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

$$A = Q \begin{bmatrix} R & \mathbf{0} \end{bmatrix}$$

where matrix $Q$ is $m \times m$ and orthogonal, 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.
for $k = 1$ to $n$

\[
\alpha_k = -\text{sign}(a_{kk}) \sqrt{a_{kk}^2 + \cdots + a_{mk}^2}
\]

\[
v_k = \begin{bmatrix} 0 & \cdots & 0 & a_{kk} & \cdots & a_{mk} \end{bmatrix}^T - \alpha_k e_k
\]

\[
\beta_k = v_k^T v_k
\]

if $\beta_k = 0$ then
  continue with next $k$

for $j = k$ to $n$

\[
\gamma_j = v_k^T a_j
\]

\[
a_j = a_j - \left(2\gamma_j / \beta_k\right)v_k
\]

end

end
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

$$\tilde{H} = \prod_{i=1}^{r} H_r = I - UV^T,$$

where $U, V \in \mathbb{R}^{n \times k}$.

Often, $V = UT$ where $T$ is upper-triangular and $U$ is lower-triangular, yielding

$$\tilde{H} = I - UT^T U^T$$

If $H_i = I - u_i u_i^T$, then the $i$th column of $U$ is $u_i$, while $T$ is defined by

$$T^{-1} + T^{-T} = U^T U$$
A basis kernel representation of Householder transformations, allows us to update a trailing matrix $B$ as

$$\bar{H}B = (I - UT^TU^T)B = B - U(T^T(U^TB))$$

with cost $O(n^2r)$

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 $u_i$ 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 $S = \Theta(\log(p))$ 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 $\bar{H}$. 
Givens Rotations

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

- For given 2-vector $\mathbf{a} = [a_1 \ a_2]^T$, 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

  \[
  G\mathbf{a} = \begin{bmatrix} \frac{a_1}{\sqrt{a_1^2 + a_2^2}} & \frac{a_2}{\sqrt{a_1^2 + a_2^2}} \\ -\frac{a_2}{\sqrt{a_1^2 + a_2^2}} & \frac{a_1}{\sqrt{a_1^2 + a_2^2}} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}
  \]

  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.
Parallel Givens QR Factorization

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
Parallel Givens QR Factorization

- 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


