Lecture 8: Fast Linear Solvers (Part 3)

Cholesky Factorization

- Matrix A is **symmetric** if $A = A^T$.
- Matrix A is **positive definite** if for all $\boldsymbol{x} \neq \mathbf{0}$, $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$.
- A symmetric positive definite matrix A has Cholesky factorization $A = LL^T$, where L is a lower triangular matrix with positive diagonal entries.
- Example.
  - $A = A^T$ with $a_{ii} > 0$ and $a_{ii} > \sum_{j \neq i} |a_{ij}|$ is positive definite.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{nn} \end{bmatrix}$$

In 2×2 matrix size case

$$\begin{bmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

$$l_{11} = \sqrt{a_{11}};\ l_{21} = a_{21}/l_{11};\ \ l_{22} = \sqrt{a_{22} - l_{21}^2}$$

Submatrix Cholesky Factorization Algorithm

for $k = 1$ **to** n
    $a_{kk} = \sqrt{a_{kk}}$
    for $i = k + 1$ **to** n
        $a_{ik} = a_{ik}/a_{kk}$
    end
    for $j = k + 1$ **to** n // reduce submatrix
        for $i = j$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{jk}$
        end
    end
end

Remark:

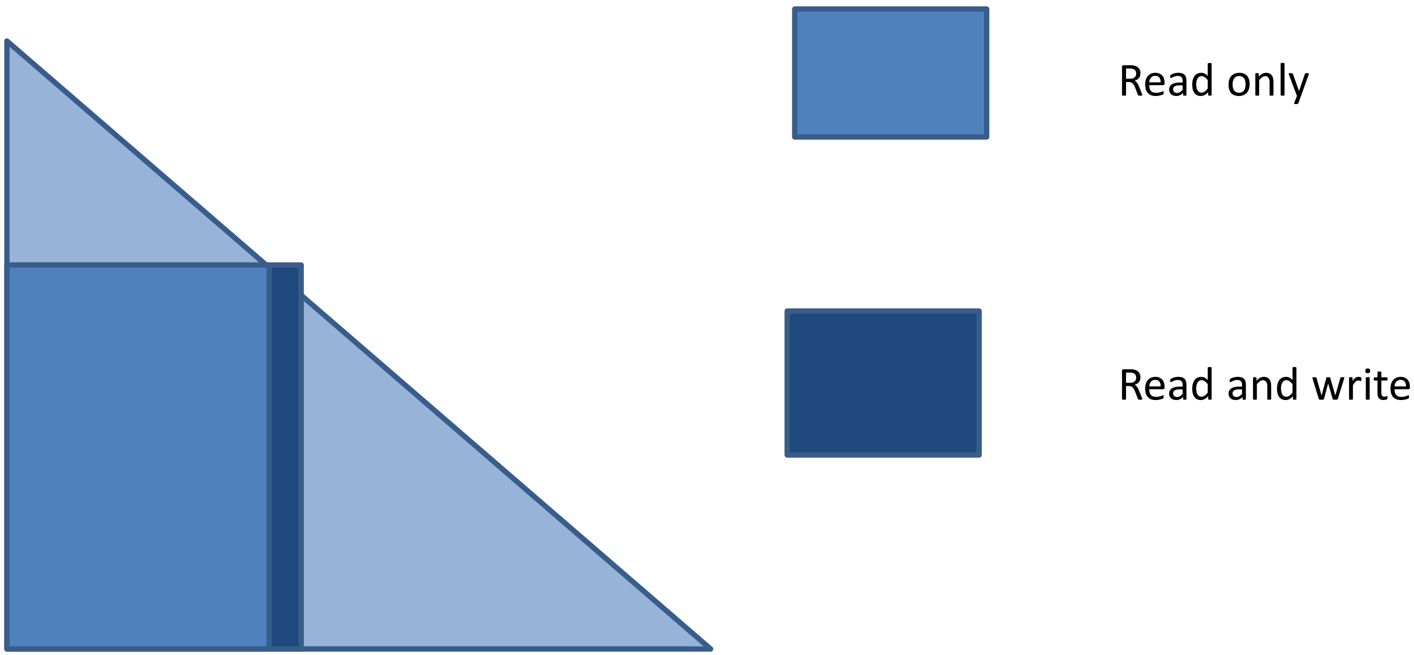
1. This is a variation of Gaussian Elimination algorithm.
2. Storage of matrix A is used to hold matrix L .
3. Only lower triangle of A is used (See $a_{ij} = a_{ij} - a_{ik}a_{jk}$).
4. Pivoting is not needed for stability.
5. About $n^3/6$ multiplications and about $n^3/6$ additions are required.

Data Access Pattern

Column Cholesky Factorization Algorithm

for $j = 1$ **to** n
    for $k = 1$ **to** $j - 1$
        for $i = j$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{jk}$
        end
    end
    $a_{jj} = \sqrt{a_{jj}}$
    for $i = j + 1$ **to** n
        $a_{ij} = a_{ij}/a_{jj}$
    end
end

Data Access Pattern



Parallel Algorithm

- Parallel algorithms are similar to those for LU factorization.

References

- X. S. Li and J. W. Demmel, SuperLU_Dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems, *ACM Trans. Math. Software* 29:110-140, 2003
- P. Hénon, P. Ramet, and J. Roman, PaStiX: A high-performance parallel direct solver for sparse symmetric positive definite systems, *Parallel Computing* 28:301-321, 2002

QR Factorization and HouseHolder Transformation

Theorem Suppose that matrix A is an $m \times n$ matrix with linearly independent columns, then A can be factored as $A = QR$

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

QR Factorization by Gram-Schmidt Process

Consider matrix $A = [\boldsymbol{a}_1|\boldsymbol{a}_2|\dots|\boldsymbol{a}_n]$

Then,

$$\boldsymbol{u}_1 = \boldsymbol{a}_1, \quad \boldsymbol{e}_1 = \frac{\boldsymbol{u}_1}{||\boldsymbol{u}_1||}$$

$$\boldsymbol{u}_2 = \boldsymbol{a}_2 - (\boldsymbol{a}_2 \cdot \boldsymbol{e}_1)\boldsymbol{e}_1, \quad \boldsymbol{e}_2 = \frac{\boldsymbol{u}_2}{||\boldsymbol{u}_2||}$$

…

$$\boldsymbol{u}_k = \boldsymbol{a}_k - (\boldsymbol{a}_k \cdot \boldsymbol{e}_1)\boldsymbol{e}_1 - \cdots - (\boldsymbol{a}_k \cdot \boldsymbol{e}_{k-1})\boldsymbol{e}_{k-1}, \boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{||\boldsymbol{u}_k||}$$

$$A = [\boldsymbol{a}_1|\boldsymbol{a}_2|\dots|\boldsymbol{a}_n] = [\boldsymbol{e}_1|\boldsymbol{e}_2|\dots|\boldsymbol{e}_n] \begin{bmatrix} \boldsymbol{a}_1 \cdot \boldsymbol{e}_1 & \boldsymbol{a}_2 \cdot \boldsymbol{e}_1 & \dots & \boldsymbol{a}_n \cdot \boldsymbol{e}_1 \\ 0 & \boldsymbol{a}_2 \cdot \boldsymbol{e}_2 & \dots & \boldsymbol{a}_n \cdot \boldsymbol{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \boldsymbol{a}_n \cdot \boldsymbol{e}_n \end{bmatrix}$$

$$= QR$$

Householder Transformation

Let $v \in R^n$ be a nonzero vector, the $n \times n$ matrix

$$H = I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}$$

is called a Householder transformation (or reflector).

- Alternatively, let $\boldsymbol{u} = \boldsymbol{v}/||\boldsymbol{v}||$, H can be rewritten as

$$H = I - 2\boldsymbol{u}\boldsymbol{u}^T.$$

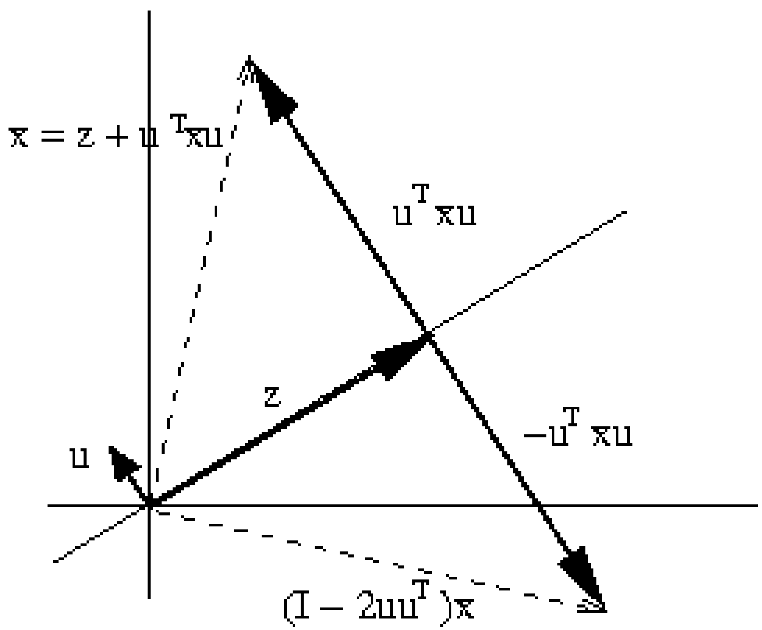
Theorem. A Householder transformation H is symmetric and orthogonal, so $H = H^T = H^{-1}$.

1. Let vector $\boldsymbol{z}$ be perpendicular to $\boldsymbol{v}$.

$$\left(I - 2\frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}\right)\boldsymbol{z} = \boldsymbol{z} - 2\frac{\boldsymbol{v}(\boldsymbol{v}^T\boldsymbol{z})}{\boldsymbol{v}^T\boldsymbol{v}} = \boldsymbol{z}$$

2. Let $\boldsymbol{u} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||}$. Any vector $\boldsymbol{x}$ can be written as $\boldsymbol{x} = \boldsymbol{z} + (\boldsymbol{u}^T\boldsymbol{x})\boldsymbol{u}$.

$$(I - 2\boldsymbol{u}\boldsymbol{u}^T)\boldsymbol{x} = (I - 2\boldsymbol{u}\boldsymbol{u}^T)(\boldsymbol{z} + (\boldsymbol{u}^T\boldsymbol{x})\boldsymbol{u}) = \boldsymbol{z} - (\boldsymbol{u}^T\boldsymbol{x})\boldsymbol{u}$$



- Householder transformation is used to selectively zero out blocks of entries in vectors or columns of matrices.
- Householder is stable with respect to round-off error.

For a given a vector $\boldsymbol{x}$, find a Householder transformation, H, such that $H\boldsymbol{x} = a \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = a\boldsymbol{e}_1$

- Previous vector reflection (case 2) implies that vector $\boldsymbol{u}$ is in parallel with $\boldsymbol{x} - H\boldsymbol{x}$.
- Let $\boldsymbol{v} = \boldsymbol{x} - H\boldsymbol{x} = \boldsymbol{x} - a\boldsymbol{e}_1$, where $a = \pm||\boldsymbol{x}||$ (when the arithmetic is exact, sign does not matter).
- $\boldsymbol{u} = \boldsymbol{v}/||\,\boldsymbol{v}||$
- In the presence of round-off error, use

$\boldsymbol{v} = \boldsymbol{x} + sign(x_1)||\boldsymbol{x}||\boldsymbol{e}_1$

to avoid catastrophic cancellation. Here x_1 is the first entry in the vector $\boldsymbol{x}$.

Thus in practice, $a = -sign(x_1)||\boldsymbol{x}||$

Algorithm for Computing the Householder Transformation

$x_m = \max\{|x_1|, |x_2|, \dots, |x_n|\}$
for $k = 1$ **to** n
    $v_k = x_k / x_m$
end
$\alpha = sign(v_1)[v_1^2 + v_2^2 + \cdots + v_n^2]^{1/2}$
$v_1 = v_1 + \alpha$
$\alpha = -\alpha x_m$
$\boldsymbol{u} = \boldsymbol{v}/||\boldsymbol{v}||$

Remark:

1. The computational complexity is $O(n)$.
2. $H\boldsymbol{x}$ is an $O(n)$ operation compared to $O(n^2)$ for a general matrix-vector product. $H\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{u}(\boldsymbol{u}^T\boldsymbol{x})$.

QR Factorization by HouseHolder Transformation

Key idea:

Apply Householder transformation successively to columns of matrix to zero out sub-diagonal entries of each column.

- **Stage 1**
  - Consider the first column of matrix A and determine a HouseHolder matrix H_1 so that $H_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$
  - Overwrite A by A_1, where

$$A_1 = \begin{bmatrix} \alpha_1 & a_{12}^* & \cdots & a_{1n}^* \\ 0 & a_{22}^* & \vdots & a_{2n}^* \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^* & \cdots & a_{nn}^* \end{bmatrix} = H_1 A$$

Denote vector which defines H_1 vector $\boldsymbol{v}_1$. $\boldsymbol{u}_1 = \boldsymbol{v}_1 / ||\boldsymbol{v}_1||$

Remark: Column vectors $\begin{bmatrix} a_{1n}^* \\ a_{2n}^* \\ \vdots \\ a_{nn}^* \end{bmatrix}$ … $\begin{bmatrix} a_{1n}^* \\ a_{2n}^* \\ \vdots \\ a_{nn}^* \end{bmatrix}$ should be computed by $H\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{u}(\boldsymbol{u}^T \boldsymbol{x})$.

- **Stage 2**
  - Consider the second column of the updated matrix $A \equiv A_1 = H_1 A$ and take the part below the diagonal to determine a Householder matrix H_2^* so that

$$H_2^* \begin{bmatrix} a_{22}^* \\ a_{32}^* \\ \vdots \\ a_{n2}^* \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Remark: size of H_2^* is $(n-1) \times (n-1)$.

Denote vector which defines H_2^* vector $\boldsymbol{v}_2$. $\boldsymbol{u}_2 = \boldsymbol{v}_2 / ||\boldsymbol{v}_2||$

  - Inflate H_2^* to H_2 where

$$H_2 = \begin{bmatrix} 1 & 0 \\ 0 & H_2^* \end{bmatrix}$$

Overwrite A by $A_2 \equiv H_2 A_1$

- **Stage k**
  - Consider the kth column of the updated matrix A and take the part below the diagonal to determine a Householder matrix H_k^* so that

$$H_k^* \begin{bmatrix} a_{kk}^* \\ a_{k+1,k}^* \\ \vdots \\ a_{n,n}^* \end{bmatrix} = \begin{bmatrix} \alpha_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Remark: size of H_k^* is $(n-k+1) \times (n-k+1)$.

Denote vector which defines H_k^* vector $\boldsymbol{v}_k$. $\boldsymbol{u}_k = \boldsymbol{v}_k / ||\boldsymbol{v}_k||$

  - Inflate H_k^* to H_k where

$H_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & H_k^* \end{bmatrix}$, I_{k-1} is $(k-1) \times (k-1)$ identity matrix.

Overwrite A by $A_k \equiv H_k A_{k-1}$

- After $(n-1)$ stages, matrix A is transformed into an upper triangular matrix R.
- $R = A_{n-1} = H_{n-1}A_{n-2} = \cdots = H_{n-1}H_{n-2} \dots H_1 A$
- Set $Q^T = H_{n-1}H_{n-2} \dots H_1$. Then $Q^{-1} = Q^T$. Thus $Q = H_1^T H_2^T \dots H_{n-1}^T$
- $A = QR$

Example. $A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$.

Find H_1 such that $H_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ 0 \\ 0 \end{bmatrix}$.

By $a = -sign(x_1)||\boldsymbol{x}||$,

$\alpha_1 = -\sqrt{3} = -1.721.$

$u_1 = 0.8881, u_2 = u_3 = 0.3250.$

By $H\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{u}\left(\boldsymbol{u}^T\boldsymbol{x}\right)$, $H_1 \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -3.4641 \\ 0.3661 \\ 1.3661 \end{bmatrix}$

$$H_1 A = \begin{bmatrix} -1.721 & -3.4641 \\ 0 & 0.3661 \\ 0 & 1.3661 \end{bmatrix}$$

Find H_2^* and H_2, where

$$H_2^* \begin{bmatrix} 0.3661 \\ 1.3661 \end{bmatrix} = \begin{bmatrix} \alpha_2 \\ 0 \end{bmatrix}$$

So $\alpha_2 = -1.4143, \boldsymbol{u}_2 = \begin{bmatrix} 0.7922 \\ 0.6096 \end{bmatrix}$

So $R = H_2 H_1 A = \begin{bmatrix} -1.7321 & -3.4641 \\ 0 & -1.4143 \\ 0 & 0 \end{bmatrix}$

$$Q = (H_2 H_1)^T = H_1 H_2$$

$$= \begin{bmatrix} -0.5774 & 0.7071 & -0.4082 \\ -0.5774 & 0 & 0.8165 \\ -0.5774 & -0.7071 & -0.4082 \end{bmatrix}$$

Parallel Householder QR

- Householder QR factorization is similar to Gaussian elimination for LU factorization.
- Forming Householder vector $\boldsymbol{v}_k$ is similar to computing multipliers in Gaussian elimination.
- Subsequent updating of remaining unreduced portion of matrix is similar to Gaussian elimination.
- Parallel Householder QR is similar to parallel LU. Householder vectors need to broadcast among columns of matrix.

References

- M. Cosnard, J. M. Muller, and Y. Robert. Parallel QR decomposition of a rectangular matrix, *Numer. Math*. 48:239-249, 1986
- A. Pothen and P. Raghavan. Distributed orthogonal factorization: Givens and Householder algorithms, *SIAM J. Sci. Stat. Comput.* 10:1113-1134, 1989