Matrix Solutions

Overview

- **Calculate** matrix **condition number** to get **error bounds** for solutions for ***Ax=b***
- **Manipulate** matrices and **determine** when a **matrix** is not **invertible**
- **Compute coordinates** using a **basis**, in particular an **orthogonal** basis
- **What PLU factorization is, how to compute it numerically and use it to solve linear systems.**
- **What banded and sparse matrices are and how the structure is used for solving large matrices.**
- **Matrix norms and condition numbers and how they are used to understand accuracy and numerical sensitivity.**
- **Flattening** higher dimensional **tensors** to **describe** linear **actions** as matrices.

Solving Ax = b

Gaussian Elimination	• **GE** is used to **solve *Ax = b*** • **transform** matrix into **Row Echelon Form (REF)** using **Elementary Row Operations (ERO)**
Sparse Matrix	• most **elements** are **zero** • **only** have to **store** the **non-zero elements** and their **locations** • **efficient** for **storage** and **computation**
Banded Matrix	• a **sparse matrix** • **non-zero** entries are **confined** to a **diagonal** band comprising the main diagonal and **zero** or more diagonals on either side

Elementary Row Operations (ERO)	1. **swap rows** ($R_i \leftrightarrow R_j$)
	   • **interchange** two **rows**
	 2. **scale row** ($R_i \rightarrow cR_i$)
	   • **multiply** a **row** by a non-zero **constant** (c)
	   • use to **make** a **pivot** 1
	 3. **add** multiple **rows** ($R_i \rightarrow R_i + c$)
	   • **replace** a **row** with **itself** plus a **multiple *(c)*** of another **row**
	   • use to **create zeros below** a **pivot**

Row Echelon Form (REF)	• **get** the first **pivot**
	   • **find** the leftmost **non-zero column**
	   • if the **top element** (pivot) is **zero, swap rows** to make it non-zero
	 • **make pivot 1**

	optional for **REF**, **required** for **RREF** **multiply** the **pivot row** by the **reciprocal** of the pivot **eliminate** below **pivot** - use **row addition** to make all **entries below** the **pivot zero** **repeat** **move** to the next **row/column** (submatrix) and **repeat** steps **1 – 3** until the **matrix** is in **REF** - produces **matrix** with **'staircase pattern'** and **zeros below pivots**
Reduced Echelon Form (RREF)	1. follow **REF** steps 2. also, **use** row **operations** to make all **entries above** the **pivots** zero 3. **result** it an **identity matrix** on the 'A side'
Back Substitution	1. once in **REF/RREF**, **start** from the **last** non-zero **equation** and **solve** for the **variable** 2. **substitute** that **value** into the **equation** above it 3. **continue** until all **variables** are **found**
Interpreting Solution	1. **unique** solution **no inconsistent rows** (like 0 = 5) **no rows** of all **zeros** (except very **last** row) 2. **infinite** solution - a **row** of all **zeros** (0 = 0) **indicates** a **free variable** (ex: x + y = 5) 3. **no solution** - an **inconsistent row** (ex: 0 = r where r ≠ 0)
Example	**Solve** the **system** of **linear equations** using **matrices** $\begin{aligned} x - y + z &= 8 \\ 2x + 3y - z &= -2 \\ 3x - 2y - 9z &= 9 \end{aligned}$ **write** the **augmented matrix** $\left[\begin{array}{ccc\|c} 1 & -1 & 1 & 8 \\ 2 & 3 & -1 & -2 \\ 3 & -2 & -9 & 9 \end{array}\right]$ **perform** row **operations** to obtain **REF**

$$-2R_1 + R_2 = R_2 \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 5 & -3 & -18 \\ 3 & -2 & -9 & 9 \end{array}\right]$$

$$-3R_1 + R_3 = R_3 \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 5 & -3 & -18 \\ 0 & 1 & -12 & -15 \end{array}\right]$$

- **interchange** two **rows**

$$\text{Interchange } R_2 \text{ and } R_3 \rightarrow \begin{bmatrix} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 5 & -3 & -18 \end{bmatrix}$$

- **perform** row **operations**

$$-5R_2 + R_3 = R_3 \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 0 & 57 & 57 \end{array}\right]$$

$$-\tfrac{1}{57}R_3 = R_3 \rightarrow \left[\begin{array}{ccc|c} 1 & -1 & 1 & 8 \\ 0 & 1 & -12 & -15 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

- the last **matrix represents** the **equivalent system**

$$\begin{aligned} x - y + z &= 8 \\ y - 12z &= -15 \\ z &= 1 \end{aligned}$$

- **solution** is **obtained** using **back-substitution**

$$(4, -3, 1).$$

LU Factorization

- any **non-singular matrix *A*** can be **factored** into a **lower** triangular matrix ***(L)*** and an **upper triangular** matrix ***(U)***

Upper Triangular Matrix

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{E_3 + 4E_2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix} \xrightarrow{E_3 + 3E_1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix}$$

$$\xrightarrow{E_2 + 2E_1} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} = \mathbf{L}$$

Lower Triangular Matrix	$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{E_3+4E_2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix} \xrightarrow{E_3+3E_1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} \xrightarrow{E_2+2E_1} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 4 & 1 \end{bmatrix} = \mathbf{L}$$

Validate Solution Using Python	`L = np.array([[1,0,0],[2,1,0],[3,4,1]])`
	`U = np.array([[1,1,0],[0,-1,-1],[0,0,3]])`
	`L @ U`
	
	`array([[ 1,  1,  0],`
	`       [ 2,  1, -1],`
	`       [ 3, -1, -1]])`

LU Transformation Using Python	See code below
Sources	https://www.cs.cornell.edu/~tomf/notes/cs421-cheat-sheet.pdf https://www.scribd.com/document/858540190/Gaussian-Elimination-and-Gauss.

```python
def lu(A):

    #Get the number of rows
    n = A.shape[0]

    U = A.copy()
    L = np.eye(n, dtype=np.double)

    #Loop over rows
    for i in range(n):

        #Eliminate entries below i with row operations
        #on U and reverse the row operations to
        #manipulate L
        factor = U[i+1:, i] / U[i, i]
        L[i+1:, i] = factor
        U[i+1:] -= factor[:, np.newaxis] * U[i]

    return L, U
```

	https://courses.lumenlearning.com/waymakercollegealgebra/chapter/solve-a-system-with-gaussian-elimination/ https://johnfoster.pge.utexas.edu/numerical-methods-book/LinearAlgebra_LU.html

Spline	
uses **systems** of **linear equations** to find **piecewise polynomial functions** (splines) that smoothly **connect data** points	

Key Concepts	• **piecewise polynomials**
	  ○ A **spline** is made of **multiple polynomial pieces** joined at **specific points** called **knots** or **breakpoints**
	• **linear splines**
	  ○ **connecting** points with **straight lines**, represented by $\boldsymbol{y = a + bx}$ for each **segment**, solvable with **basic linear equations**
	• **higher-order splines**
	  ○ use **higher degree polynomials** for smoother **transitions**
	• **knots/breakpoints**
	  ○ points where the **polynomial pieces connect**, often **data points** themselves
	• **continuity conditions**
	  ○ to **ensure** smoothness, **derivatives** must **match** at the **knots**, creating the **system** of **linear equations**

Solve Using Linear Algebra	• **define polynomials**
	  • ***N*** data points produce ***N* – *1*** segments
	  • each **segment** has its own **polynomial**
	• **interpolation constraints**
	  • set up **equations** ensuring $P_k(x_k) = y_k$ and $P_k(x_{k+1}) = y_{k+1}$
	  • the **spline** passes **through** the **data** points
	• **smoothness constraints**
	  • add **equations** for **derivative continuity** at **interior knots**
	  • **introduces** more **unknowns** and **equations**
	• **boundary conditions**
	  • add **conditions** for the **ends**

	- natural **cubic splines** have **zero second derivatives** at the **ends** **solve** the **system** - the **result** is a **large system** of **linear equations** - often **represented** as ***Ax = b*** **solve** for **unknown coefficients**
Sources	https://en.wikipedia.org/wiki/Spline https://people.computing.clemson.edu/
Condition Number	
the **condition number *(CN)*** of a matrix **measures** how **sensitive** the solution of a **linear system** or the **inverse** of the matrix is to **changes** in the **input data**	
Low Condition Number	- indicates a ***well-conditioned* matrix** - means that **small input errors** lead to **small output errors**
High Condition Number	- indicates an ***ill-conditioned*** matrix that is **nearly singular** **small input errors** can cause **massive errors** in the solution - potentially leads to inaccurate results
Matrix Norm	- a **measure** of **how large** a matrix's **elements** are - the **magnitude** of the **matrix** - is a **real number** between ***1 and ∞*** - denoted: **|| *A* ||**
Key Points	- for any **matrix *A***, **cond(A) ≥ 1** - for the **identity matrix *I, cond(I) = 1*** - for any **matrix *A*** and a **non-zero** scalar ***γ***, ***cond(γA) = cond(A)*** - for any **diagonal matrix *D***, ***cond(D) = (max(|d_i|) / min(|d_i|))*** - The **condition number** is a measure of **how close** a matrix is to being **singular**: a matrix with **large** condition **number** is nearly **singular**, whereas a **matrix** with a condition **number close to *1*** is **far** from being **singular** - The **determinant** of a **matrix** is **NOT** a **good** indicator to **check** whether a **matrix** is near **singularity**
General Method	**calculate** the **product** of the matrix **norm** of ***A*** and the matrix **norm** of ***A^{-1}*** ***cond(A)* = ||*A*|| • ||*A^{-1}*||** using the **same** matrix **norm**

2-Norm Method (Singular Values)	• **compute Singular Value Decomposition** ○ **perform the**
Sources	https://courses.grainger.illinois.edu/cs357/fa2023/notes/ref-10-condition.html

Live Session Notes	13 Jan 2026
• **Matrix Solutions** • •	
*	• • •
*	• • •
*	• • •
*	• • •