Lecture 18
Linear Algebra, II
Lecture 18 Goals

• Matrix inversion, singularity, rank, and determinants
• Solving systems of linear equations
Problem of inversion

Solve for unknown vector $x$:

$$Ax = y$$

where $A$ is $m \times n$, $x$ is $n \times 1$, and $y$ is $m \times 1$. 
Problem of inversion

Solve for unknown vector $x$:

$$Ax = y$$

where $A$ is $mxn$, $x$ is $nx1$, and $y$ is $mx1$.

That’s basic algebra!

$$x = y/A$$
Problem of inversion

Solve for unknown vector $x$:

$$Ax = y$$

where $A$ is $m \times n$, $x$ is $n \times 1$, and $y$ is $m \times 1$.

Or is it?

$$x = y/A$$

Recall that matrix multiplication is not commutative, so we cannot divide $B=C/A$ since $AB/A$ is not equal to $B$.

What is Matrix “division” anyway?
Problem of inversion

Solve for unknown vector $x$:

$$Ax = y$$

where $A$ is $mxn$, $x$ is $nx1$, and $y$ is $mx1$.

Correct solution (use the Matrix inverse):

$$A\backslash Ax = A\backslash y \quad \Rightarrow \quad x = A\backslash y$$

Order matters, i.e.

$$AB = C \Rightarrow A\backslash AB = A\backslash C \Rightarrow B = A\backslash C$$
Problem of inversion

Solve for unknown vector $x$:

$$Ax = y$$

where $A$ is $mxn$, $x$ is $nx1$, and $b$ is $mx1$.

Correct solution (use the Matrix inverse):

$$A\backslash Ax = A\backslash y \quad \Rightarrow \quad x = A\backslash y$$

For matrices, the definition of the “inverse”, or “one over” the matrix, has to be defined properly.

Key question: When does the inverse exist?
Answer: The Determinant

If the determinant is **non-zero**, the matrix can be **inverted** and unique solution exists for $Ax=y$.

If the determinant is **zero**, the matrix **cannot be inverted**, there can be either 0 or an infinite number of solutions to $Ax=y$. 
Wait. What’s the Determinant?

Geometric interpretation: The determinant describes how the area (2D), volume (3D), or hypervolume (4D or higher) defined by set of points X changes when matrix A is applied to X.

Determinants (square matrices)

Consider a 2 x 2 matrix

\[
\begin{bmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{bmatrix}
\]

The determinant of a 2 x 2 matrix is

\[
\begin{vmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}
\]
Determinants (square matrices)

Finding the determinant of a general n x n square matrix requires evaluation of a complicated polynomial of the coefficients of the matrix, but there is a simple recursive approach.

$$|B| = \text{det}(B) = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & k \end{vmatrix} = a \begin{vmatrix} e & f \\ h & k \end{vmatrix} - b \begin{vmatrix} d & f \\ g & k \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$
Solving $Ax=y$

- When $A$ is non-singular (has non-zero determinant) $A^{-1}$ exists, and one can find $x$ via
  $$ x = \text{inv}(A) \ast y $$

- However, depending on $A$, this can be computationally inefficient and or less precise than using $x = A \backslash y$

- The MATLAB \ operation (called mldivide) takes the form of $A$ into account while trying to solve $A \backslash y$

- doc mldivide
Operation of $A\backslash y$ in MATLAB

1. Is $A$ square?
   - Yes: Use triangular solver
   - No: Is $A$ triangular?
     - Yes: Use triangular solver
     - No: Is $A$ permuted triangular?
       - Yes: Use permuted triangular solver
       - No: Is $A$ upper Hessenberg?
         - Yes: Use Hessenberg solver
         - No: Is $A$ Hermitian?
           - Yes: Use LDL solver
           - No: Does $A$ have a real and positive diagonal?
             - Yes: Use Cholesky solver
             - No: Does Cholesky succeed?
               - Yes: Use LDL solver
               - No: Use LU solver

Linear Equations

- $y = mx + b$ is a linear function, $y(x)$
- Setting $mx + b = c$ is a linear equation
- Systems of equations
  Have multiple Equations

\[
\begin{align*}
2x_1 + 9x_2 &= 5 \\
3x_1 - 4x_2 &= 7
\end{align*}
\]

Solving a system of equations involves finding a set (or sets) of values that allow all the equations to hold.

These are not always solvable!
Linear equations - Example

Linear equations

\[
\begin{aligned}
2x_1 + 9x_2 &= 5 \\
3x_1 - 4x_2 &= 7
\end{aligned}
\]

can be written in matrix form

\[
\begin{bmatrix}
2 & 9 \\
3 & -4
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
5 \\
7
\end{bmatrix}
\]

Or more symbolically as

\[y = Ax\]

where

\[
A = \begin{bmatrix}
2 & 9 \\
3 & -4
\end{bmatrix}
\quad x = \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
\quad y = \begin{bmatrix}
5 \\
7
\end{bmatrix}
\]
Linear equations – General Form

m linear equations with n variables:

\[
\begin{align*}
  y_1 &= a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n \\
  y_2 &= a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n \\
  \vdots \\
  y_m &= a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n
\end{align*}
\]

Can be written in matrix as

\[
y = Ax
\]

where

\[
y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
\]
Linear equations

For the equation $Ax = y$, there are 3 distinct cases:

- **Square**, equal number of unknowns and equations

- **Underdetermined**: more unknowns than equations

- **Overdetermined**: fewer unknowns than equations
Types of solutions with “random” data

“Generally” the following observations would hold:

- **One solution** (e.g., 2 lines intersect at one point)

- **Infinite solutions** (e.g., 2 planes intersect at many points)

- **No solutions** (e.g., 3 lines don’t intersect at a point)
Linear equations

But other things can happen. For example:

No solution (2 parallel lines)
Many solutions (2 parallel lines)

No solutions (2 parallel planes)
Solutions (3 lines that do intersect at a point)
What is a linear function?

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. It is said to be linear if

- $f(x + y) = f(x) + f(y)$, $\forall x, y \in \mathbb{R}^n$
- $f(\alpha x) = \alpha f(x)$, $\forall x \in \mathbb{R}^n \forall \alpha \in \mathbb{R}$

Superposition can be applied to linear systems!
Matrix representation of a linear function

- consider function $f : \mathbb{R}^n \to \mathbb{R}^m$ given by $f(x) = Ax$, where $A \in \mathbb{R}^{m \times n}$

- matrix multiplication function $f$ is linear

- converse is true: any linear function $f : \mathbb{R}^n \to \mathbb{R}^m$ can be written as $f(x) = Ax$ for some $A \in \mathbb{R}^{m \times n}$

- representation via matrix multiplication is unique: for any linear function $f$ there is only one matrix $A$ for which $f(x) = Ax$ for all $x$

- $y = Ax$ is a concrete representation of a generic linear function
What is a linear function?

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. It is said to be linear if

- $f(x + y) = f(x) + f(y), \quad \forall x, y \in \mathbb{R}^n$
- $f(\alpha x) = \alpha f(x), \quad \forall x \in \mathbb{R}^n, \ \forall \alpha \in \mathbb{R}$

This is sometimes called superposition.

When $f$ is represented with matrix $A$, it is clear it satisfies the properties above, i.e.,

$A(x+y) = Ax + Ay$

$A(cx) = cAx$
Solving $Ax=b$

Two key questions

1) How can we tell when no solutions exist?
2) When solutions exist how can we find and represent them?
Range of a matrix

The range of \( A \in \mathbb{R}^{m \times n} \) is defined as

\[
\mathcal{R}(A) = \{ Ax \mid x \in \mathbb{R}^n \} \subseteq \mathbb{R}^m
\]

\( \mathcal{R}(A) \) can be interpreted as

- the set of vectors that can be ‘hit’ by linear mapping \( y = Ax \)
- the span of columns of \( A \)
- the set of vectors \( y \) for which \( Ax = y \) has a solution
Ax=b has a solution when b is in R(A)
Linear independence

A set of vectors \( \{v_1, v_2, \ldots, v_k\} \) is independent if

\[
\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = 0 \implies \alpha_1 = \alpha_2 = \cdots = 0
\]

Some equivalent conditions:

- Coefficients of \( \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k \) are uniquely determined, i.e.,

\[
\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = \beta_1 v_1 + \beta_2 v_2 + \cdots + \beta_k v_k
\]

implies \( \alpha_1 = \beta_1, \alpha_2 = \beta_2, \ldots, \alpha_k = \beta_k \)

- No vector \( v_i \) can be expressed as a linear combination of the other vectors \( v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k \)
Rank of a matrix

The rank of a matrix is equal to the maximal number of linearly independent vectors in the range of the matrix.

A square matrix $A$ of dimension $n \times n$ is said to be **full rank** if the rank of the matrix is $n$, i.e. $\text{rank}(A) = n$

If a matrix is full rank, it can be inverted.

The rank of a matrix can be computed using the command `rank`.

```
A =
1 1 1 1
1 -2 -3 -4
1 2 3 4
1 2 2 1

>> rankOfA = rank(A)

rankOfA =
4
```
Key Question #1: Existence

How can we tell when no solution exists to $Ax=b$?

- When $\text{rank}([A\ b]) > \text{rank}(A)$ there isn’t a linear combination of the columns of $A$ that can be used to represent $y$, i.e. $Ax = b$ has no solution.
Key Question #2: Uniqueness?

How can we tell the solution to $Ax=b$ is unique? When multiple solutions exists how can we find and represent them?
Nullspace of a matrix

The *nullspace* of $A \in \mathbb{R}^{m \times n}$ is defined as

$$\mathcal{N}(A) = \{ x \in \mathbb{R}^n \mid Ax = 0 \}$$

- $\mathcal{N}(A)$ is set of vectors mapped to zero by $y = Ax$
- $\mathcal{N}(A)$ is set of vectors orthogonal to all rows of $A$
Nullspace $\neq \{0\} \Rightarrow$ Infinite Solutions

the nullspace of $A \in \mathbb{R}^{m \times n}$ is defined as

$$\mathcal{N}(A) = \{ x \in \mathbb{R}^n \mid Ax = 0 \}$$

- $\mathcal{N}(A)$ is set of vectors mapped to zero by $y = Ax$
- $\mathcal{N}(A)$ is set of vectors orthogonal to all rows of $A$

$\mathcal{N}(A)$ gives ambiguity in $x$ given $y = Ax$:

- if $y = Ax$ and $z \in \mathcal{N}(A)$, then $y = A(x + z)$
- conversely, if $y = Ax$ and $y = A\tilde{x}$, then $\tilde{x} = x + z$ for some $z \in \mathcal{N}(A)$
Existence and Uniqueness

• **Ax=b**

• **Existence:** Having a solution means $b$ in $R(A)$

• **Having a unique solution means** $Az=0$ iff $z$ is an appropriately sized zero vector
  
  • Otherwise $x+z \neq x$, and $A(x+z)=Ax+Az = Ax = b$

• **Uniqueness:** Having a unique solution requires $N(A)=\{0\}$
When do we need to compute $N(A)$?

For $\text{dim}(A)$ $m$ by $n$,

$\text{dim}(\text{Null}(A))$ is $m$ by $n-\text{rank}(A)$.

So when $A$ is full rank (i.e. $\text{rank}(A) = n$), $N(A) = \{0\}$
Finding the nullspace in MATLAB

null(A) returns an orthonormal basis for the null space of A
null(A,’r’) returns a “rational” basis for the null space of A

For illustrative purposes, use the script initializeMatrices.m, which puts 4 matrices of size 4 x 4 into the workspace:

A is of rank 4
B is of rank 3
C is of rank 2
D is of rank 1

```
>> initializeMatrices
>> whos

Name      Size      Bytes  Class          Attributes
--------- --------    ------  --------------
A         4x4      128    double array
B         4x4      128    double array
C         4x4      128    double array
D         4x4      128    double array

Grand total is 64 elements using 512 bytes
```
Finding the null space in MATLAB

Since A is of rank 4, the null space of A is reduced to zero.

The null space of the matrix B (of rank 3) is of dimension 1

The command `null(A,'r')` will take the matrix A and return the set of vectors which belong to the null space of A. The ‘r’ is added so Matlab returns fractional vectors when possible.

Null space of A is empty, because the matrix is invertible

Null space of B is of dimension 1
Sanity checks

```matlab
>> initializeMatrices
>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>B</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>C</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>D</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
</tbody>
</table>

Grand total is 64 elements using 512 bytes

>> nullB = null(B,'r')

nullB =

-1
3
-3
1

>> B*nullB

ans =

0
0
0
0
Sanity checks

>> initializeMatrices
>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>B</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>C</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>D</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
</tbody>
</table>

Grand total is 64 elements using 512 bytes

>> nullC = null(C,'r')

nullC =

```
0   -1
-1    0
 1    0
 0    1
```

>> C*nullC

ans =

```
 0   0
 0   0
0   0
0   0
```
Sanity checks

>> initializeMatrices

>> whos

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>B</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>C</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
<tr>
<td>D</td>
<td>4x4</td>
<td>128</td>
<td>double array</td>
</tr>
</tbody>
</table>

Grand total is 64 elements using 512 bytes

nullID = null(D,'r')

nullID =

-1 -1 -1
1 0 0
0 1 0
0 0 1

>> D*nullID

ans =

0 0 0
0 0 0
0 0 0
0 0 0
Procedure to solve a linear system

1. $\text{rank}(A) = \text{rank}([A, y])$
   - Yes: Go to next step.
   - No: Go to step 2.

2. Number of unknowns is equal to $\text{rank}(A)$
   - Yes: Infinite number of solutions. Use rref and null to compute solutions.
   - No: Go to step 3.

3. Single solution
   - Use $A \backslash y$ to compute solution.
   - No solution

There is no solution
Procedure to solve a linear system

1. Calculate \( \text{rank}(A) = \text{rank}([A,y]) \)
   - if yes, proceed to the next step
   - if no, there is no solution

2. Check if the number of unknowns is equal to the rank of \( A \)
   - if yes, there is a single solution
     - Use \( A\backslash y \) to compute the solution
   - if no, there is an infinite number of solutions
     - Use rref and null to compute solutions

Infinite number of solutions
Use rref and null to compute solutions

Single solution
Use \( A\backslash y \) to compute solution

There is no solution
Step: check if \( \text{rank}(A) = \text{rank}([A,y]) \rightarrow \text{no} \)

If \( \text{rank}(A) \) is not equal to \( \text{rank}([A,y]) \), \( y \) is not in the range of \( A \)

i.e. There is no \( x \) such that \( Ax = y \)

System has no solution
Procedure to solve a linear system

\[ \text{rank}(A) = \text{rank}([A,y]) \]

- **If yes:**
  - **Yes:** Single solution
    - Use \( A \backslash y \) to compute solution
  - **No:** Infinite number of solutions
    - Use rref and null to compute solutions

- **If no:**
  - **Yes:** number of unknowns is equal to \( \text{rank}(A) \)
  - **No:** There is no solution
Procedure to solve a linear system

1. Check if \( \text{rank}(A) = \text{rank}([A,y]) \)
   - No: Continue with the next step.
   - Yes: Proceed to the next check.

2. Check if the number of unknowns is equal to \( \text{rank}(A) \)
   - No: There is no solution.
   - Yes: Proceed to the next check.

3. Use rref and null to compute solutions if there are infinite number of solutions.
   - Use \( A\backslash y \) to compute solution if there is a single solution.
   - There is no solution if the number of unknowns is not equal to \( \text{rank}(A) \).

Infinite number of sols
Use rref and null to compute solutions

Single solution
Use \( A\backslash y \) to compute solution

There is no solution
Step: rank(A) = number of unknowns? → yes

If the rank of A is the same as the number of unknowns, the system can be inverted, and the system has a unique solution, which can be computed by A\y
Procedure to solve a linear system

1. Check if \( \text{rank}(A) = \text{rank}([A,y]) \).
2. If yes, check if the number of unknowns is equal to \( \text{rank}(A) \).
   - If yes, there is a single solution. Use \( A\backslash y \) to compute the solution.
   - If no, there is no solution.
3. If no, there is an infinite number of solutions. Use rref and null to compute solutions.

Infinite number of sols
Use rref and null to compute solutions

Single solution
Use \( A\backslash y \) to compute solution

There is no solution
Procedure to solve a linear system

\[ \text{rank}(A) = \text{rank}([A,y]) \]

- Yes
  - Number of unknowns is equal to rank(A)
    - Yes
      - Infinite number of sols
        - Use rref and null to compute solutions
    - No
      - Single solution
        - Use A\y to compute solution
  - No
    - There is no solution
Step: \( \text{rank}(A) = \text{number of unknowns?} \rightarrow \text{no} \)

In this case \( \text{rank}(A) < \text{number of unknowns} \) (i.e. it is smaller). An infinite number of solutions exist.

Imagine you can find a particular solution to this problem

\[
y = Ax_{\text{particular}}
\]

Then if you take any vector in the null space, it is also a solution to this problem:

\[
y = A(x_{\text{particular}} + v_{\text{null space}})
\]

Because

\[
Au_{\text{null space}} = 0
\]
Step: \( \text{rank}(A) = \text{number of unknowns} \? \rightarrow \text{no} \)

The rank of \( A \) is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

\[
\begin{align*}
  y &= A(x_{\text{particular}} + v_1) \\
  y &= A(x_{\text{particular}} + 2v_1) \\
  y &= A(x_{\text{particular}} + 3v_1) \\
  y &= A(x_{\text{particular}} + \lambda_1 v_1)
\end{align*}
\]
Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

\[
\begin{align*}
  y &= A(x_{\text{particular}} + v_1 + v_2) \\
  y &= A(x_{\text{particular}} + 2v_1 + 2v_2) \\
  y &= A(x_{\text{particular}} + 3v_1 + 3v_2) \\
  y &= A(x_{\text{particular}} + \lambda_1 v_1 + \lambda_2 v_2)
\end{align*}
\]
Step: rank(A) = number of unknowns? → no

The rank of A is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

\[
\begin{align*}
y &= A(x_{\text{particular}} + v_1 + v_2 + v_3) \\
y &= A(x_{\text{particular}} + 2v_1 + 2v_2 + 2v_3) \\
y &= A(x_{\text{particular}} + 3v_1 + 3v_2 + 3v_3) \\
y &= A(x_{\text{particular}} + \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3)
\end{align*}
\]
Step: $\text{rank}(A) = \text{number of unknowns?} \rightarrow \text{no}$

The rank of $A$ is not equal to the number of unknowns (i.e. it is smaller).

An infinite number of solutions exist. Now you might have to construct this infinite amount of solutions:

$$y = A(x_{\text{particular}} + v_1 + v_2 + v_3 + \cdots + v_k)$$
$$y = A(x_{\text{particular}} + 2v_1 + 2v_2 + 2v_3 + \cdots + 2v_k)$$
$$y = A(x_{\text{particular}} + 3v_1 + 3v_2 + 3v_3 + \cdots + 3v_k)$$
$$y = A(x_{\text{particular}} + \lambda_1 v_1 + \lambda_2 v_2 + \lambda_3 v_3 + \cdots + \lambda_k v_k)$$

How far can you go?

You can go as far as the null space permits, i.e. pick $v$’s from columns of null spaces. Their linear combinations span the null space.
Step: rank(A) = number of unknowns? → no

How to do this with matlab?

Use the command rref

```matlab
>> A
A =
  1  2  3
  0  3  1
  1 14  7

>> rank(A)
ans =
   2

>> y
y =
   1
   3
  13

>> rank([A,y])
ans =
   2
```

y is in the range of A. There is an infinity of solutions

How to compute them

For this, you need to use

B = rref([A,y])
Step: rank(A) = number of unknowns? → no

\[
>> \text{B=rref([A,y])}
\]

\[
\begin{bmatrix}
1.0000 & 0 & 2.3333 \\
0 & 1.0000 & 0.3333 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

Gauss form of A

\[
\begin{bmatrix}
-1.0000 & 0 \\
0 & 1.0000 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

Particular solution

Find a solution of this system (Gauss elimination has already been done for you)
Step: \( \text{rank}(A) = \text{num. of unk.?} \rightarrow \text{no} \)

Find a solution of this system (Gauss elimination has already been done for you)

Obvious solution: \([-1;1;0]\)
Step: rank(A) = number of unknowns? → no

How do I find all solutions?

Add all the vectors from the null space

vNull =

-0.9113
-0.1302
0.3906

>> vParticular = [-1;1;0]

vParticular =

-1
1
0
Step: $\text{rank}(A) = \text{number of unknowns?} \rightarrow \text{no}$

How do I find all solutions?

Add all the vectors from the null space

\[
\begin{align*}
\text{vNull} &= \\
&= [-0.9113, -0.1302, 0.3906] \\
\text{vParticular} &= [-1, 1, 0] \\
\text{ans} &= \\
&= [1.0000, 3.0000, 13.0000]
\end{align*}
\]

\[
\text{A*(vParticular+2*vNull)} \\
\text{ans} &= \\
&= [1.0000, 3.0000, 13.0000]
\]
Gauss Elimination (non-singular A)

Want to solve $Ax = b$

- **Forward elimination**
  - Starting with the first row, add or subtract multiples of that row to eliminate the first coefficient from the second row and beyond.
  - Continue this process with the second row to remove the second coefficient from the third row and beyond.
  - Stop when an upper triangular matrix remains.

- **Back substitution**
  - Starting with the last row, solve for the unknown, then substitute that value into the next highest row.
  - Because of the upper-triangular nature of the matrix, each row will contain only one more unknown.

\[
\begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \vdots & b_1 \\
  a_{21} & a_{22} & a_{23} & \vdots & b_2 \\
  a_{31} & a_{32} & a_{33} & \vdots & b_3 \\
\end{bmatrix}
\]

\[
\begin{align*}
x_3 &= b''_3/a''_{33} \\
x_2 &= (b'_2 - a'_{23}x_3)/a'_{22} \\
x_1 &= (b_1 - a_{13}x_3 - a_{12}x_2)/a_{11}
\end{align*}
\]
Order of Elimination

1 ? ? ? ? ?
2 4 ? ? ? ?
3 5 6 ? ?

=
Gaussian Elimination in 3D

\[2x + 4y - 2z = 2\]
\[4x + 9y - 3z = 8\]
\[-2x - 3y + 7z = 10\]

• Using the first equation to eliminate \(x\) from the next two equations
Gaussian Elimination in 3D

\[\begin{align*}
2x & \quad + \quad 4y \quad - \quad 2z \quad = \quad 2 \\
\quad y \quad + \quad z \quad = \quad 4 \\
\quad y \quad + \quad 5z \quad = \quad 12
\end{align*}\]

• Using the second equation to eliminate \( y \) from the third equation
Gaussian Elimination in 3D

\[ 2x + 4y - 2z = 2 \]
\[ y + z = 4 \]
\[ 4z = 8 \]

• Using the second equation to eliminate \( y \) from the third equation
Solving Triangular Systems

• We now have a triangular system which is easily solved using a technique called *Backward-Substitution*.

\[
\begin{align*}
2x + 4y - 2z &= 2 \\
y + z &= 4 \\
4z &= 8
\end{align*}
\]
Solving Triangular Systems

• If $A$ is upper triangular, we can solve $Ax = b$ by:

$$x_n = b_n / A_{nn}$$

$$x_i = \left( b_i - \sum_{j=i+1}^{n} A_{ij} x_j \right) / A_{ii}, \quad i = n-1, \ldots, 1$$
Backward Substitution

• From the previous work, we have

\[ 2x + 4y - 2z = 2 \]
\[ y + z = 4 \]
\[ z = 2 \]

• And substitute \( z \) in the first two equations
Backward Substitution

\[ 2x + 4y - 4 = 2 \]

\[ y + 2 = 4 \]

\[ z = 2 \]

• We can solve \( y \)
Backward Substitution

\[ 2x + 4y - 4 = 2 \]
\[ y = 2 \]
\[ z = 2 \]

• Substitute to the first equation
Gauss-Jordan Elimination

Keep going until augmented matrix is reduced row echelon form (rref):

1. The rows (if any) consisting entirely of zeros are grouped together at the bottom of the matrix.
2. In each row that does not consist entirely of zeros, the leftmost nonzero element is a 1 (called a leading 1 or a pivot).
3. Each column that contains a leading 1 has zeros in all other entries.
4. The leading 1 in any row is to the left of any leading 1’s in the rows below it.

Stop process in step 2 if you obtain a row whose elements are all zeros except the last one on the right. In that case, the system is inconsistent and has no solutions. Otherwise, finish step 2 and read the solutions of the system from the final matrix.
Overdetermined systems

consider \( y = Ax \) where \( A \in \mathbb{R}^{m \times n} \) is (strictly) skinny, i.e., \( m > n \)

- called overdetermined set of linear equations (more equations than unknowns)
- for most \( y \), cannot solve for \( x \)

one approach to approximately solve \( y = Ax \):

- define residual or error \( r = Ax - y \)
- find \( x = x_{ls} \) that minimizes \( \|r\| \)

\( x_{ls} \) called least-squares (approximate) solution of \( y = Ax \)
Overdetermined systems

\(Ax_{ls}\) is point in \(\mathcal{R}(A)\) closest to \(y\) (\(Ax_{ls}\) is projection of \(y\) onto \(\mathcal{R}(A)\))
The “Least Squares” Problem

If $A$ is an $n$-by-$m$ array, and $b$ is an $n$-by-1 vector, let $c^*$ be the smallest possible (over all choices of $m$-by-1 vectors $x$) mismatch between $Ax$ and $b$ (i.e., pick $x$ to make $Ax$ as much like $b$ as possible).

$$c^* := \min_{x, m \times 1} \| Ax - b \|$$

“is defined as”

“the minimum, over all $m$-by-1 vectors $x$”

“the length (i.e., norm) of the difference/mismatch between $Ax$ and $b$. “
Four cases for Least Squares

Recall least squares formulation

\[ c^* := \min_{x, m \times 1} \| Ax - b \| \]

There are 4 scenarios

\( c^* = 0 \): the equation \( Ax = b \) has at least one solution
  • only one \( x \) vector achieves this minimum
  • many different \( x \) vectors achieves the minimum

\( c^* > 0 \): the equation \( Ax = b \) has no solutions
  • only one \( x \) vector achieves this minimum
  • many different \( x \) vectors achieves the minimum
Four cases: \( x = A \backslash b \) as solution of

\[
c^* := \min_{x, m \times 1} \| Ax - b \|
\]

No Mismatch:

\( c^* = 0 \), and **only one** \( x \) vector achieves this minimum

Choose this \( x \)

\( c^* = 0 \), and **many different** \( x \) vectors achieve the minimum

From all these minimizers, choose smallest \( x \) (i.e., norm)

Mismatch:

\( c^* > 0 \), and **only one** \( x \) vector achieves this minimum

Choose this \( x \)

\( c^* > 0 \), and **many different** \( x \) vectors achieve the minimum

From all minimizers, choose an \( x \) with the smallest norm
The backslash operator

If $A$ is an $n$-by-$m$ array, and $b$ is an $n$-by-1 vector, then

\[
\begin{align*}
\text{>>} \quad x &= A\backslash b \\
\end{align*}
\]
solves the “least squares” problem. Namely

- If there is an $x$ which solves $Ax=b$, then this $x$ is computed
- If there is no $x$ which solves $Ax=b$, then an $x$ which minimizes the mismatch between $Ax$ and $b$ is computed.

In the case where many $x$ satisfy one of the criterion above, then a smallest (in terms of vector norm) such $x$ is computed.

So, mismatch is handled first. Among all equally suitable $x$ vectors that minimize the mismatch, choose a smallest one.