Dense LU and QR factorizations

Laura Grigori
INRIA and LJLL, Sorbonne Université

October 2020
Plan

Direct methods of factorization
  LU factorization
  Block LU factorization
  QR factorization
  Block QR factorization
Norms and other notations

\[ \|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}|^2} \]

\[ \|A\|_2 = \sigma_{\text{max}}(A) \]

\[ \|A\|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}| \]

\[ \|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}| \]

Inequalities \(|x| \leq |y|\) and \(|A| \leq |B|\) hold componentwise.
Plan

Direct methods of factorization
- LU factorization
- Block LU factorization
- QR factorization
- Block QR factorization
LU factorization
Compute the factorization $PA = LU$

Example
Given the matrix

$$A = \begin{pmatrix} 3 & 1 & 3 \\ 6 & 7 & 3 \\ 9 & 12 & 3 \end{pmatrix}$$

Let

$$M_1 = \begin{pmatrix} 1 & -2 & 1 \\ -3 & 1 & 1 \end{pmatrix}, \quad M_1A = \begin{pmatrix} 3 & 1 & 3 \\ 0 & 5 & -3 \\ 0 & 9 & -6 \end{pmatrix}$$
Algebra of the LU factorization

- In general

\[
A^{(k+1)} = M_k A^{(k)} := \begin{pmatrix}
I_{k-1} & 1 \\
-m_{k+1,k} & 1 \\
. & . & . \\
-m_{n,k} & . & 1
\end{pmatrix} A^{(k)}, \text{ where}
\]

\[
M_k = I - m_k e_k^T, \quad M_k^{-1} = I + m_k e_k^T
\]

where \( e_k \) is the k-th unit vector, \( m_k = (0, \ldots, 0, 1, m_{k+1,k}, \ldots, m_{n,k})^T \), \( e_i^T m_k = 0, \forall i \leq k \)

- The factorization can be written as

\[
M_{n-1} \ldots M_1 A = A^{(n)} = U
\]
We obtain

\[ A = M_1^{-1} \ldots M_{n-1}^{-1} U \]
\[ = (I + m_1 e_1^T) \ldots (I + m_{n-1} e_{n-1}^T) U \]
\[ = \left( I + \sum_{i=1}^{n-1} m_i e_i^T \right) U \]
\[ = \begin{pmatrix}
1 & 1 \\
m_{21} & 1 \\
: & : & : \\
m_{n1} & m_{n2} & \ldots & 1
\end{pmatrix} U = LU \]
The need for pivoting

- For stability, avoid division by small diagonal elements
- For example

\[ A = \begin{pmatrix} 0 & 3 & 3 \\ 3 & 1 & 3 \\ 6 & 2 & 3 \end{pmatrix} \quad (1) \]

has an LU factorization if we permute the rows of matrix \( A \)

\[ PA = \begin{pmatrix} 6 & 2 & 3 \\ 0 & 3 & 3 \\ 3 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0.5 \end{pmatrix} \begin{pmatrix} 6 & 2 & 3 \\ 3 & 3 & 1.5 \end{pmatrix} \quad (2) \]

- Partial pivoting allows to bound the multipliers \( m_{ik} \leq 1 \) and hence \( |L| \leq 1 \)
Theorem

Given a full rank matrix $A$ of size $m \times n$, $m \geq n$, the matrix $A$ can be decomposed as $A = PLU$ where $P$ is a permutation matrix of size $m \times m$, $L$ is a unit lower triangular matrix of size $m \times n$ and $U$ is a nonsingular upper triangular matrix of size $n \times n$.

Proof: simpler proof for the square case. Since $A$ is full rank, there is a permutation $P_1$ such that $P_1a_{11}$ is nonzero. Write the factorization as

$$P_1A = \begin{pmatrix} a_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ A_{21}/a_{11} & I \end{pmatrix} \begin{pmatrix} a_{11} & A_{12} \\ 0 & A_{22} - a_{11}^{-1}A_{21}A_{12} \end{pmatrix},$$

where $S = A_{22} - a_{11}^{-1}A_{21}A_{12}$.

Since $\det(A) \neq 0$, then $\det(S) \neq 0$. Continue the proof by induction on $S$. 

Existence of the LU factorization
Solving $Ax = b$ by using Gaussian elimination

Composed of 4 steps
1. Factor $A = PLU$, $(2/3)n^3$ flops
2. Compute $P^T b$ to solve $LUx = P^T b$
3. Forward substitution: solve $Ly = P^T * b$, $n^2$ flops
4. Backward substitution: solve $Ux = y$, $n^2$ flops
Algorithm to compute the LU factorization

- Algorithm for computing the in place LU factorization of a matrix of size \( n \times n \).
- \#flops = \( 2n^3/3 \)

1: \textbf{for } k = 1:n-1 \textbf{ do}
2: \quad \text{Let } a_{ik} \text{ be the element of maximum magnitude in } A(k : n, k)
3: \quad \text{Permute row } i \text{ and row } k
4: \quad A(k + 1 : n, k) = A(k + 1 : n, k)/a_{kk}
5: \quad \textbf{for } i = k + 1 : n \textbf{ do}
6: \quad \quad \textbf{for } j = k + 1 : n \textbf{ do}
7: \quad \quad \quad a_{ij} = a_{ij} - a_{ik}a_{kj}
8: \quad \quad \textbf{end for}
9: \quad \textbf{end for}
10: \textbf{end for}
Wilkinson’s backward error stability result

Growth factor $g_W$ defined as

$$g_W = \frac{\max_{i,j,k} |a_{ij}^k|}{\max_{i,j} |a_{ij}|}$$

Note that

$$|u_{ij}| = |a_{ij}^i| \leq g_W \max_{i,j} |a_{ij}|$$

Theorem (Wilkinson’s backward error stability result, see also [N.J.Higham, 2002] for more details)

Let $A \in \mathbb{R}^{n \times n}$ and let $\hat{x}$ be the computed solution of $Ax = b$ obtained by using GEPP. Then

$$(A + \Delta A)\hat{x} = b, \quad \|\Delta A\|_\infty \leq n^2 \gamma_3 n g_W(n) \|A\|_\infty,$$

where $\gamma_n = nu/(1 - nu)$, $u$ is machine precision and assuming $nu < 1$. 

The growth factor

- The LU factorization is backward stable if the growth factor is small (grows linearly with $n$).
- For partial pivoting, the growth factor $g(n) \leq 2^{n-1}$, and this bound is attainable.
- In practice it is on the order of $n^{2/3} - n^{1/2}$

Exponential growth factor for Wilkinson matrix

$$A = \text{diag}(\pm 1)$$

$$\begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & 1 \\
-1 & 1 & 0 & \cdots & 0 & 1 \\
-1 & -1 & 1 & \ddots & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
-1 & -1 & \cdots & -1 & 1 & 1 \\
-1 & -1 & \cdots & -1 & -1 & 1 \\
\end{bmatrix}$$
Experimental results for special matrices

Several error bounds for GEPP, the normwise backward error $\eta$ and the componentwise backward error $w$ ($r = b - Ax$).

$$\eta = \frac{||r||_1}{||A||_1 ||x||_1 + ||b||_1},$$

$$w = \max_i \frac{|r_i|}{(|A||x| + |b|)_i}.$$
Block formulation of the LU factorization

Partitioning of matrix \( A \) of size \( n \times n \)

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\]

where \( A_{11} \) is of size \( b \times b \), \( A_{21} \) is of size \( (m - b) \times b \), \( A_{12} \) is of size \( b \times (n - b) \) and \( A_{22} \) is of size \( (m - b) \times (n - b) \).

Block LU algebra

The first iteration computes the factorization:

\[
P_1^T A = \begin{bmatrix}
\tilde{A}_{11} & \tilde{A}_{12} \\
\tilde{A}_{21} & \tilde{A}_{22}
\end{bmatrix} = \begin{bmatrix}
L_{11} & I_{n-b} \\
L_{21} & A^{1}
\end{bmatrix} \cdot \begin{bmatrix}
U_{11} & U_{12} \\
U_{21} & A^{1}
\end{bmatrix}
\]

The algorithm continues recursively on the trailing matrix \( A^{1} \).
Block LU factorization - the algorithm

1. Compute the LU factorization with partial pivoting of the first block column

\[ P_1 \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \begin{pmatrix} U_{11} \end{pmatrix} \]

2. Pivot by applying the permutation matrix \( P_1^T \) on the entire matrix,

\[ \tilde{A} = P_1^T A. \]

3. Solve the triangular system

\[ L_{11} U_{12} = \tilde{A}_{12} \]

4. Update the trailing matrix,

\[ A^1 = \tilde{A}_{22} - L_{21} U_{12} \]

5. Compute recursively the block LU factorization of \( A^1 \).
LU Factorization as in ScaLAPACK

LU factorization on a $P = P_r \times P_c$ grid of processors

For $ib = 1$ to $n-1$ step $b$

$A(ib) = A(ib : n, ib : n)$

1. Compute panel factorization
   - find pivot in each column, swap rows

2. Apply all row permutations
   - broadcast pivot information along the rows
   - swap rows at left and right

3. Compute block row of $U$
   - broadcast right diagonal block of $L$ of current panel

4. Update trailing matrix
   - broadcast right block column of $L$
   - broadcast down block row of $U$
LU factorization on a $P = P_r \times P_c$ grid of processors
For $ib = 1$ to $n-1$ step $b$
$A(ib) = A(ib : n, ib : n)$

1. Compute panel factorization
   - $\#messages = O(n \log_2 P_r)$

2. Apply all row permutations
   - $\#messages = O(n/b(\log_2 P_r + \log_2 P_c))$

3. Compute block row of $U$
   - $\#messages = O(n/b \log_2 P_c)$

4. Update trailing matrix
   - $\#messages = O(n/b(\log_2 P_r + \log_2 P_c))$
Consider that we have a $\sqrt{P} \times \sqrt{P}$ grid, block size $b$

$$\gamma \cdot \left( \frac{2/3n^3}{P} + \frac{n^2b}{\sqrt{P}} \right) + \beta \cdot \frac{n^2 \log P}{\sqrt{P}} + \alpha \cdot \left( 1.5n \log P + \frac{3.5n}{b} \log P \right).$$
The QR factorization

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

$$A = QR = (Q_1 \quad Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular.

If $A$ has full rank, the factorization $Q_1 R_1$ is essentially unique (modulo signs of diagonal elements of $R$).

- $A^T A = R_1^T R_1$ is a Cholesky factorization and $A = AR_1^{-1} R_1$ is a QR factorization.
- $A = Q_1 D \cdot DR_1$, $D = \text{diag}(\pm1)$ is a QR factorization.
Householder transformation

The Householder matrix

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

has the following properties:

- is symmetric and orthogonal, \( P^2 = I \),
- is independent of the scaling of \( v \),
- it reflects \( x \) about the hyperplane \( \text{span}(v)^\perp \)

\[ Px = x - \frac{2v^T x}{v^T v} v = x - \alpha v \]

Presentation of Householder transformations and stability analysis from [N.J.Higham, 2002].
Householder for the QR factorization

We look for a Householder matrix that allows to annihilate the elements of a vector $x$, except first one.

$$Px = y, \quad \|x\|_2 = \|y\|_2, \quad y = \sigma e_1, \quad \sigma = \pm \|x\|_2$$

With the choice of sign made to avoid cancellation when computing $v_1 = x_1 - \sigma$ (where $v_1, x_1$ are the first elements of vectors $v, x$ respectively), we have

$$v = x - y = x - \sigma e_1,$$

$$\sigma = -\text{sign}(x_1)\|x\|_2, \quad v = x - \sigma e_1,$$

$$P = I - \beta vv^T, \quad \beta = \frac{2}{v^Tv}$$
Householder based QR factorization

\[
A = \begin{pmatrix}
x & x & x \\
x & x & x \\
x & x & x
\end{pmatrix}
\]

\[
P_1A = \begin{pmatrix}
x & x & x \\
0 & x & x \\
0 & x & x
\end{pmatrix}, \quad \begin{pmatrix}
1 \\
\tilde{P}_2
\end{pmatrix} P_1 = \begin{pmatrix}
x & x & x \\
0 & x & x \\
0 & 0 & x
\end{pmatrix} = R
\]

So we have

\[
Q^T A = P_n P_{n-1} \ldots P_1 A = R,
\]

\[
Q = (I - \beta_1 v_1 v_1^T) \ldots (I - \beta_{n-1} v_{n-1} v_{n-1}^T)(I - \beta_n v_n v_n^T)
\]

\[
\# \text{flops} = 2n^2 (m - n/3)
\]
The following result follows

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

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

$$A + \Delta A = Q\hat{R}, \text{ where } \|\Delta a_j\|_2 \leq \tilde{\gamma}_{mn}\|a_j\|_2, \quad j = 1 : n,$$

where $\tilde{\gamma}_{mn} = cmnu/(1 - cmnu)$, $c$ is a constant, $u$ is machine precision, $mnu < 1$, $a_j$ denotes the $j$-th column of $A$. 
Householder-QR factorization

Require: $A \in \mathbb{R}^{m \times n}$

1: Let $R \in \mathbb{R}^{n \times n}$ be initialized with zero matrix
2: for $k = 1$ to $n$ do
3: \hspace{1em} $\triangleright$ Compute Householder matrix $P_k = I - \beta_k v_k v_k^T$ s.t. $P_k A(k : m, k) = \pm \|A(k : m, k)\|_2 e_1$. Store $v_k$ in $Y()$ and $\beta_k$ in $T(k)$
4: $R(k, k) = -\text{sgn}(A(k, k)) \cdot \|A(k : m, k)\|_2$
5: $T(k) = \frac{R(k,k) - A(k,k)}{R(k,k)}$
6: $Y(k + 1 : m, k) = \frac{1}{R(k,k) - A(k,k)} \cdot A(k + 1 : m, k)$
7: \hspace{1em} $\triangleright$ Update trailing matrix
8: $A(k : m, k + 1 : n) = (I - Y(k + 1 : m, k) T(k) Y(k + 1 : m, k)^T) \cdot A(k : m, k + 1 : n)$
9: $R(k, k + 1 : n) = A(k, k + 1 : n)$
10: end for

Assert: $A = QR$, where $Q = P_1 \ldots P_n = (I - \beta_1 v_1 v_1^T) \ldots (I - \beta_n v_n v_n^T)$, the Householder vectors $v_k$ are stored in $Y$ and $T$ is an array of size $n$. 

25 of 34
Computational complexity

- Flops per iterations
  - Dot product $w = v_k^T A(k : m, k + 1 : n) : 2(m - k)(n - k)$
  - Outer product $v_k w : (m - k)(n - k)$
  - Subtraction $A(k : m, k + 1 : n) - \ldots : (m - k)(n - k)$

- Flops of Householder-QR

\[
\sum_{k=1}^{n} 4(m - k)(n - k) = 4 \sum_{k=1}^{n} (mn - k(m + n) + k^2)
\approx 4mn^2 - 4(m + n)n^2/2 + 4n^3/3 = 2mn^2 - 2n^3/3
\]
Algebra of block QR

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

$$Q = Q_1 Q_2 \cdots Q_k = (I - \beta_1 v_1 v_1^T) \cdots (I - \beta_k v_k v_k^T) = I - YTY^T$$

Example for $k = 2$

$$Y = (v_1 | v_2), \quad T = \begin{pmatrix} \beta_1 & -\beta_1 v_1^T v_2 \beta_2 \\ 0 & \beta_2 \end{pmatrix}$$

Example for combining two compact representations

$$Q = (I - Y_1 T_1 Y_1^T)(I - Y_2 T_2 Y_2^T)$$

$$T = \begin{pmatrix} T_1 & -T_1 Y_1^T Y_2 T_2 \\ 0 & T_2 \end{pmatrix}$$
Partitioning of matrix $A$ of size $m \times n$

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where $A_{11}$ is of size $b \times b$, $A_{21}$ is of size $(m - b) \times b$, $A_{12}$ is of size $b \times (n - b)$ and $A_{22}$ is of size $(m - b) \times (n - b)$.

**Block QR algebra**

The first step of the block QR factorization algorithm computes:

$$Q_1^T A = \begin{pmatrix} R_{11} & R_{12} \\ A^1 \end{pmatrix}$$

The algorithm continues recursively on the trailing matrix $A^1$. 
Algebra of block QR factorization

\[
A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = Q_1 \begin{pmatrix} R_{11} & R_{12} \\ A^1 \end{pmatrix}
\]

Block QR algebra

1. Compute the factorization

\[
\begin{pmatrix} A_{11} \\ A_{12} \end{pmatrix} = Q_1 R_{11}
\]

2. Compute the compact representation \( Q_1 = I - YTY^T \)

3. Apply \( Q_1^T \) on the trailing matrix

\[
(I - YT^TY^T) \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} - Y \left( T^T \left( YT \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} \right) \right)
\]

4. The algorithm continues recursively on the trailing matrix \( A^1 \).
Parallel implementation of the QR factorization

QR factorization on a \( P = P_r \times P_c \) grid of processors
For \( ib = 1 \) to \( n-1 \) step \( b \)
1. Compute panel factorization on \( P_r \) processors
   \[
   \begin{pmatrix}
   A_{11} \\
   A_{12}
   \end{pmatrix}
   = Q_1 R_{11} = (I - YTY^T) R_{11}
   \]
2. The \( P_r \) processors broadcast along the rows their parts of \( Y \) and \( T \)
3. Apply \( Q_1^T \) on the trailing matrix:
   □ All processors compute their local part of
   \[
   W_i = Y_i^T (A_{12i}; A_{22i})
   \]
   □ The processors owning block row \( ib \) compute the sum over \( W_i \), that is
   \[
   W = Y^T (A_{12}; A_{22})
   \]
   and then compute \( W' = T^T W \)
   □ The processors owning block row \( ib \) broadcast along the columns their part of \( W' \)
4. All processors compute
   \[
   (A_{12}^1; A_{22}^1) = (A_{12}; A_{22}) - Y \ast W'
   \]
Cost of parallel QR factorization

\[
\gamma \cdot \left( \frac{6mnb - 3n^2b}{2p_r} + \frac{n^2b}{2p_c} + \frac{2mn^2 - 2n^3/3}{p} \right) \\
+ \beta \cdot \left( nb \log p_r + \frac{2mn - n^2}{p_r} + \frac{n^2}{p_c} \right) \\
+ \alpha \cdot \left( 2n \log p_r + \frac{2n}{b} \log p_c \right)
\]
Solving least squares problems

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

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

Using the QR factorization of $A$

$$A = QR = (Q_1 \quad Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \quad (3)$$

We obtain

$$\|r\|_2^2 = \|b - Ax\|_2^2 = \|b - (Q_1 \quad Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x\|_2^2$$

$$= \| (Q_1^T \quad Q_2^T) b - \begin{pmatrix} R_1 \\ 0 \end{pmatrix} x\|_2^2 = \| \begin{pmatrix} Q_1^T b - R_1 x \\ Q_2^T b \end{pmatrix}\|_2^2$$

$$= \|Q_1^T b - R_1 x\|_2^2 + \|Q_2^T b\|_2^2$$

Solve $R_1 x = Q_1^T b$ to minimize $\|r\|_2$. 

Acknowledgement

- Some of the examples taken from [Golub and Van Loan, 1996]
*Matrix Computations (3rd Ed.).*
Johns Hopkins University Press, Baltimore, MD, USA.

N.J.Higham (2002).
*Accuracy and Stability of Numerical Algorithms.*
SIAM, second edition.

A storage efficient WY representation for products of Householder transformations.

Optimization of collective communication operations in mpich.