AMS526: Numerical Analysis I
(Numerical Linear Algebra)
Lecture 2: Structure and Efficiency;
Block Matrices and Algorithms;
Range and Null Space

Xiangmin Jiao

Stony Brook University
Outline

1. Structure and Efficiency
2. Block Matrices and Algorithms
3. Fast Matrix Multiplication
4. Range, Null Space and Rank
Banded Matrices

- Matrix is *sparse* if a large fraction of its entries are zero
- *Band matrix* is an important special case of *sparse matrices*
- For $A \in \mathbb{R}^{m \times n}$,
  - Lower bandwidth is $p$ if $a_{ij} = 0$ whenever $i > j + p$
  - Upper bandwidth is $q$ if $a_{ij} = 0$ whenever $j > i + q$
  - Bandwidth is $p + q + 1$

- Example: $\times$ designates arbitrary nonzero value

$$
\begin{bmatrix}
\times & \times & \times \\
\times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times & \times & \times \\
\end{bmatrix}
$$

What are lower and upper bandwidths, respectively?
Some Special Triangular Matrices

What are lower and upper bandwidths of $A \in \mathbb{R}^{m \times n}$ of following types?

<table>
<thead>
<tr>
<th>Matrix type</th>
<th>Lower bw</th>
<th>Upper bw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper triangular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower triangular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tridiagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper bidiagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower bidiagonal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper Hessenberg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower Hessenberg</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Classical linear algebra algorithms typically involve transformation into one of these forms
- Modern linear algebra algorithms often involve one of these forms in conjunction with orthogonal matrices (next lecture)
Some Special Triangular Matrices

What are lower and upper bandwidths of $A \in \mathbb{R}^{m\times n}$ of following types?

<table>
<thead>
<tr>
<th>Matrix type</th>
<th>Lower bw</th>
<th>Upper bw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagonal</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Upper triangular</td>
<td>0</td>
<td>$n - 1$</td>
</tr>
<tr>
<td>Lower triangular</td>
<td>$m - 1$</td>
<td>0</td>
</tr>
<tr>
<td>Tridiagonal</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Upper bidiagonal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Lower bidiagonal</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Upper Hessenberg</td>
<td>1</td>
<td>$n - 1$</td>
</tr>
<tr>
<td>Lower Hessenberg</td>
<td>$m - 1$</td>
<td>1</td>
</tr>
</tbody>
</table>

- Classical linear algebra algorithms typically involve transformation into one of these forms
- Modern linear algebra algorithms often involve one of these forms in conjunction with *orthogonal matrices* (later lectures)
Why Take Advantage of Structures? Reason 1: Efficiency

- Example: Matrix multiplication update \( C = C + AB \), where \( A, B, C \in \mathbb{R}^{n \times n} \) and are upper triangular. 3 \( \times \) 3 example:

\[
AB = \begin{bmatrix}
a_{11}b_{11} & a_{11}b_{12} + a_{12}b_{22} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\
a_{21}b_{12} + a_{22}b_{22} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\
a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33}
\end{bmatrix}
\]

- Consider only nonzeros in computation

```plaintext
for i = 1 : n
    for j = i : n
        c_{ij} = c_{ij} + a_{i;i:j} b_{i:j;j}
```

Note: colon notation: \( n_1 : n_2 \) (e.g., 1 : 6 \( \equiv [1, 2, 3, 4, 5, 6] \))

- Instead of 2\( n^3 \) flops with standard algorithm, \#flops is approximately

\[
\sum_{i=1}^{n} \sum_{j=i}^{n} 2(j - i + 1) = \sum_{i=1}^{n} \sum_{j=1}^{n-i+1} 2j \approx \sum_{i=1}^{n} (n - i + 1)^2 = \sum_{i=1}^{n} i^2 \approx \frac{n^3}{3}
\]
Why Take Advantage of Structures? Reason 2: Storage

- Example: Suppose $A \in \mathbb{R}^{n \times n}$ has lower bandwidth $p$ and upper bandwidth $q$, and $p, q \ll n$
- **Band Storage**: Store $A$ in $(p + q + 1)$-by-$n$ array $A.band$, such as

$$
\begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} & a_{24} \\
    a_{32} & a_{33} & a_{34} & a_{35} \\
    a_{43} & a_{44} & a_{45} & a_{46} \\
    a_{54} & a_{55} & a_{56} \\
    a_{65} & a_{66}
\end{bmatrix}
\Rightarrow
\begin{bmatrix}
    - & - & a_{13} & a_{24} & a_{35} & a_{46} \\
    - & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\
    a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\
    a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & -
\end{bmatrix}
$$

- $a_{ij} = A.band(i - j + q + 1, j)$ for $i, j = 1, \ldots, n$
- Instead of $n^2$ numbers with standard storage, it now requires $n(p + q + 1)$ numbers
- However, algorithm must be adapted to use new storage format
Diagonal Matrices

- A diagonal matrix has upper and lower bandwidths zero
- If $D \in \mathbb{R}^{m \times n}$, denote it by

$$D = \text{diag}(d_1, d_2, \ldots, d_q), \quad q = \min\{m, n\}$$

where $d_i = d_{ii}$, and $d$ is vector in band storage
- If $m = n$ and $x \in \mathbb{R}^n$, $Dx = d \ast x$
- Scaling by diagonal matrices
  - If $A \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{m \times m}$, $DA$ scales rows of $A$
  - If $A \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{n \times n}$, $AD$ scales columns of $A$
  - Both requires $mn$ flops
Symmetry

- For real matrices $A \in \mathbb{R}^{n \times n}$,
  - symmetric: $A = A^T$
  - skew-symmetric: $A = -A^T$

- For complex matrices $A \in \mathbb{C}^{n \times n}$,
  - Hermitian: $A = A^H$
  - skew-Hermitian: $A = -A^H$

- Storage can be halved by storing only lower triangular part. E.g.,

  $$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \iff A.vec = [ 1 \ 2 \ 3 \ 4 \ 5 \ 6 ]$$

- $a_{ij} = \begin{cases} A.vec(n(j - 1) - j(j - 1)/2 + i) & i \geq j \\ A.vec(n(i - 1) - i(i - 1)/2 + j) & i < j \end{cases}$

- Note: $n(j - 1) - j(j - 1)/2 \neq (n - j/2)(j - 1)$ with integer operation

- Exploiting symmetry may also reduce computational cost sometimes
Permutation Matrices

- *Permutation matrix* is reordering of identity matrix
- A permutation matrix has a single 1 per row and per column. E.g.,

\[
P = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\]

- \(P^{-1} = P^T\), so that \(P^T(Px) = x\) (an example of *orthogonal matrices*);
- \(P\) can be stored in integer vector storing column index, such as \(v = [2 \ 4 \ 3 \ 1]\)
  - \(y = Px \Rightarrow y = x(v)\)
  - \(y = P^T x \Rightarrow y(v) = x\)
- Multiply \(A \in \mathbb{R}^{m \times n}\) by permutation matrix
  - Left-multiplying by \(P \in \mathbb{R}^{m \times m}\) exchanges rows
  - Right-multiplying by \(P \in \mathbb{R}^{n \times n}\) exchanges columns
Outline

1. Structure and Efficiency
2. Block Matrices and Algorithms
3. Fast Matrix Multiplication
4. Range, Null Space and Rank
Block Matrices

- Matrix $A \in \mathbb{R}^{m \times n}$ can be partitioned into blocks
  
  $$A = \begin{bmatrix} A_{11} & \cdots & A_{1r} \\ \vdots & \ddots & \vdots \\ A_{q1} & \cdots & A_{qr} \end{bmatrix}^{\begin{bmatrix} m_1 \\ \vdots \\ m_q \end{bmatrix}},$$

  where $m = \sum_{i=1}^{q} m_i$, $n = \sum_{j=1}^{r} n_j$, and

  $A_{ij}$ designates the $(i, j)$ block (submatrix)

- Matrix structures also generalize block matrices, such as

  - Block diagonal
    
    $$\begin{bmatrix} D_{11} & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & D_{33} \end{bmatrix}$$

  - Block triangular: lower
    
    $$\begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix}$$

    ; upper

    $$\begin{bmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{bmatrix}$$

- Blocks do not need to be square

- Algorithm descriptions can often be simplified using block matrices
Block Matrix Operators

Basic matrix operations generalize to block matrices

- Scaling: $B = \mu A \Rightarrow B_{ij} = \mu A_{ij}$
- Addition: $C = A + B \Rightarrow C_{ij} = A_{ij} + B_{ij}$
- Transposition: $B = A^T \Rightarrow B_{ij} = A_{ji}^T$
- Matrix multiplication update: Suppose $A$ has $s_1$ block rows and $s_2$ block columns, and $B$ has $s_2$ block rows and $s_3$ block columns

$$C = C + AB \text{ using block-matrix operators}$$

for $i = 1 : s_1$

for $j = 1 : s_3$

for $k = 1 : s_2$

$$C_{ij} \leftarrow C_{ij} + A_{ik}B_{kj}$$

- Block matrix operations are richer in level-2 and 3 operations, which can improve cache performance and reduce data movement
Example: Matrix Multiplication

- Matrix multiplication: Suppose

\[
A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} m_1 \\ r_1 \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} r_2 \\ n_2 \end{bmatrix}
\]

and then \( C = AB \) is

\[
\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}
\]

where

\[
C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}, \quad i, j = 1, 2
\]

- Block-matrix operations are similar to matrix operations, except that
  1. Dimensions of sub-matrices must match (column partition of \( A \) must match row partition of \( B \))
  2. Matrix multiplication do not commute (\( XY \neq YX \))
Notation of Submatrices

- Suppose \( \alpha = [\alpha_1, \ldots, \alpha_s] \) and \( \beta = [\beta_1, \ldots, \beta_t] \) are integer vectors with distinct components, where \( 1 \leq \alpha_i \leq m \) and \( 1 \leq \beta_i \leq n \).

- For \( A \in \mathbb{R}^{m \times n} \), \( A(\alpha, \beta) \) (or \( A_{\alpha,\beta} \)) denote \( s \)-by-\( t \) submatrix

\[
A(\alpha, \beta) = \begin{bmatrix}
a_{\alpha_1\beta_1} & \cdots & a_{\alpha_1\beta_t} \\
\vdots & \ddots & \vdots \\
a_{\alpha_s\beta_1} & \cdots & a_{\alpha_s\beta_t}
\end{bmatrix}
\]

- Integer vectors may be denoted by colon notation
  - \( n_1 : n_3 \) (e.g., \( 1 : 6 \equiv [1, 2, 3, 4, 5, 6] \))
  - \( n_1 : s : n_3 \), where \( s \) is stride (e.g., \( 1 : 2 : 6 \equiv [1, 3, 5] \))

- \( A_{ij} = A(\tau + 1 : \tau + m_i, \mu + l : \mu + n_j) \) for

\[
A = \begin{bmatrix}
A_{11} & \cdots & A_{1r} \\
\vdots & \ddots & \vdots \\
A_{q1} & \cdots & A_{qr}
\end{bmatrix}
\begin{bmatrix}
m_1 \\
\vdots \\
m_q
\end{bmatrix}
\]

where \( \tau = \sum_{k=1}^{i-1} m_k \) and \( \mu = \sum_{k=1}^{j-1} n_k \)
Example Algorithm Using Submatrices

- Matrix multiplication update: Suppose $A$, $B$, and $C$ are $N$-by-$N$ block matrices with $\ell$-by-$\ell$ blocks

\[
C = C + AB \text{ using block-matrix operators}
\]

1: \textbf{for} $i = 1 : N$
2: \quad \alpha \leftarrow (i - 1)\ell + l : i\ell$
3: \textbf{for} $j = 1 : N$
4: \quad \beta \leftarrow (j - 1)\ell + l : j\ell$
5: \textbf{for} $k = 1 : N$
6: \quad \gamma \leftarrow (k - 1)\ell + l : k\ell$
7: \quad $C(\alpha, \beta) \leftarrow C(\alpha, \beta) + A(\alpha, \gamma)B(\gamma, \beta)$

- Note: ‘:\’ has lower precedence than arithmetic operator
- Line 7 involves level-3 operation gaxpy on submatrices. This incurs extra implicit loops, but overall performance can be improved significantly, especially when each block fits into a cache line
Outline

1. Structure and Efficiency
2. Block Matrices and Algorithms
3. Fast Matrix Multiplication
4. Range, Null Space and Rank
Strassen Matrix Multiplication

- Multiplying two $n \times n$ matrices requires $\sim 2n^3$ flops using inner-product-based algorithm
- Is this optimal?
Strassen Matrix Multiplication

- Multiplying two $n \times n$ matrices requires $\sim 2n^3$ flops using inner-product-based algorithm
- Is this optimal?

- Strassen’s method requires $O(n^s)$ flops, where $s = \log_2 7 \approx 2.807$
  - It is recursive algorithm applied to matrices of size $2^k \times 2^k$
  - For matrices of sizes not of $2^k \times 2^k$, fill missing rows and columns with zeros

- Asymptotically faster algorithm, due to D. Coppersmith and S. Winograd (1990), requires $O(n^{2.3755})$ flops

- Asymptotically fastest algorithm currently known, due to V. Williams (2011), requires $O(n^{2.3727})$ flops
Matrix Multiplication in Practice

- In practice, inner-product-based algorithm is almost always used
  - Earlier authors estimated that Strassen’s algorithm is faster for matrices with widths of $n \gtrsim 100$ for optimized implementations
  - Compared to a highly optimized traditional multiplication on current architectures, Strassen’s algorithm is not faster unless $n \gtrsim 1000$, and the benefit is marginal for matrix sizes of several thousand
  - Coppersmith and Winograd’s algorithm has a huge constant $C$ in front of it, so it is never fast enough in practice for realistic problem sizes
  - The asymptotically “faster” methods are less accurate than standard multiplication when using floating-point numbers (more prone to rounding errors and cancellation errors)

- Lower-complexity algorithms are sometimes used to prove theoretical time bounds of other algorithms

- For sparse matrices, efficient matrix-matrix multiplication should take advantage of sparsity (see SMMP for further reading)
Notes on Other Practical Ways to Improve Performance

- Exploit special properties of matrices
  - E.g., multiplying of Fourier matrix with a vector can be done in $O(n \log n)$ time using FFT (covered in AMS 527)
  - Similar algorithms include fast sine and cosine transformations and wavelet transformations
  - These algorithms are useful in fast algorithms for solving linear systems of elliptic PDEs on regular grids with periodic boundary conditions

- Vectorization and Parallelization
  - Some modern processors support vectorization (such as Intel Xeon and Intel Phi processors)
  - Use multiple processors to perform multiple operations concurrently
  - Both of these involve block-matrix operations

- Using efficient software libraries BLAS and LAPACK
Outline

1. Structure and Efficiency
2. Block Matrices and Algorithms
3. Fast Matrix Multiplication
4. Range, Null Space and Rank
Terminology of Vector Space

- **Vector space** is closed under addition and scalar multiplication, with zero vector as a member
- Vector space span by a set of vectors \( \{a_j\} \) is

\[
\text{span}\{a_1, \ldots, a_n\} = \left\{ \sum_{j=1}^{n} \beta_j a_j \mid \beta_j \in \mathbb{R}\right\}
\]

- Space spanned by \( n \)-vectors is a *subspace* of \( \mathbb{R}^n \)
- If \( \{a_1, \ldots, a_n\} \) is *linearly independent*, then the \( a_j \) are the *basis* of \( S = \text{span}\{a_1, \ldots, a_n\} \), *dimension* of \( S \) is \( \dim(S) = n \), and each \( b \in S \) is unique linear combination of the \( a_j \)

- If \( S_1 \) and \( S_2 \) are two subspaces, then \( S_1 \cap S_2 \) is a subspace, so is \( S_1 + S_2 = \{b_1 + b_2 \mid b_1 \in S_1, b_2 \in S_2\} \)
  - Note: \( S_1 + S_2 \) is different from \( S_1 \cup S_2 \); latter may not be a subspace

- Two subspaces \( S_1 \) and \( S_2 \) of \( \mathbb{R}^n \) are *complementary subspaces* of each other if \( S_1 + S_2 = \mathbb{R}^n \) and \( S_1 \cap S_2 = \{0\} \)
  - In other words, \( \dim(S_1) + \dim(S_2) = n \) and \( S_1 \cap S_2 = \{0\} \)
Range and Null Space

Definition

The *range* of a matrix $A$, written as $\text{range}(A)$ or $\text{ran}(A)$, is the set of vectors that can be expressed as $Ax$ for some $x$

$$\{y \in \mathbb{R}^m | y = Ax \text{ for some } x \in \mathbb{R}^n\}.$$
Range and Null Space

**Definition**

The *range* of a matrix $A$, written as $\text{range}(A)$ or $\text{ran}(A)$, is the set of vectors that can be expressed as $Ax$ for some $x$

$$\{ y \in \mathbb{R}^m | y = Ax \text{ for some } x \in \mathbb{R}^n \}.$$ 

**Theorem**

$\text{range}(A)$ *is the space spanned by the columns of A.*

Therefore, the *range* of $A$ is also called the *column space* of $A$. 
Range and Null Space

**Definition**
The *range* of a matrix $A$, written as $\text{range}(A)$ or $\text{ran}(A)$, is the set of vectors that can be expressed as $Ax$ for some $x$
\[ \{ y \in \mathbb{R}^m | y = Ax \text{ for some } x \in \mathbb{R}^n \}. \]

**Theorem**
$\text{range}(A)$ is the space spanned by the columns of $A$.

Therefore, the *range* of $A$ is also called the *column space* of $A$.

**Definition**
The *null space* of $A \in \mathbb{R}^{m \times n}$, written as $\text{null}(A)$, is the set of vectors $x$ that satisfy $Ax = 0$.

Entries of $x \in \text{null}(A)$ give coefficient of $\sum x_i a_i = 0$.

Note: The null space of $A$ is in general *not* a complementary subspace of $\text{range}(A)$. 
Relationship Between Null and Range Space

- For real matrices \( A \in \mathbb{R}^{m \times n} \)
  - \( \text{null}(A) \) and \( \text{range}(A^T) \) are complementary subspaces
  - For symmetric matrices \( (A = A^T) \), \( \text{null}(A) \) and \( \text{range}(A) \) are complementary subspaces

- For complex matrices \( A \in \mathbb{C}^{m \times n} \)
  - \( \text{null}(A) \) and \( \text{range}(A^H) \) are complementary subspaces
  - For Hermitian matrices \( (A = A^H) \), \( \text{null}(A) \) and \( \text{range}(A) \) are complementary subspaces
Rank

**Definition**

The *column rank* of a matrix is the dimension of its column space. The *row rank* is the dimension of the space spanned by its rows.

**Question**: Can the column rank and the row rank be different?
Rank

### Definition

The *column rank* of a matrix is the dimension of its column space. The *row rank* is the dimension of the space spanned by its rows.

#### Question: Can the column rank and the row rank be different?

#### Answer: No. We will give a proof in future lectures.

- We therefore simply say the *rank* of a matrix.

#### Question: Given $A \in \mathbb{R}^{m \times n}$, what is $\dim(\text{null}(A)) + \text{rank}(A)$ equal to?
**Definition**

The *column rank* of a matrix is the dimension of its column space. The *row rank* is the dimension of the space spanned by its rows.

**Question:** Can the column rank and the row rank be different?

**Answer:** No. We will give a proof in future lectures.

- We therefore simply say the *rank* of a matrix.

**Question:** Given $A \in \mathbb{R}^{m \times n}$, what is $\dim(\text{null}(A)) + \text{rank}(A)$ equal to?

**Answer:** $n$.

- A real matrix $A$ is rank-1 if it can be written as $A = uv^T$, where $u$ and $v$ are nonzero vectors
Full Rank

Definition

A matrix has full rank if it has the maximal possible rank, i.e., \( \min\{m, n\} \).

Otherwise, it is called rank deficient.

Theorem

A matrix \( A \in \mathbb{R}^{m \times n} \) with \( m \geq n \) has full rank if and only if it maps no two distinct vectors to the same vector.

In other word, the linear mapping defined by \( Ax \) for \( x \in \mathbb{R}^n \) is one-to-one.

Proof.

(\( \Rightarrow \)) Column vectors of \( A \) forms a basis of range \((A)\), so every \( b \in \text{range}(A) \) has a unique linear expansion in terms of the columns of \( A \).

(\( \Leftarrow \)) If \( A \) does not have full rank, then its column vectors are linear dependent, so its vectors do not have a unique linear combination.
Full Rank

Definition
A matrix has full rank if it has the maximal possible rank, i.e., \( \min \{m, n\} \).

Otherwise, it is called rank deficient.

Theorem
A matrix \( A \in \mathbb{R}^{m \times n} \) with \( m \geq n \) has full rank if and only if it maps no two distinct vectors to the same vector.

In other word, the linear mapping defined by \( Ax \) for \( x \in \mathbb{R}^n \) is one-to-one.
Full Rank

**Definition**
A matrix has *full rank* if it has the maximal possible rank, i.e., \( \min\{m, n\} \).

Otherwise, it is called *rank deficient*.

**Theorem**
A matrix \( A \in \mathbb{R}^{m \times n} \) with \( m \geq n \) has full rank if and only if it maps no two distinct vectors to the same vector.

In other words, the linear mapping defined by \( Ax \) for \( x \in R^n \) is one-to-one.

**Proof.**

(⇒) Column vectors of \( A \) forms a basis of range(\( A \)), so every \( b \in \text{range}(A) \) has a unique linear expansion in terms of the columns of \( A \).

(⇐) If \( A \) does not have full rank, then its column vectors are linear dependent, so its vectors do not have a unique linear combination.
Full Rank vs. Non-singularity

- If \( A \in \mathbb{R}^{n \times n} \) and \( AX = I \), then \( X \) is the inverse of \( A \), denoted by \( A^{-1} \)
  
  \[ (AB)^{-1} = B^{-1}A^{-1} \]
  
  \[ (A^{-1})^T = (A^T)^{-1} = A^{-T} \]
  
  \[ (A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1} U)^{-1} V^T A^{-1} \]
  
  for \( U, V \in \mathbb{R}^{n \times k} \) (Sherman-Morrison-Woodbury formula)

- If \( A^{-1} \) exists, \( A \) is nonsingular. \( A \) is square and has full rank.

- In \( A \) is nonsingular, linear system \( Ax = b \) results in \( x = A^{-1}b \), and it is the inverse problem of matrix-vector multiplication

- If \( A \in \mathbb{R}^{m \times n} \) where \( n \neq m \) and \( A \) has full rank, what is the inverse problem of matrix-vector multiplication?

- If \( A \) is rank deficient, what is the inverse problem of matrix-vector multiplication?

- We will need progressively more advanced linear algebra concepts to answer these questions in later part of this semester.