Matrix Factorizations
LU and *LDL^T*

Section 8.1

CS 3010

Numerical Methods

Dr. Amar Raheja

LU Factorization

- Can a matrix be factored into a multiplication of a lower and upper triangle matrices ?
- An $n \times n$ system of linear equations can be written in matrix form $\boldsymbol{Ax} = \boldsymbol{b}$

  where the coefficient matrix $\boldsymbol{A}$ has the form

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \qquad \boldsymbol{L} = \begin{bmatrix} 1 & & & & \\ \ell_{21} & 1 & & & \\ \ell_{31} & \ell_{32} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \ell_{n1} & \ell_{n2} & \ell_{n3} & \cdots & 1 \end{bmatrix} \qquad \boldsymbol{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ & u_{22} & u_{23} & \cdots & u_{2n} \\ & & u_{33} & \cdots & u_{3n} \\ & & & \ddots & \vdots \\ & & & & u_{nn} \end{bmatrix}$$

- Objective: show that the naive Gaussian algorithm applied to ***A*** yields a factorization of ***A*** into a product of two simple matrices, one unit *lower triangular* (***L***) and the other *upper triangular* (***U***):
- In short, we refer to this as an ***LU* factorization** of ***A***; that is, ***A*** = ***LU***

Numerical Example to Understand

- Linear System to solve

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ 26 \\ -19 \\ -34 \end{bmatrix}$$

- Gaussian Elimination

(forward process only gives)

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

- The forward elimination phase can be interpreted as starting from ***Ax***=***B*** and proceeding to ***MAx* = *Mb***
- where ***M*** is a matrix chosen so that ***M A*** is the coefficient matrix for the upper triangular matrix shown above, ***MA = U***

Numerical Example to Understand

$$MA = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \equiv U$$

- The first step of naive Gaussian elimination results in the system

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -27 \\ -18 \end{bmatrix}$$

- This step can be accomplished by multiplying $\boldsymbol{Ax} = \boldsymbol{b}$ by a lower triangular matrix $\boldsymbol{M}_1 \Rightarrow \boldsymbol{M}_1\,\boldsymbol{Ax} = \boldsymbol{M}_1\boldsymbol{b}$

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \xrightarrow{\text{Multiply 6 by appropriate factor in 1st column}} \boldsymbol{M}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

- Special form of $\boldsymbol{M}_1$: diagonal elements are all 1's, and the only other nonzero elements are in the first column.
- These numbers are the *negatives of the multipliers* located in the positions where they created 0's as coefficients in step 1 of the forward elimination phase.

Numerical Example to Understand

- Continue, step 2 of FE results in

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -21 \end{bmatrix}$$

which is equivalent to $\boldsymbol{M}_2\boldsymbol{M}_1\,\boldsymbol{Ax} = \boldsymbol{M}_2\boldsymbol{M}_1\boldsymbol{b}$ where

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \xrightarrow{\text{Multiply -4 by appropriate factor in 2}^{\text{nd}}\text{ column}} \boldsymbol{M}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix}$$

- These numbers are the *negatives of the multipliers* located in the positions where they created 0's as coefficients in step 2 of the forward elimination phase

Numerical Example to Understand

- Continue, step 3 of FE results in

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 16 \\ -6 \\ -9 \\ -3 \end{bmatrix}$$

which is equivalent to $\boldsymbol{M_3 M_2 M_1 Ax} = \boldsymbol{M_3 M_2 M_1 b}$ where

$$\begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \xrightarrow{\text{Multiply 2 by appropriate factor in 3rd column}} \boldsymbol{M}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{bmatrix}$$

- These numbers are the *negatives of the multipliers* located in the positions where they created 0's as coefficients in step 3 of the forward elimination phase

Numerical Example to Understand

- the forward elimination phase is complete, and with

$$\boldsymbol{M} = \boldsymbol{M}_3\boldsymbol{M}_2\boldsymbol{M}_1$$

$$\boldsymbol{M}\boldsymbol{A} = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \equiv \boldsymbol{U}$$

- From our equation $\boldsymbol{MA} = \boldsymbol{U} \Rightarrow \boldsymbol{A} = \boldsymbol{M}^{-1}\boldsymbol{U}$

$$\Rightarrow \boldsymbol{A} = \boldsymbol{M}_1^{-1}\boldsymbol{M}_2^{-1}\boldsymbol{M}_3^{-1}\boldsymbol{U}$$

- Verify this by multiplying all the $\boldsymbol{M}$ matrices
- Since each $\boldsymbol{M}_k$ has such a special form, its inverse is obtained by simply changing the signs of the negative multiplier entries! Hence, we have

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & -\frac{1}{2} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix}$$

LU Factorization

- We see that ***A*** is **factored** or **decomposed** into a unit lower triangular matrix ***L*** and an upper triangular matrix ***U***.
- The matrix ***L*** consists of the multipliers located in the positions of the elements they annihilated from ***A***, of unit diagonal elements, and of 0 upper triangular elements.
- In fact, we now know the general form of ***L*** and can just write it down directly using the multipliers *without* forming the $\boldsymbol{M_k^{-1}}$'s and the $\boldsymbol{M_k}$'s. The matrix ***U*** is upper triangular (not generally having unit diagonal) and is the final coefficient matrix after the forward elimination phase is completed.

$$\boldsymbol{LU} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ \frac{1}{2} & 3 & 1 & 0 \\ -1 & -\frac{1}{2} & 2 & 1 \end{bmatrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} = \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} = \boldsymbol{A}$$

LU Factorization Theorem

LU FACTORIZATION THEOREM

Let $\boldsymbol{A} = (a_{ij})$ be an $n \times n$ matrix. Assume that the forward elimination phase of the naive Gaussian algorithm is applied to $\boldsymbol{A}$ without encountering any 0 divisors. Let the resulting matrix be denoted by $\widetilde{A} = (\widetilde{a}_{ij})$. If

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \widetilde{a}_{21} & 1 & 0 & \cdots & 0 \\ \widetilde{a}_{31} & \widetilde{a}_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \widetilde{a}_{n1} & \widetilde{a}_{n2} & \cdots & \widetilde{a}_{n,n-1} & 1 \end{bmatrix}$$

and

$$\boldsymbol{U} = \begin{bmatrix} \widetilde{a}_{11} & \widetilde{a}_{12} & \widetilde{a}_{13} & \cdots & \widetilde{a}_{1n} \\ 0 & \widetilde{a}_{22} & \widetilde{a}_{23} & \cdots & \widetilde{a}_{2n} \\ 0 & 0 & \widetilde{a}_{33} & \cdots & \widetilde{a}_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \widetilde{a}_{nn} \end{bmatrix}$$

then $\boldsymbol{A} = \boldsymbol{LU}$.

Proof of Theorem

- We define the Gaussian algorithm formally as follows. Let $\boldsymbol{A}^{(1)} = \boldsymbol{A}$. Then we compute $\boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}, \ldots, \boldsymbol{A}^{(n)}$ recursively by the naive Gaussian algorithm, following these equations:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} \qquad (\text{if } i \leqq k \text{ or } j < k) \tag{1}$$

$$a_{ij}^{(k+1)} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \qquad (\text{if } i > k \text{ and } j = k) \tag{2}$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \left(\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) a_{kj}^{(k)} \qquad (\text{if } i > k \text{ and } j > k) \tag{3}$$

- Equation (7) states that in proceeding from $\boldsymbol{A}^{(k)}$ to $\boldsymbol{A}^{(k+1)}$, we do not alter rows $1, 2, \ldots, k$ or columns $1, 2, \ldots, k - 1$.
- Equation (8) shows how the multipliers are computed and stored in passing from $\boldsymbol{A}^{(k)}$ to $\boldsymbol{A}^{(k+1)}$.
- Equation (9) shows how multiples of row k are subtracted from rows $k + 1, k + 2, \ldots, n$ to produce $\boldsymbol{A}^{(k+1)}$ from $\boldsymbol{A}^{(k)}$.

Proof of Theorem

- $\boldsymbol{A}^{(n)}$ is the final result of the process. (It was referred to as $\tilde{A}$ in the statement of the theorem.)
- The formal definitions of $\boldsymbol{L} = (l_{ik})$ and $\boldsymbol{U} = (u_{kj})$ are therefore

$$\ell_{ik} = 1 \qquad (i = k) \tag{4}$$
$$\ell_{ik} = a_{ik}^{(n)} \qquad (k < i) \tag{5}$$
$$\ell_{ik} = 0 \qquad (k > i) \tag{6}$$
$$u_{kj} = a_{kj}^{(n)} \qquad (j \geqq k) \tag{7}$$
$$u_{kj} = 0 \qquad (j < k) \tag{8}$$

- Now we draw some consequences of these equations. First, it follows immediately from Equation (1) that

$$a_{ij}{}^{(i)} = a_{ij}{}^{(i+1)} = \cdots = a_{ij}{}^{(n)} \tag{9}$$
$$a_{ij}{}^{(j+1)} = a_{ij}{}^{(j+2)} = \cdots = a_{ij}{}^{(n)} \text{ for } (j < n) \tag{10}$$

Proof of Theorem

- From equations (10) and (2), we have

$$a_{ij}^{(n)} = a_{ij}^{(j+1)} = \frac{a_{ij}^{(j)}}{a_{jj}^{(j)}} \qquad (j < n) \tag{11}$$

- From equations (11) and (5), we have

$$\ell_{ik} = a_{ik}^{(n)} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \qquad (k < i) \tag{12}$$

- From equations (7) and (9), we have

$$u_{kj} = a_{kj}^{(n)} = a_{kj}^{(k)} \qquad (k \leqq j) \tag{13}$$

Proof of Theorem

- With the aid of all these equations, we can now prove that ***LU*** = ***A***.
- For the case $i \leqq j$

$$
\begin{aligned}
(\boldsymbol{LU})_{ij} &= \sum_{k=1}^{n} \ell_{ik} u_{kj} \\
&= \sum_{k=1}^{i} \ell_{ik} u_{kj} && \text{by Equation (6)} \\
&= \sum_{k=1}^{i-1} \ell_{ik} u_{kj} + u_{ij} && \text{by Equation (4)} \\
&= \sum_{k=1}^{i-1} \left[\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right] a_{kj}^{(k)} + a_{ij}^{(i)} && \text{by Equation (12) and (17)} \\
&= \sum_{k=1}^{i-1} \left[a_{ij}^{(k)} - a_{ij}^{(k+1)} \right] + a_{ij}^{(i)} && \text{by Equation (3)} \\
&= a_{ij}^{(1)} = a_{ij}
\end{aligned}
$$

Proof of Theorem

- In the remaining case, *i* > *j*, we have

$$
\begin{aligned}
(\boldsymbol{LU})_{ij} &= \sum_{k=1}^{n} \ell_{ik} u_{kj} && \text{by Equation (8)} \\
&= \sum_{k=1}^{j} \ell_{ik} u_{kj} && \text{by Equation (12) and (13)} \\
&= \sum_{k=1}^{j} \left[\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right] a_{kj}^{(k)} \\
&= \sum_{k=1}^{j-1} \left[\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right] a_{kj}^{(k)} + a_{ij}^{(j)} \\
&= \sum_{k=1}^{j-1} \left[a_{ij}^{(k)} - a_{ij}^{(k+1)} \right] + a_{ij}^{(j)} && \text{by Equation (3)} \\
&= a_{ij}^{(1)} = a_{ij}
\end{aligned}
$$

Solving Linear System using *LU* Factorization

- pseudocode for carrying out the ***LU*** factorization, which is sometimes called the **Doolittle factorization**

integer i, k, n; **real array** $(a_{ij})_{1:n\times 1:n}, (\ell_{ij})_{1:n\times 1:n}, (u_{ij})_{1:n\times 1:n}$
for $k = 1$ **to** n **do**
    $\ell_{kk} \leftarrow 1$
    for $j = k$ **to** n **do**

$$u_{kj} \leftarrow a_{kj} - \sum_{s=1}^{k-1} \ell_{ks} u_{sj}$$

    end do
    for $i = k + 1$ **to** n **do**

$$\ell_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} \ell_{is} u_{sk} \right) \Big/ u_{kk}$$

    end do
end do

Solving Linear Systems using *LU* Factorization

- When ***LU*** factorization of ***A*** is available, we can solve the system ***Ax*** = ***b***

by writing

$$\boldsymbol{LUx} = \boldsymbol{b}$$

Then we solve two triangular systems:

$$\boldsymbol{Lz} = \boldsymbol{b} \qquad \text{for } \boldsymbol{z}$$

Likewise, ***x*** is obtained by the pseudocode

$$\boldsymbol{Ux} = \boldsymbol{z}$$

This is particularly useful for problems that involve the same coefficient matrix ***A*** and many different right-hand vectors ***b***.

Solving Linear Systems using *LU* Factorization

- Since ***L*** is unit lower triangular, ***z*** is obtained by the pseudocode

integer i, n; **real array** $(b_i)_{1:n}, (\ell_{ij})_{1:n\times 1:n}, (z_i)_{1:n}$
$z_1 \leftarrow b_1$
for $i = 2$ **to** n **do**

$$z_i \leftarrow b_i - \sum_{j=1}^{i-1} \ell_{ij} z_j$$

end for

- This algorithm applies the forward phase of Gaussian elimination to the right-hand-side vector ***b***. [Recall that the l_{ij}'s are the *multipliers* that have been stored in the array (a_{ij}).]

Solving Linear Systems using *LU* Factorization

- Likewise, ***x*** is obtained by the pseudocode

$$
\begin{array}{l}
\textbf{integer } i, n; \quad \textbf{real array } (u_{ij})_{1:n\times 1:n}, (x_i)_{1:n}, (z_i)_{1:n} \\
x_n \leftarrow z_n / u_{nn} \\
\textbf{for } i = n-1 \textbf{ to } 1 \textbf{ step } -1 \textbf{ do} \\
\qquad x_i \leftarrow \left(z_i - \displaystyle\sum_{j=i+1}^{n} u_{ij} x_j \right) \Bigg/ u_{ii} \\
\textbf{end for}
\end{array}
$$

- This algorithm for solving ***Ux*** = ***z*** is the back substitution phase of the naive Gaussian elimination process.

Example for LU factorization

- Find ***L*** and ***U*** for

$$A = \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 1 & 3 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 0 & 3 & -1 \end{bmatrix} \qquad \boldsymbol{U}\ \begin{bmatrix} 3 & 0 & 3 \\ 0 & -1 & 3 \\ 0 & 0 & 8 \end{bmatrix}$$

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/3 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/3 & -3 & 1 \end{bmatrix}$$

Class Exercises Section 8.1

- Problem 1b

$$A = \begin{bmatrix} 1 & 0 & \frac{1}{3} & 0 \\ 0 & 1 & 3 & -1 \\ 3 & -3 & 0 & 6 \\ 0 & 2 & 4 & -6 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 3 & -3 & 1 & 0 \\ 0 & 2 & -1/4 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 1 & 0 & 1/3 & 0 \\ 0 & 1 & 3 & -1 \\ 0 & 0 & 8 & 3 \\ 0 & 0 & 0 & -13/4 \end{bmatrix}$$

LDL^T Factorization

- In the $\boldsymbol{LDL^T}$ factorization, $\boldsymbol{L}$ is unit lower triangular, and $\boldsymbol{D}$ is a diagonal matrix.
- This factorization can be carried out if $\boldsymbol{A}$ is symmetric and has an ordinary $\boldsymbol{LU}$ factorization, with $\boldsymbol{L}$ unit lower triangular.
- To see this, we start with

$$\boldsymbol{LU} = \boldsymbol{A} = \boldsymbol{A}^T = (\boldsymbol{LU})^T = \boldsymbol{U}^T\boldsymbol{L}^T$$

- Since $\boldsymbol{L}$ is unit lower triangular, it is invertible, and we can write

$$\boldsymbol{U} = \boldsymbol{L}^{-1}\boldsymbol{U}^T\boldsymbol{L}^T \quad \Rightarrow \quad \boldsymbol{U}(\boldsymbol{L}^T)^{-1} = \boldsymbol{L}^{-1}\boldsymbol{U}^T$$

- Since the right side of this equation is lower triangular and the left side is upper triangular, both sides are diagonal, say, $\boldsymbol{D}$. From the equation $\boldsymbol{U}(\boldsymbol{L}^T)^{-1} = \boldsymbol{D}$, we have

$$\boldsymbol{U} = \boldsymbol{DL}^T \text{ and } \boldsymbol{A} = \boldsymbol{LU} = \boldsymbol{LDL}^T$$

Derivation of *LDL^T* Psuedocode

- we write a_{ij} as generic elements of ***A*** and l_{ij} as generic elements of ***L*** . The diagonal of ***D*** has elements d_{ii}, or d_i
- From the equation ***A*** = ***LDL^T*** , we have

$$
\begin{aligned}
a_{ij} &= \sum_{\nu=1}^{n} \sum_{\mu=1}^{n} \ell_{i\nu} d_{\nu\mu} \ell^T_{\mu j} \\
&= \sum_{\nu=1}^{n} \sum_{\mu=1}^{n} \ell_{i\nu} d_{\nu} \delta_{\nu\mu} \ell_{j\mu} \\
&= \sum_{\nu=1}^{n} \ell_{i\nu} d_{\nu} \ell_{j\nu} \qquad (1 \leqq i,\ j \leqq n)
\end{aligned}
$$

Derivation of LDL^T Psuedocode

- Use the fact that l_{ij} = 0 when $j > i$ and l_{ii} = 1 to continue the argument

$$a_{ij} = \sum_{\nu=1}^{\min(i,j)} \ell_{i\nu} d_\nu \ell_{j\nu} \qquad (1 \leqq i,\ j \leqq n)$$

- Assume now that $j < i$. Then

$$\begin{aligned} a_{ij} &= \sum_{\nu=1}^{j} \ell_{i\nu} d_\nu \ell_{j\nu} \\ &= \sum_{\nu=1}^{j-1} \ell_{i\nu} d_\nu \ell_{j\nu} + \ell_{ij} d_j \ell_{jj} \\ &= \sum_{\nu=1}^{j-1} \ell_{i\nu} d_\nu \ell_{j\nu} + \ell_{ij} d_j \qquad (1 \leqq j \leqq i \leqq n) \end{aligned}$$

Derivation of LDL^T Pseudocode

- In particular, let $j = i$. We get

$$a_{ii} = \sum_{\nu=1}^{i-1} \ell_{i\nu} d_\nu \ell_{i\nu} + d_i \qquad (1 \leqq i \leqq n)$$

- Equivalently, we have

$$d_i = a_{ii} - \sum_{\nu=1}^{i-1} d_\nu \ell_{i\nu}^2 \qquad (1 \leqq i \leqq n)$$

- Particular cases of this are

$$\begin{aligned} d_1 &= a_{11} \\ d_2 &= a_{22} - d_1 \ell_{21}^2 \\ d_3 &= a_{33} - d_1 \ell_{31}^2 - d_2 \ell_{32}^2 \\ &\text{etc.} \end{aligned}$$

Derivation of LDL^T Psuedocode

- Now we can limit our attention to the cases $1 \leqq j \leqq n$, where we have

$$a_{ij} = \sum_{\nu=1}^{j-1} \ell_{i\nu} d_\nu \ell_{j\nu} + \ell_{ij} d_j \qquad (1 \leqq j < i \leqq n)$$

- Solving for l_{ij}, we obtain

$$\ell_{ij} = \left[a_{ij} - \sum_{\nu=1}^{j-1} \ell_{i\nu} d_\nu \ell_{j\nu} \right] \bigg/ d_j \qquad (1 \leqq j < i \leqq n)$$

- Taking $j = 1$, we have

$$\ell_{i1} = a_{i1}/d_1 \qquad (2 \leqq i \leqq n)$$

- This formula produces column one in $\boldsymbol{L}$. Taking $j = 2$, we have

$$\ell_{i2} = (a_{i2} - \ell_{i1} d_1 \ell_{21})/d_2 \qquad (3 \leqq i \leqq n)$$

LDL^T Pseudocode

- This formula produces column two in ***L***. The formal algorithm for the ***LDL***^{T} factorization is as follows:

integer i, j, n, ν; **real array** $(a_{ij})_{1:n\times 1:n}, (\ell_{ij})_{1:n\times 1:n}, (d_i)_{1:n}$
for $j = 1$ **to** n
$\quad \ell_{jj} = 1$

$$d_j = a_{jj} - \sum_{\nu=1}^{j-1} d_\nu \ell_{j\nu}^2$$

$\quad$**for** $i = j + 1$ **to** n
$\quad\quad \ell_{ji} = 0$

$$\ell_{ij} = \left(a_{ij} - \sum_{\nu=1}^{j-1} \ell_{i\nu} d_\nu \ell_{j\nu} \right) \Big/ d_j$$

$\quad$**end for**
end for

Example of LDL^T

- Determine the $\boldsymbol{LDL^T}$ factorization of the matrix

$$A = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Note: symmetric matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{3}{4} & 1 & 0 & 0 \\ \frac{1}{2} & \frac{2}{3} & 1 & 0 \\ \frac{1}{4} & \frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 & 2 & 1 \\ 0 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} = LU$$

$$U = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & \frac{3}{4} & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & 1 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 1 & \frac{1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} = DL^T$$

Clearly, we have $\boldsymbol{A = LDL^T}$

Class Exercise

- Determine the LDL^T factorization for the following matrix:

$$A = \begin{bmatrix} 1 & 2 & -1 & 1 \\ 2 & 3 & -4 & 3 \\ -1 & -4 & -1 & 3 \\ 1 & 3 & 3 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 2 & 1 & 0 \\ 1 & -1 & 1 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}$$