HIGH PERFORMANCE NUMERICAL LINEAR ALGEBRA

Chao Yang
Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA, USA

Solving Dense Linear System of Equations

- Gauss elimination with partial pivoting
- Error analysis
- Iterative refinement
- LAPACK
- Choleksy & LDL^T factorization
- Left-looking, right-looking and Crout algorithms
- Block algorithms
- Parallel Cholesky factorization
- Parallel triangular substitution
- Communication avoiding algorithms
- ScaLAPACK

Linear Least Squres and Eigenvalue Problems

- QR factorization
- The QR algorithm
- Hessenberg reduction
- Bulge chase
- Divide conquer algorithm for symmetric tridiagonal eigenvalue problem

Gauss elimination with partial pivoting (GEPP) for solving Ax=b

- Factorization and partial pivoting

  $PA = LU$, where P is a permutation matrix
- Forward substitution:

  Solve $Ly = Pb$
- Backward substitution

  Solve $Ux = y$

LU without pivoting

- Basic algorithm (recursive)
  - Partition the matrix A as

$$A = \begin{pmatrix} \alpha_{11} & b^T \\ a & \hat{A} \end{pmatrix}$$

  - First step:

$$A = \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & b^T \\ 0 & S \end{pmatrix},$$

where $l = \frac{a}{\alpha_{11}}$, $S = \hat{A} - lb^T$ (Schur complement, rank-1 update)

  - Apply the same procedure recursively on S until it becomes 1×1
- Not accurate in floating point arithmetic,cancellation error in Schur complement

Example (from Demmel's Applied Linear Algebra)

- Assume 3-decimal-digit floating point unit
- $A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}$,
- $L = \begin{pmatrix} 1 & 0 \\ 1/10^{-4} & 1 \end{pmatrix}$, $U = \begin{pmatrix} 10^{-4} & 1 \\ 0 & fl(1 - 10^4 \cdot 1) \end{pmatrix}$
- Multiply L and U back
- $LU = \begin{pmatrix} 1 & 0 \\ 1/10^{-4} & 1 \end{pmatrix} \begin{pmatrix} 10^{-4} & 1 \\ 0 & fl(1 - 10^4 \cdot 1) \end{pmatrix} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 0 \end{pmatrix}$

Partial pivoting

- Algorithm:

```
for j = 1:n-1
    [amax,p(j)] = max(abs(A(j:n,j)));
    p(j) = p(j)+j-1;
    %swap A(j,j:n) with A(p(j),j:n)
    if (p(j)~=j)
        a = A(j,j:n);
        A(j,j:n) = A(p(j),j:n);
        A(j,j:n) = a;
    end
    A(j:n,j) = A(j:n,j)/A(j,j);
    A(j+1:n,j+1:n) = A(j+1:n,j+1:n) -
A(j:n,j)*A(j,j:n)';
end
```

Error Analysis Basics

- Perturbation analysis: If the matrix A is perturbed by ΔA, and the right-hand side is perturbed by δb, what is the maximum amount of error δx we expect from the computed solution x
- Condition number $\kappa(A) = ||A|| \cdot ||A^{-1}||$. In 2-norm, $\kappa(A) = \frac{\lambda_{maxs}}{\lambda_{min}}$. It is an intrinsic property of the problem. Error bound in the computed solution is often related to the perturbation of the data through $\kappa(A)$
- Forward error analysis: analyze floating point error in each step and examine the cumulative effect
- Backward error: treat floating point error as perturbation of the original matrix and/or data. Backward stable if $||\Delta A||/||A||$ and $||\delta b||/||b||$ are on the order of machine precision $O(\epsilon)$

Matrix and vector norms

- Vector norms
  - $\|x\|_\infty = \sqrt{x^T x}$, $\|x\|_1 = \sum_{i=1}^{n} |x_i|$, $\|x\|_\infty = \max_i |x_i|$
  - Equivalence of norms, e.g.: $\|x\|_2 \le \|x\|_1 \le \sqrt{n}\|x\|_2$
- Matrix norm
  - $\|A\|_F = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\mathrm{trace}(A^T A)}$
  - $\|A\| = \min_{\|x\|=1} \|Ax\|$, e.g.
    - $\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$
    - $\|A\|_\infty = \max_i j \sum \, |a_{ij}|$
    - $\|A\|_1 = \max_j \sum_i \, |a_{ij}|$
- $\||X|\| = \|X\|$ holds for $\|\cdot\|_F$, $\|\cdot\|_\infty$, $\|\cdot\|_1$ but not for $\|\cdot\|_2$

Backward error analysis of GEPP

- Residual: $r = b - A\hat{x}$
- Solving $Ax = b$ in floating point arithmetic is equivalent to solving $(A + \Delta A)\hat{x} = b + \delta b$ in exact arithmetic with

$$\omega_\infty = \max\left(\frac{\|\Delta A\|_\infty}{\|A\|_\infty}, \frac{\|\delta b\|_\infty}{\|b\|_\infty}\right) \le \frac{\|r\|_\infty}{\|A\|_\infty \cdot \|\hat{x}\| + \|b\|_\infty}$$
$$\le p(n) \cdot \text{machine precision}$$

- The factor $p(n)$ is related to the growth factor of GEPP defined by $g = \|U\|/\|A\|$. In practice, $p(n)$ often satisfies $p(n) \le n$. In rare cases, $p(n) \sim 2^n$
- Gauss elimination with complete pivoting has a lower growth factor, but too costly in practice

Error bound and condition number estimation

- $\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \le 2\omega_\infty \kappa_\infty(A) = 2\frac{\|r\|_\infty \|A\|_\infty \|A^{-1}\|_\infty}{\|A\|_\infty \cdot \|\hat{x}\| + \|b\|_\infty}$
- Conditioner number estimator: Need to estimate $\|A^{-1}\|_\infty$
  - Solve an optimization problem:
$$\max_{x \neq 0} \frac{\|A^{-1}x\|_\infty}{\|x\|_\infty}$$
  - Convex relaxation
$$\max_{\|x\|_\infty \le 1} \|A^{-1}x\|_\infty$$
- Practical bounds:
$$\frac{\|x-\hat{x}\|_\infty}{\|x\|_\infty} \le \|A^{-1}\|_\infty \frac{\|r\|_\infty}{\|\hat{x}\|_\infty}$$

Iterative refinement and Equibration

- What can we do when $\kappa(A)$ is large, and error in the computed solution is relatively large?
- Use Newton's method to refine the root of $f(x) = Ax - b$, starting from the previously computed solution

For i = 1, 2, …
1. Comput residual $r = Ax_i - b$
2. Solve $Ad = r$;
3. Make correction $x_{i+1} = x_i - d$

- Solve $D_r A D_c (D_c^{-1} x) = D_r b$. Choose D_r and D_c to reduce condition number, balance the matrix elements

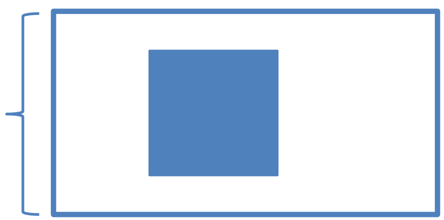
LAPACK

- Assume matrix stored in A and right-hand side stored in B
- Solve system; The solution X overwrites B

CALL SGESV(N, 1, A, LDA, IPIV, B, LDB, INFO)

number of right-hand sides

Leading dimension



permutation vector

- Get reciprocal condition number RCOND of A

CALL SGECON('I', N, A, LDA, ANORM, RCOND, WORK, IWORK, INFO) where

ANORM = SLANGE('I', N, N, A, LDA, WORK) is infinity-norm of A

Cholesky factorization

- If A is symmetric positive definite $A = LL^T$, where L is lower triangular
- Cholesky factorization

$$A = \begin{pmatrix} \alpha_{11} & a^T \\ a & \hat{A} \end{pmatrix} = \begin{pmatrix} 1 & \\ a/\alpha_{11} & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & a^T \\ & \hat{A} - \frac{aa^T}{\alpha_{11}} \end{pmatrix}$$

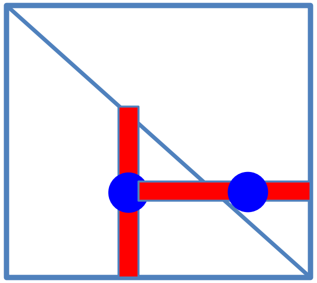
$$A = \begin{pmatrix} 1 & \\ a/\alpha_{11} & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & \\ & I \end{pmatrix} \begin{pmatrix} 1 & a^T/\alpha_{11} \\ & \hat{A} - \frac{aa^T}{\alpha_{11}} \end{pmatrix}$$

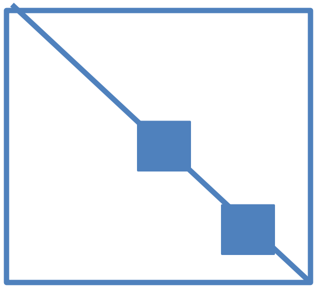
$$= \begin{pmatrix} \sqrt{\alpha_{11}} & \\ a/\sqrt{\alpha_{11}} & I \end{pmatrix} \begin{pmatrix} 1 & \\ & \hat{A} - \frac{aa^T}{\alpha_{11}} \end{pmatrix} \begin{pmatrix} \sqrt{\alpha_{11}} & a^T/\sqrt{\alpha_{11}} \\ & I \end{pmatrix}$$

- No pivot is need, algorithm stable, grow factor moderate

LDLT factorization

- Symmetric indefinite matrices can be factored as $A = LDL^T$, where D may contain negative entries
- $A = \begin{pmatrix} 1 & \\ a/\alpha_{11} & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & \\ & I \end{pmatrix} \begin{pmatrix} 1 & a^T/\alpha_{11} \\ & \hat{A} - \frac{aa^T}{\alpha_{11}} \end{pmatrix}$ may not be numerically stable
- Use Bunch-Kaufman algorithm to create 1x1 or 2x2 pivot, so that the D matrix contains 1x1 and 2x2 blocks.
  - $PAP^T = LDL^T$





Right-looking, Left-looking and Crout

- Right-looking is usually how the algorithm is presented

$$A = \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & b^T \\ 0 & S \end{pmatrix},$$

where $l = \frac{a}{\alpha_{11}}$, the Schur complement update $S = \hat{A} - lb^T$ is to the right of the column being eliminated

- Left-looking: delay the update of the Schur complement until a column of L is to be eliminated.

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & S \end{pmatrix}$$

Assume L_{11}, L_{21}, U_{11}, U_{12} are available, but not S. We now compute only the first column of S:

$$Se_1 = A_{22}e_1 - L_{21}U_{12}e_1$$

Block algorithms

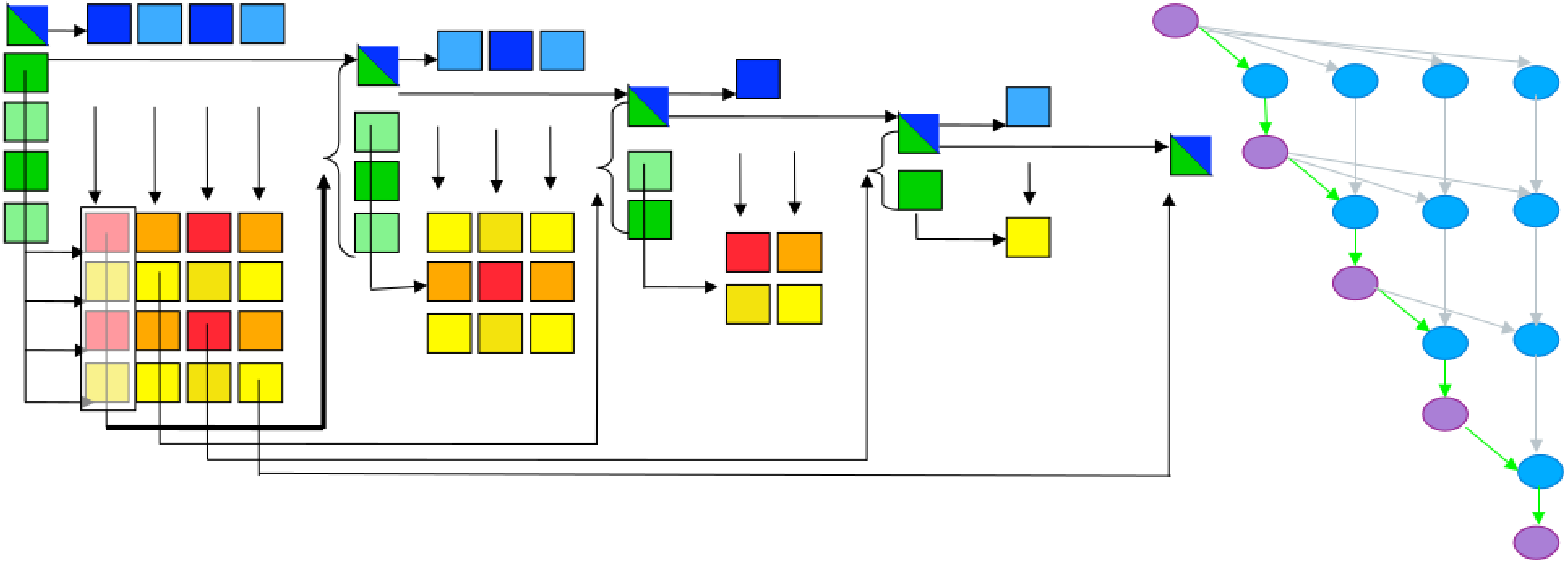
- Block LU factorization

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ A_{21}L_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{11}^{-1}A_{12} \\ & \hat{A} - A_{21}U_{11}^{-1}L_{11}^{-1}A_{12} \end{pmatrix}$$

- Blocking factorization to improve memory locality
- Leverage BLAS3 performance
- Block size can be tuned

Parallelization for shared memory machines

- LAPACK (thread parallelism): rely on threaded BLAS, limited scalability (because BLAS is used to multiply matrix blocks that may be too small for parallelism)
- Exploit concurrency at block (tile level) level (triple loop)



Courtesy: J. Dongarra

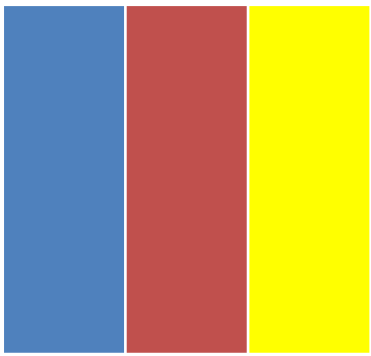
PLASMA & MAGMA

- PLASMA: Parallel Linear Algebra Software for Multi-core Architectures
- http://icl.cs.utk.edu/plasma
- Dynamic DAG (direct acyclic graph) scheduling (using QUARK)
- Fine granularity (to ensure load balance)
- Block data layout to promote locality

- MAGMA:
- http://icl.cs.utk.edu/magma
- For heterogeneous systems (e.g. systems that contain GPUs)

Parallel factorization for distributed memory machines

- Data decomposition

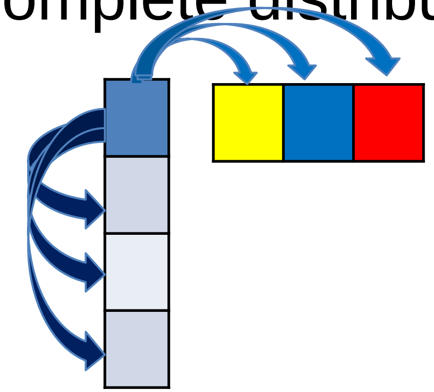
1D 

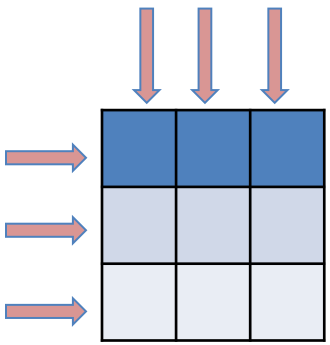
2D 

0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3
0	1	0	1	0	1	0	1
2	3	2	3	2	3	2	3

- Block cyclic to achieve load balance (no processor should be sitting idle while others complete distributed tasks)
- Right-looking (fan-out)

$$A = \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \begin{pmatrix} \alpha_{11} & b^T \\ 0 & S \end{pmatrix}$$





Algorithm

for k = 1 to n − 1
    broadcast $\{a_{kj}$: *j* ∈ mycols, *j* ≥ *k*} in process column
    if *k* ∈ mycols then
        for *i* ∈ myrows, *i* > *k*
            $l_{ik} = a_{ik}/a_{kk}$ { multipliers }
        end
    end
    broadcast $\{l_{ik}$: i ∈ myrows, i > k} in process row
    for j ∈ mycols, j > k
        for i ∈ myrows, i > k,
            $a_{ij} = a_{ij} - l_{ik}a_{kj}$ { update }
        end
    end
end

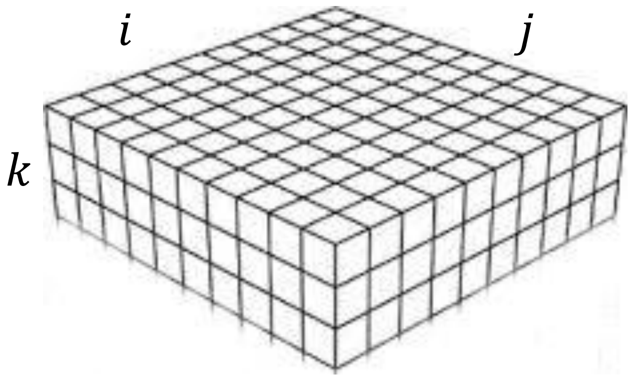
Cost analysis

- Flops:
  - Updating by each process at step k requires about $(n-k)^2/p$ operations
  - Summing over $n-1$ steps

    $T \approx t_c \sum_{k=1}^{n-1} \frac{(n-k)^2}{p} \approx \frac{t_c n^3}{3p}$
- Communication:
  - data broadcast at step k along each process row/column is about $(n-k)/\sqrt{p}$
  - Bandwidth: $\Omega(\log p \frac{n^2}{\sqrt{p}})$ latency: $\Omega(n \log p)$

How far are we from optimal performance

- Metric for optimal (lower bound for communication volume and frequency) See J. Demmel's SC14 turtorial http://www.cs.berkeley.edu/~demmel/SC14_tutorial/Demmel_SC14_Tutorial_final_v2_2pp.pdf
  - Let M be "fast" memory (e.g. cache) size per processor
  - **#words moved (per processor) = Ω(#flops (per processor) / $M^{1/2}$)**
  - **#messages sent = Ω(#flops (per processor) / $M^{3/2}$)**
- Schur complement updated by 2.5 matrix-matrix multiplication algorithm (require extra memory)



- Initially processor $P(i,j,0)$ owns $A(i,j)$ and $B(i,j)$ each of size $n\sqrt{\frac{c}{P}} \times n\sqrt{\frac{c}{P}}$
- $P(i,j,0)$ broadcasts $A(i,j)$ and $B(i,j)$ to $P(i,j,k)$
- Processors at level *k* perform 1/*c*-th of SUMMA, i.e. 1/*c*-th of $\sum_m A(i,m)B(m,j)$

(3) Sum-reduce partial sums $\sum_m A(i,m)B(m,j)$ along k-axis so $P(i,j,0)$ owns $C(i,j)$

Reported Performance improvement

- 2.5D SUMMA GEMM on 16,384 nodes of BlueGene/P with c=16, i.e., 32x32x16 processor grid
  - 12x speedup for matrices of size $n = 8{,}192$, 95% reduction in communication
  - 2.7x speedup fo rmatrices of size $n = 131{,}072$
- LU on 16,384 BlueGene/P nodes, for $n = 131{,}072$, observe 2x speedup using 2.5D algorithm with and without pivoting

ScaLAPACK

- Extension of LAPACK for distributed-memory parallel computers
- Build on top of BLACS (Basic Linear Algebra Communication Subroutine) and PBLAS (parallel BLAS)
- Example:

```
CALL PDGEMM( TRANSA, TRANSB, M, N, K, ALPHA, A, IA,
JA, DESC_A, B, IB, JB, DESC_B, BETA, C, IC, JC,
DESC_C )
```

Array descriptors: **DESC_A, DESC_B, DESC_C** specifies

- ✓the communication (BLACS) context/group (no inter-context comm)
- ✓#of rows/columns in the distributed matrix,
- ✓row/col block size
- ✓Leading dimension

References

- J. W. Demmel, M. T. Heath, and H. A. van der Vorst, Parallel numerical linear algebra, Acta Numerica 2:111-197, 1993
- G. A. Geist and C. H. Romine, LU factorization algorithms on distributed-memory multiprocessor architectures, SIAM J. Sci. Stat. Comput. 9:639-649, 1988
- L. Grigori, J. Demmel, and H. Xiang, CALU: A communication optimal LU factorization algorithm, SIAM J. Matrix Anal. Appl. 32:1317-1350, 2011
- B. A. Hendrickson and D. E. Womble, The torus-wrap mapping for dense matrix calculations on massively parallel computers, SIAM J. Sci. Stat. Comput. 15:1201-1226, 1994

- J. M. Ortega, Introduction to Parallel and Vector Solution of Linear Systems, Plenum Press, 1988
- J. M. Ortega and C. H. Romine, The ijk forms of factorization methods II: parallel systems, Parallel Comput. 7:149-162, 1988
- Y. Robert, The Impact of Vector and Parallel Architectures on the Gaussian Elimination Algorithm, Wiley, 1990
- E. Solomonik and J. Demmel, Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms, 17th Euro-Par Conf. on Parallel Processing, LNCS 6853, Springer, 2011
- S. A. Vavasis, Gaussian elimination with pivoting is P-complete, SIAM J. Disc. Math. 2:413-423, 1989

Linear Least Squares Problem

- $\min_x \|b - Ax\|_2$, A is $m \times n$, with $m > n$
- Application in (high-dimensional) data/curve fitting, tomography, statistical estimation (inference)
- Weighted least square: replace 2-norm with another norm induced by a positive definite matrix W
- A can be full-rank or rank-deficient (numerically)

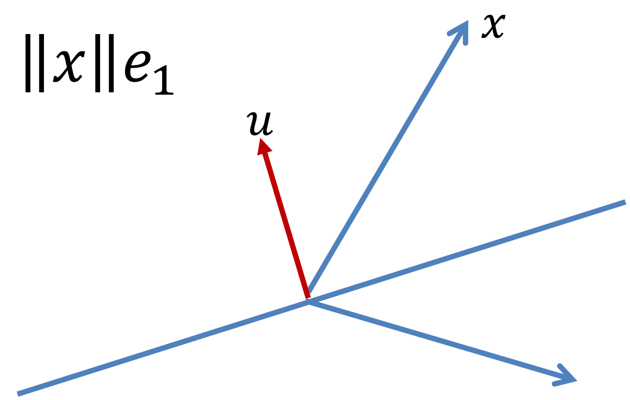
Basic Strategies

- Normal equation:
  - Optimality condition:
    $$\nabla[\|b - Ax\|^2] = 0 \rightarrow A^T(b - Ax) = 0 \rightarrow A^T Ax = A^T b$$
  - Not preferred due to the squaring of the condition number $\kappa(A^T A) = \kappa(A)^2$
- QR factorization
  - $A = QR$, where $Q^T Q = I$, R is upper triangular
  - $\|b - Ax\| = \|Q^T b - Rx\|$
  - Rank-revealing QR $AP = QR$, diagonal of R decreasing
- Singular Value Decomposition
  - $A = U\Sigma V^T$, $U^T U = I$, $VV^T = I$, Σ diagonal with possibly zeros on the diagonal
  - $\|b - Ax\| = \|U^T b - \Sigma(Vx)\|$

QR factorization

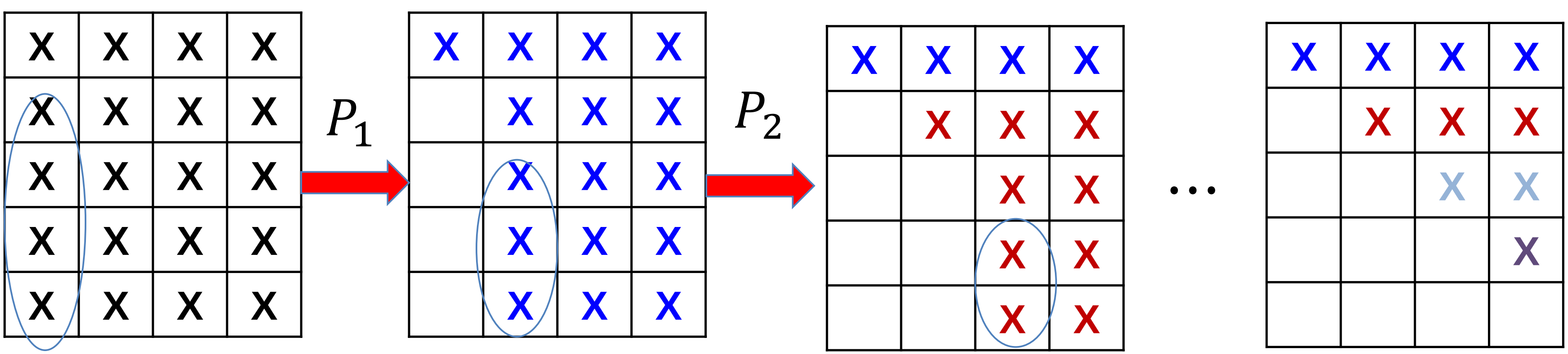
- Householder reflector

$$Px = (I - 2uu^T)x = \|x\|e_1$$



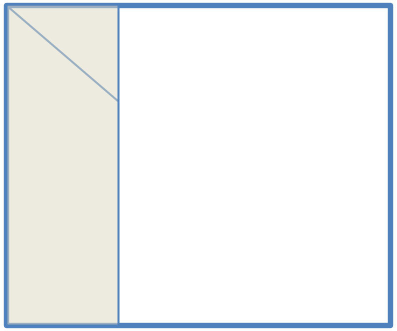
- Successive elimination

$$P = P_1 P_2 \cdots$$



Block Householder transform

- Accumulate several householder transformation into a single block low-rank update
$$(I - \alpha u_1 u_1^T)(I - \alpha u_2 u_2^T) \cdots = I - YTY^T = I - YW^T$$
- Obtain u_1, u_2,… by constructing and apply Householder reflectors from/to the first few columns of A
- Apply the transform $I - YTY^T$ using GEMM (BLAS3) to subsequent columns of A
$$\hat{A} = \hat{A} - YT(Y^T \hat{A})$$



Other ways to perform QR

- Given's rotation

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\ \theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2 + y^2} \\ 0 \end{pmatrix}$$

  Applying Given's rotation (BLAS1 operation)

- Gram-Schmidt

$$q \leftarrow (I - QQ^T)a_j,\ q \leftarrow q/||q||$$

  BLAS2 operation

- Cholesky QR

$$A^T A = LL^T,\ Q = AL^{-T}$$

  Less stable numerically

Rank-revealing QR

- QR with column pivoting $AP = QR$

  Choose the column with the largest norm in the trailing (unfinished) part of the matrix

- Rank-revealing QR (M. Gu, SIAM J. Sci. Comp, vol 17, 1996)
  - Additional permutations to make the algorithm more stable
- Randomized algorithm

Talk skinny QR (TSQR)

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} Q_1 R_1 \\ Q_2 R_2 \\ Q_3 R_3 \\ Q_4 R_4 \end{pmatrix} = \begin{pmatrix} Q_1 & & & \\ & Q_2 & & \\ & & Q_3 & \\ & & & Q_4 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{pmatrix}$$

$$\begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{pmatrix} = \begin{pmatrix} \hat{Q}_1 \hat{R}_1 \\ \hat{Q}_2 \hat{R}_2 \end{pmatrix} = \begin{pmatrix} \hat{Q}_1 & \\ & \hat{Q}_2 \end{pmatrix} \begin{pmatrix} \hat{R}_1 \\ \hat{R}_2 \end{pmatrix}$$

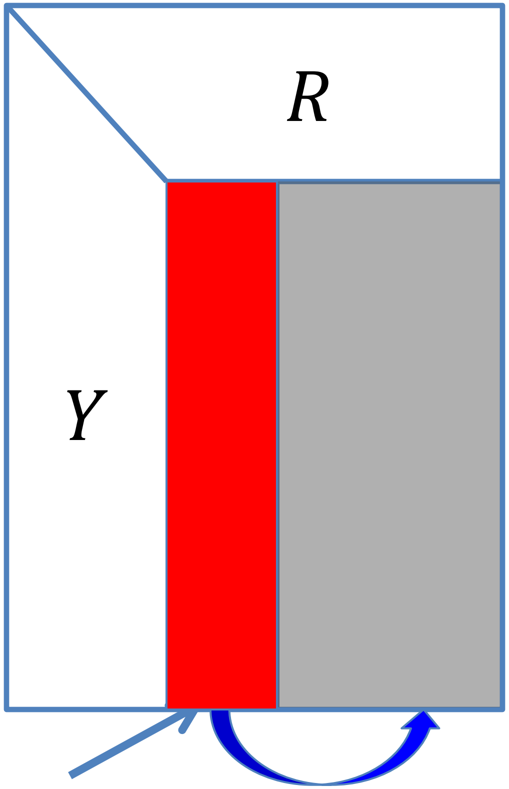
$$\begin{pmatrix} \hat{R}_1 \\ \hat{R}_2 \end{pmatrix} = \tilde{Q} \tilde{R}$$

flops: $\frac{2mn^2}{P} + \frac{2}{3} n^3 \log P$ # **words**: $\frac{n^2}{2} \log P$ # **messages**: log P

Communication avoiding QR

- Based on TSQR $A = \begin{pmatrix} Q_1 R_{11} & \hat{A} \end{pmatrix}$
- Right-looking update (GEMM)
- For details: see LAWN204

	CAQR	ScaLAPACK
# flops	$\frac{2mn^2}{P} - \frac{2n^3}{3P}$	same
# words	$\sqrt{\frac{mn^3}{P}} \log P - \frac{1}{4}\sqrt{\frac{n^5}{mP}} \log \frac{nP}{m}$	same
# messages	$\sqrt{\frac{nP}{m}} \log^2\left(\frac{mP}{n}\right) \log\left(P\sqrt{\frac{mP}{n}}\right)$	



Panel factorization by TSQR

Eigenvalue problem

- Standard $Ax = \lambda x$
- Generalized $Ax = \lambda Bx$
- A can be symmetric, nonsymmetric, B often symmetric positive definite

The QR algorithm

- Hessenberg reduction: $AV = VH$
- Shifted QR algorithm:

for j = 1, 2, … until convergence
    μ=select_shift(H);
    QR factorization: $H - \mu I = QR$;
    $H^{+} = RQ + \mu I = Q^{*}HQ$;
    $V \leftarrow VQ$;
end

Hessenberg reduction

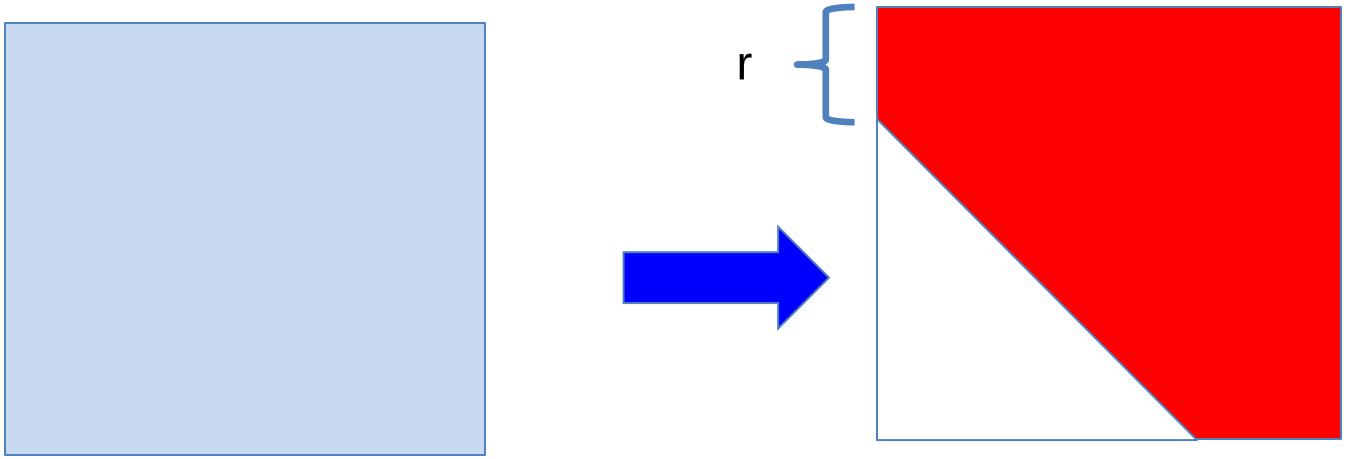
- Use Householder transformation
- Apply from both sides (two sided transformation)

$$Q_1 A = \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \end{pmatrix} \qquad A_1 = Q_1 A Q_1^T = \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \\ 0 & X & X & X & X \end{pmatrix}$$

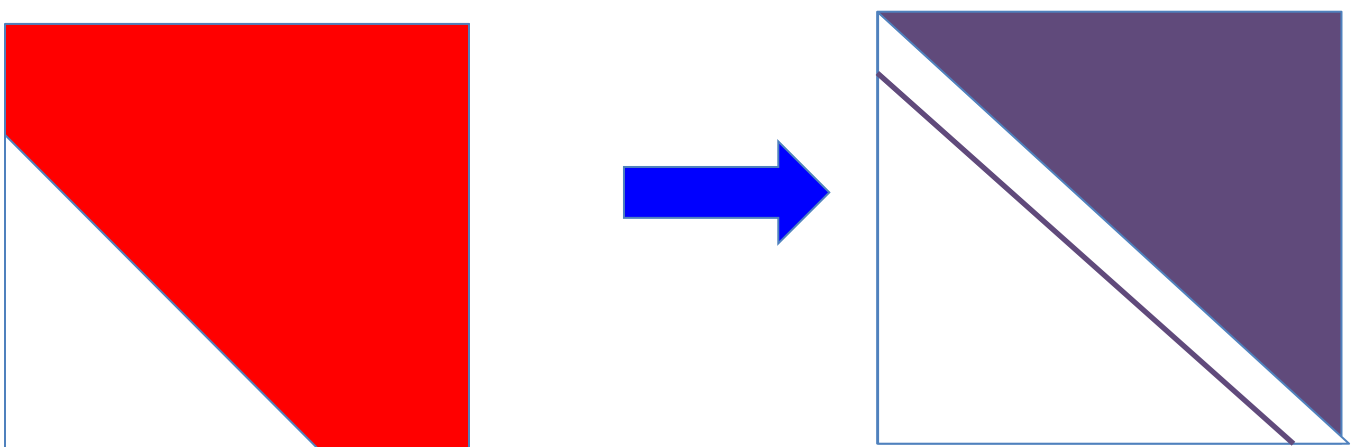
$$Q_2 A_1 = \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & X & X & X \end{pmatrix} \qquad Q_2 A_1 Q_2^T = \begin{pmatrix} X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \\ 0 & 0 & X & X & X \\ 0 & 0 & X & X & X \end{pmatrix}$$

2-stage algorithm and parallelization

- Reduce to r-Hessenberg form first



- From r-Hessenberg to Hessenberg



Bulge chase

Symmetric tridiagonal eigensolver