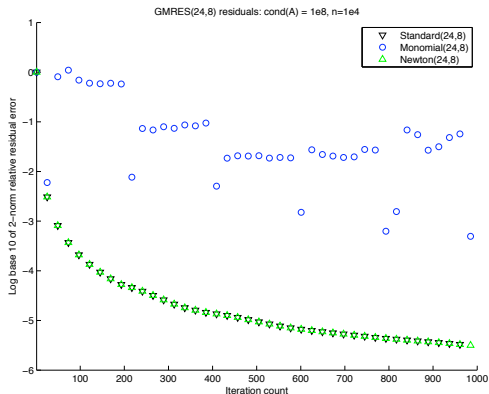


Figure: Restart after 8 groups of $s = 24$ steps.



Krylov methods we can rearrange

- s -step Arnoldi / GMRES
- s -step symmetric Lanczos / CG
- Need not restart after each group of s
 - Just update TSQR factorization



Previous work: s-step CG, part 1

- Van Rosendale 1983, Chronopoulos 1989, ...
 - Compute $W = [v, Av, A^2v, \dots, A^sv]$
 - Get solution update coefficients from $W^T W$
- Unstable
 - Monomial basis ($\kappa_2(W)$ is $\Theta(2^s)$)
 - Gram matrix $W^T W$ (squares $\kappa_2(A)$)
- No matrix powers kernel
- No preconditioning



Previous work: s-step GMRES

- De Sturler 1991, Bai et al. 1991, et al.
- More stable
 - Newton basis, not monomial
 - QR, not Gram matrix
- No matrix powers kernel
- No preconditioning
- Must restart after each group of s



Previous work: s-step CG, part 2

- Toledo 1995 (PhD thesis)
- Developed as part of a matrix powers kernel
 - For (un)structured low-dimensional grids
 - Also for multigrid-like hierarchical graphs
- Based on Chronopoulos 1989
- Suggested change of basis for stability
- Formed Gram matrix $W^T W$ (squares $\kappa_2(A)$)
- No preconditioning



Preconditioning: matrix powers kernel changes

- GMRES with left preconditioning (or any kind)
 - $v, M^{-1}Av, (M^{-1}A)^2v, \dots, (M^{-1}A)^sv$
- Symmetric Lanczos / CG with split preconditioning
 - $v, L^{-1}AL^{-T}v, \dots, (L^{-1}AL^{-T})^sv$
- Symmetric Lanczos / CG with left preconditioning
 - $V = [v, M^{-1}Av, \dots, (M^{-1}A)^sv]$, and
 - $W = [Av, AM^{-1}Av, \dots, (AM^{-1})^sAv]$
- Works with any basis



Preconditioning: matrix powers kernel changes

- GMRES with left preconditioning (or any kind)
 - $v, M^{-1}Av, (M^{-1}A)^2v, \dots, (M^{-1}A)^sv$
- Symmetric Lanczos / CG with split preconditioning
 - $v, L^{-1}AL^{-T}v, \dots, (L^{-1}AL^{-T})^sv$
- Symmetric Lanczos / CG with left preconditioning
 - $V = [v, M^{-1}Av, \dots, (M^{-1}A)^sv]$, and
 - $W = [Av, AM^{-1}Av, \dots, (AM^{-1})^sAv]$
- Works with any basis



Effective preconditioning

- Easy to limit communication if connectivity local
- Sparse: “looks like a low-dimensional mesh”
- General: low-rank off-diagonal blocks
 - Rank only grows linearly in s
 - Matrix *and* preconditioner
 - e.g., hierarchical matrices, semiseparable, fast multipole

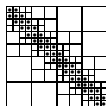


Figure: Discretization of $\log(|x - y|)$ on interval.



Future work

- Preconditioner implementations
- Performance tuning (choosing s)
- Extension to eigensolvers
- Lanczos biorthogonalization (e.g., Bi-CG)
- Combine with block Krylov methods
 - Block methods can already use TSQR
 - Does combining block and s -step pay?






Summary

- s -step Krylov methods incomplete before:
 - Either not stable, not scalable, or both
 - Had to restart between groups of s
 - No preconditioning / not part of optimizations
- Now we have all the pieces!
 - Stable, optimized kernels
 - Can do restarting or not
 - Preconditioning






Bibliography I

-  Z. BAI, D. HU, AND L. REICHTEL, *A Newton basis GMRES implementation*, IMA Journal of Numerical Analysis, 14 (1994), pp. 563–581.
-  A. H. BAKER, J. M. DENNIS, AND E. R. JESSUP, *On improving linear solver performance: A block variant of GMRES*, SIAM J. Sci. Comp., 27 (2006), pp. 1608–1626.
-  S. BÖRM, L. GRASEDYCK, AND W. HACKBUSCH, *Hierarchical matrices*.
http://www.mis.mpg.de/scicomp/Fulltext/WS_HMatrices.pdf, 2004.






Bibliography II

-  S. CHANDRASEKARAN, M. GU, AND W. LYONS, *A fast and stable adaptive solver for hierarchically semi-separable representations*, May 2004.
-  A. T. CHRONOPOULOS AND C. W. GEAR, *s-step iterative methods for symmetric linear systems*, J. Comput. Appl. Math., 25 (1989), pp. 153–168.
-  A. T. CHRONOPOULOS AND A. B. KUCHEROV, *A parallel Krylov-type method for nonsymmetric linear systems*, in High Performance Computing - HiPC 2001: Eighth International Conference, Hyderabad, India, December 17-20, 2001. Proceedings, Springer, 2001, pp. 104–114.



Bibliography III

-  E. DE STURLER, *A parallel variant of GMRES(m)*, in Proceedings of the 13th IMACS World Congress on Computation and Applied Mathematics, J. J. H. Miller and R. Vichnevetsky, eds., Dublin, Ireland, 1991, Criterion Press.
-  J. ERHEL, *A parallel GMRES version for general sparse matrices*, Electronic Transactions on Numerical Analysis, 3 (1995), pp. 160–176.
-  W. GAUTSCHI AND G. INGLESE, *Lower bounds for the condition number of Vandermonde matrices*, Numer. Math., 52 (1988), pp. 241–250.



Bibliography IV




-  W. HACKBUSCH, *Hierarchische Matrizen – Algorithmen und Analysis*.

<http://www.mis.mpg.de/scicomp/Fulltext/hmvorlesung.ps>, last accessed 22 May 2006, Jan. 2006.

-  W. D. JOUBERT AND G. F. CAREY, *Parallelizable restarted iterative methods for nonsymmetric linear systems, Part I: Theory*, International Journal of Computer Mathematics, 44 (1992), pp. 243–267.






Bibliography V

-  ———, *Parallelizable restarted iterative methods for nonsymmetric linear systems, Part II: Parallel implementation*, International Journal of Computer Mathematics, 44 (1992), pp. 269–290.
-  C. E. LEISERSON, S. RAO, AND S. TOLEDO, *Efficient out-of-core algorithms for linear relaxation using blocking covers*, Journal of Computer and System Sciences, 54 (1997), pp. 332–344.
-  G. MEURANT, *The block preconditioned conjugate gradient method on vector computers*, BIT, 24 (1984), pp. 623–633.



Bibliography VI

-  D. P. O'LEARY, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.
-  S. A. TOLEDO, *Quantitative performance modeling of scientific computations and creating locality in numerical algorithms*, PhD thesis, Massachusetts Institute of Technology, 1995.
-  J. VAN ROSENDALE, *Minimizing inner product data dependence in conjugate gradient iteration*, in Proc. IEEE Internat. Confer. Parallel Processing, 1983.



Why not use block Krylov methods?

- Solve $Ax = B$ for multiple right-hand sides
- Useful for eigenproblems (original use)
- No extra latency cost
- Bandwidth cost scales linearly w/ # RHS's
- Can use if only one right-hand side



Problems with block methods for $Ax = b$

- If only one right-hand side:
 - Start with one right-hand side
 - After each restart cycle, add error vector to RHS block
 - High startup cost
 - Need s cycles of s until at full block size
 - Whereas, s -step always at full optimization
- More complicated convergence & breakdown conditions



Preconditioning

- Modifications to matrix powers kernel
- Low off-diagonal rank characterization
- Possible preconditioners



Preconditioning and matrix powers

- GMRES or split-preconditioner Lanczos
 - Standard matrix powers kernel
 - Just replace A with preconditioned operator $L^{-1}AL^{-T}$
- Left-preconditioned CG: need new kernel!



New kernel for left-preconditioned CG

- For a basis p_0, p_1, p_2, \dots , define “left shift” operator Lshift :
 - In that basis’ coordinate system, $\text{Lshift } e_i = e_{i+1}$ (“multiply by x ”)
 - $\text{Lshift}_A(v)$ means replace x with matrix A
- Left-preconditioned CG: need

$$V_{s+1} = [v, \text{Lshift}_{M^{-1}A}(v), \dots, \text{Lshift}_{M^{-1}A}^s(v)], \text{ and}$$

$$W_s = [Av, \text{Lshift}_{AM^{-1}}(Av), \dots, \text{Lshift}_{AM^{-1}}^{s-1}(Av)]$$



Preconditioning and orthogonalization

- GMRES or split-preconditioned CG: no change
- Left-preconditioned CG:
 - $M^{-1}A$ usually nonsymmetric
 - Basis vectors not orthogonal
 - M -orthogonal (“conjugate”) instead
 - Can’t use QR to orthogonalize
 - Must rely on CG recurrence instead
 - Gram matrix $V_{s+1}^* W_s$ squares $\kappa(A)$ – bad!
 - Avoid by using generalized QR or SVD instead



Preconditioning and orthogonalization

- GMRES or split-preconditioned CG: no change
- Left-preconditioned CG:
 - $M^{-1}A$ usually nonsymmetric
 - Basis vectors not orthogonal
 - M -orthogonal (“conjugate”) instead
 - Can’t use QR to orthogonalize
 - Must rely on CG recurrence instead
 - Gram matrix $V_{s+1}^* W_s$ squares $\kappa(A)$ – bad!
 - Avoid by using generalized QR or SVD instead



Preconditioning: Low off-diagonal rank

- Matrix powers: depends on boundaries being “lower dimension” than interiors
- Boundary edges of graph are off-diagonal nonzeros
- Generalization: *low-rank off-diagonal blocks*
- Can do matrix powers kernel with SVD-like representation of partitioned matrix



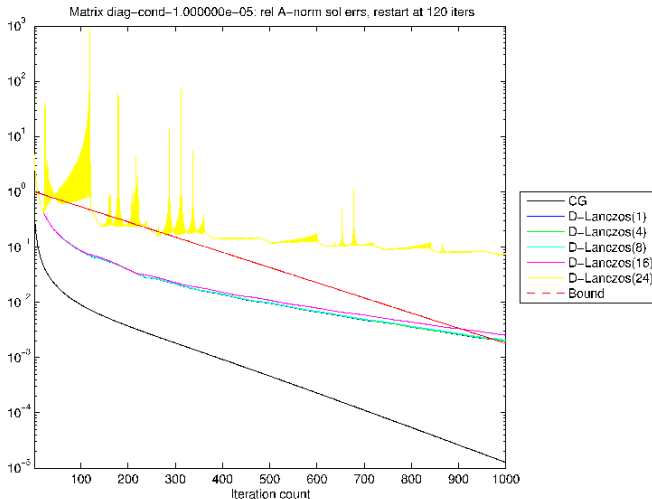
Possible preconditioners

Right generalization: *low-rank off-diagonal blocks*

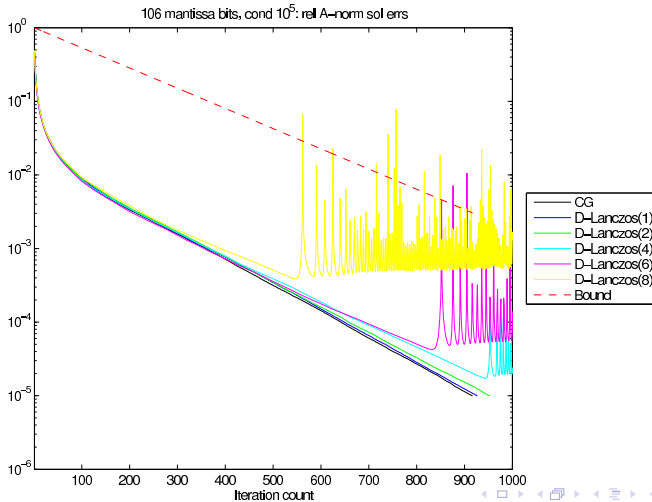
- Rank 0: block diagonal (a.k.a. block Jacobi)
 - Blocks can be arbitrarily complex
- But effective preconditioning needs some communication!
- Sparse approximate inverse (SPAI) – constrain low off-diag rank
- \mathcal{H} , \mathcal{H}^2 , HSS matrices
 - From integral equations with separable kernels
 - Continuous analogue to discrete “low-rank off-diagonal blocks” condition



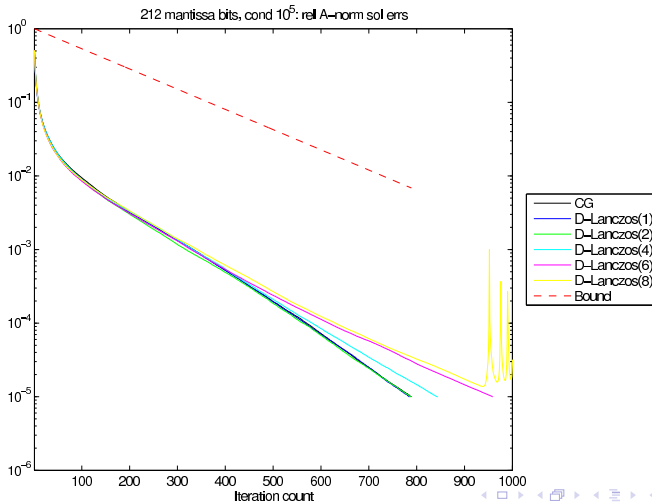
Restarting for stability



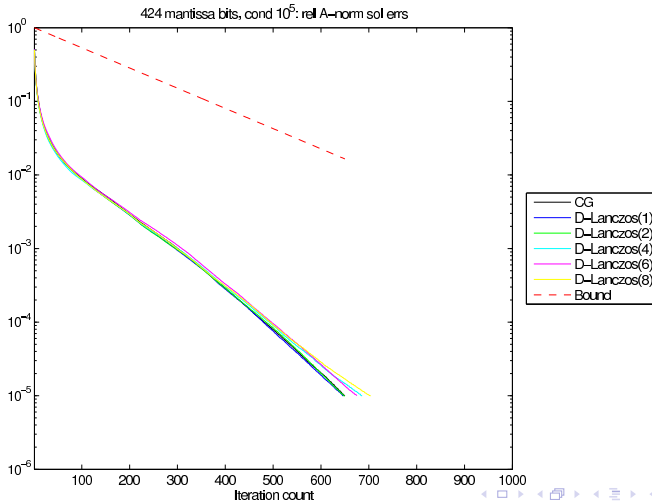
Extra precision for stability (1 of 3)



Extra precision for stability (2 of 3)



Extra precision for stability (3 of 3)



Lanczos(s,t) w/ reorthogonalization

- Get orthogonality estimates from Lanczos recurrence (Paige)
- Each group of s basis vectors is a TSQR Q factor
- Best reorthogonalization:
 - Do TSQR of last group to compute Lanczos coefficients
 - Use Lanczos coeffs in Paige's recurrence
 - If last group not orthogonal w.r.t. previous groups
 - Compute it explicitly
 - Orthogonalize against previous $t - 1$ groups
 - Finally take TSQR again of last group
- Converting all groups of s to explicit storage and redoing TSQR on them all is too expensive & unnecessary



Components

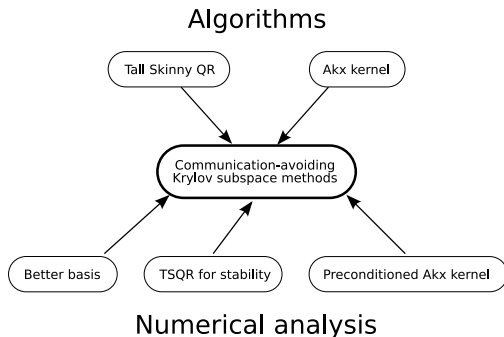


Figure: Components of communication-avoiding Krylov methods.



Acknowledgments

- NSF
- DoE
- ACM/IEEE