Communication-Avoiding Nonsymmetric Eigensolver using Spectral Divide & Conquer

Grey Ballard[1] Jim Demmel[1] Ioana Dumitriu[2]

[1]UC Berkeley

[2]University of Washington

Feb 17, 2012

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (CAREER Award #DIG07-10227), as well as the National Science Foundation (Award #DMS-0847661)

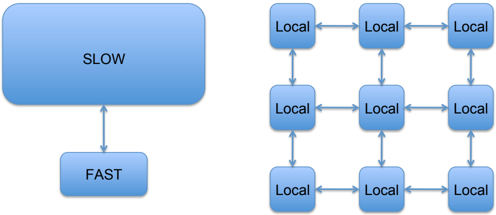
Summary

- Goal: solve nonsymmetric eigenproblem using only communication-efficient algorithms
  - matrix multiplication and QR decomposition
- We take the approach of spectral divide & conquer
  - instead of reduction to Hessenberg and QR iteration
- For communication optimality, we need randomization

Memory Models

By *communication* we mean

- moving data within memory hierarchy on a sequential computer
- moving data between processors on a parallel computer



Sequential

Parallel

Communication Cost Model

Measure communication in terms of *messages* and *words*

- Flop cost: γ
- Cost of message of size *w* words: $\alpha + \beta w$
- Total running time of an algorithm (ignoring overlap):

$$\alpha \cdot (\text{\# messages}) + \beta \cdot (\text{\# words}) + \gamma \cdot (\text{\# flops})$$

- think of α as latency+overhead cost, β as inverse bandwidth

As flop rates continue to improve more quickly than data transfer rates, the relative cost of communication (the first two terms) grows larger

Motivation

Sequential

	Flops	**Words**	**Messages**
Matmul QR/LU/Chol Sym Eig	$O(n^3)$	$O\left(\frac{n^3}{\sqrt{M}}\right)$	$O\left(\frac{n^3}{M^{3/2}}\right)$
NonSym Eig	$O(n^3)$	?	?

n = matrix dimension M = fast memory size

- We know how to avoid communication for matrix multiplication, one-sided factorizations, and the symmetric eigenproblem
  - algorithms match theoretical lower bounds
- It's not clear how to obtain optimal communication efficiency using standard approaches to the nonsymmetric eigenproblem
- We will use alternative approach: spectral divide & conquer

Motivation

Parallel

	Flops	**Words**	**Messages**
Matmul QR/LU/Chol Sym Eig	$O\left(\frac{n^3}{P}\right)$	$O\left(\frac{n^2}{\sqrt{P}}\right)$	$O(\sqrt{P})$
NonSym Eig	$O\left(\frac{n^3}{P}\right)$	?	?

$n =$ matrix dimension $\quad P =$ processors $\quad M = O(n^2/P)$

- We know how to avoid communication for matrix multiplication, one-sided factorizations, and the symmetric eigenproblem
  - algorithms match theoretical lower bounds
- It's not clear how to obtain optimal communication efficiency using standard approaches to the nonsymmetric eigenproblem
- We will use alternative approach: spectral divide & conquer

History of Spectral Divide & Conquer

- Ideas go back to Bulgakov, Godunov, Malyshev [BG88], [Mal89]
- Bai, Demmel, Gu [BDG97]
  - reduced to matmul, QR, generalized QR with pivoting (bug)
- Demmel, Dumitriu, Holtz [DDH07]
  - instead of QR with pivoting, use RURV (randomized URV) (no bug)
  - requires matmul and QR, no column pivoting
- Demmel, Grigori, Hoemmen, Langou [DGHL12]
  - communication-optimal QR decomposition ("CAQR")
- New communication-optimal algorithm
  - use generalized RURV for better rank-detection than [DDH07]
  - use communication-optimal implementations for matrix multiplication and QR as subroutines
  - use randomization in divide and conquer

Overview of Algorithm

One step of divide and conquer:

1. Compute $\left(I + (A^{-1})^{2^k}\right)^{-1}$ implicitly
   - maps eigenvalues of A to 0 and 1 (roughly)
2. Compute rank-revealing decomposition to find invariant subspace
3. Output block-triangular matrix

$$A_{\text{new}} = U^* A U = \begin{bmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{bmatrix}$$

- block sizes chosen so that norm of E_{21} is small
- eigenvalues of A_{11} all lie outside unit circle, eigenvalues of A_{22} lie inside unit circle, subproblems A_{11} and A_{22} solved recursively
- stable, but progress guaranteed only with high probability

Implicit Repeated Squaring

$A_0 = A$, $B_0 = I$

Repeat

1. $$\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = \mathrm{qr}\left(\begin{bmatrix} B_j \\ -A_j \end{bmatrix} \right)$$
2. $A_{j+1} = {Q_{12}}^* \cdot A_j$
3. $B_{j+1} = {Q_{22}}^* \cdot B_j$

until R_j converges

Output is A_k, B_k such that

$$ {A_k}^{-1} B_k = \left(A^{-1} \right)^{2^k} $$

Implicit Repeated Squaring

$A_0 = A$, $B_0 = I$

Repeat

1. $\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = \mathrm{qr}\left(\begin{bmatrix} B_j \\ -A_j \end{bmatrix} \right)$
2. $A_{j+1} = {Q_{12}}^* \cdot A_j$
3. $B_{j+1} = {Q_{22}}^* \cdot B_j$

until R_j converges

Output is A_k, B_k such that

$$ {A_k}^{-1} B_k = \left(A^{-1} \right)^{2^k} $$

- Next step is to compute a rank-revealing decomposition of

$$ \left(I + (A^{-1})^{2^k} \right)^{-1} = \left(I + A_k^{-1} B_k \right)^{-1} = (A_k + B_k)^{-1} A_k $$

Randomized Rank-Revealing QR (RURV)

Use a Haar-distributed random matrix:

1. generate random matrix B with i.i.d. $N(0,1)$ entries
2. $V \cdot R_1 = \text{qr}(B)$
3. $U \cdot R = \text{qr}(A \cdot V^*)$

so that

$$A = U \cdot R \cdot V$$

where U and V are orthogonal and R is upper triangular

- this decomposition is rank-revealing with high probability
- deterministic algorithm involves column pivoting and is communication-inefficient
    - could use tournament pivoting idea

Generalized RURV (GRURV)

We want to compute RURV of matrices of the form $C^{-1}D$:

$$(A_k + B_k)^{-1} A_k$$

We can do it implicitly:

1. $U_2 \cdot R_2 \cdot V = \mathrm{rurv}(D)$
2. $R_1 \cdot U_1 = \mathrm{rq}(U_2{}^* \cdot C)$

so that

$$C^{-1} D = (U_2 R_1 U_1)^{-1} (U_2 R_2 V) = U_1{}^* (R_1^{-1} R_2) V$$

- No inverses computed (we need only the orthogonal matrix U_1)
- Computing $U_1 \cdot A \cdot U_1{}^*$ completes one step of divide and conquer

Overview of Algorithm

One step of divide and conquer:

1. Compute $\left(I + (A^{-1})^{2^k}\right)^{-1}$ implicitly
   - maps eigenvalues of A to 0 and 1 (roughly)
2. Compute rank-revealing decomposition to find invariant subspace
3. Output block-triangular matrix

$$A_{\text{new}} = U^* A U = \begin{bmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{bmatrix}$$

- block sizes chosen so that norm of E_{21} is small
- eigenvalues of A_{11} all lie outside unit circle, eigenvalues of A_{22} lie inside unit circle, subproblems A_{11} and A_{22} solved recursively
- stable, but progress guaranteed only with high probability

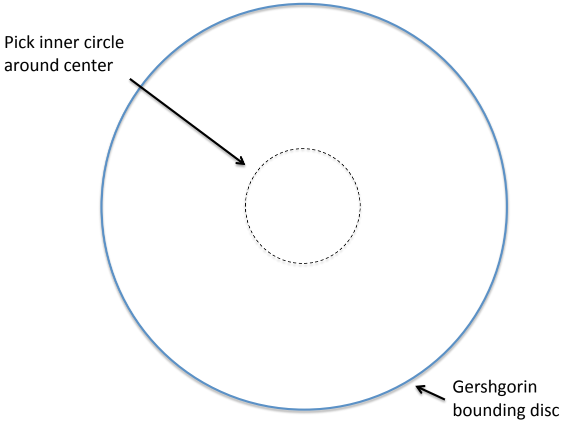
Choosing splitting lines

- Computing $\left(I + \left(A^{-1}\right)^{2^k}\right)^{-1}$ splits spectrum along unit circle
- Use Moebius transformation to split along any circle or line in complex plane
  - set $A_0 = wA + xI$, $B_0 = yA + zI$
- Continue splitting until subproblem fits
  - on one processor or
  - in fast memory

  and use standard algorithms (no extra communication costs)

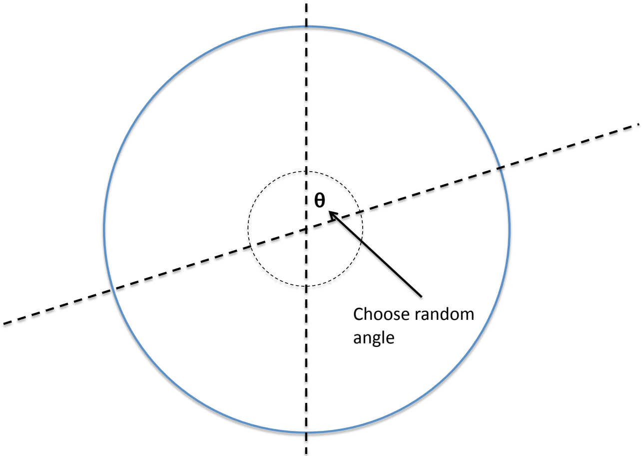
Randomized Bisection

Goals: split spectrum or split bounding region



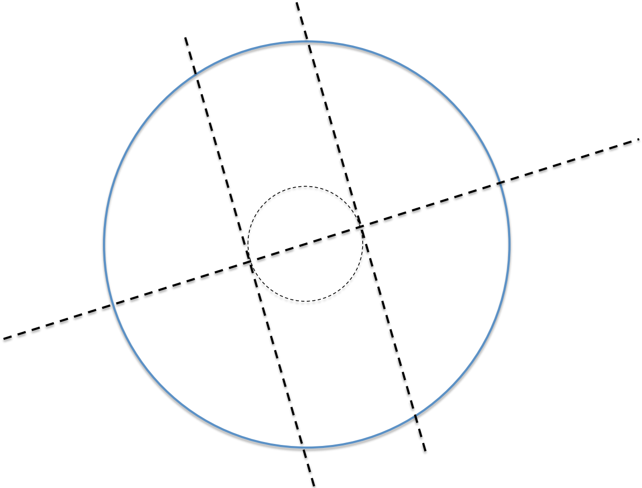
Randomized Bisection

Goals: split spectrum or split bounding region



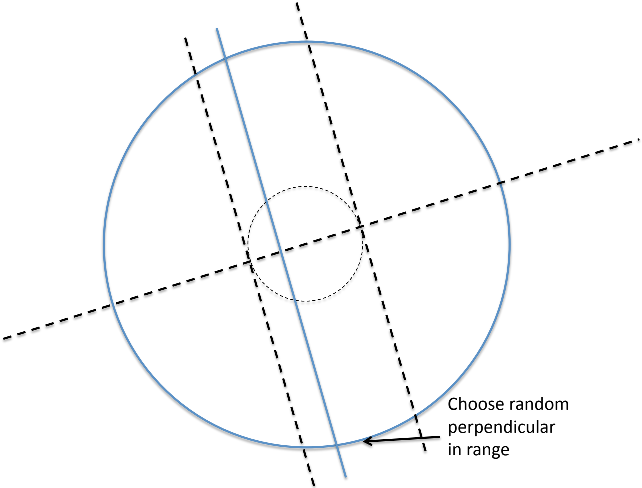
Randomized Bisection

Goals: split spectrum or split bounding region



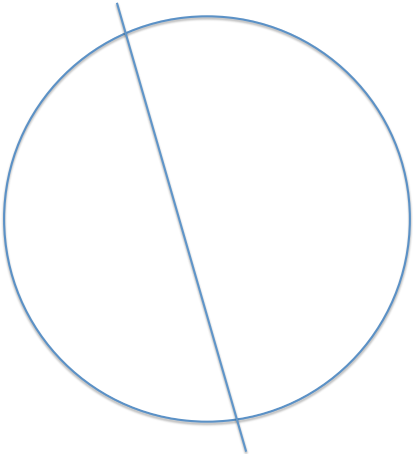
Randomized Bisection

Goals: split spectrum or split bounding region



Randomized Bisection

Goals: split spectrum or split bounding region



Probability of Success

- "Success" means iterative process converges
  - either we split the spectrum, or
  - we narrow down the region containing all the eigenvalues

- If the splitting line does not intersect the $(\epsilon \cdot \|A\|)$-pseudospectrum, then convergence occurs within a constant number of iterations
  - number of iterations depends on smallest relative perturbation that moves an eigenvalue onto splitting line (it does not depend on n)

- For the case of normal matrices, the probability of not intersecting the pseudospectrum with randomized bisection is

$$1 - O(n \cdot \epsilon)$$

(ϵ is machine precision)

Communication Upper Bound (sequential case)

- M = memory size, γ = cost of flop, β = inverse bandwidth, α = latency

Assuming constant number of iterations, cost of one step of divide-and-conquer is

$$C_{D+C}(n) = \alpha \cdot O\left(\frac{n^3}{M^{3/2}}\right) + \beta \cdot O\left(\frac{n^3}{\sqrt{M}}\right) + \gamma \cdot O(n^3)$$

Assuming we split the spectrum by some fraction each time, the total cost of the entire algorithm is asymptotically the same

- same communication complexity as matrix multiplication and QR
- attains lower bound

Communication Upper Bound (parallel case)

- P = # processors, γ = cost of flop, β = inverse bandwidth, α = latency

Assuming constant number of iterations, cost of one step of divide-and-conquer is

$$C_{\mathrm{D+C}}(n, P) = \alpha \cdot O\left(\sqrt{P} \log^2 P\right) + \beta \cdot O\left(\frac{n^2}{\sqrt{P}} \log P\right) + \gamma \cdot O\left(\frac{n^3}{P}\right)$$

By assigning disjoint subsets of processors to two subproblems after each split, subproblems can be solved in parallel yielding the same asymptotic cost for the entire algorithm

- same communication complexity as QR
- attains lower bound (to within logarithmic factors)

Numerical Experiments / Stopping Criteria

Repeat

1. $\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = \text{qr}\left(\begin{bmatrix} B_j \\ -A_j \end{bmatrix}\right)$
2. $A_{j+1} = {Q_{12}}^* \cdot A_j$
3. $B_{j+1} = {Q_{22}}^* \cdot B_j$

until $\frac{\|R_j - R_{j-1}\|}{\|R_{j-1}\|}$ is small

4. $U = \text{GRURV}(A_j + B_j, A_j)$
5. $A_{\text{new}} = U \cdot A \cdot U^* = \begin{bmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{bmatrix}$

check that $\frac{\|E_{21}\|}{\|A\|}$ is small

Repeat

1. $\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \cdot \begin{bmatrix} R_j \\ 0 \end{bmatrix} = \text{qr}\left(\begin{bmatrix} B_j \\ -A_j \end{bmatrix}\right)$
2. $A_{j+1} = {Q_{12}}^* \cdot A_j$
3. $B_{j+1} = {Q_{22}}^* \cdot B_j$
4. $U = \text{GRURV}(A_j + B_j, A_j)$
5. $A_{\text{new}} = U \cdot A \cdot U^* = \begin{bmatrix} A_{11} & A_{12} \\ E_{21} & A_{22} \end{bmatrix}$

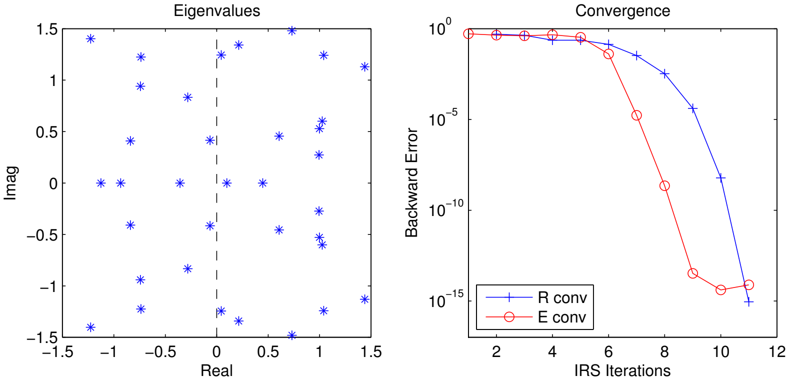
until $\frac{\|E_{21}\|}{\|A\|}$ is small

R conv $= \frac{\|R_j - R_{j-1}\|}{\|R_{j-1}\|}$ is cheaper to compute

E conv $= \frac{\|E_{21}\|}{\|A\|}$ is relative backward error

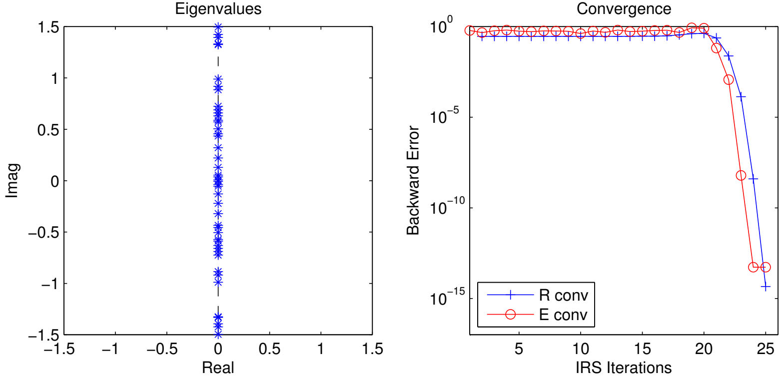
Random Matrix

- Random matrix $A = \texttt{randn}(50)$



Try a tougher matrix

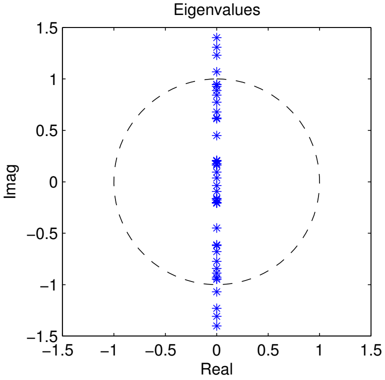
- Half the eigenvalues have real part 10^{-5}
- Other half of eigenvalues have real part -10^{-5}
- Normal matrix

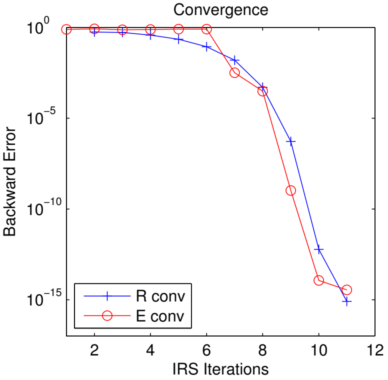


- Imaginary axis worst choice for splitting line

Try a different splitting curve

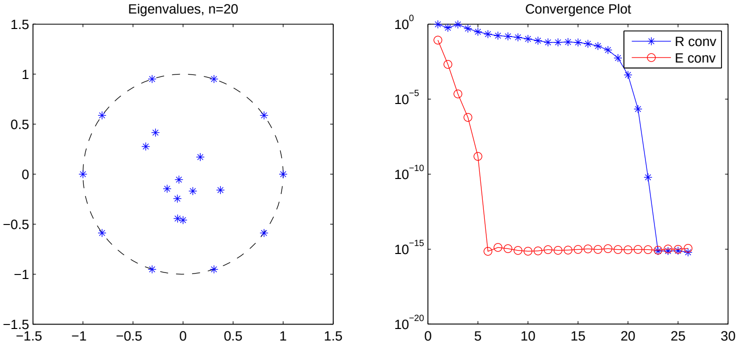
- Half the eigenvalues have real part 10^{-5}
- Other half of eigenvalues have real part -10^{-5}
- Normal matrix





R conv vs E conv

- Half the eigenvalues lie at distance 10^{-5} outside unit circle
- Other half of eigenvalues $< .5$ in absolute value
- Normal matrix



Convergence for Normal Matrices

		Distance to splitting line						
		1e+00	1e-02	1e-04	1e-06	1e-08	1e-10	1e-12
Dimension	**10**	8	15	21	28	35	41	48
	100	8	15	21	28	35	41	48
	500	9	15	22	28	35	42	48
	1000	9	15	22	28	35	42	48
	5000	10	15	22	30	35	42	49
	10000	10	15	22	30	35	42	49

Table: Number of iterations to convergence for normal matrices

- Number of iterations to convergence depends on distance between the splitting line and the nearest eigenvalue
  - not on matrix dimension
- In these experiments, all eigenvalues are at specified distance from splitting line (and all eigenvalues are well-conditioned)
- Convergence means relative backward error of $O(n \cdot \epsilon)$

Conclusions / Summary

- New divide-and-conquer approach communication-optimal
  - minimizes words and messages, in sequential and parallel
  - constant factor more flops than standard algorithms
  - requires randomization
- Convergence depends on distance of splitting line to eigenvalues
- Progress involves
  - splitting the spectrum (reducing the problem size) or
  - splitting the complex plane (localizing the eigenvalues)
- Stability is guaranteed, progress occurs with high probability
- Still working on high performance implementation
  - haven't plugged in fastest QR code, just multithreaded MKL

Thank You!

Please contact me with questions!

ballard@cs.berkeley.edu

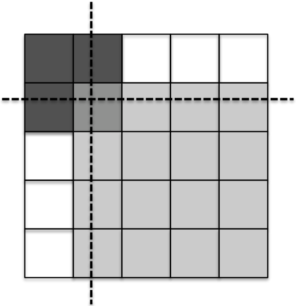
http://www.eecs.berkeley.edu/~ballard

Find links to papers and other resources at the BEBOP webpage:

http://bebop.cs.berkeley.edu/

Parallel subproblem assignment

- Assign number of processors proportional to size of subproblem



- assuming 2D blocked layout, at most one processor owns pieces of both subproblems
- use one of the idle processors to help out
- cost of larger subproblem dominates cost of smaller subproblem

Convergence for Non-normal Matrices

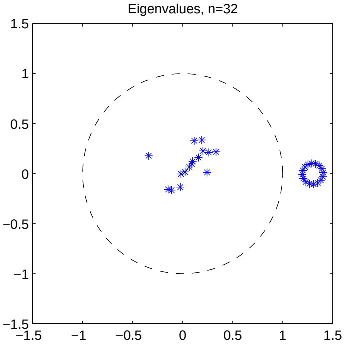
		Distance to splitting line									
		1.0e+00		**1.0e-02**		**1.0e-04**		**1.0e-06**		**1.0e-08**	
Condition #	**1.0e+00**	8	5e-15	15	4e-15	21	2e-14	27	2e-13	35	5e-14
	1.0e+02	8	6e-16	15	1e-14	21	1e-14	27	1e-13	34	3e-14
	1.0e+04	9	2e-13	14	5e-13	22	1e-12	28	2e-12	34	2e-12
	1.0e+06	9	9e-12	14	4e-11	22	6e-10	30	2e-10	32	1e-06
	1.0e+08	9	7e-10	16	9e-09	18	9e-09	18	8e-09	24	5e-09

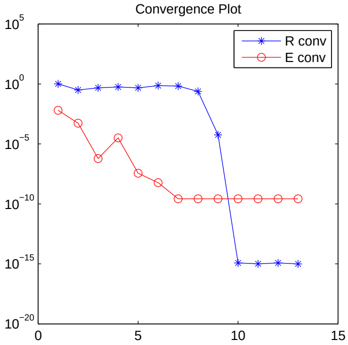
Table: Number of iterations to convergence and relative backward error after convergence for non-normal matrices ($n = 100$)

- In these experiments, all eigenvalues are at specified distance from splitting line and one eigenvalue has specified condition #
- Relative backward error is measured by $\frac{\|E_{21}\|}{\|A\|}$
- In case of large error after convergence, can try restarting

Non-normal Matrix with Jordan block

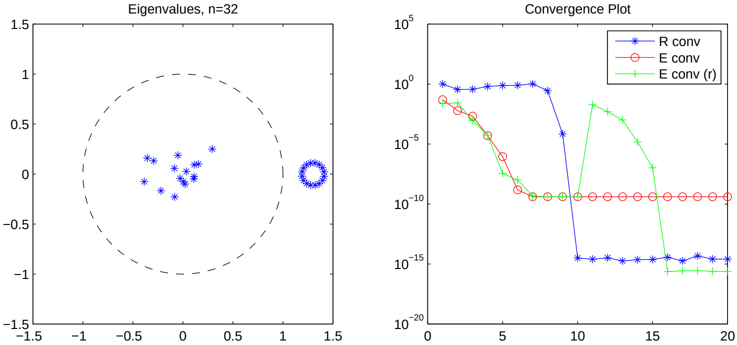
- Half the eigenvalues form Jordan block at 1.3
- Other half of eigenvalues $< .5$ in absolute value





Try restarting

- Half the eigenvalues form Jordan block centered at 1.3
- Other half of eigenvalues $< .5$ in absolute value



- Restart iteration with nearly block triangular matrix

About RURV

If $\sigma_r \sim \sigma_1$ and $\sigma_{r+1} \sim \frac{1}{\text{poly}(n)}\sigma_r$, then with high probability

$$\begin{aligned} \sigma_{\min}(R_{11}) &\geq O\left(\frac{1}{\sqrt{rn}}\right)\sigma_r \\ \sigma_{\max}(R_{22}) &\leq O\left((rn)^2\right)\sigma_{r+1} \end{aligned}$$

- first inequality matches best deterministic URV algorithms
- second inequality is much weaker, but proof is lax (actual bound may be linear)
- repeated squaring will drive σ_r and σ_{r+1} very far apart

About GRURV

- Generalized RURV works for arbitrary products of matrices:

$$A_1^{\pm 1} \cdot A_2^{\pm 1} \cdots A_k^{\pm 1}$$

  - requires one RURV (or RULV) and $k - 1$ QR's (or RQ's)
  - output is $U(R_1^{\pm 1} \cdot R_2^{\pm 1} \cdots R_k^{\pm 1})V$
  - rank-revealing properties same as for RURV (on one matrix)

- Deterministic rank-revealing QR (for one matrix) doesn't suffice in generalized case

Sequential Algorithm for TREVC

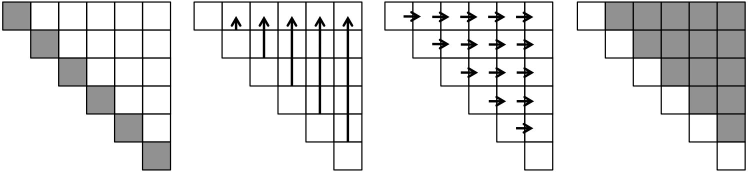
Algorithm 1 Blocked Iterative Algorithm

for $j = 1$ to n/b **do**
  solve $T[j,j] \cdot X[j,j] = X[j,j] \cdot D[j,j]$ for $X[j,j]$
  for $i = j-1$ down to 1 **do**
    $S = 0$
    for $k = i+1$ to j **do**
      $S = S + T[i,k] \cdot X[k,j]$
    end for
    solve $T[i,i] \cdot X[i,j] + S = X[i,j] \cdot D[j,j]$ for $X[i,j]$
  end for
end for

- notation: $T[i,j]$ is a $b \times b$ block
- use blocksize $b = \Theta(\sqrt{M})$ and block-contiguous DS for optimality
- this algorithm ignores need for scaling to prevent under/overflow
- a recursive, cache-oblivious algorithm also achieves optimality
- LAPACK's TREVC solves for one eigenvector at a time

Parallel Algorithm for PTREVC

- Using 2D blocked layout for T on square grid of processors, compute X with same layout
- Iterate over block diagonals, updating trailing matrix each step
  - local computation occurs in gray: (a) and (d)
  - communication occurs along arrows: (b) is a broadcast of X block, (c) is a nearest-neighbor pass of T block



(a) (b) (c) (d)

- Communication costs within log P of optimality
- ScaLAPACK's PTREVC solves for one eigenvector at a time

References I

Z. Bai, J. Demmel, and M. Gu.
An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems.
Numerische Mathematik, 76(3):279–308, 1997.

A. Ya. Bulgakov and S. K. Godunov.
Circular dichotomy of a matrix spectrum.
Sibirsk. Mat. Zh., 29(5):59–70, 237, 1988.

J. Demmel, I. Dumitriu, and O. Holtz.
Fast linear algebra is stable.
Numer. Math., 108(1):59–91, 2007.

J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.
Communication-optimal parallel and sequential QR and LU factorizations.
SIAM Journal on Scientific Computing, 2012.
To appear.

A.N. Malyshev.
Computing invariant subspaces of a regular linear pencil of matrices.
Siberian Math. J., 30:559–567, 1989.