Orthogonalization of a set of vectors and the QR factorization

Laura Grigori

EPFL and PSI

September 24/31, 2024

Plan

Orthogonalization processes
- Background on QR factorization

Gram-Schmidt (GS) orthogonalization process

Cholesky-QR

Householder QR
- Compact representation

Communication avoiding QR factorization
- TSQR: QR factorization of a tall skinny matrix

Summary of cost and stability of the different algorithms

BLAS: optimized routines for basic linear algebra

- Industry standard interface
  www.netlib.org/blas, www.netlib.org/blas/blast–forum
- Vendors, others supply optimized implementations
- BLAS1: 15 different operations
  - vector-vector operations: dot product, saxpy (y=a*x+y)
  - not very efficient since few operations performed on each element of the vectors, cost of data transfer through memory hierarchies is important
- BLAS2: 25 different operations
  - matrix-vector operations: matrix vector multiply, etc
  - slightly faster than BLAS1
- BLAS3 : 9 different operations
  - matrix-matrix operations: matrix matrix multiply, etc
  - better usage of caches, n^3 operations for matrices of dimensions $n \times n$, potentially much faster than BLAS2 and BLAS1

LAPACk and ScaLAPACK for linear algebra

- LAPACK : linear algebra package
  www.netlib.org/lapack,scalapack
- ScaLAPACK: scalable linear algebra package
  - designed for distributed memory machines
- Vendors provide those libraries optimized
- Both libraries rely as much as possible on BLAS3

Plan

The QR factorization

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, its QR factorization is

$$\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}} = \begin{pmatrix} \mathbf{Q} & \tilde{\mathbf{Q}} \end{pmatrix} \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} = \mathbf{QR}$$

where $\hat{\mathbf{Q}} \in \mathbb{R}^{m \times m}$ is orthogonal and $\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$ is upper triangular.
The thin factorization has $\mathbf{Q} \in \mathbb{R}^{m \times n}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$.

If $\mathbf{A}$ has full rank, the factorization QR is essentialy unique (modulo signs of diagonal elements of $\mathbf{R}$).

- $\mathbf{A}^T\mathbf{A} = \mathbf{R}^T\mathbf{R}$ is a Cholesky factorization and $\mathbf{A} = \mathbf{A}\mathbf{R}^{-1}\mathbf{R}$ is a QR factorization.
- $\mathbf{A} = \mathbf{QD} \cdot \mathbf{DR}$, $\mathbf{D} = diag(\pm 1)$ is a QR factorization.

Importance of orthogonalization

- Used in many different applications, as:
  - Solving least squares problems
  - Computing an orthogonal basis of a set of vectors as in Krylov subspace methods (they will be discussed in later lectures)
  - Computing the low rank approximation of a matrix

Solving least squares problems

Given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $rank(\mathbf{A}) = n$, vector $\mathbf{b} \in \mathbb{R}^{m \times 1}$, the unique solution to $\arg\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2$ is

$$\mathbf{x} = \mathbf{A}^{+}\mathbf{b}, \quad \mathbf{A}^{+} = (\mathbf{A}^{T}\mathbf{A})^{-1}\mathbf{A}^{T}$$

Using the QR factorization of $\mathbf{A}$

$$\mathbf{A} = \hat{\mathbf{Q}}\hat{\mathbf{R}} = \begin{pmatrix} \mathbf{Q} & \tilde{\mathbf{Q}} \end{pmatrix} \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} \tag{1}$$

$$\begin{aligned} ||\mathbf{r}||_2^2 &= ||\mathbf{Ax} - \mathbf{b}||_2^2 = || \begin{pmatrix} \mathbf{Q} & \tilde{\mathbf{Q}} \end{pmatrix} \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} \mathbf{x} - \mathbf{b}||_2^2 \\ &= || \begin{pmatrix} \mathbf{R} \\ 0 \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{Q}^T \\ \tilde{\mathbf{Q}}^T \end{pmatrix} b||_2^2 = || \begin{pmatrix} \mathbf{Rx} - \mathbf{Q}^T\mathbf{b} \\ \tilde{\mathbf{Q}}^T\mathbf{b} \end{pmatrix} ||_2^2 \\ &= ||\mathbf{Rx} - \mathbf{Q}^T\mathbf{b}||_2^2 + ||\tilde{\mathbf{Q}}^T\mathbf{b}||_2^2 \end{aligned}$$

$$\implies \arg\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\| = \arg\min_{\mathbf{x}} \|\mathbf{Rx} - \mathbf{Q}^T\mathbf{b}\|.$$

Algorithms for orthogonalization

Many algorithms available. In this class we study:

- Gram-Schmidt process (GS): Classical GS (CGS), Modified GS (MGS)
- Cholesky-QR
- Householder QR: QR factorization based on Householder transformations

Two different cases arise:

- The vectors to be orthogonalized are known all at once
  - all algorithms discussed can be used
- The vectors to be orthogonalized are given progressively
  - MGS, CGS could be used, Householder QR with modifications
  - Cholesky-QR cannot be used

Algorithms for orthogonalization

Many algorithms available. In this class we study:

- Gram-Schmidt process (GS): Classical GS (CGS), Modified GS (MGS)
- Cholesky-QR
- Householder QR: QR factorization based on Householder transformations

Two different cases arise:

- The vectors to be orthogonalized are known all at once
  - all algorithms discussed can be used
- The vectors to be orthogonalized are given progressively
  - MGS, CGS could be used, Householder QR with modifications
  - Cholesky-QR cannot be used

Numerical stability

We will discuss the numerical stability of these algorithms as:

- Loss of orthogonality $\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\|$ and condition number of the basis $\kappa(\mathbf{Q})$
- Accuracy of the factorization $\|\mathbf{A} - \mathbf{QR}\|$

Data distribution

We consider first the the case when $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \gg n$
Matrix A distributed using a 1D distribution by blocks of rows, that is:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} \text{ is distributed on } \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

that is $\mathbf{A}_I \in \mathbb{R}^{m/P \times n}$ is assigned to processor P_i, $i = 1 : 4$

- Note that processors are numbered 1 to P, while in MPI they are numbered 0 to $P - 1$,
- Matrices are 1-indexed (as in Matlab), while they are 0-indexed in Python

Plan

Gram-Schmidt (GS) orthogonalization process

Given set of linearly independent vectors $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n]$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \gg n$.
Construct an orthogonal basis $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ such that $\mathbf{A} = \mathbf{QR}$, $\mathbf{Q} \in \mathbb{R}^{m \times n}$, $\mathbf{R} \in \mathbb{R}^{n \times n}$.

For each a_j **do**

1. Given P_{j-1} projector on $\mathrm{span}(\mathbf{Q}(:, 1 : j-1))^{\perp}$, compute $\mathbf{q}_j \perp \mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$ as

$$\mathbf{q}_j = P_{j-1}\mathbf{a}_j$$

2. Normalize $\mathbf{q}_j$

End For

Orthogonal projector

$$
\begin{aligned}
P_{j-1} &= \mathbf{I} - \mathbf{Q}(:, 1:j-1)(\mathbf{Q}(:, 1:j-1)^T \mathbf{Q}(:, 1:j-1))^{-1} \mathbf{Q}(:, 1:j-1)^T \\
&= \mathbf{I} - \mathbf{Q}(:, 1:j-1)\mathbf{Q}(:, 1:j-1)^\dagger
\end{aligned}
$$

- Expensive to compute, hence two different projectors very used, leading to:
  - Classical Gram-Schmidt
  - Modified Gram-Schmidt

Projectors P_j used in Gram-Schmidt

- **Classical Gram-Schmidt (CGS)**: BLAS-2 matrix-vector operations
  - one synchronization for each new vector $\mathbf{a}_j$

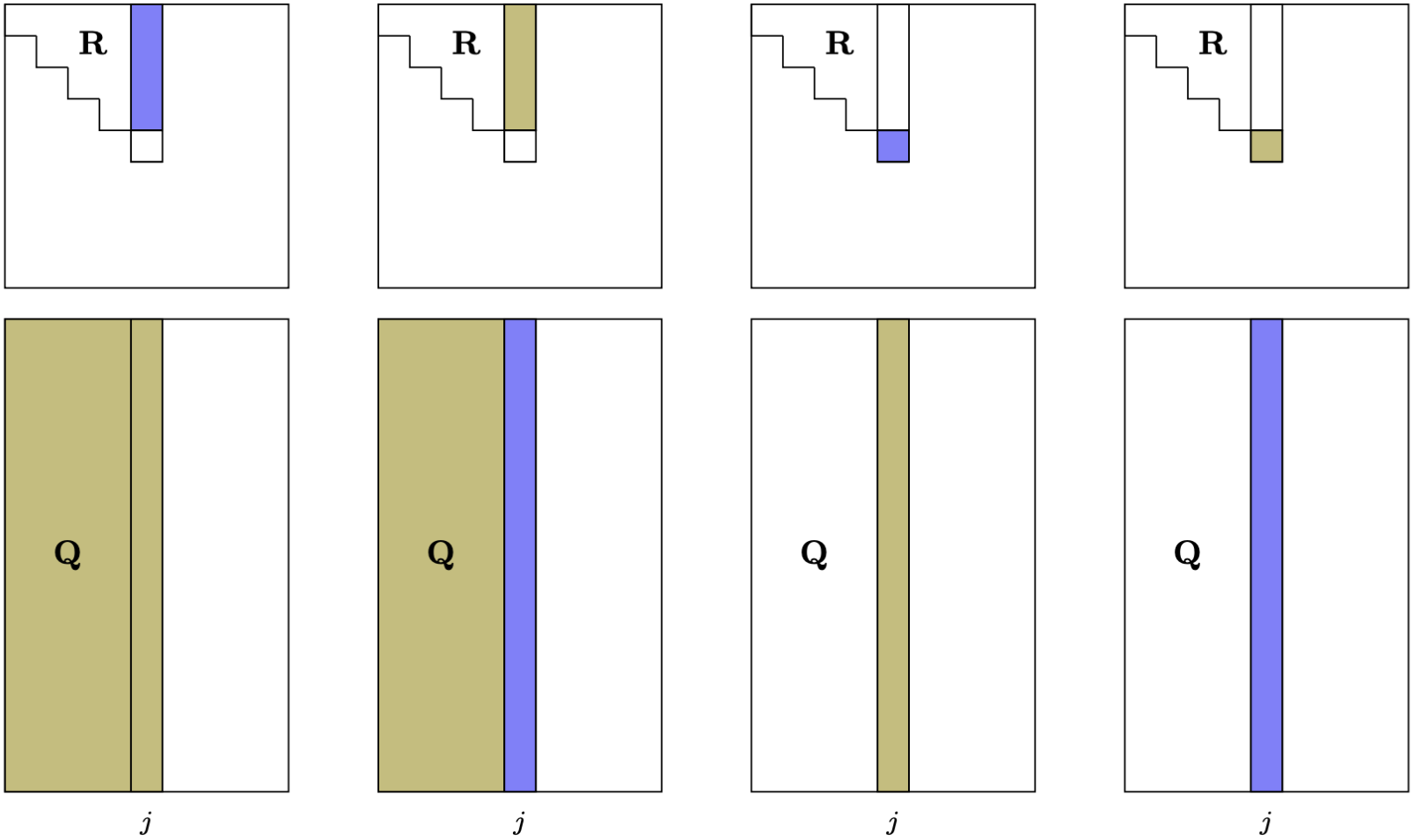
$$P_{j-1} = I - \mathbf{Q}(:, 1:j-1)\mathbf{Q}(:, 1:j-1)^T$$

  - Numerical stability: assume $O(\epsilon)\kappa^2(\mathbf{A}) < 1$

$$\|I - \mathbf{Q}^T\mathbf{Q}\|_2 = O(\varepsilon)\kappa^2(\mathbf{A})$$

Note: ε machine precision, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\kappa(\mathbf{A})$ is condition number of $\mathbf{A}$

Classical Gram-Schmidt



(a) Comp. projections (line 6)

(b) Update from left (line 7)

(c) Compute norm (line 8)

(d) Normalize $\mathbf{Q}(:, j)$ (line 9)

CGS: algorithm

Algorithm 1 Classical Gram-Schmidt (Left-looking, BLAS-2 version)

Require: $\mathbf{A}$ is an $m \times n$ matrix with $m \geq n$
Assert: $\mathbf{QR} = \mathbf{A}$ where $\mathbf{R}$ is upper triangular

1: **function** $[\mathbf{Q}, \mathbf{R}] = \mathrm{CGS}(\mathbf{A})$
2: $\quad \mathbf{R} = \mathbf{0}$
3: $\quad \mathbf{R}(1,1) = \|\mathbf{A}(:,1)\|_2$
4: $\quad \mathbf{Q}(:,1) = \mathbf{A}(:,1) \,/\, \mathbf{R}(1,1)$
5: $\quad$ **for** $j = 2$ to n **do**
6: $\quad\quad \mathbf{R}(1:j-1,j) = \mathbf{Q}(:,1:j-1)^T \cdot \mathbf{A}(:,j)$
7: $\quad\quad \mathbf{Q}(:,j) = \mathbf{A}(:,j) - \mathbf{Q}(:,1:j-1) \cdot \mathbf{R}(1:j-1,j)$
8: $\quad\quad \mathbf{R}(j,j) = \|\mathbf{Q}(:,j)\|_2$
9: $\quad\quad \mathbf{Q}(:,j) = \mathbf{Q}(:,j) \,/\, \mathbf{R}(j,j)$
10: $\quad$ **end for**
11: **end function**

Parallel Classical Gram-Schmidt

Algorithm 2 Parallel Classical Gram-Schmidt

Require: $\mathbf{A}$ is an $m \times n$ matrix 1D-row-distributed over P processors
Assert: $\mathbf{QR} = \mathbf{A}$ where $\mathbf{R}$ is upper triangular and $\mathbf{Q}$ is $m \times n$
Assert: $\mathbf{R}$ is stored redundantly on all processors and $\mathbf{Q}$'s distribution matches $\mathbf{A}$

1: **function** $[\mathbf{Q}, \mathbf{R}] = \text{1D-CGS}(\mathbf{A})$
2: $I = \text{MyProcID}$
3: $\mathbf{R} = \mathbf{0}$
4: $\overline{\beta} = \|\mathbf{A}_I(:,1)\|_2^2$
5: All-reduce $\overline{\beta}$ over all processors, take square root, and store in $\mathbf{R}(1,1)$
6: $\mathbf{Q}_I(:,1) = \mathbf{A}_I(:,1) \,/\, \mathbf{R}(1,1)$
7: **for** $j = 2$ to n **do**
8: $\bar{\mathbf{r}} = \mathbf{Q}_I(:,1:j-1)^T \cdot \mathbf{A}_I(:,j)$
9: All-reduce $\bar{\mathbf{r}}$ over all processors, store in $\mathbf{R}(1:j-1,j)$
10: $\mathbf{Q}_I(:,j) = \mathbf{A}_I(:,j) - \mathbf{Q}_I(:,1:j-1) \cdot \mathbf{R}(1:j-1,j)$
11: $\overline{\beta} = \|\mathbf{Q}_I(:,j)\|_2^2$
12: All-reduce $\overline{\beta}$ over all processors, take square root, and store in $\mathbf{R}(j,j)$
13: $\mathbf{Q}_I(:,j) = \mathbf{Q}_I(:,j) \,/\, \mathbf{R}(j,j)$
14: **end for**
15: **end function**

Cost of parallel CGS

- Computation: each iteration j does a matrix-vector product with $\mathbf{Q}_l$ with m/P rows:

$$\gamma \cdot \frac{2mn^2}{P}$$

- Communication:
  all-reduce at each iteration j with data of size $j-1$ and data of size 1

$$\alpha \cdot O(n \log P) + \beta \cdot O(n^2 + n \log P)$$

Projectors P_j used in Gram-Schmidt

- **Modified Gram-Schmidt (MGS)** : BLAS-1 vector-vector operations
  - $(j-1)$ synchronizations for each vector w_j

$$P_{j-1} = (I - q_{j-1}q_{j-1}^T)\dots(I - q_1 q_1^T)$$

  - Numerical stability: assume $O(\varepsilon)\kappa(W) < 1$,

$$\|I - \mathbf{Q}^T\mathbf{Q}\|_2 = O(\varepsilon)\kappa(W)$$

$\rightarrow$ better numerical stability, but poor efficiency

Note: ε machine precision, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\kappa(\mathbf{A})$ is condition number of $\mathbf{A}$

Projectors P_j used in Gram-Schmidt

- **Modified Gram-Schmidt (MGS)** : BLAS-1 vector-vector operations
  - $(j-1)$ synchronizations for each vector w_j

$$P_{j-1} = (I - q_{j-1}q_{j-1}^T)\dots(I - q_1 q_1^T)$$

  - Numerical stability: assume $O(\varepsilon)\kappa(W) < 1$,

$$\|I - \mathbf{Q}^T\mathbf{Q}\|_2 = O(\varepsilon)\kappa(W)$$

$\rightarrow$ better numerical stability, but poor efficiency

Note: ε machine precision, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\kappa(\mathbf{A})$ is condition number of $\mathbf{A}$

Parallel Modified Gram-Schmidt

Algorithm 3 Parallel Modified Gram-Schmidt

Require: $\mathbf{A}$ is an $m \times n$ matrix 1D-row-distributed over P processors
Assert: $\mathbf{QR} = \mathbf{A}$ where $\mathbf{R}$ is upper triangular and $\mathbf{Q}$ is $m \times n$
Assert: $\mathbf{R}$ is stored redundantly on all procs and $\mathbf{Q}$'s distribution matches $\mathbf{A}$

1: **function** $[\mathbf{Q}, \mathbf{R}] = \text{1D-MGS}(\mathbf{A})$
2: $I =$ MyProcID
3: $\mathbf{R} = \mathbf{0}$
4: $\mathbf{Q}_I = \mathbf{A}_I$
5: **for** $j = 1$ to n **do**
6: **for** $i = 1$ to $j - 1$ **do**
7: $\overline{\rho} = \mathbf{Q}_I(:, i)^T \cdot \mathbf{Q}_I(:, j)$
8: All-reduce $\overline{\rho}$ over all processors, store in $\mathbf{R}(i, j)$
9: $\mathbf{Q}_I(:, j) = \mathbf{Q}_I(:, j) - \mathbf{Q}_I(:, i) \cdot \mathbf{R}(i, j)$
10: **end for**
11: $\overline{\beta} = \|\mathbf{Q}_I(:, j)\|_2^2$
12: All-reduce $\overline{\beta}$ over all processors, take square root, and store in $\mathbf{R}(j, j)$
13: $\mathbf{Q}_I(:, j) = \mathbf{Q}_I(:, j) \ / \ \mathbf{R}(j, j)$
14: **end for**
15: **end function**

Cost of parallel MGS

- Computation:
  $n^2/2$ inner iterations
  inner loop: dot product and axpy of vectors of local dimension m/P

$$\gamma \cdot \frac{2mn^2}{P}$$

- Communication:
  $n^2/2$ inner iterations
  inner loop: all-reduce of a single element

$$\alpha \cdot O(n^2 \log P) + \beta \cdot O(n^2 \log P)$$

Plan

Cholesky-QR

- **Cholesky-QR** : BLAS-3 matrix-matrix operations
  Compute $\mathbf{A} = \mathbf{QR}$ as:
  1. Compute the Cholesky factorization of $\mathbf{A}^T\mathbf{A}$ as $\mathbf{A}^T\mathbf{A} = \mathbf{G} = \mathbf{R}^T\mathbf{R}$
  2. Compute the orthogonal factor as $\mathbf{Q} = \mathbf{AR}^{-1}$
- Numerical stability: assume $O(\epsilon)\kappa^2(\mathbf{A}) < 1$, then

$$\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\|_2 = O(\epsilon)\kappa^2(\mathbf{A})$$

Cholesky-QR: algorithm

Algorithm 4 Parallel Cholesky-QR

Require: $\mathbf{A}$ is an $m \times n$ matrix 1D-row-distributed over P processors
Assert: $\mathbf{QR} = \mathbf{A}$ where $\mathbf{R}$ is upper triangular and $\mathbf{Q}$ is $m \times n$
Assert: $\mathbf{R}$ is stored redundantly on all processors and $\mathbf{Q}$'s distribution matches $\mathbf{A}$

1: **function** $[\mathbf{Q}, \mathbf{R}] = \textsc{ParCholQR}(\mathbf{A})$
2: $I = \text{MyProcID}$
3: $\overline{\mathbf{G}} = \mathbf{A}_I^T \mathbf{A}_I$
4: All-reduce $\overline{\mathbf{G}}$ over all processors, store in $\mathbf{G}$
5: $\mathbf{R} = \text{Cholesky}(\mathbf{G})$ ▷ Performed redundantly on all processors
6: $\mathbf{Q}_I = \mathbf{A}_I \mathbf{R}^{-1}$ ▷ Local triangular solve with multiple RHS
7: **end function**

Cost of Cholesky-QR

- Computation:
  matrix multiplication + Cholesky factorization:

$$\gamma \cdot \frac{2mn^2}{P} + O(n^3)$$

- Communication:
  all-reduce of Cholesky factor

$$\alpha \cdot O(\log P) + \beta \cdot O(n^2) \text{ when } n^2 \geq P$$

Plan

Householder transformation

The Householder matrix

$$\mathbf{H} = \mathbf{I}_m - \frac{2}{\mathbf{y}^T\mathbf{y}}\mathbf{y}\mathbf{y}^T$$

has the following properties:

- is symmetric and orthogonal, $\mathbf{H}^2 = \mathbf{H}^T\mathbf{H} = \mathbf{I}_m$,
- is independent of the scaling of $\mathbf{y}$,
- it reflects a vector $\mathbf{a}_1$ about the hyperplane $span(\mathbf{y})^\perp$

$$\mathbf{H} \cdot \mathbf{a}_1 = \mathbf{a}_1 - \frac{2\mathbf{y}^T\mathbf{a}_1}{\mathbf{y}^T\mathbf{y}}\mathbf{y} = \mathbf{a}_1 - \xi\mathbf{y}$$

- By taking $\xi = 1$ we obtain $\mathbf{y} = \mathbf{a}_1 - \mathbf{H} \cdot \mathbf{a}_1$

Householder for the QR factorization

We look for a Householder matrix that allows to annihilate the elements of a vector $\mathbf{a}_1$, except first one.

$$\mathbf{H} \cdot \mathbf{a}_1 = \tilde{\beta}\mathbf{e}_1, \quad \tilde{\beta} = \pm\|\mathbf{a}_1\|_2$$

With the choice of sign made to avoid cancellation when computing $\mathbf{y}(1) = \tilde{\alpha} - \tilde{\beta}$ (where $\mathbf{y}(1), \mathbf{a}_1(1) = \tilde{\alpha}$ are the first elements of vectors $\mathbf{y}, \mathbf{a}_1$ respectively), we have

$$\begin{aligned} \mathbf{y} &= \mathbf{a}_1 - \tilde{\beta}\mathbf{e}_1, \quad \tilde{\beta} = -sign(\mathbf{a}_1(1))\|\mathbf{a}_1\|_2, \\ \mathbf{H} &= I - \tau\mathbf{y}\mathbf{y}^T, \tau = \frac{2}{\mathbf{y}^T\mathbf{y}} \end{aligned}$$

Householder based QR factorization

- Let $\mathbf{a}_1$ be the first column of $\mathbf{A}$ and the Householder matrix be:

$$\mathbf{H}_1 = \mathbf{I}_m - \tau_1 \mathbf{y}_1 \mathbf{y}_1^T$$

- Apply the same reasoning to subsequent columns.
- Consider the j-th column $\mathbf{a}_j$ and $\mathbf{H}_j$ that annihilates all the elements below the diagonal,

$$\mathbf{H}_j \cdot \mathbf{a}_j = (\mathbf{I}_m - \tau_j \mathbf{y}_j \mathbf{y}_j^T) \cdot \mathbf{a}_j = \begin{bmatrix} \mathbf{a}_j(1:j-1) \\ \pm \|\mathbf{a}_j(j:m)\| \\ \mathbf{0}_{m-j} \end{bmatrix}, \quad \text{where } \mathbf{y}_j = \begin{bmatrix} \mathbf{0}_{j-1} \\ \mathbf{a}_j(1) + sgn(\mathbf{a}_j(1)) \|\mathbf{a}_j(j:m)\|_2 \\ \mathbf{a}_j(j+1:m) \end{bmatrix}$$

Householder based QR factorization

$$\mathbf{A} = \begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}$$

$$\mathbf{H}_3\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \mathbf{H}_3\mathbf{H}_2 \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix} = \mathbf{H}_3 \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{pmatrix} = \hat{\mathbf{R}}$$

So we have

$$\begin{aligned} \mathbf{A} &= \mathbf{H}_1 \cdots \mathbf{H}_n \begin{bmatrix} \mathbf{R} \\ \mathbf{0}_{n-m,m} \end{bmatrix} = \hat{\mathbf{Q}}\hat{\mathbf{R}}, \\ \hat{\mathbf{Q}} &= (I - \tilde{\beta}_1 \mathbf{y}_1 \mathbf{y}_1^T) \ldots (I - \tilde{\beta}_{n-1} \mathbf{y}_{n-1} \mathbf{y}_{n-1}^T)(I - \tilde{\beta}_n \mathbf{y}_n \mathbf{y}_n^T) \end{aligned}$$

$\#flops = 2n^2(m - n/3)$

Error analysis of the QR factorization

Theorem ([N.J.Higham, 2002])

Let $\tilde{\mathbf{R}} \in \mathbb{R}^{m\times n}$ *be the computed factor of* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *obtained by using Householder transformations. Then there is an orthogonal* $\hat{\mathbf{Q}} \in \mathbb{R}^{m\times m}$ *such that*

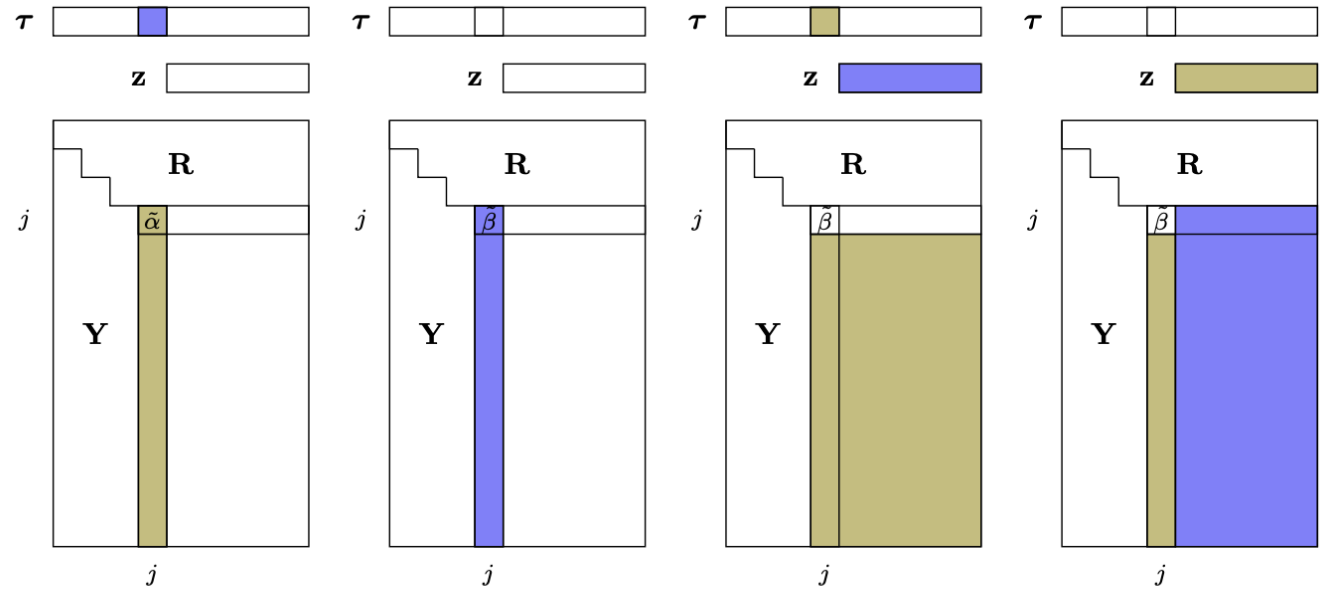
$$\mathbf{A} + \Delta\mathbf{A} = \hat{\mathbf{Q}}\tilde{\mathbf{R}}, \text{ where } \|\Delta\mathbf{a}_j\|_2 \leq O(mn\varepsilon)\|\mathbf{a}_j\|_2, \quad j = 1:n,$$

where ε *is machine precision,* $mn\varepsilon < 1$, a_j *denotes the j-th column of A.*

Loss of orthogonality:

$\|\mathbf{I} - \hat{\mathbf{Q}}^T\hat{\mathbf{Q}}\| = O(\varepsilon)$, with no constraints on $\kappa(\mathbf{A})$ (unconditionally stable)

Householder QR - algorithm



(a) Norm of subcolumn (lines 5 and 6)

(b) Scale subcolumn (lines 7 and 8)

(c) Trailing matrix-vector product (line 9)

(d) Outer product update (lines 10 and 11)

Householder QR - algorithm

Algorithm 5 Householder QR

Require: $\mathbf{A}$ is an $m \times n$ matrix with $m \geq n$

Assert: $\hat{\mathbf{Q}}\hat{\mathbf{R}} = \mathbf{A}$ where $\hat{\mathbf{R}}$ is upper triangular $(m \times n)$, $\mathbf{R}$ is its upper triangular part $(n \times n)$, and $\hat{\mathbf{Q}} = (\mathbf{I} - \tau_1 \mathbf{y}_1 \mathbf{y}_1^T) \cdots (\mathbf{I} - \tau_n \mathbf{y}_n \mathbf{y}_n^T)$

function $[\mathbf{Y}, \boldsymbol{\tau}, \mathbf{R}] = \textsc{HouseholderQR}(\mathbf{A})$

  $\mathbf{Y} = \mathbf{0}$, $\mathbf{R} = \mathbf{0}$

  for $j = 1$ to n **do**

    ▷ Compute the Householder vector

    $\tilde{\alpha} = \mathbf{A}(j, j)$

    $\tilde{\beta} = -\operatorname{sgn}(\tilde{\alpha}) \cdot \|\mathbf{A}(j : m, j)\|_2$

    $\boldsymbol{\tau}(j) = (\tilde{\beta} - \tilde{\alpha})/\tilde{\beta}$

    $\mathbf{Y}(j{+}1 : m, j) = 1/(\tilde{\alpha} - \tilde{\beta}) \cdot \mathbf{A}(j{+}1 : m, j)$

    $\mathbf{R}(j, j) = \tilde{\beta}$

    ▷ Apply the Householder transformation to the trailing matrix

    $\mathbf{z} = \boldsymbol{\tau}(j) \cdot (\mathbf{A}(j, j{+}1 : n) + \mathbf{Y}(j{+}1 : m, j)^T \cdot \mathbf{A}(j{+}1 : m, j{+}1 : n))$

    $\mathbf{R}(j, j{+}1 : n) = \mathbf{A}(j, j{+}1 : n) - \mathbf{z}$

    $\mathbf{A}(j{+}1 : m, j{+}1 : n) = \mathbf{A}(j{+}1 : m, j{+}1 : n) - \mathbf{Y}(j{+}1 : m, j) \cdot \mathbf{z}$

  end for

end function

Computational complexity

- Flops per iterations
  - Dot product: $2(m-j)(n-j)$
    $\mathbf{z} = \boldsymbol{\tau}(j) \cdot (\mathbf{A}(j, j{+}1 : n) + \mathbf{Y}(j{+}1 : m, j)^T \cdot \mathbf{A}(j{+}1 : m, j{+}1 : n))$
  - Outer product and Subtraction: $2(m-j)(n-j)$
    $\mathbf{A}(j{+}1 : m, j{+}1 : n) = \mathbf{A}(j{+}1 : m, j{+}1 : n) - \mathbf{Y}(j{+}1 : m, j) \cdot \mathbf{z}$
- Flops of Householder-QR

$$\sum_{j=1}^{n} 4(m-j)(n-j) = 4\sum_{j=1}^{n}(mn - j(m+n) + j^2)$$
$$\approx 4mn^2 - 4(m+n)n^2/2 + 4n^3/3 = 2mn^2 - 2n^3/3$$

Applying/obtaining the $\mathbf{Q}$ factor

- Apply directly $\hat{\mathbf{Q}}$ to a vector
- Sometimes necessary to obtain $\mathbf{Q}$, the first n columns of $\hat{\mathbf{Q}}$, as

$$\hat{\mathbf{Q}}(1:m, 1:n) = (\mathbf{I} - \tau_1 \mathbf{y}_1 \mathbf{y}_1^T) \cdots (\mathbf{I} - \tau_n \mathbf{y}_n \mathbf{y}_n^T) \mathbf{I}_{m,n}.$$

Parallelization of Householder QR

- Not discussed in the lectures
- Expensive in terms of communication since each iteration requires computing the norm of a vector and updating the trailing matrix
- Thus number of messages is $O(n \log_2(P))$

Compact representation

Storage efficient representation for Q [Schreiber and Loan, 1989]

$$\mathbf{Q} = \mathbf{H}_1\mathbf{H}_2\ldots\mathbf{H}_n = (\mathbf{I} - \tilde{\beta}_1\mathbf{y}_1\mathbf{y}_1^T)\ldots(\mathbf{I} - \tilde{\beta}_n\mathbf{y}_n\mathbf{y}_n^T) = \mathbf{I} - \mathbf{Y}\mathbf{T}\mathbf{Y}^T$$

Example for $k = 2$

$$\mathbf{Y} = (\mathbf{y}_1|\mathbf{y}_2), \quad \mathbf{T} = \begin{pmatrix} \tilde{\beta}_1 & -\tilde{\beta}_1\mathbf{y}_1^T\mathbf{y}_2\tilde{\beta}_2 \\ 0 & \tilde{\beta}_2 \end{pmatrix}$$

Example for combining two compact representations

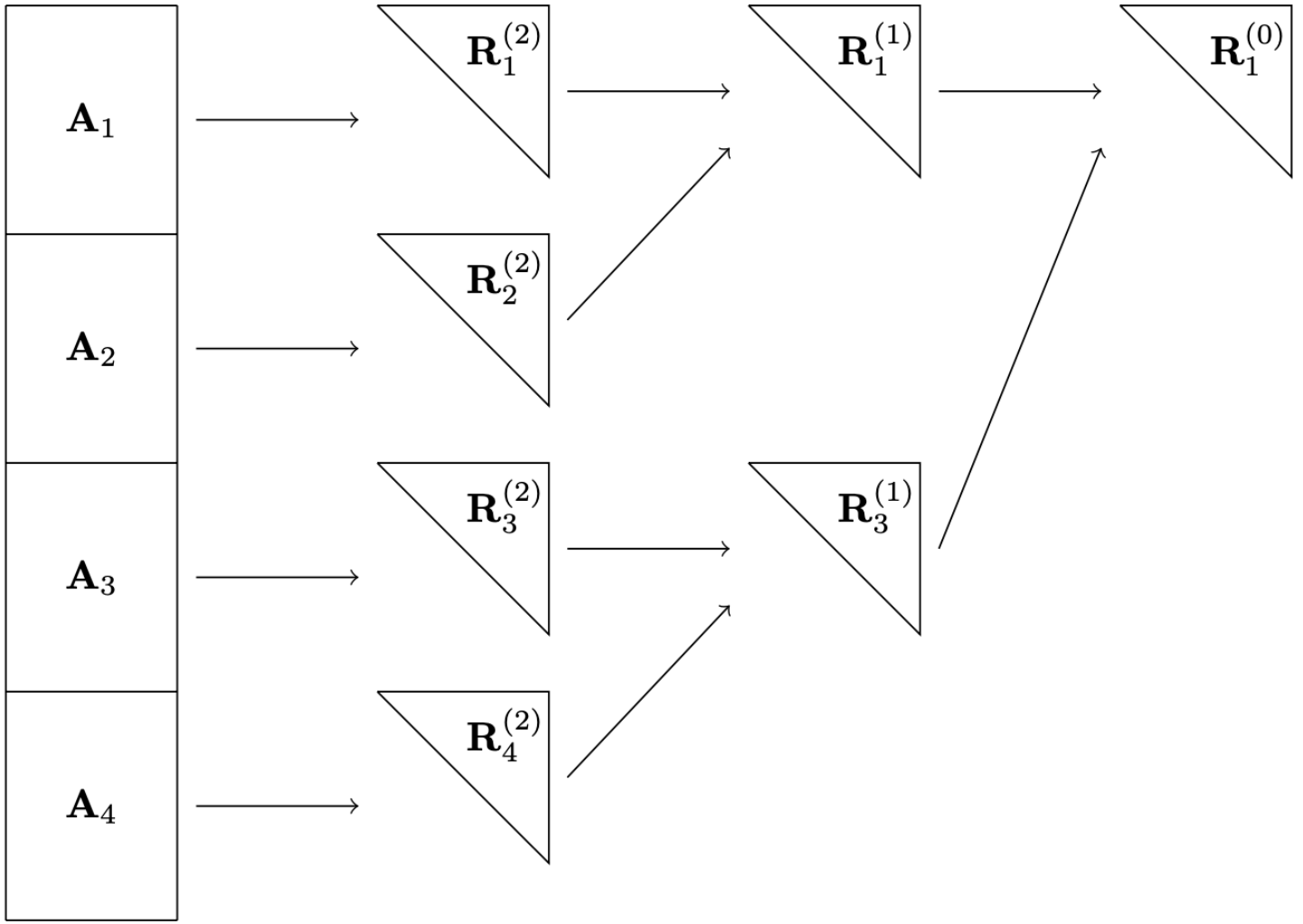
$$\begin{aligned} \mathbf{Q} &= (\mathbf{I} - \mathbf{Y}_1\mathbf{T}_1\mathbf{Y}_1^T)(\mathbf{I} - \mathbf{Y}_2\mathbf{T}_2\mathbf{Y}_2^T) \\ \mathbf{T} &= \begin{pmatrix} \mathbf{T}_1 & -\mathbf{T}_1\mathbf{Y}_1^T\mathbf{Y}_2\mathbf{T}_2 \\ 0 & \mathbf{T}_2 \end{pmatrix} \end{aligned}$$

Plan

TSQR: QR factorization of a tall skinny matrix

- QR decomposition of m x n matrix **A**, $m \gg n$ using Householder transformations
  - P processors, block row layout
- Classic Parallel Algorithm
  - Compute Householder vector for each column
  - Number of messages $\approx n \cdot log_2 P$
- Communication Avoiding TSQR Algorithm
  - Reduction operation, with QR as operator
  - Number of messages $log_2 P$

TSQR: QR factorization of a tall skinny matrix



J. Demmel, LG, M. Hoemmen, J. Langou, 08

References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02

Algebra of TSQR with binomial tree and 4 procs

- At the leaves of the binomial tree, perform in parallel 4 local QR factorizations,
  $\mathbf{A}_l = \hat{\mathbf{Q}}_l^{(2)} \hat{\mathbf{R}}_l^{(2)}$ for each processor l, $\hat{\mathbf{Q}}_l^{(2)} \in \mathbb{R}^{(m/4)\times(m/4)}$ and $\hat{\mathbf{R}}_l^{(2)} \in \mathbb{R}^{(m/4)\times n}$.
- Write the algebra as:

$$\hat{\mathbf{Q}}^{(2)\,T} \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{R}}_1^{(2)} \\ \hat{\mathbf{R}}_2^{(2)} \\ \hat{\mathbf{R}}_3^{(2)} \\ \hat{\mathbf{R}}_4^{(2)} \end{bmatrix}, \quad \text{where} \quad \hat{\mathbf{Q}}^{(2)} = \begin{bmatrix} \hat{\mathbf{Q}}_1^{(2)} & & & \\ & \hat{\mathbf{Q}}_2^{(2)} & & \\ & & \hat{\mathbf{Q}}_3^{(2)} & \\ & & & \hat{\mathbf{Q}}_4^{(2)} \end{bmatrix},$$

and $\hat{\mathbf{R}}_l^{(2)} = \begin{bmatrix} \mathbf{R}_l^{(2)} \\ \mathbf{0} \end{bmatrix}$.

Algebra of TSQR with binomial tree and 4 procs

- Second level of the binomial tree, eliminate $\mathbf{R}_2^{(2)}$ and $\mathbf{R}_4^{(2)}$ in parallel,

$$\begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(1)} & \hat{\mathbf{Q}}_{12}^{(1)} \\ \hat{\mathbf{Q}}_{21}^{(1)} & \hat{\mathbf{Q}}_{22}^{(1)} \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_1^{(2)} \\ \mathbf{R}_2^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^{(1)} \\ \mathbf{0} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \hat{\mathbf{Q}}_{33}^{(1)} & \hat{\mathbf{Q}}_{34}^{(1)} \\ \hat{\mathbf{Q}}_{43}^{(1)} & \hat{\mathbf{Q}}_{44}^{(1)} \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_3^{(2)} \\ \mathbf{R}_4^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_3^{(1)} \\ \mathbf{0} \end{bmatrix}.$$

  Here, each $\hat{\mathbf{Q}}_{IJ}^{(1)}$ is $n \times n$.

- Write the algebra as:

$$\hat{\mathbf{Q}}^{(1)\,T} \begin{bmatrix} \hat{\mathbf{R}}_1^{(2)} \\ \hat{\mathbf{R}}_2^{(2)} \\ \hat{\mathbf{R}}_3^{(2)} \\ \hat{\mathbf{R}}_4^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^{(1)} \\ \mathbf{0} \\ \mathbf{R}_3^{(1)} \\ \mathbf{0} \end{bmatrix}, \quad \text{where } \hat{\mathbf{Q}}^{(1)} = \begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(1)} & & \hat{\mathbf{Q}}_{12}^{(1)} & \\ & \mathbf{I} & & \\ \hat{\mathbf{Q}}_{21}^{(1)} & & \hat{\mathbf{Q}}_{22}^{(1)} & \\ & & & \mathbf{I} \\ & & & & \hat{\mathbf{Q}}_{33}^{(1)} & & \hat{\mathbf{Q}}_{34}^{(1)} \\ & & & & & \mathbf{I} & \\ & & & & \hat{\mathbf{Q}}_{43}^{(1)} & & \hat{\mathbf{Q}}_{44}^{(1)} \\ & & & & & & & \mathbf{I} \end{bmatrix}.$$

Algebra of TSQR with binomial tree and 4 procs

- At the root of the tree: determining a $2n \times 2n$ orthogonal matrix that satisfies

$$\begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(0)} & \hat{\mathbf{Q}}_{13}^{(0)} \\ \hat{\mathbf{Q}}_{31}^{(0)} & \hat{\mathbf{Q}}_{33}^{(0)} \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_1^{(1)} \\ \mathbf{R}_3^{(1)} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^{(0)} \\ \mathbf{0} \end{bmatrix}$$

(with the same Householder vector structure as the 2nd step) so that

$$\hat{\mathbf{Q}}^{(0)\,T} \begin{bmatrix} \mathbf{R}_1^{(1)} \\ \mathbf{0} \\ \mathbf{R}_3^{(1)} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1^{(0)} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \text{where} \quad \hat{\mathbf{Q}}^{(0)} = \begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(0)} & & \hat{\mathbf{Q}}_{13}^{(0)} & \\ & \mathbf{I} & & \\ \hat{\mathbf{Q}}_{31}^{(0)} & & \hat{\mathbf{Q}}_{33}^{(0)} & \\ & & & \mathbf{I} \end{bmatrix}.$$

Algebra of TSQR with binomial tree and 4 procs

- $\hat{Q}$ is represented implicitly as a product of orthogonal matrices

$$\hat{\mathbf{Q}} = \hat{\mathbf{Q}}^{(2)}\hat{\mathbf{Q}}^{(1)}\hat{\mathbf{Q}}^{(0)}$$

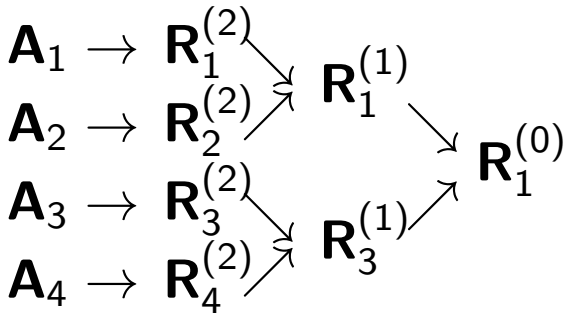
$$= \begin{bmatrix} \hat{\mathbf{Q}}_1^{(2)} & & & \\ & \hat{\mathbf{Q}}_2^{(2)} & & \\ & & \hat{\mathbf{Q}}_3^{(2)} & \\ & & & \hat{\mathbf{Q}}_4^{(2)} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(1)} & & \hat{\mathbf{Q}}_{12}^{(1)} & \\ & \mathbf{I} & & \\ \hat{\mathbf{Q}}_{21}^{(1)} & & \hat{\mathbf{Q}}_{22}^{(1)} & \\ & & & \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{Q}}_{33}^{(1)} & & \hat{\mathbf{Q}}_{34}^{(1)} & \\ & \mathbf{I} & & \\ \hat{\mathbf{Q}}_{43}^{(1)} & & \hat{\mathbf{Q}}_{44}^{(1)} & \\ & & & \mathbf{I} \end{bmatrix}$$

$$\begin{bmatrix} \hat{\mathbf{Q}}_{11}^{(0)} & & \hat{\mathbf{Q}}_{13}^{(0)} & \\ & \mathbf{I} & & \\ \hat{\mathbf{Q}}_{31}^{(0)} & & \hat{\mathbf{Q}}_{33}^{(0)} & \\ & & & \mathbf{I} \end{bmatrix}$$

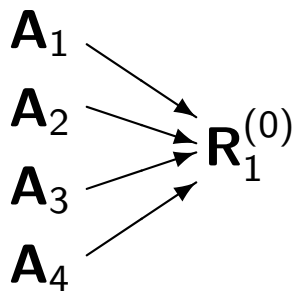
- For the products to make sense, dimensions of intermediate I matrices have appropriate dimensions

Flexibility of TSQR

- Reduction tree will depend on the underlying architecture, could be chosen dynamically



(a) Parallel TSQR



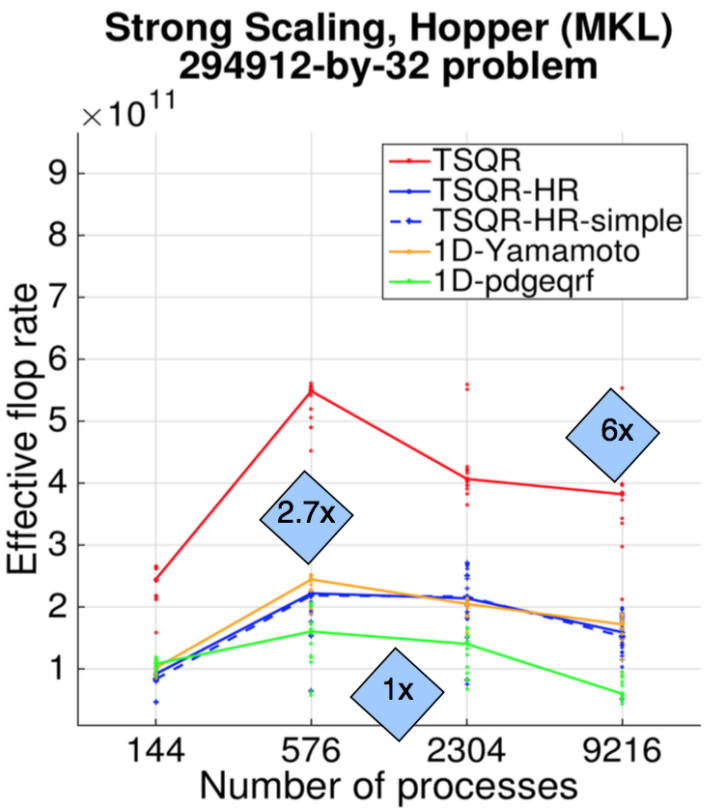
(b) Householder QR

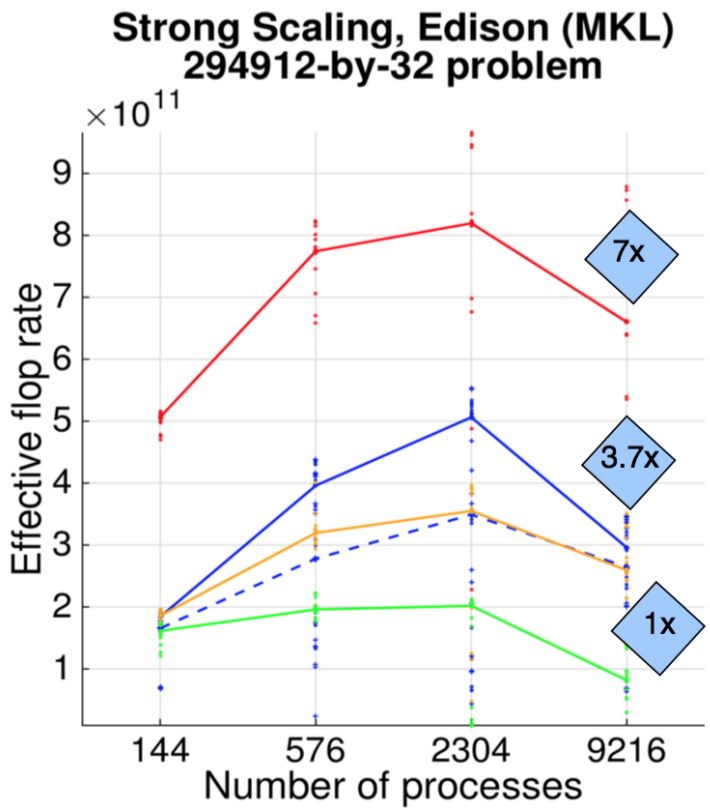
TSQR

Require: $\mathbf{A}$ is an $m \times n$ matrix 1D-row-distributed over power-of-two P processors
Assert: $\hat{\mathbf{Q}}\hat{\mathbf{R}} = \mathbf{A}$ where $\hat{\mathbf{R}}$ is upper triangular ($m \times n$), $\mathbf{R}$ is its upper triangular part ($n \times n$), and $\hat{\mathbf{Q}} = \hat{\mathbf{Q}}^{(\log P)} \cdots \hat{\mathbf{Q}}^{(0)}$
Assert: $\mathbf{R}$ is stored on processor 1 and each $\mathbf{Y}^{(k)}$ is distributed across 2^k processors

1: **function** $\left[\left\{\mathbf{Y}_I^{(k)}\right\}, \mathbf{R}\right] = \textsc{ParTSQR}(\mathbf{A})$
2: $I = \text{MyProcID}$
3: $\left[\mathbf{Y}_I^{(\log P)}, \mathbf{R}_I^{(\log P)}\right] = \text{HouseholderQR}\mathbf{A}_I$ ▷ Eliminate lower triangle of local block
4: **for** $k = \log P - 1$ down to 0 **do**
5: Break if I doesn't have a partner proc
6: Determine J, partner proc ID
7: **if** $I > J$ **then**
8: Send $\mathbf{R}_I^{(k+1)}$ to processor J
9: **else**
10: Receive $\mathbf{R}_J^{(k+1)}$ from processor J
11: $\left[\mathbf{Y}_I^{(k)}, \mathbf{R}_I^{(k)}\right] = \text{HouseholderQR}\left(\begin{bmatrix}\mathbf{R}_I^{(k+1)} \\ \mathbf{R}_J^{(k+1)}\end{bmatrix}\right)$ ▷ Eliminate Jth triangle
12: **end if**
13: **end for**
14: **if** $I = 1$ **then**
15: $\mathbf{R} = \mathbf{R}_1^{(0)}$
16: **end if**
17: **end function**

Strong scaling of TSQR





- Hopper: Cray XE6 (NERSC) – 2 x 12-core AMD Magny-Cours (2.1 GHz)
- Edison: Cray CX30 (NERSC) – 2 x 12-core Intel Ivy Bridge (2.4 GHz)
- Effective flop rate, computed by dividing $2mn^2 - 2n^3/3$ by measured runtime
- 1D-pdgeqrf corresponds to Householder QR implemented in Scalapack, TSQR corresponds to the algorithm learned in class, the other algorithms plotted in this graph are not studied in this class.

In libraries

- TSQR and its extended version for square matrices implemented in
  - Intel Data analytics library
  - GNU Scientific Library
  - ScaLAPACK
  - Spark for data mining
- CALU (introduced in next lectures) implemented in
  - Cray's libsci
  - To be implemented in lapack/scapalack

Plan

Summary of cost of the different algorithms

Algorithmic costs for various parallel orthogonalization routines ($P < m/n$)
Cost of CholQR assumes $n^2 \geq P$.

Algorithm	**# flops**	**# words**	**# messages**
CGS	$\frac{2mn^2}{P}$	$O(n^2 + n \log P)$	$O(n \log P)$
MGS	$\frac{2mn^2}{P}$	$O(n^2 \log P)$	$O(n^2 \log P)$
Cholesky-QR	$\frac{2mn^2}{P} + \frac{n^3}{3}$	$O(n^2)$	$O(\log P)$
Householder QR	$\frac{2mn^2}{P}$	$O(n^2)$	$O(n \log P)$
TSQR	$\frac{2mn^2}{P} + \frac{2n^3}{3} \log P$	$O(n^2 \log P)$	$O(\log P)$

Summary of stability of the different algorithms

Various orthogonalization routines and their stability in terms of loss of orthogonality, associated constraints on condition number, and references. Note that there are dimensional constants hidden in the $O(\varepsilon)$ factors.

Algorithm	$\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\|$	Constraint	References
CGS	$O(\varepsilon)\kappa^2(\mathbf{A})$	$O(\varepsilon)\kappa^2(\mathbf{A}) < 1$	[Giraud et al., 2005]
MGS	$O(\varepsilon)\kappa(\mathbf{A})$	$O(\varepsilon)\kappa(\mathbf{A}) < 1$	[Björck, 1967]
Cholesky-QR	$O(\varepsilon)\kappa^2(\mathbf{A})$	$O(\varepsilon)\kappa^2(\mathbf{A}) < 1$	[Yamamoto et al., 2015]
Householder QR	$O(\varepsilon)$	none	[Wilkinson, 1965]
TSQR	$O(\varepsilon)$	none	[Demmel et al., 2012],[Mori et al., 2012]

Acknowledgement

Many figures and algorithms taken from upcoming book on communication avoiding algorithms with G. Ballard, E. Carson, and J. Demmel.

References (1)

Björck, Å. (1967).

Solving linear least squares problems by Gram-Schmidt orthogonalization.
BIT, 7:1–21.

Demmel, J. W., Grigori, L., Hoemmen, M., and Langou, J. (2012).

Communication-optimal parallel and sequential QR and LU factorizations.
SIAM J. Sci. Comput., 34(1):A206–A239.

Giraud, L., Langou, J., Rozložník, M., and Van Den Eshof, J. (2005).

Rounding error analysis of the classical Gram-Schmidt orthogonalization process.
Numer. Math., 101:87–100.

Mori, D., Yamamoto, Y., and Zhang, S. L. (2012).

Backward error analysis of the AllReduce algorithm for Householder QR decomposition.
Jpn. J. Ind. Appl. Math., 29(1):111–130.

N.J.Higham (2002).

Accuracy and Stability of Numerical Algorithms.
SIAM, second edition.

Schreiber, R. and Loan, C. V. (1989).

A storage efficient *WY* representation for products of Householder transformations.
SIAM J. Sci. Stat. Comput., 10(1):53–57.

Wilkinson, J. H. (1965).

The algebraic eigenvalue problem, volume 87.
Oxford University Press.

References (2)

Yamamoto, Y., Nakatsukasa, Y., Yanagisawa, Y., and Fukaya, T. (2015).
Roundoff error analysis of the CholeskyQR2 algorithm.
Electron. Trans. Numer. Anal., 44:306–326.