Algorithmic Improvements for Dense Symmetric Tridiagonalization

Grey Ballard

EPASA 2015
International Workshop on Eigenvalue Problems:
Algorithms, Software, and Applications

September 14, 2015

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

(SAND2015-8055 C)

Summary

- We want to solve the dense symmetric eigenvalue problem
  - most/all of the eigenvalues and (possibly) eigenvectors
- We're targeting large distributed-memory parallel machines
  - seeking scalable, communication-efficient algorithms
- Overall approach is two-phase tridiagonalization
- We propose two algorithmic improvements
  - Householder vector reconstruction
  - Communication-avoiding successive band reduction

Collaborators

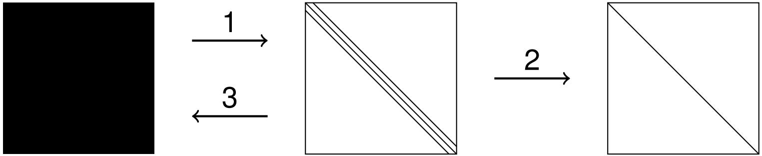
Based on joint work with

- James Demmel — UC Berkeley
- Laura Grigori — INRIA
- Nick Knight — NYU
- Mathias Jacquelin — LBNL
- Hong Diep Nguyen — UC Berkeley
- Edgar Solomonik — ETH Zurich

Outline

Tridiagonalization

We can solve the dense symmetric eigenvalue problem with 3 steps:



1. Reduction-to-tridiagonal via orthogonal similarity transformations
2. Solve the symmetric tridiagonal eigenvalue problem
3. Back-transformation of eigenvectors (if desired)

Two-Phase Tridiagonalization (SBR)

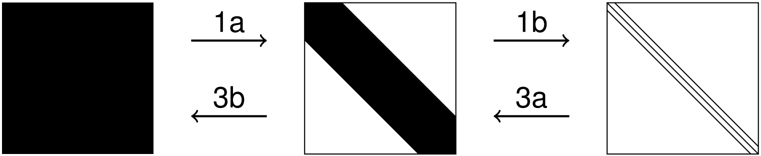
Tridiagonalization can be done over two (or more) phases in procedure known as Successive Band Reduction (SBR) [BLS00]:

1a Full-to-band via orthogonal similarity transformations

1b Band-to-tridiagonal using bulge-chasing transformations

Two-Phase Tridiagonalization (SBR)

Tridiagonalization can be done over two (or more) phases in procedure known as Successive Band Reduction (SBR) [BLS00]:



1a Full-to-band via orthogonal similarity transformations

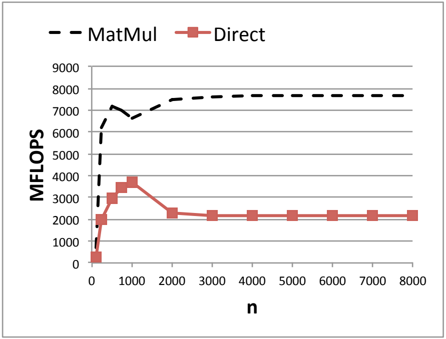
1b Band-to-tridiagonal using bulge-chasing transformations

- Two-phase back-transformation required for eigenvectors

Two-Phase Performance Benefits

Two-phase tridiagonalization avoids communication bottlenecks of direct approach

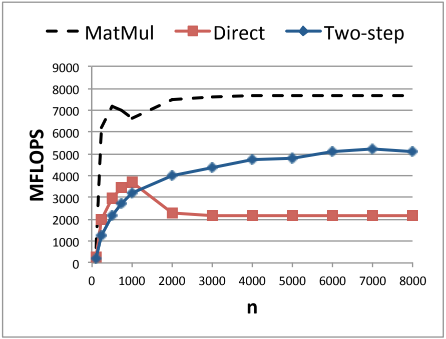
- Sequential performance example
- Direct approach suffers poor cache performance



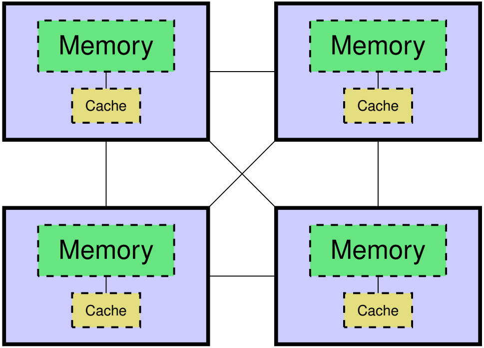
Two-Phase Performance Benefits

Two-phase tridiagonalization avoids communication bottlenecks of direct approach

- Sequential performance example
- Direct approach suffers poor cache performance
- Two-phase approach already available in MKL



Model of Distributed-Memory Parallel Computation



To analyze algorithms, we are interested in the following quantities

- **flops** — floating point operations
- **memory bandwidth cost** — words moved between memory and cache
- **interprocessor bandwidth cost** — words communicated between processors
- **latency cost** — messages communicated between processors

Related Work

Two-phase tridiagonalization is proven to be effective in practice, achieving better performance than direct tridiagonalization

- despite requiring more flops for eigenvectors

- Sequential
  - Successive Band Reduction [BLS00], Intel MKL
- Multicore
  - PLASMA [LLD11], CA-SBR [BDK12]
- GPU
  - MAGMA [HSG+13], Eigen-G [IYM14]
- Distributed-memory parallel
  - ELPA [MBJ+14], Eigen-Exa [IYM11]

Outline

(One-Phase) Householder Tridiagonalization

For $i = 1$ **to** $n - 2$

1. compute Householder vector y_i to annihilate column i
2. apply two-sided symmetric update

$$\tilde{A} = (I - \tau_i y_i y_i^T) \cdot A \cdot (I - \tau_i y_i y_i^T)$$

   cast as symmetric rank-2 update

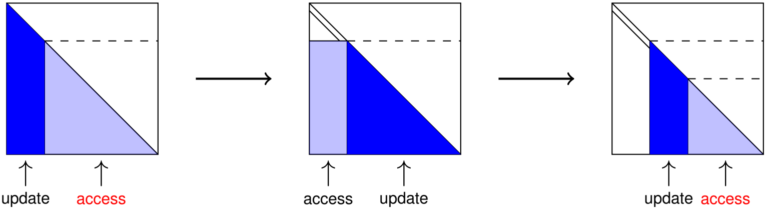
$$\tilde{A} = A - y_i v_i^T - v_i y_i^T$$

End

Blocked Direct Tridiagonalization Algorithm

Direct tridiagonalization performed with blocked algorithm:

- panel factorization + (two-sided symmetric) trailing matrix update

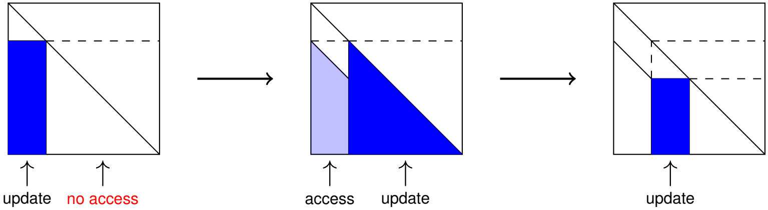


- Panel factorization requires BLAS 2 (matrix-vector) operations
  - total of $O(n^3)$ operations
- Trailing matrix update uses BLAS 3 (matrix-matrix) operations
  - total of $O(n^3)$ operations

Blocked Full-to-Band Algorithm

Full-to-band also performed with blocked algorithm:

- panel factorization + (two-sided symmetric) trailing matrix update



- Panel factorization is tall-skinny QR factorization
  - total of $O(n^2 b)$ operations
- Trailing matrix update uses BLAS 3 (matrix-matrix) operations
  - total of $O(n^3)$ operations

Blocked Full-to-Band Algorithm

For $i = 1$ **to** $\frac{n}{b} - 2$

1. QR factorization to generate Y_i and annihilate block column i
2. apply two-sided symmetric update

$$\tilde{A} = (I - Y_i T_i Y_i^T) \cdot A \cdot (I - Y_i T_i^T Y_i^T)$$

cast as symmetric rank-$2b$ update

$$\tilde{A} = A - Y_i V_i^T - V_i Y_i^T$$

End

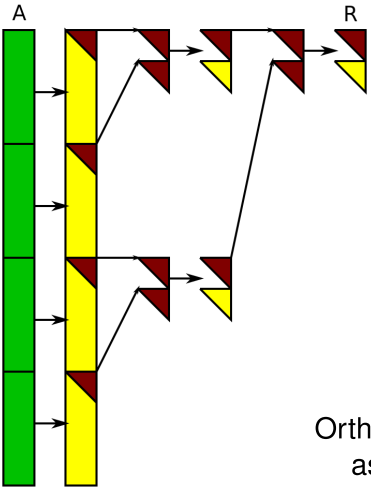
Direct Tridiagonalization vs Full-to-Band

Full-to-band communicates less than direct tridiagonalization, reducing **interprocessor latency** and **local memory bandwidth** costs

However,

- tall-skinny QR can still be a latency bottleneck
  - Householder QR requires a synchronization for every column
- (and we still have to reduce the band matrix to tridiagonal form)

Tall-Skinny QR (TSQR) Algorithm [DGHL12]



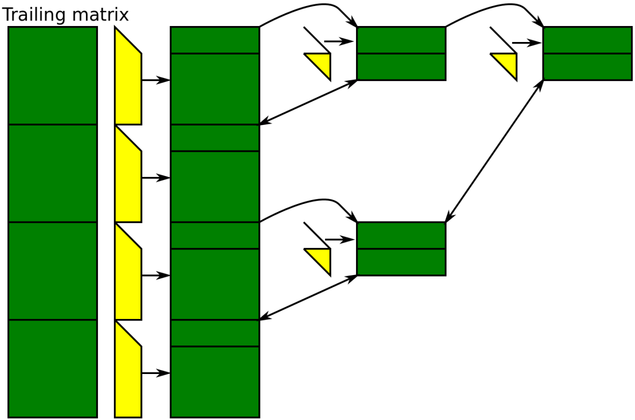
Key benefit of TSQR:
one parallel reduction

Householder QR:
one reduction *per column*

Orthogonal factor stored implicitly
as tree of Householder vectors

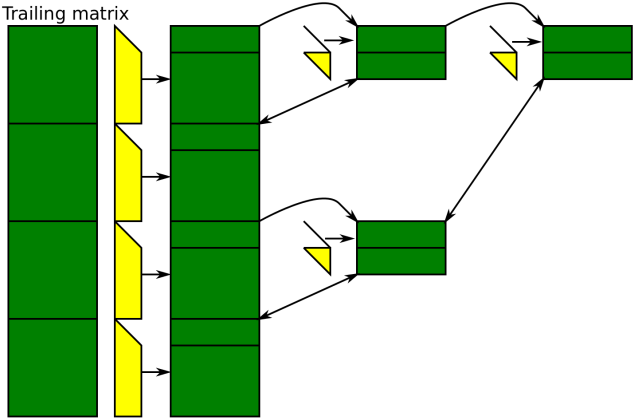
Communication-Avoiding QR (CAQR)

CAQR is designed for general matrices, using TSQR for panel factorization and applying the *one-sided* update using implicit structure



Communication-Avoiding QR (CAQR)

CAQR is designed for general matrices, using TSQR for panel factorization and applying the *one-sided* update using implicit structure



Performing a symmetric two-sided update is much more complicated

TSQR within Full-to-Band

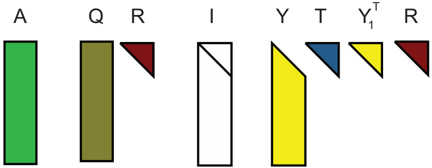
- TSQR is best algorithm for panel factorization, but
- Two-sided symmetric trailing matrix update is easier with Householder vectors

Can we get the best of both worlds:
can we perform TSQR but then recover Householder vectors?

Key Idea

Compute a QR decomposition using Householder vectors*:

$$A = QR = (I - YTY_1^T)R$$

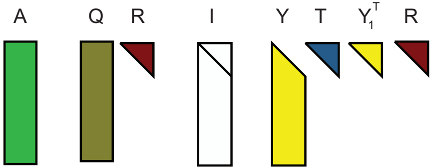


*$I - YTY_1^T$ known as compact WY representation

Key Idea

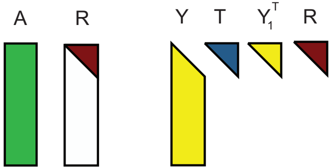
Compute a QR decomposition using Householder vectors*:

$$A = QR = (I - YTY_1^T)R$$



Re-arrange the equation and we have an LU decomposition:

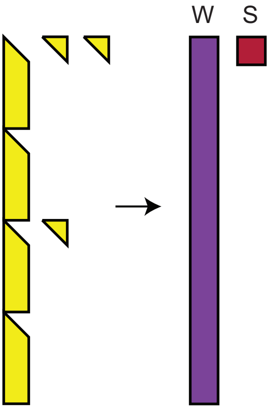
$$A - R = Y \cdot (-TY_1^T R)$$



*$I - YTY_1^T$ known as compact WY representation

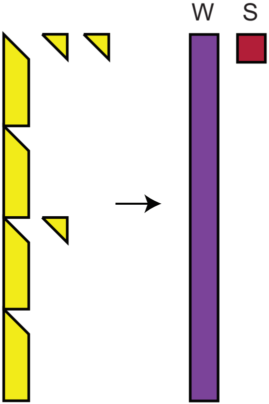
Yamamoto's Idea for QR Decomposition

- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the one-sided trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea was to convert implicit tree into compact WY-like representation

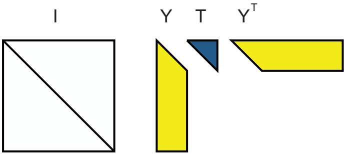


Yamamoto's Idea for QR Decomposition

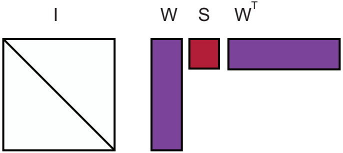
- Y. Yamamoto gave a talk at SIAM ALA 2012: he wanted to use TSQR but offload the one-sided trailing matrix update to a GPU
- To make CAQR's trailing matrix update more like matrix multiplication, his idea was to convert implicit tree into compact WY-like representation



Compact WY representation: $I - YTY^T$



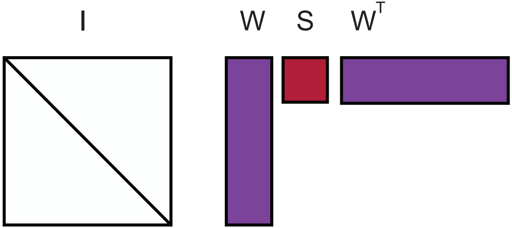
Basis-kernel representation: $I - WSW^T$



Yamamoto's Algorithm

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Set $W = Q - I$
4. Set $S = (I - Q_1)^{-1}$

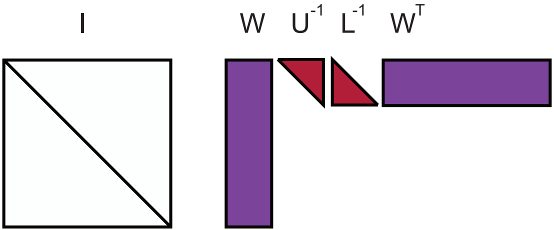
$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1 \right]^{-1} \left[(Q_1 - I)^T \quad Q_2^T \right]$$



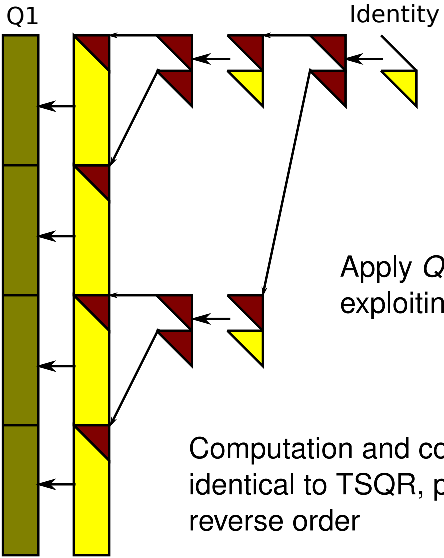
Yamamoto's Algorithm

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Set $W = Q - I$
4. Set $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1 \right]^{-1} \left[(Q_1 - I)^T \quad Q_2^T \right]$$



How is Q formed?



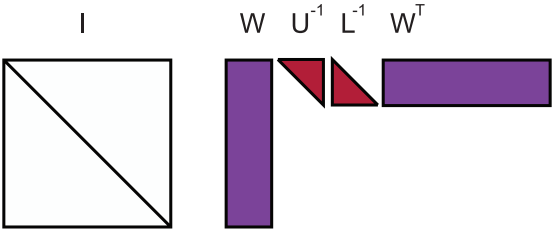
Apply Q to the identity, exploiting sparsity

Computation and communication identical to TSQR, performed in reverse order

Yamamoto's Algorithm

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Set $W = Q - I$
4. Set $S = (I - Q_1)^{-1}$

$$I - WSW^T = I - \begin{bmatrix} Q_1 - I \\ Q_2 \end{bmatrix} \left[I - Q_1 \right]^{-1} \left[(Q_1 - I)^T \quad Q_2^T \right]$$

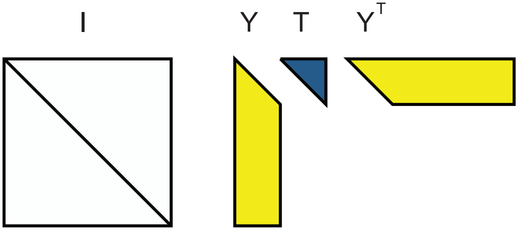


Reconstructing Householder Vectors (TSQR-HR)

With a little more effort, we can obtain the compact WY representation:

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Perform LU decomposition: $Q - I = LU$
4. Set $Y = L$
5. Set $T = -UY_1^{-T}$

$$I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}$$



Why form Q?

Cheaper approach based on $A - R = Y \cdot (-TY_1^T R)$:

1. Perform TSQR
2. Perform LU decomposition: $A - R = LU$
3. Set $Y = L$
4. Set $T = -UR^{-1}Y_1^{-T}$ (or compute T from Y)

Why form Q?

Cheaper approach based on $A - R = Y \cdot (-TY_1^T R)$:

1. Perform TSQR
2. Perform LU decomposition: $A - R = LU$
3. Set $Y = L$
4. Set $T = -UR^{-1}Y_1^{-T}$ (or compute T from Y)

This approach is similar to computing R using TSQR and Q using Householder QR

- if A is well-conditioned, works fine
- if A is ill-conditioned, R matrix is sensitive to roundoff
- more on less-stable approaches later...

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
    - what if $Q - I$ is singular?

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
  - what if $Q - I$ is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition: $Q - Sgn = LU$
  - Sgn matrix corresponds to sign choice in Householder QR
  - guarantees $Q - Sgn$ is nonsingular
  - guarantees maximum element on the diagonal (no pivoting)

What about pivoting in LU?

Third step in reconstructing Householder vectors:

- Perform LU decomposition: $Q - I = LU$
    - what if $Q - I$ is singular?

Actually, we need to make a sign choice:

- Perform LU decomposition: $Q - Sgn = LU$
    - Sgn matrix corresponds to sign choice in Householder QR
    - guarantees $Q - Sgn$ is nonsingular
    - guarantees maximum element on the diagonal (no pivoting)

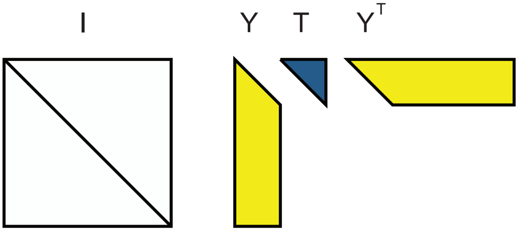
No pivoting makes LU of tall-skinny matrix very easy

- LU of top block followed by triangular solve for all other rows

Reconstructing Householder Vectors (TSQR-HR)

1. Perform TSQR
2. Form Q explicitly (tall-skinny orthonormal factor)
3. Perform LU decomposition: $Q - Sgn = LU$
4. Set $Y = L$
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$

$$I - YTY^T = I - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} Y_1^T & Y_2^T \end{bmatrix}$$



Numerical Stability

Theorem

Let $\hat{R}$ be the computed upper triangular factor of $m \times b$ matrix A obtained via the TSQR algorithm with p processors using a binary tree (assuming $m/p \geq b$), and let $\tilde{Q} = I - \tilde{Y}\tilde{T}\tilde{Y}_1^T$ and $\tilde{R} = Sgn \cdot \hat{R}$ where $\tilde{Y}$, $\tilde{T}$, and Sgn are the computed factors obtained from Householder reconstruction. Then

$$\|A - \tilde{Q}\tilde{R}\|_F \leq F_1(m, b, p, \varepsilon)\|A\|_F$$

and

$$\|I - \tilde{Q}^T\tilde{Q}\|_F \leq F_2(m, b, p, \varepsilon)$$

where $F_1, F_2 = O\left(\left(b^{3/2}(m/p) + b^{5/2}\log p + b^3\right)\epsilon\right)$ *for* $b(m/p)\epsilon \ll 1$.

*Result based on the stability of TSQR [MYZ12]

Numerical Experiments for Tall-Skinny Matrices

ρ	$\kappa(A)$	$\|A - \tilde{Q}\tilde{R}\|_F$	$\|I - \tilde{Q}^T\tilde{Q}\|_F$
1e-01	5.1e02	2.2e-15	6.8e-15
1e-03	5.2e04	2.3e-15	9.3e-15
1e-05	5.2e06	2.4e-15	9.5e-15
1e-07	5.1e08	2.3e-15	9.1e-15
1e-09	5.2e10	2.3e-15	9.3e-15
1e-11	5.2e12	2.2e-15	8.8e-15
1e-13	5.0e14	2.7e-15	1.2e-14
1e-15	4.7e15	2.3e-15	8.7e-15

Error of TSQR-HR on tall and skinny matrices ($m = 1000, b = 200$)

Costs of Householder Reconstruction

Let A be $m \times b$

Householder Reconstruction

1. Perform TSQR — $2mb^2$ flops, one QR reduction
2. Form Q — $2mb^2$ flops, one QR reduction
3. $\mathrm{LU}(Q - Sgn)$ — mb^2 flops, one broadcast
4. Set $Y = L$
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Costs of Householder Reconstruction

Let A be $m \times b$

Improved Householder Reconstruction

1. Perform TSQR — $2mb^2$ flops, one QR reduction
2. Form Q_1 — $O(b^3)$ flops
3. Compute $LU = Q_1 - Sgn$ — $O(b^3)$ flops
4. Apply Q to U^{-1} to get Y — $2mb^2$ flops, one QR reduction
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Costs of Householder Reconstruction

Let A be $m \times b$

Improved Householder Reconstruction

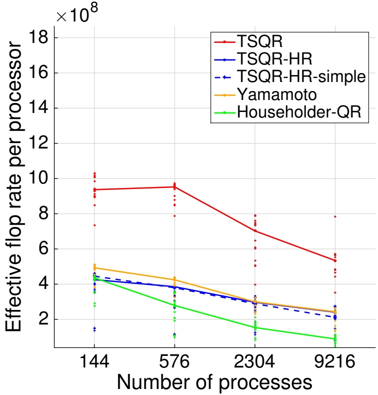
1. Perform TSQR — $2mb^2$ flops, one QR reduction
2. Form Q_1 — $O(b^3)$ flops
3. Compute $LU = Q_1 - Sgn$ — $O(b^3)$ flops
4. Apply Q to U^{-1} to get Y — $2mb^2$ flops, one QR reduction
5. Set $T = -U \cdot Sgn \cdot Y_1^{-T}$ — $O(b^3)$ flops

Alternative Stable Algorithms

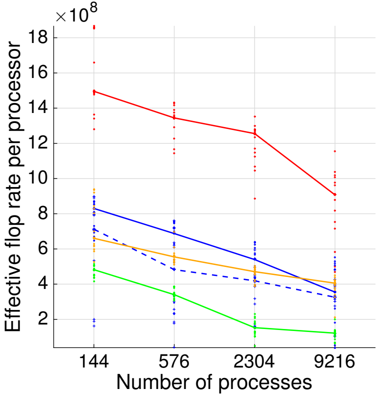
- TSQR — $2mb^2$ flops, one QR reduction
- HhQR (and form T) — $3mb^2$ flops, $2b$ reductions
- Yamamoto's — $4mb^2$ flops, two QR reductions

Performance of Stable Tall-Skinny QR Algorithms

Weak Scaling, Hopper (MKL)
512*p-by-32 problem



Weak Scaling, Edison (MKL)
512*p-by-32 problem



Some Cheaper and Less-Stable Alternatives

Alternative Less-Stable Algorithms

- TSQR-AR
  - Use TSQR to get R
  - Perform LU($A - R$) to get Y
- CholQR-HR
  - $R = \text{Chol}(A^T A)$
  - Perform LU($A - R$) to get Y

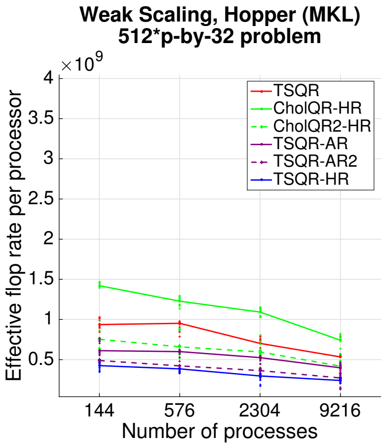
...or run these with a step of iterative refinement

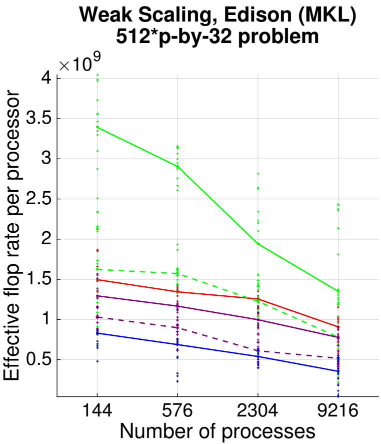
Numerical Stability of Less-Stable Alternatives

		TSQR-AR with it. refinement		CholQR-HR with it. refinement	
ρ	κ	$\|A - \tilde{Q}\tilde{R}\|_F$	$\|I - \tilde{Q}^T\tilde{Q}\|_F$	$\|A - \tilde{Q}\tilde{R}\|_F$	$\|I - \tilde{Q}^T\tilde{Q}\|_F$
1e-01	5.1e02	1.1e-15	2.6e-15	1.1e-15	2.6e-15
1e-03	5.2e04	1.0e-15	2.7e-15	1.0e-15	2.8e-15
1e-05	5.2e06	1.1e-15	2.6e-15	1.1e-15	3.0e-15
1e-07	5.1e08	1.1e-15	2.8e-15	1.0e-15	2.6e-15
1e-09	5.2e10	1.1e-15	2.8e-15	1.0e-15	2.8e-15
1e-11	5.2e12	1.1e-15	2.7e-15	1.0e-15	2.8e-15
1e-13	5.0e14	1.1e-15	2.7e-15	$+\infty$	$+\infty$
1e-15	4.7e15	1.1e-15	4.6e-15	$+\infty$	$+\infty$

Errors on tall and skinny matrices ($m = 1000, b = 200$)

Performance of Less-Stable Tall-Skinny QR Algorithms





Leading Order Costs for Full-to-Band

Panel Factorization	Flops	Words	Messages
Householder QR		$O\left(\frac{n^2}{\sqrt{p}}\right)$	$O\left(n \log p\right)$
TSQR	$\frac{4}{3}\frac{n^3}{p}$	$O\left(\frac{n^2}{\sqrt{p}} \log p\right)$	$O\left(\sqrt{p} \log^3 p\right)$
TSQR-HR		$O\left(\frac{n^2}{\sqrt{p}}\right)$	$O\left(\sqrt{p} \log^2 p\right)$

Costs of full-to-band reduction of $n \times n$ matrix
to band matrix with bandwidth $b \ll n$
distributed over p processors in 2D fashion.

Outline

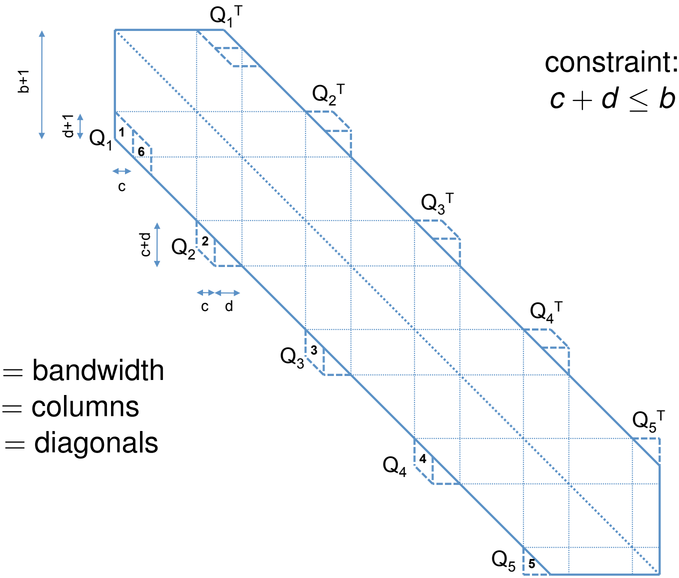
Band-to-Tridiagonal Reduction

Maintaining band structure during orthogonal similarity transformations is trickier

- Annihilating entries within band causes fill-in outside the band
- Bulge-chasing process is required to maintain band structure

- Ideas go back a long way:
  - Rutishauser [Rut63]
  - Schwarz [Sch63]
  - Murata and Horikoshi [MH75]
  - Kaufman [Kau84]
  - Bischof, Lang, and Sun [BLS00]

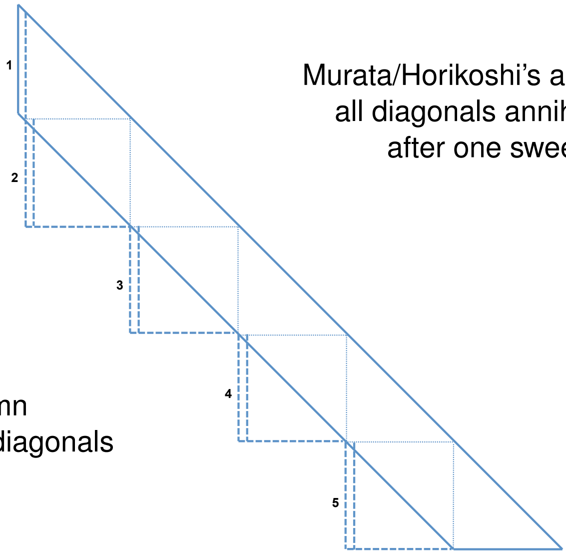
Band-to-Tridiagonal Bulge Chasing



constraint:
$c + d \leq b$

$b =$ bandwidth
$c =$ columns
$d =$ diagonals

1-Sweep Band-to-Tridiagonal [MH75]

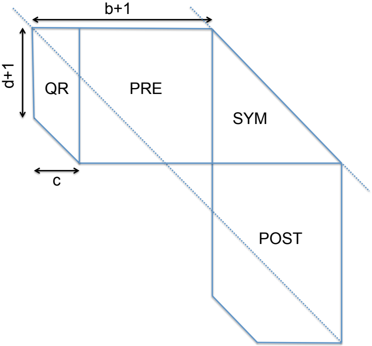


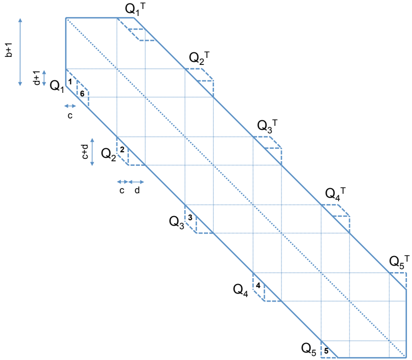
Murata/Horikoshi's algorithm: all diagonals annihilated after one sweep

$c = 1$ column
$d = b - 1$ diagonals

How do we get data re-use?

1. Increase number of columns in parallelogram (c)
   - permits blocking Householder updates: $O(c)$ re-use
   - constraint $c + d \leq b \implies$ trade-off between re-use and progress
   - requires multiple “sweeps”
2. Chase multiple bulges at a time (ω)
   - apply several updates to band while it’s in local memory: $O(\omega)$ re-use
   - bulges cannot overlap, need working set to fit in local memory





Communication-Avoiding SBR

We propose an algorithm that balances the two techniques for getting data re-use (CA-SBR)

Theoretically optimal approach: cut bandwidth in half at every sweep

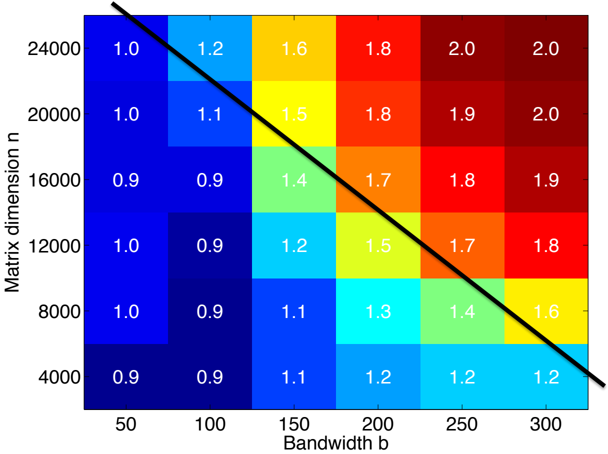
- $\log b$ sweeps

Sequential and shared-memory implementation exist [BDK12]

- number of sweeps is tuning parameter

Performance Results in Shared Memory

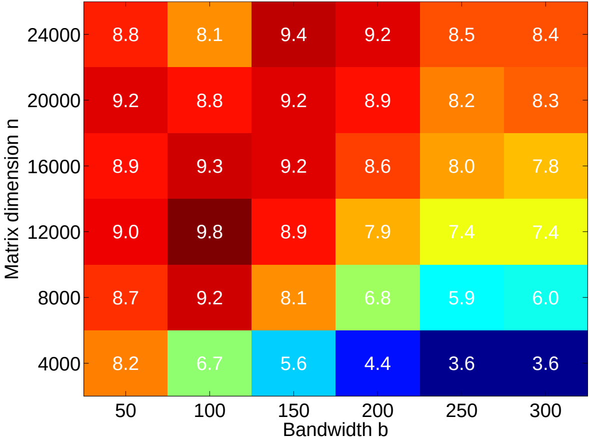
Speedup of sequential CASBR over Intel's Math Kernel Library



Benchmarked on 10-core Intel Westmere [BDK12]

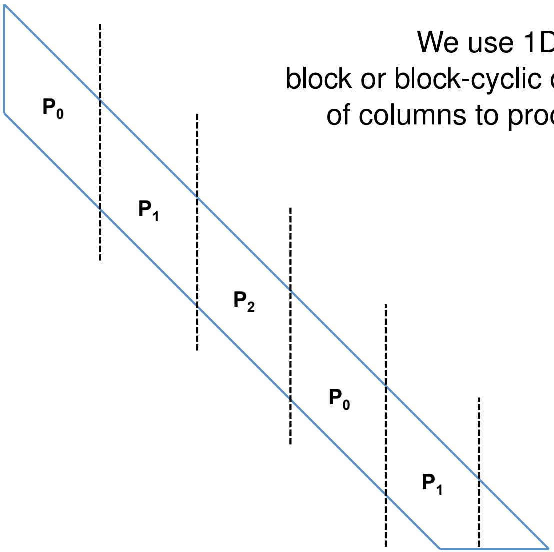
Performance Results in Shared Memory

Speedup of parallel CASBR (10 threads) over sequential CASBR



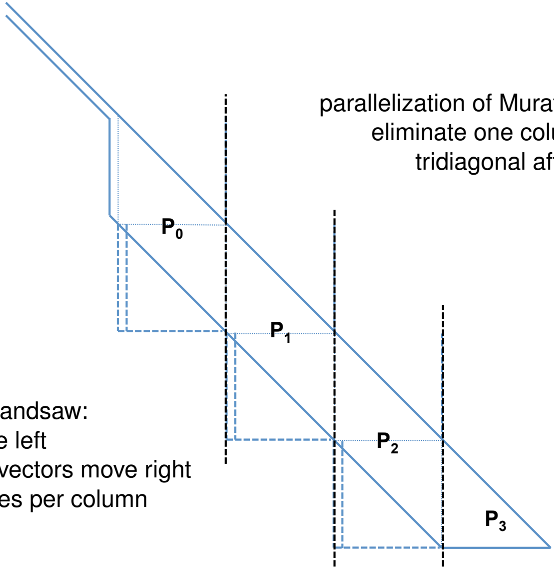
Benchmarked on 10-core Intel Westmere [BDK12]

Distributed-Memory Parallel Distribution



We use 1D
block or block-cyclic distribution
of columns to processors

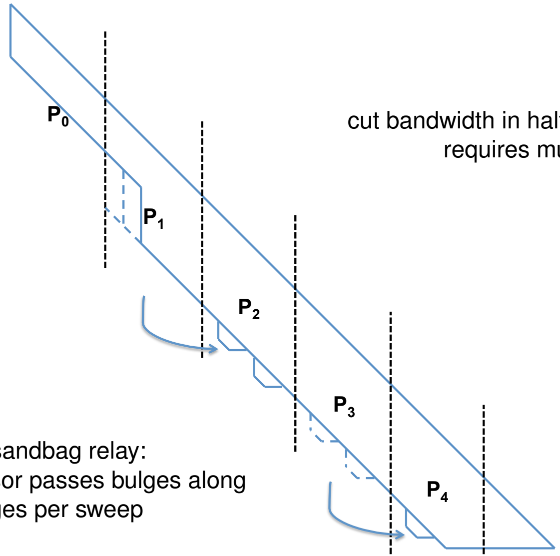
Lang's Algorithm [Lan93, Auc12]



parallelization of Murata/Horikoshi's:
eliminate one column at a time,
tridiagonal after one sweep

works like a bandsaw:
columns move left
Householder vectors move right
$O(1)$ messages per column

Communication-Avoiding SBR [BDK15]



cut bandwidth in half each sweep;
requires multiple sweeps

works like a sandbag relay:
each processor passes bulges along
$O(p)$ messages per sweep

Leading Order Costs for Band-to-Tridiagonal

Algorithm	Flops	Words	Messages
Lang's [Lan93, Auc12]	$O\left(\frac{n^2 b}{p}\right)$	$O(nb)$	$O(n)$
CA-SBR [BDK15]			$O(p \log b)$

Costs of band-to-tridiagonal reduction of band matrix with bandwidth $b \ll n$ distributed over p processors in 1D fashion.

Back-Transformation for Eigenvectors

If only eigenvalues are desired, CA-SBR gives a theoretical net win, but . . .

Back-Transformation for Eigenvectors

If only eigenvalues are desired, CA-SBR gives a theoretical net win, but . . .

. . . what if we want eigenvectors?

- we must accumulate all the orthogonal transformations from the band-to-tridiagonal reduction
- we generate $O(n^2)$ data per sweep
- naively, we need $O(n^3)$ computation per sweep

Back-Transformation for Eigenvectors

If only eigenvalues are desired, CA-SBR gives a theoretical net win, but ...

...what if we want eigenvectors?

- we must accumulate all the orthogonal transformations from the band-to-tridiagonal reduction
- we generate $O(n^2)$ data per sweep
- naively, we need $O(n^3)$ computation per sweep

In order to achieve $O(p \log b) \ll O(n)$ messages, we need to take $O(\log b)$ sweeps instead of 1 sweep

- tradeoff between latency cost and flops/bandwidth cost

Outline

Open Problem

In two-phase tridiagonalization, we compute the following matrices:

$$A = Q_1 B {Q_1}^T = Q_1 (Q_2 T Q_2^T) Q_1^T = Q_1 Q_2 (V \Lambda V^T) {Q_2}^T Q_1^T$$

where A is dense, B is banded, T is tridiagonal, Λ is diagonal

We can compute A, B, T, Λ, V stably and efficiently, we seek $Q_1 Q_2 V$

Open Problem

In two-phase tridiagonalization, we compute the following matrices:

$$A = Q_1 B {Q_1}^T = Q_1 (Q_2 T Q_2^T) Q_1^T = Q_1 Q_2 (V \Lambda V^T) {Q_2}^T Q_1^T$$

where A is dense, B is banded, T is tridiagonal, Λ is diagonal

We can compute A, B, T, Λ, V stably and efficiently, we seek $Q_1 Q_2 V$

Can we compute

- $Q_1 Q_2$ from A and T; or
- Q_2 from B and T

stably (as stable as direct tridiagonalization) and efficiently ($\ll O(n)$ messages)?

Open Problem

Problem: Given A and (similar) T, compute Q such that $A = QTQ^T$

In the first part of the talk, we established a connection between $A = QR$ and $A = LU$

Is there an analogous connection between $A = QTQ^T$ and $A = L\tilde{T}L^T$ (or $A = LDL^T$)?

- $\tilde{T}$ is tridiagonal from Aasen's factorization
- D is block-diagonal from Bunch-Kaufman's factorization

Summary

- We want to solve the dense symmetric eigenvalue problem
  - most/all of the eigenvalues and (possibly) eigenvectors
- We're targeting large distributed-memory parallel machines
  - seeking scalable, communication-efficient algorithms
- Overall approach is two-phase tridiagonalization
- We propose two algorithmic improvements
  - Householder vector reconstruction
  - Communication-avoiding successive band reduction

Thanks!

For more details:

Reconstructing Householder Vectors from Tall-Skinny QR
Grey Ballard, Jim Demmel, Laura Grigori, Mathias Jacquelin, Nick Knight, Hong Diep Nguyen and Edgar Solomonik
Journal on Parallel and Distributed Computing 2015
`http://dx.doi.org/10.1016/j.jpdc.2015.06.003`

Avoiding Communication in Successive Band Reduction
Grey Ballard, Jim Demmel, and Nick Knight
ACM Transactions on Parallel Computing 2015
`http://doi.acm.org/10.1145/2686877`

References I

T. Auckenthaler.
Highly Scalable Eigensolvers for Petaflop Applications.
PhD thesis, Fakultät für Informatik, Technische Universität München, 2012.

G. Ballard, J. Demmel, and N. Knight.
Communication avoiding successive band reduction.
In *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 35–44, New York, NY, USA, 2012. ACM.

Grey Ballard, James Demmel, and Nicholas Knight.
Avoiding communication in successive band reduction.
ACM Transactions on Parallel Computing, 1(2):11:1–11:37, February 2015.

References II

C. Bischof, B. Lang, and X. Sun.
A framework for symmetric band reduction.
ACM Transactions on Mathematical Software, 26(4):581–601, December 2000.

J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.
Communication-optimal parallel and sequential QR and LU factorizations.
SIAM Journal on Scientific Computing, 34(1):A206–A239, 2012.

Azzam Haidar, Raffaele Solcá , Mark Gates, Stanimire Tomov, Thomas Schulthess, and Jack Dongarra.
Leading edge hybrid multi-GPU algorithms for generalized eigenproblems in electronic structure calculations.
In J.M. Kunkel, T. Ludwig, and H. W. Meuer, editors, *Supercomputing*, volume 7905 of *Lecture Notes in Computer Science*, pages 67–80. Springer Berlin Heidelberg, 2013.

References III

Toshiyuki Imamura, Susumu Yamada, and Masahiko Machida.
Development of a high performance eigensolver on the petascale next generation supercomputer system.
In *Proceedings of Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA+ MC2010)*, 2011.

Toshiyuki Imamura, Susumu Yamada, and Masahiko Machida.
Eigen-G: GPU-based eigenvalue solver for real-symmetric dense matrices.
In Roman Wyrzykowski, Jack Dongarra, Konrad Karczewski, and Jerzy Waśniewski, editors, *Parallel Processing and Applied Mathematics*, volume 8384 of *Lecture Notes in Computer Science*, pages 673–682. Springer Berlin Heidelberg, 2014.

L. Kaufman.
Banded eigenvalue solvers on vector machines.
ACM Transactions on Mathematical Software, 10:73–86, 1984.

References IV

B. Lang.
A parallel algorithm for reducing symmetric banded matrices to tridiagonal form.
SIAM Journal on Scientific Computing, 14(6):1320–1338, November 1993.

P. Luszczek, H. Ltaief, and J. Dongarra.
Two-stage tridiagonal reduction for dense symmetric matrices using tile algorithms on multicore architectures.
In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 944–955, Washington, DC, USA, 2011. IEEE Computer Society.

References V

A. Marek, V. Blum, R. Johanni, V. Havu, B. Lang, T. Auckenthaler, A. Heinecke, H.-J. Bungartz, and H. Lederer.
The ELPA library: scalable parallel eigenvalue solutions for electronic structure theory and computational science.
Journal of Physics: Condensed Matter, 26(21), 2014.

K. Murata and K. Horikoshi.
A new method for the tridiagonalization of the symmetric band matrix.
Information Processing in Japan, 15:108–112, 1975.

Daisuke Mori, Yusaku Yamamoto, and Shao-Liang Zhang.
Backward error analysis of the AllReduce algorithm for Householder QR decomposition.
Japan Journal of Industrial and Applied Mathematics, 29(1):111–130, 2012.

References VI

H. Rutishauser.
On Jacobi rotation patterns.
In *Proceedings of Symposia in Applied Mathematics*, volume 15, pages 219–239. AMS, 1963.

H. Schwarz.
Algorithm 183: reduction of a symmetric bandmatrix to triple diagonal form.
Communications of the ACM, 6(6):315–316, June 1963.

Experimental Platforms

Hopper

- Compute nodes:
  2 12-core AMD MagnyCours
- Peak flop rate:
  8.4 Gflops/core
- Memory bandwidth:
  53.9 GB/s
- Interconnect:
  Gemini 3D-torus

Edison

- Compute nodes:
  2 12-core Intel Ivy Bridge
- Peak flop rate:
  19.2 Gflops/core
- Memory bandwidth:
  103.3 GB/s
- Interconnect:
  Aries dragonfly