Parallel Numerical Algorithms
Chapter 5 – Eigenvalue Problems
Section 5.1 – QR Factorization

Michael T. Heath and Edgar Solomonik

Department of Computer Science
University of Illinois at Urbana-Champaign

CS 554 / CSE 512
Outline

1. QR Factorization

2. Householder Transformations
   - Recursive TSQR
   - 2D and 3D Householder QR

3. Givens Rotations
For given $m \times n$ matrix $A$, with $m > n$, QR factorization has form

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix}$$

where matrix $Q$ is $m \times m$ with orthonormal columns, and $R$ is $n \times n$ and upper triangular

- Can be used to solve linear systems, least squares problems, and eigenvalue problems
- As with Gaussian elimination, zeros are introduced successively into matrix $A$, eventually reaching upper triangular form, but using orthogonal transformations instead of elementary eliminators
Methods for QR Factorization

- Householder transformations (elementary reflectors)
- Givens transformations (plane rotations)
- Gram-Schmidt orthogonalization
Householder Transformations

- **Householder transformation** has form

\[
H = I - 2 \frac{vv^T}{v^Tv}
\]

where \(v\) is nonzero vector

- From definition, \(H = H^T = H^{-1}\), so \(H\) is both orthogonal and symmetric

- For given vector \(a\), choose \(v\) so that

\[
Ha = \begin{bmatrix}
\alpha \\
0 \\
\vdots \\
0
\end{bmatrix} = \alpha \begin{bmatrix}
1 \\
0 \\
\vdots \\
0
\end{bmatrix} = \alpha e_1
\]
Substituting into formula for $H$, we see that we can take

$$v = a - \alpha e_1$$

and to preserve norm we must have $\alpha = \pm \|a\|_2$, with sign chosen to avoid cancellation.
Householder QR Factorization

\[
\text{for } k = 1 \text{ to } n \\
\alpha_k = -\text{sign}(a_{kk}) \sqrt{a_{kk}^2 + \cdots + a_{mk}^2} \\
\mathbf{v}_k = [0 \cdots 0 \ a_{kk} \cdots \ a_{mk}]^T - \alpha_k \mathbf{e}_k \\
\beta_k = \mathbf{v}_k^T \mathbf{v}_k \\
\text{if } \beta_k = 0 \text{ then} \\
\quad \text{continue with next } k \\
\text{for } j = k \text{ to } n \\
\quad \gamma_j = \mathbf{v}_k^T \mathbf{a}_j \\
\quad \mathbf{a}_j = \mathbf{a}_j - \left(2 \frac{\gamma_j}{\beta_k}\right) \mathbf{v}_k \\
\text{end} \\
\text{end}
\]
Basis-Kernel Representations

- A Householder matrix $H$ is represented by $H = I - uu^T$, i.e. a rank-1 perturbation of the identity.

- We can combine $r$ Householder matrices $H_1, \ldots, H_r$ into a rank-$r$ perturbation of the identity:

$$
\bar{H} = \prod_{i=1}^{r} H_i = I - YV^T,
$$

where $Y, V \in \mathbb{R}^{n \times r}$.

- Often, $V = YT$ where $T$ is upper-triangular and $Y$ is lower-triangular, yielding

$$
\bar{H} = I - YT^TY^T
$$

- If $H_i = I - y_iy_i^T$, then the $i$th column of $Y$ is $y_i$, while $T$ is defined by $T^{-1} + T^{-T} = Y^TY$.
A basis kernel representation of Householder transformations, allows us to update a trailing matrix \( B \) as

\[
\bar{H}B = (I - YT^T Y^T) B = B - Y (T^T (Y^T B))
\]

with cost \( O(n^2 r) \)

Performing such updates is essentially as hard as Schur complement updates in LU

Forming Householder vector \( v_k \) is also analogous to computing multipliers in Gaussian elimination

Thus, parallel implementation is similar to parallel LU, but with Householder vectors broadcast horizontally instead of multipliers
Panel QR Factorization

- Finding Householder vector $y_i$ requires computation of the norm of the leading vector of the $i$th trailing matrix, creating a latency bottleneck much like that of pivot row selection in partial pivoting.
- Other methods need $L = \Theta(\log(p))$ rather than $\Theta(n)$ msgs.
- For example, Cholesky-QR and Cholesky-QR2 perform $R = \text{Cholesky}(A^T A)$, $Q = AR^{-1}$ (Cholesky-QR2 does one step of refinement), requiring only a single allreduce, but losing stability.
- Unconditional stability and $O(\log(p))$ messages achieved by TSQR algorithm with row-wise recursion (akin to tournament pivoting).
- Basis-kernel representation can be recovered by constructing first $r$ columns of $\bar{H}$. 
Cholesky QR2

Cholesky-QR can be made more stable [Yamamoto et al 2014]

- As before, compute \( \{\bar{Q}, \bar{R}\} = \text{Cholesky-QR}(A) \)
- Then, iterate \( \{Q, \hat{R}\} = \text{Cholesky-QR}(\bar{Q}) \)
- \( R = \hat{R}\bar{R} \)
- \( A = QR \)

Solution still bad when \( \kappa(A) \geq 1/\sqrt{\epsilon_{\text{mach}}} \)

- But if \( \kappa(A) < 1/\sqrt{\epsilon_{\text{mach}}} \), it is numerically stable because \( \kappa(\bar{Q}) \approx 1 \)

- For QR of a tall-skinny \( A \) with \( \kappa(A) < 1/\sqrt{\epsilon_{\text{mach}}} \), this algorithm is easy to implement, stable, and scalable
Block Givens rotations yield another idea

- We can also employ a recursive scheme analogous to tournament pivoting for LU
- Subdivide \( A = \begin{bmatrix} A_U \\ A_L \end{bmatrix} \) and recursively compute
  \[ \{Q_U, R_U\} = QR(A_U), \{Q_L, R_L\} = QR(A_L) \] concurrently with \( P/2 \) processors each

- We have
  \[ A = \begin{bmatrix} Q_U & R_U \\ Q_L & R_L \end{bmatrix} = \begin{bmatrix} Q_U & \\ & Q_L \end{bmatrix} \begin{bmatrix} R_U \\ R_L \end{bmatrix} \]

- Gather \( R_U \) and \( R_L \) and compute sequentially,
  \[ \begin{bmatrix} R_U \\ R_L \end{bmatrix} = \tilde{Q}R \]

- We now have \( A = QR \) where
  \[ Q = \begin{bmatrix} Q_U & Q_L \end{bmatrix} \tilde{Q} \]
Recursive TSQR, Binary (Binomial) Tree

A

R

Householder vectors are denoted in yellow (\(R\) is \(R_1\)).
Cost Analysis of Recursive TSQR

We can subdivide the cost into base cases (tree leaves) and internal nodes

- Every processor computes a QR of their $m/P \times n$ leaf matrix block

\[ T_{\text{Rec-TSQR}}(m, n, P) = T_{\text{Rec-TSQR}}(nP, n, 1) + (m/P)n^2 \cdot \gamma \]

- Subsequently for each tree node, each processor sends/receives a message of size $O(n^2)$ and performs $O(n^3)$ work to factorize $2n \times n$ matrix

- The total cost is

\[ T_{\text{Rec-TSQR}}(m, n, P) = O\left([mn^2/P + n^3 \log(P)] \cdot \gamma + n^2 \log(P) \cdot \beta + \log(P) \cdot \alpha\right) \]

- Communication cost is higher than of Cholesky-QR2, which is $2T_{\text{allreduce}}(n^2/2, P) = O(n^2 \beta + \log(P)\alpha)$
Recovering $Q$ in Recursive TSQR

Michael T. Heath and Edgar Solomonik
Given $m \times n$ matrix $Q_1$, we can construct $Y$ such that

$$Q = (I - YTY^T) = [Q_1, Q_2]$$

and $Q$ is orthogonal.

- note that in the Householder representation, we have $I - Q = Y \cdot TY^T$, where $Y$ is lower-trapezoidal and $TY^T$ is upper-trapezoidal.

- Let $Q_1 = \begin{bmatrix} Q_{11} \\ Q_{21} \end{bmatrix}$ where $Q_{11}$ is $n \times n$, compute

$$\{Y, TY_1^T\} = LU\left( \begin{bmatrix} I - Q_{11} \\ Q_{21} \end{bmatrix} \right),$$

where $Y_1$ is the upper-triangular $n \times n$ leading block of $Y^T$. 

Michael T. Heath and Edgar Solomonik

Parallel Numerical Algorithms 16 / 31
Householder reconstruction can be done with unconditional stability

- We need to be just a little more careful

\[ \{Y, TY_1^T\} = LU \left( \begin{bmatrix} S - Q_{11} \\ Q_{21} \end{bmatrix} \right), \]

where \( S \) is a sign matrix (each value in \( \{-1, 1\} \)) with values picked to match the sign of the diagonal entry within LU

- These are the sign choices we need to make for regular Householder factorization

- Since all entries of \( Q \) are \( \leq 1 \), pivoting is unnecessary (partial pivoting would do nothing)

- Since \( \kappa(Q) \approx 1 \), Householder reconstruction is stable
2D Householder QR, Basis-Kernel Representation

Transpose and Broadcast $Y$
Reduce $W = Y^T A$
Transpose $W$ and Compute $T^T W$

$T^T W = T^T Y^T A$

Transpose and multiply by $T^T$
Compute $YT^T Y^T A$ and subsequently $Q^T A = A - YT^T Y^T A$

$Y(T^TW) = YT^TY^TA$

Broadcast and multiply by $Y$
Elmroth-Gustavson Algorithm (3Dx2Dx1D)

One approach is to use column-recursion \( A = [A_1, A_2] \)

- Compute \( \{Y_1, T_1, R_1\} = \text{QR}(A_1) \) recursively with \( P \) processors
- Perform rectangular matrix multiplications with communication-avoiding algorithms to compute \( B_2 = (I - Y_1 T_1 Y_1^T)T A_2 \)

- Compute \( \{Y_2, T_2, R_2\} = \text{QR}(B_{22}) \) where \( B_2 = \begin{bmatrix} R_{12} \\ B_{22} \end{bmatrix} \) recursively
- Concatenate \( Y_1 \) and \( Y_2 \) into \( Y \) and compute \( T \) from \( Y \) via rectangular matrix multiplication
- Output \( \{Y, T, \begin{bmatrix} R_1 & R_{12} \\ R_{21} & R_2 \end{bmatrix} \} \)
- Pick an appropriate number of columns for a TSQR base-case

Michael T. Heath and Edgar Solomonik
Parallel Numerical Algorithms
**Givens Rotations**

- **Givens rotation** operates on pair of rows to introduce single zero

- For given 2-vector \( a = [a_1 \ a_2]^T \), if

  \[
  c = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}
  \]

  then

  \[
  G a = \begin{bmatrix}
  c & s \\
  -s & c
  \end{bmatrix}
  \begin{bmatrix}
  a_1 \\
  a_2
  \end{bmatrix} = \begin{bmatrix}
  \alpha \\
  0
  \end{bmatrix}
  \]

- Scalars \( c \) and \( s \) are cosine and sine of angle of rotation, and \( c^2 + s^2 = 1 \), so \( G \) is orthogonal
Givens QR Factorization

- Givens rotations can be systematically applied to successive pairs of rows of matrix $A$ to zero entire strict lower triangle.

- Subdiagonal entries of matrix can be annihilated in various possible orderings (but once introduced, zeros should be preserved).

- Each rotation must be applied to all entries in relevant pair of rows, not just entries determining $c$ and $s$.

- Once upper triangular form is reached, product of rotations, $Q$, is orthogonal, so we have QR factorization of $A$. 
Parallel Givens QR Factorization

- With 1-D partitioning of $A$ by columns, parallel implementation of Givens QR factorization is similar to parallel Householder QR factorization, with cosines and sines broadcast horizontally and each task updating its portion of relevant rows.

- With 1-D partitioning of $A$ by rows, broadcast of cosines and sines is unnecessary, but there is no parallelism unless multiple pairs of rows are processed simultaneously.

- Fortunately, it is possible to process multiple pairs of rows simultaneously without interfering with each other.
Stage at which each subdiagonal entry can be annihilated is shown here for $8 \times 8$ example

\[
\begin{bmatrix}
\times & & & & & & & \\
7 & \times & & & & & & \\
6 & 8 & \times & & & & & \\
5 & 7 & 9 & \times & & & & \\
4 & 6 & 8 & 10 & \times & & & \\
3 & 5 & 7 & 9 & 11 & \times & & \\
2 & 4 & 6 & 8 & 10 & 12 & \times & \\
1 & 3 & 5 & 7 & 9 & 11 & 13 & \times
\end{bmatrix}
\]

Maximum parallelism is $n/2$ at stage $n - 1$ for $n \times n$ matrix.
Parallel Givens QR Wavefront
Communication cost is high, but can be reduced by having each task initially reduce its entire local set of rows to upper triangular form, which requires no communication.

Then, in subsequent phase, task pairs cooperate in annihilating additional entries using one row from each of two tasks, exchanging data as necessary.

Various strategies can be used for combining results of first phase, depending on underlying network topology.

Parallel partitioning with slanted-panels (slope -2) achieve same scalability as parallel algorithms for LU without pivoting (see [Tiskin 2007]).
With 2-D partitioning of $A$, parallel implementation combines features of 1-D column and 1-D row algorithms.

In particular, sets of rows can be processed simultaneously to annihilate multiple entries, but updating of rows requires horizontal broadcast of cosines and sines.
References


References