

ENGG 5781 Matrix Analysis and Computations

Lecture 7: Linear Systems

Wing-Kin (Ken) Ma

2024–25 First Term

Department of Electronic Engineering
The Chinese University of Hong Kong

Lecture 7: Linear Systems

- triangular systems and LU decomposition
- LDM decomposition, LDL decomposition and Cholesky factorization
- iterative methods for linear systems

Main Results

- a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is said to have an **LU decomposition** if it can be factored as

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is lower triangular; $\mathbf{U} \in \mathbb{R}^{n \times n}$ is upper triangular

- does not always exist
- pivoting: there exists a permutation matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$ such that $\mathbf{PA} = \mathbf{LU}$
- if $\mathbf{A} \in \mathbb{S}^{n \times n}$ has an LU decomposition, then $\mathbf{U} = \mathbf{D}\mathbf{L}^T$ where \mathbf{D} is diagonal
- **Cholesky factorization:** if $\mathbf{A} \in \mathbb{S}^{n \times n}$ is PD, it can always be factored as

$$\mathbf{A} = \mathbf{G}\mathbf{G}^T,$$

where \mathbf{G} is lower triangular.

The System of Linear Equations

Consider the system of linear equations

$$\mathbf{Ax} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ are given, and $\mathbf{x} \in \mathbb{R}^n$ is the solution to the system.

- \mathbf{A} will be assumed to be nonsingular (unless specified)
- we consider the real case for convenience; extension to the complex case is simple

Solving the Linear System

Problem: compute the solution to $\mathbf{Ax} = \mathbf{b}$ in a numerically efficient manner.

- the problem is easy if \mathbf{A}^{-1} is known
 - but computing \mathbf{A}^{-1} also costs computations...
 - do you know how to compute \mathbf{A}^{-1} efficiently?
- here, \mathbf{A} is assumed to be a general nonsingular matrix.
 - the problem may become easy in some special cases, e.g., orthogonal \mathbf{A} , circulant \mathbf{A} .

LU Decomposition

LU decomposition: given $\mathbf{A} \in \mathbb{R}^{n \times n}$, find two matrices $\mathbf{L}, \mathbf{U} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{A} = \mathbf{LU},$$

where

$\mathbf{L} \in \mathbb{R}^{n \times n}$ is lower triangular with unit diagonal elements (i.e., $\ell_{ii} = 1$ for all i);
 $\mathbf{U} \in \mathbb{R}^{n \times n}$ is upper triangular.

Idea: **Suppose** that \mathbf{A} has an LU decomposition. Then, solving $\mathbf{Ax} = \mathbf{b}$ can be recast as two linear system problems:

1. solve $\mathbf{Lz} = \mathbf{b}$ for \mathbf{z} , and then
2. solve $\mathbf{Ux} = \mathbf{z}$ for \mathbf{x} .

Questions:

1. how to solve $\mathbf{Lz} = \mathbf{b}$, and then $\mathbf{Ux} = \mathbf{z}$?
2. how to perform $\mathbf{A} = \mathbf{LU}$? Does LU decomposition exist?

Backward Substitution

Example: a 3×3 upper triangular system

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}.$$

If $u_{11}, u_{22}, u_{33} \neq 0$, then x_1, x_2, x_3 can be solved by, in sequence,

$$x_3 = z_3 / u_{33}$$

$$x_2 = (z_2 - u_{23}x_3) / u_{22}$$

$$x_1 = (z_1 - u_{12}x_2 - u_{13}x_3) / u_{11}$$

Backward Substitution

Backward substitution for solving $Ux = z$:

$$x_i = \left(z_i - \sum_{j=i+1}^n u_{ij}x_j \right) / u_{ii}, \quad \text{for } i = n, n-1, \dots, 1.$$

Backward substitution in MATLAB form:

```
function x= back_subs(U,z)
n= length(z);
x= zeros(n,1);
x(n)= z(n)/U(n,n);
for i= n-1:-1:1,
    x(i)= ( z(i)- U(i,i+1:n)*x(i+1:n) )/U(i,i);
end;
```

- complexity: $\mathcal{O}(n^2)$

Forward Substitution

Example: a 3×3 lower triangular system

$$\begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

If $\ell_{11}, \ell_{22}, \ell_{33} \neq 0$, then z_1, z_2, z_3 can be solved by

$$z_1 = b_1 / \ell_{11}$$

$$z_2 = (b_2 - \ell_{21}z_1) / \ell_{22}$$

$$z_3 = (b_3 - \ell_{31}z_1 - \ell_{32}z_2) / \ell_{33}$$

Forward Substitution

Forward substitution for solving $\mathbf{Lz} = \mathbf{b}$:

$$z_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} z_j \right) / \ell_{ii}, \quad \text{for } i = 1, 2, \dots, n.$$

Forward substitution in MATLAB form:

```
function z= for_subs(L,b)
n= length(b);
z= zeros(n,1);
z(1)= b(1)/L(1,1);
for i=2:1:n
    z(i)= (b(i)-L(i,1:i-1)*z(1:i-1))/L(i,i);
end;
```

- complexity: $\mathcal{O}(n^2)$

Gauss Transformations: the Key Building Block for LU

Observation: given $\mathbf{x} \in \mathbb{R}^n$ that has $x_k \neq 0$, $1 \leq k \leq n$,

$$\underbrace{\begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -\frac{x_{k+1}}{x_k} & 1 & \\ & & \vdots & & \ddots \\ & & -\frac{x_n}{x_k} & & & 1 \end{bmatrix}}_{=\mathbf{M}} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The above \mathbf{M} also satisfies

$$\mathbf{M}\mathbf{y} = \mathbf{y}, \quad \text{for any } \mathbf{y} = [y_1, \dots, y_{k-1}, 0, \dots, 0]^T, \quad y_i \in \mathbb{R}.$$

Characterization of \mathbf{M} :

$$\mathbf{M} = \mathbf{I} - \boldsymbol{\tau} \mathbf{e}_k^T, \quad \boldsymbol{\tau} = [0, \dots, 0, x_{k+1}/x_k, \dots, x_n/x_k]^T.$$

Finding \mathbf{U} by Gauss Elimination

Problem: find Gauss transformations $\mathbf{M}_1, \dots, \mathbf{M}_{n-1} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} = \mathbf{U}, \quad \mathbf{U} \text{ being upper triangular.}$$

Step 1: choose \mathbf{M}_1 such that $\mathbf{M}_1 \mathbf{a}_1 = [a_{11}, 0, \dots, 0]^T$

- **if** $a_{11} \neq 0$, then we can choose

$$\mathbf{M}_1 = \mathbf{I} - \boldsymbol{\tau}^{(1)} \mathbf{e}_1^T, \quad \boldsymbol{\tau}^{(1)} = [0, a_{21}/a_{11}, \dots, a_{n1}/a_{11}]^T.$$

- result:

$$\mathbf{M}_1 \mathbf{A} = \begin{bmatrix} a_{11} & \times & \dots & \times \\ 0 & \times & \dots & \times \\ \vdots & \vdots & & \vdots \\ 0 & \times & \dots & \times \end{bmatrix}$$

Finding \mathbf{U} by Gauss Elimination

Step 2: let $\mathbf{A}^{(1)} = \mathbf{M}_1 \mathbf{A}$. Choose \mathbf{M}_2 such that $\mathbf{M}_2 \mathbf{a}_2^{(1)} = [a_{12}^{(1)}, a_{22}^{(1)}, 0, \dots, 0]^T$.

- **if** $a_{22}^{(1)} \neq 0$, then we can choose

$$\mathbf{M}_2 = \mathbf{I} - \boldsymbol{\tau}^{(2)} \mathbf{e}_2^T, \quad \boldsymbol{\tau}^{(2)} = [0, 0, a_{32}^{(1)}/a_{22}^{(1)}, \dots, a_{n,2}^{(1)}/a_{22}^{(1)}]^T.$$

- result:

$$\mathbf{M}_2 \mathbf{A}^{(1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \times & \dots & \times \\ 0 & a_{22}^{(1)} & \times & \dots & \times \\ \vdots & 0 & \times & & \times \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \times & \dots & \times \end{bmatrix}$$

Finding \mathbf{U} by Gauss Elimination

Let $\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)}$, $\mathbf{A}^{(0)} = \mathbf{A}$. Note $\mathbf{A}^{(k)} = \mathbf{M}_k \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A}$.

Step k : Choose \mathbf{M}_k such that $\mathbf{M}_k \mathbf{a}_k^{(k-1)} = [a_{1k}^{(k-1)}, \dots, a_{kk}^{(k-1)}, 0, \dots, 0]^T$.

- **if** $a_{kk}^{(k-1)} \neq 0$, then

$$\mathbf{M}_k = \mathbf{I} - \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T, \quad \boldsymbol{\tau}^{(k)} = [0, \dots, 0, a_{k+1,k}^{(k-1)} / a_{kk}^{(k-1)}, \dots, a_{n,k}^{(k-1)} / a_{kk}^{(k-1)}]^T,$$

- result:

$$\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)} = \begin{bmatrix} a_{11}^{(k-1)} & \cdots & a_{1k}^{(k-1)} & \times & \cdots & \times \\ 0 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & & a_{kk}^{(k-1)} & \vdots & & \vdots \\ \vdots & & 0 & \times & & \times \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \times & \cdots & \times \end{bmatrix}$$

- $\mathbf{A}^{(n-1)} = \mathbf{U}$ is upper triangular

Where is \mathbf{L} ?

We have seen that under the assumption of $a_{kk}^{(k-1)} \neq 0$ for all k ,

$$\mathbf{U} = \mathbf{M}_{n-1} \cdots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} \text{ is upper triangular.}$$

But where is \mathbf{L} ?

Property 7.1. Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ be lower triangular. Then, \mathbf{AB} is lower triangular. Also, if \mathbf{A}, \mathbf{B} have unit diagonal entries, then \mathbf{AB} has unit diagonal entries.

Property 7.2. If $\mathbf{A} \in \mathbb{R}^{n \times n}$ is lower triangular, then $\det(\mathbf{A}) = \prod_{i=1}^n a_{ii}$.

Property 7.3. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be nonsingular lower triangular. Then, \mathbf{A}^{-1} is lower triangular with $[\mathbf{A}^{-1}]_{ii} = 1/a_{ii}$.

Suppose that every \mathbf{M}_k is invertible. Then,

$$\mathbf{L} = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \cdots \mathbf{M}_{n-1}^{-1}$$

satisfies $\mathbf{A} = \mathbf{LU}$, and is lower triangular with unit diagonal entries.

A Naive Implementation of LU (Don't Use It)

```
function [L,U]= my_naive_lu(A)
n= size(A,1);
L= eye(n); t= zeros(n,1); U= A;
for k=1:1:n-1,
    rows= k+1:n;
    t(rows)= U(rows,k)/U(k,k);
    M= eye(n); M(rows,k)= -t(rows);
    U= M*U;           % compute  $\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)}$ 
    L= L*inv(M);       % to eventually obtain  $\mathbf{L} = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} \cdots \mathbf{M}_{n-1}^{-1}$ 
end;
```

Weaknesses:

- the above code treats each $\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{A}^{(k-1)}$ as a general matrix multiplication process, which takes $\mathcal{O}(n^3)$ flops. It does not utilize structures of \mathbf{M}_k .
- (more serious) to compute \mathbf{L} , the above code calls inverse $n - 1$ times. If the problem is to solve $\mathbf{Ax} = \mathbf{b}$, then why not just call inverse once for \mathbf{A} ?

Computing \mathbf{L}

Fact: $\mathbf{M}_k^{-1} = \mathbf{I} + \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T$.

Verification: by noting $[\boldsymbol{\tau}^{(k)}]_k = 0$,

$$\begin{aligned} (\mathbf{I} + \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T) \mathbf{M}_k &= (\mathbf{I} + \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T) (\mathbf{I} - \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T) \\ &= \mathbf{I} + \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T - \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T - \underbrace{\boldsymbol{\tau}^{(k)} \mathbf{e}_k^T \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T}_{=0} = \mathbf{I}. \end{aligned}$$

By the same spirit, it can be verified that

$$\mathbf{L} = \mathbf{M}_1^{-1} \cdots \mathbf{M}_{n-1}^{-1} = \mathbf{I} + \sum_{k=1}^{n-1} \boldsymbol{\tau}^{(k)} \mathbf{e}_k^T$$

A More Mature LU Code (Still Not the LU inside MATLAB)

```
function [L,U]= my_lu(A)
n= size(A,1);
L= eye(n); t= zeros(n,1); U= A;
for k=1:1:n-1,
    rows= k+1:n;
    t(rows)= U(rows,k)/U(k,k);
    U(rows,rows)= U(rows,rows)- t(rows)*U(k,rows);
    U(rows,k)= 0;
    L(rows,k)= t(rows);
end;
```

- complexity: $\mathcal{O}(2n^3/3)$
- works as long as $a_{kk}^{(k-1)}$ —the so-called **pivots**—are all nonzero

Existence of LU Decomposition

Theorem 7.1. A matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ has an LU decomposition if every principal submatrix $\mathbf{A}_{\{1, \dots, k\}}$ satisfies

$$\det(\mathbf{A}_{\{1, \dots, k\}}) \neq 0,$$

for $k = 1, 2, \dots, n - 1$. If the LU decomposition of \mathbf{A} exists and \mathbf{A} is nonsingular, then (\mathbf{L}, \mathbf{U}) is unique.

- the proof is essentially about when $a_{kk}^{(k-1)} \neq 0$.

Discussion

- the LU algorithm described above requires nonzero pivots, $a_{kk}^{(k-1)} \neq 0$ for all k .
- Gauss elimination is known to be numerically unstable when a pivot is close to zero
- **pivoting:** at each Gauss elimination step, interchange the rows of $\mathbf{A}^{(k)}$ to obtain better pivots.
 - when you call `lu(A)` or `A\b` in MATLAB, it always perform pivoting
- besides solving $\mathbf{Ax} = \mathbf{b}$, LU decomposition can also be used to
 - compute \mathbf{A}^{-1} : let $\mathbf{B} = \mathbf{A}^{-1}$.

$$\mathbf{AB} = \mathbf{I} \iff \mathbf{Ab}_i = \mathbf{e}_i, \quad i = 1, \dots, n \text{ (i.e., solve } n \text{ linear systems).}$$

- compute $\det(\mathbf{A})$: $\det(\mathbf{A}) = \det(\mathbf{L})\det(\mathbf{U}) = \prod_{i=1}^n u_{ii}$ (cf. Property 7.2).

LDM Decomposition

LDM decomposition: given $\mathbf{A} \in \mathbb{R}^{n \times n}$, find matrices $\mathbf{L}, \mathbf{D}, \mathbf{M} \in \mathbb{R}^{n \times n}$ such that

$$\mathbf{A} = \mathbf{LDM}^T,$$

where

\mathbf{L} is lower triangular with unit diagonal elements;

$\mathbf{D} = \text{Diag}(d_1, \dots, d_n)$;

\mathbf{M} is lower triangular with unit diagonal elements.

- a different way of writing the LU decomposition: if $\mathbf{A} = \mathbf{LU}$ is the LU decomposition, then the same \mathbf{L} ,

$$\mathbf{D} = \text{Diag}(u_{11}, \dots, u_{nn}), \quad \mathbf{M} = \mathbf{U}^T \mathbf{D}^{-1},$$

form the LDM decomposition.

- the existence of LDM decomposition follows that of LU.

Solving LDM Decomposition

Notation: $\mathbf{A}_{i:j,k:l}$ denotes a submatrix of \mathbf{A} obtained by keeping $i, i+1, \dots, j$ rows and $k, k+1, \dots, l$ columns of \mathbf{A} .

Idea: examine $\mathbf{A} = \mathbf{LDM}^T$ column by column:

$$\mathbf{A}_{1:n,j} = \mathbf{A}\mathbf{e}_j = \mathbf{L}\mathbf{v}, \quad (\star)$$

where $1 \leq j \leq n$,

$$\mathbf{v} = \mathbf{DM}^T\mathbf{e}_j.$$

Observations:

1. $v_i = d_j m_{ji}$;
2. $v_i = 0, i = j+1, \dots, n$;
3. (\star) can be expanded as

$$\begin{bmatrix} \mathbf{A}_{1:j,j} \\ \mathbf{A}_{j+1:n,j} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{1:j,1:j} & \mathbf{0} \\ \mathbf{L}_{j+1:n,1:j} & \mathbf{L}_{j+1:n,j+1:n} \end{bmatrix} \begin{bmatrix} \mathbf{v}_{1:j} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{1:j,1:j}\mathbf{v}_{1:j} \\ \mathbf{L}_{j+1:n,1:j}\mathbf{v}_{1:j} \end{bmatrix}$$

Solving LDM Decomposition

Recall from the last page that

$$\mathbf{A}_{1:j,j} = \mathbf{L}_{1:j,1:j} \mathbf{v}_{1:j}$$

$$\mathbf{A}_{j+1:n,j} = \mathbf{L}_{j+1:n,1:j} \mathbf{v}_{1:j}$$

Problem: suppose that $\mathbf{L}_{1:n,1:j-1}$, the first $j-1$ columns of \mathbf{L} , is known. Find $\mathbf{L}_{1:n,j}$, the j th column of \mathbf{L} .

1. $\mathbf{L}_{1:j,1:j}$ is known (why?)
2. solve $\mathbf{A}_{1:j,j} = \mathbf{L}_{1:j,1:j} \mathbf{v}_{1:j}$ for $\mathbf{v}_{1:j}$
3. $\mathbf{L}_{j+1:n,j} = (\mathbf{A}_{j+1:n,j} - \mathbf{L}_{j+1:n,1:j-1} \mathbf{v}_{1:j-1}) / v_j$.
4. (bonus) $d_j = v_j$, $m_{ji} = v_i / d_i$ for $i = 1, \dots, j-1$.

An LDM Decomposition Code

```
function [L,D,M]= my_ldm(A)
n= size(A,1);
L= eye(n); d= zeros(n,1); M= eye(n);
v= zeros(n,1);
for j=1:n,
    % solve  $\mathbf{A}_{1:j,j} = \mathbf{L}_{1:j,1:j} \mathbf{v}_{1:j}$  by forward substitution
    v(1:j)= for_subs(L(1:j,1:j),A(1:j,j));
    d(j)= v(j);
    for i=1:j-1,
        M(j,i)= v(i)'/d(i);
    end;
    L(j+1:n,j)= (A(j+1:n,j)-L(j+1:n,1:j-1)*v(1:j-1))/v(j);
end;
D= diag(d);
```

- complexity: $\mathcal{O}(2n^3/3)$ (same as the previous LU code)

LDL Decomposition for Symmetric Matrices

If \mathbf{A} is symmetric, then the LDM decomposition may be reduced to

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T.$$

Theorem 7.2. If $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{M}^T$ is the LDM decomposition of a nonsingular symmetric \mathbf{A} , then $\mathbf{L} = \mathbf{M}$.

Solving LDL:

- recall that in the previous LDM decomposition, the key is to find the unknown

$$\mathbf{v} = \mathbf{D}\mathbf{M}^T \mathbf{e}_j$$

by solving $\mathbf{A}_{1:j} = \mathbf{L}_{1:j,1:j} \mathbf{v}_{1:j}$ via forward substitution.

- Now, since $\mathbf{M} = \mathbf{L}$,

$$v_i = d_i \ell_{ji}.$$

Finding \mathbf{v} is much easier and there is no need to run forward substitution.

An LDL Decomposition Code

```
function [L,D]= my_ldl(A)
n= size(A,1);
L= eye(n); d= zeros(n,1); M= eye(n);
v= zeros(n,1);
for j=1:n,
    v(1:j)= for_subs(L(1:j,1:j),A(1:j,j));
    v(1:j-1)= L(j,1:j-1)' .* d(1:j-1); % replace for_subs.
    v(j)= A(j,j)- L(j,1:j-1)*v(1:j-1); % replace for_subs.
    d(j)= v(j);
    for i=1:j-1,
        M(j,i)= v(i)'/d(i);
    end;
    L(j+1:n,j)= (A(j+1:n,j)-L(j+1:n,1:j-1)*v(1:j-1))/v(j);
end;
D= diag(d);
```

- complexity: $\mathcal{O}(n^3/3)$, half of LU or LDM

Cholesky Factorization for PD Matrices

Cholesky factorization: given a PD $\mathbf{A} \in \mathbb{S}^n$, factorize \mathbf{A} as

$$\mathbf{A} = \mathbf{G}\mathbf{G}^T,$$

where $\mathbf{G} \in \mathbb{R}^{n \times n}$ is lower triangular with positive diagonal elements.

Theorem 7.3. If $\mathbf{A} \in \mathbb{S}^n$ is PD, then there exists a unique lower triangular $\mathbf{G} \in \mathbb{R}^{n \times n}$ with positive diagonal elements such that $\mathbf{A} = \mathbf{G}\mathbf{G}^T$.

- idea: if \mathbf{A} is symmetric and PD, then its LDL decomposition

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

has $d_i > 0$ for all $i = 1, \dots, n$ (as an exercise, verify this). Putting $\mathbf{G} = \mathbf{L}\mathbf{D}^{\frac{1}{2}}$ yields the Cholesky factorization.

- can be computed in $\mathcal{O}(n^3/3)$ (similar to LDL and details skipped), no pivoting required, numerically very stable

Iterative Methods for Linear Systems

- solving linear systems via LU requires $\mathcal{O}(n^3)$
- $\mathcal{O}(n^3)$ is too much for large-scale linear systems
- the motivation behind iterative methods is to seek less expensive ways to find an (approximate) linear system solution
- note: see also Lecture 1 for ideas of handling large-scale LS problems, which is relevant to the context here

The Key Insight of Iterative Methods

- assume $a_{ii} \neq 0$ for all i
- observe

$$\begin{aligned}\mathbf{b} = \mathbf{Ax} &\iff b_i = a_{ii}x_i + \sum_{j \neq i} a_{ij}x_j, \quad i = 1, \dots, n \\ &\iff x_i = \left(b_i - \sum_{j \neq i} a_{ij}x_j \right) / a_{ii}, \quad i = 1, \dots, n \quad (\dagger)\end{aligned}$$

- idea: find an \mathbf{x} that fulfils the equations in (\dagger)

Jacobi Iterations

input: a starting point $\mathbf{x}^{(0)}$
for $k = 0, 1, 2, \dots$
 $x_i^{(k+1)} = \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right) / a_{ii}$, for $i = 1, \dots, n$
end

- complexity per iteration: $\mathcal{O}(n^2)$ for dense \mathbf{A} , $\mathcal{O}(\text{nnz}(\mathbf{A}))$ for sparse \mathbf{A}
- the Jacobi update step can be computed in a parallel or distributed fashion
 - same idea appeared in distributed power control in 2G or 3G wireless networks
- a natural idea, heuristic at first glance
- does the Jacobi iterations converge to the linear system solution?
 - it does not, in general
 - it does if the diagonal elements a_{ii} 's are “dominant” compared to the off-diagonal elements; see Theorem 10.1.1 in [\[Golub-van-Loan'12\]](#) for details

Gauss-Seidel Iterations

input: a starting point $\mathbf{x}^{(0)}$

for $k = 0, 1, 2, \dots$

for $i = 1, 2, \dots, n$

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right) / a_{ii}$$

end

end

- use the most recently available \mathbf{x} to perform update
- sequential, cannot be computed in a distributed or parallel manner
- guaranteed to converge to the linear system solution if
 - \mathbf{A} has diagonally dominant characteristics (similar to the Jacobi iterations)
 - \mathbf{A} is symmetric PD; see Theorem 10.1.2 in [\[Golub-van-Loan'12\]](#)

References

[Golub-van-Loan'12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd edition, JHU Press, 2012.