Ch.2 Systems of Equations

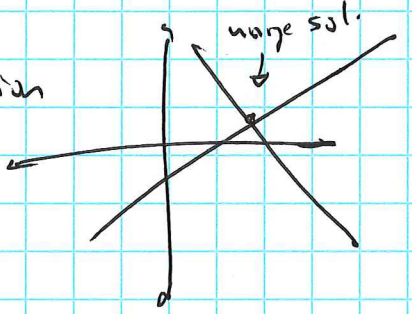
2.1 Gaussian Elimination

Some essential aspects of Linear Algebra ...

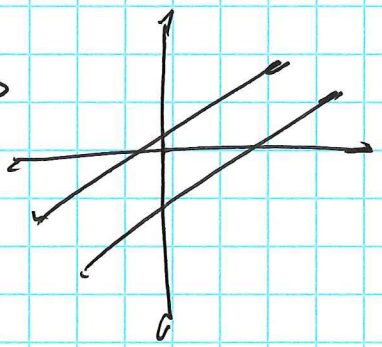
(3) scenarios for solving systems of linear equations:

2×2 system $\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$

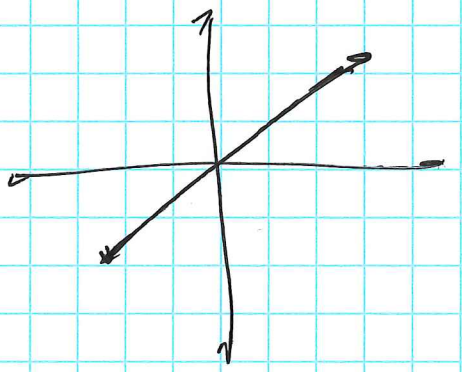
(1) unique solution



(2) No solutions



(3) ∞ solutions



The Matrix equation: $\boxed{A\vec{x} = \vec{b}}$

$$\left.\begin{array}{l} x + y = 3 \\ 3x - 4y = 2 \end{array}\right\} \longrightarrow \begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

"A" "$\vec{x}$" "$\vec{b}$"

"A" = coefficient Matrix

In "augmented form" we write:

$$\left[\begin{array}{cc|c} 1 & 1 & 3 \\ 3 & -4 & 2 \end{array}\right]$$

Recall that the system: $A\vec{x} = \vec{b}$ has a unique solution if & only if the coefficient matrix A is invertible.

In this case, the solution is given by: $\boxed{\vec{x} = A^{-1}\vec{b}}$

By the Fundamental Theorem of Invertible Matrices, we note that A is invertible iff. $\det(A) \neq 0$.

If A is invertible it is called $\boxed{\text{non-singular}}$.

When A is non-singular, we may find its inverse, A^{-1} in the 2x2 with the classic formula!

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \boxed{A^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}}$$

For larger square matrices (nxn) we may use a "super" augmented matrix:

$$\left[\, A \,|\, I_n \,\right] \overset{\text{Row-Reduction}}{\rightsquigarrow} \left[\, I \,|\, A^{-1} \,\right]$$

which yields A^{-1} for any non-singular, square matrix. ↑ inverse!

"Naive" Gaussian Elimination

Recall from elementary Linear Algebra: There are (3) so-called elementary row operations, performed on matrices.

3 elementary row ops

(1) Row swap $r_i \leftrightarrow r_j$

(2) Non-zero scalar multiplication $r_i^* \to kr_i$ $(k \neq 0)$

(3) Add or subtract a multiple of one row from another (Replacement) $r_i^* \to r_i + kr_j$

Any matrix A' that results from a (finite) sequence of elementary row operations applied to a matrix A is said to be row equivalent with A; we just write: $A' \sim A$.

The main point of row operations is that when we apply them in succession to an augmented matrix/system, these operations preserve the solution set of the system.

Ex. $\begin{matrix} x+y=2 \\ 2x+y=3 \end{matrix} \to \left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 1 & 3 \end{array}\right]$ (make pivot) $\sim r_2^* \to r_2 - 2r_1$

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & -1 & -1 \end{array}\right]$$

Notice: coefficient matrix has a pivot in each row/col., so we stop.

Gaussian Elimination (2 Phases)

(1) Apply sequence of elem. row ops. until coefficient matrix is in upper-triangular

(2) Use back-substitution to solve system.

Note:
A matrix is said to be in **upper-Triangular form** if: $a_{ij} = 0$ for all $0 < j < i \leq n$; in other words, all entries below the main diagonal are zero!

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$; In general → $$\begin{bmatrix} * & * & * \\ & * & * \\ 0 & & * \end{bmatrix}$$

↑ upper Trigular

Note That **lower-Triangular Matrices** are defined analogously

$$\begin{bmatrix} * & & 0 \\ * & * & \\ * & * & * \end{bmatrix}$$ → lower-Triangular $$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

Returning to the previous example...

Phase II: Back-su

$$\left[\begin{array}{cc|c} 1 & 1 & 2 \\ 2 & 1 & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & 0 & 2 \\ 0 & -1 & -1 \end{array}\right]$$

↳ Stop row reduction here since coeff. matrix is upper-Triangular!

Back-sub: automating

To "reading" reduced Matrix (backwards), i.e. bottom to to

$-y = -1 \rightarrow \boxed{y = 1}$ (row 2)

"reading" row 1: $x + y = 2 \rightarrow \boxed{x = 1}$

check: $x+y = 2$ ✓

$2x+y = 3$ ✓

Another example of Gaussian elimination...

Ex.

$$\left.\begin{aligned} x+2y-z&=3\\ 2x+y-2z&=3\\ -3x+y+z&=-6 \end{aligned}\right\} \rightsquigarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3\\ 2 & 1 & -2 & 3\\ -3 & 1 & 1 & -6 \end{array}\right] \begin{array}{l} r_2^* \to r_2 - 2r_1\\ r_3^* \to r_3 - (-3r_1) \end{array}$$

pivot!

$$\sim \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3\\ 0 & -3 & 0 & -3\\ 0 & 7 & -2 & 3 \end{array}\right] \quad r_3^* \to r_3 - \left(\tfrac{-7}{3} r_2\right) \quad \sim \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3\\ 0 & -3 & 0 & -3\\ 0 & 0 & -2 & -4 \end{array}\right]$$

move on to next row.

pivot in each row/col

coeff. Matrix is in triangular-form

Phase II: Back-sub

Row 3: $-2z = -4 \to \boxed{z=2}$

Row 2: $-3y = -3 \to \boxed{y=1}$

Row 1: $x+2y-z=3 \to x+2(1)-2=3 \to \boxed{x=3}$. □

Operation Counts

Recall that the general form/Tableau for an $n \times n$ system

$$\left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1\\ \vdots & \ddots & \vdots & b_2\\ a_{n1} & & a_{nn} & b_n \end{array}\right]$$

Let's consider the # of "flops" (basic floating point operations) required for Gaussian elimination

$$\left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ (a_{21}) & \cdots & a_{2n} & b_2 \\ \vdots & & \vdots & \vdots \end{array}\right]$$

How many computations to eliminate a_{21}?

$$a_{21}^{+} \rightarrow a_{21} - \left(\frac{a_{21}}{a_{11}}\right) \cdot a_{11}$$

one addition, one multiplication, one division

store this #

$\therefore$ Total # comps (for entire row) to eliminate "a_{21}": n additions + n mult + 1 division

$$= \boxed{2n+1}$$

Thus To eliminate each element of column 1 (except the pivot) requires: $\boxed{(n-1)(2n+1)}$ Total computations.

In general, then, the elimination of each a_{ij} requires the following # of operations:

$$\left[\begin{array}{ccccc} 0 & & & & \\ 2n+1 & 0 & & & \\ 2n+1 & 2(n-1)+1 & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ 2n+1 & 2(n-1)+1 & 2(n-2)+1 & \cdots & 2(2)+1 \quad 0 \end{array}\right]$$

Total # comps for Gauss elimination is thus:

$$\sum_{j=1}^{n-1} \sum_{i=1}^{j} \left(2\,(j+1)\right)+1 = \sum_{j=1}^{n-1} 2j(j+1)+j$$

(since: $\sum_{i=1}^{j} = j$)

over each column; over # entries in column

$$= 2\sum_{j=1}^{n-1} j^2 + 3\sum_{j=1}^{n-1} j = 2\,\frac{(n-1)\cdot n\,(2n-1)}{6} + 3\,\frac{(n-1)n}{2}$$

(Recall: $\sum_{i=1}^{n} i^2 = \frac{(n+1)n(2n+1)}{6}$, $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$)

$$= \frac{2}{3}n^2 + \frac{1}{2}n^2 - \frac{7}{6}n \sim \boxed{\mathcal{O}(n^3)}$$

Question II: How many ops. in, general, for backward substitution?

Because of the triangular shape of the reduced coefficient matrix, we are able to "use" x_i values once they have been learned, with each successive step.

e.g. $\begin{bmatrix} a_{11} & & a_{1n} & b_1 \\ 0 & \ddots & & \vdots \\ \vdots & & & \vdots \\ 0 & & a_{nn} & b_n \end{bmatrix}$ →

$x_n = \frac{b_n}{a_{nn}}$: 1 op

$x_{n-1} = \frac{b_{n-1} - x_n \cdot a_{n-1,n}}{a_{n-1,n-1}}$ (known) 1 mult, 1 add, 1 div → 3 ops

$x_{n-2} = \cdots =$ 5 ops ..., etc.

In this way the total # of operations required for back-substitution is: $1+3+5+7+\cdots+(2n-1)$

$$= \sum_{i=1}^{n} 2i-1 = 2\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 = \frac{2n(n+1)}{2} - n = \boxed{n^2} \sim \boxed{\mathcal{O}(n^2)}.$$

In Total, we may say Gaussian elimination requires:

$$\frac{2}{3}n^3 + n^2 \approx \frac{2}{3}n^3 \sim \underline{\mathcal{O}\left(\frac{2}{3}n^3\right)} \quad \text{operations.}$$

Ex. Estimate the time required to carry out back substitution on a system of 500 equations in 500 unknowns on a computer where elimination takes 1 second.

Estimated Time for back sub: $\dfrac{n^2 \to \#\text{ ops back sub}}{\frac{2}{3}n^3 \to \#\text{ ops per elimination (1 sec)}} = \dfrac{500^2}{\frac{2}{3}(500)^3} \approx \boxed{.003 \text{ sec}}$

So Total Time for 500×500 system $\approx 1 + .003 = \boxed{1.003 \text{ sec}}$

Ex. On a particular computer, back sub of a 5000×5000 Triangular matrix takes 0.1 seconds. Estimate the time needed to solve a general system of 3000×3000 eqtns.

(comp. carries out $(5000)^2$ in .1 sec, or $\boxed{2.5 * 10^8 \text{ ops/sec.}}$ $\quad (5000)^2 * 10 \text{ ops/sec}$)

Solving a general system w/ Gaussian elim requires: $\frac{2(3000)^3}{3}$ ops $\to$ $\dfrac{2(3000)^3}{(5000)^2 \cdot 10} \approx \boxed{\geq 2 \text{ sec.}}$

2.3 The LU Factorization

In short, the so-called **LU factorization** is a **matrix representation of Gaussian elimination.**

Where "L" is a **lower-Triangular** ($n \times n$) matrix ($\begin{bmatrix} * & & 0 \\ * & * & \\ * & * & * \end{bmatrix}$)

where $l_{ij} = 0$ for $i < j$.

"U" is an **upper-Triangular** matrix ($\begin{bmatrix} * & * & * \\ & * & * \\ 0 & & * \end{bmatrix}$)

where $u_{ij} = 0$ for $i > j$.

Ex. We find the LU factorization for:

$$A = \begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix} \sim r_2^* \to r_2 - 3r_1 \underset{\text{Note}}{\sim} \begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix} = U$$

For the matrix "L" we defined it as lower-Triangular (with all diagonal entries $= 1$); Note that because **3** was the multiplier used to eliminate entry $(2,1)$ of A, we accordingly store **3** $\to$ $(2,1)$ of L.

So, $L = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}$ ↑ stored.

check:

$$LU = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix} = A. \checkmark$$

Ex. Find the LU factorization of:

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

We first perform the steps of Gaussian elimination:

$$\begin{bmatrix} 1 & 2 & -1 \\ \textcircled{2} & 1 & -2 \\ -3 & 1 & 1 \end{bmatrix} \sim r_2^* \to r_2 - \tfrac{2}{=} r_1 \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ -3 & 1 & 1 \end{bmatrix}$$

$$\sim r_3^* \to r_3 - \underset{=}{(-3)} r_1 \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 7 & -2 \end{bmatrix}$$

$$\sim r_3^* \to r_3 - \underset{=}{\left(-\tfrac{7}{3}\right)} r_2 \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix} = U.$$

From this sequence of Gaussian elimination we have:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \textcircled{2} & 1 & 0 \\ \textcircled{-3} & \textcircled{-\tfrac{7}{3}} & 1 \end{bmatrix}$$

(2,1) (3,1) (3,2)

Notice: $LU =$

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -\frac{7}{3} & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}}_{U} =$$

$$\begin{bmatrix} 1 & 2 & -1 \\ 2 & 1 & -1 \\ -3 & 1 & 1 \end{bmatrix} = A. \checkmark$$

Q: Why does LU factorization work?

A: Due to (3) facts related to Triangular matrices.

Fact (1): Let $L_{ij}(-c)$ represent the lower-triangular matrix with $(-c)$ in the (i,j) coordinate. Then $L_{ij}(-c)A$ represents the row operation: $r_i^* \to r_i - cr_j$

For example:

$$\begin{bmatrix} 1 & 0 & 0 \\ -c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21}-ca_{11} & a_{22}-ca_{12} & a_{23}-ca_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$L_{21}(-c)$ · A

Fact (2): $\boxed{L_{ij}(-c)^{-1} = L_{ij}(c)}$ check: $L_{21}(-c) = \begin{bmatrix} 1 & 0 & 0 \\ -c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3$ ✓

$L_{21}(c)$

Thus: $L_{21}(3)A = \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix} = U$ (A)

Solve for A: $A = L_{21}(3)^{-1}U = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix}$

L U

Fact (3): Products of lower Triangular matrices are lower triangular.

Ex.

$$\begin{bmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ c_2 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & c_3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ c_1 & 1 & 0 \\ c_2 & c_3 & 1 \end{bmatrix} = U.$$ ✓

Back substitution with LU factorization.

- Once L & U are known, the problem $A\vec{x} = \vec{b}$ can be written as: $LU\vec{x} = \vec{b}$; we then solve "back-sub" w/ two additional steps:

  (1) Solve $L\vec{c} = \vec{b}$ for $\vec{c}$.

  (2) Solve $U\vec{x} = \vec{c}$ for $\vec{x}$.

Note: Since L, U are triangular matrices, these steps are straightforward.

Ex. Solve the system using the LU factorization.

$$\begin{bmatrix} 1 & 1 \\ 3 & -4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$A \cdot \vec{x} \quad \vec{b}$

Note: $A = LU = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix}$

(I) Solve: $\begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \rightarrow \boxed{c_1 = 3,\ c_2 = -7}$

(II) Solve: $\begin{bmatrix} 1 & 1 \\ 0 & -7 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -7 \end{bmatrix} \rightarrow \boxed{x_1 = 1,\ x_2 = 2}$

Ex. Solve with LU: $\begin{bmatrix} 1 & 2 & -1 \\ 2 & 1 & -2 \\ -3 & 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ -6 \end{bmatrix}$

$A \cdot \vec{x} = \vec{b}$

$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -\frac{7}{3} & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}$

(I) Solve: $L\vec{c} = \vec{b} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -\frac{7}{3} & 1 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ -6 \end{bmatrix} \rightarrow \boxed{c_1 = 3,\ c_2 = -3,\ c_3 = -4}$

(II) Solve: $U\vec{x} = \vec{c} \rightarrow \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 0 \\ 0 & 0 & -2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ -4 \end{bmatrix} \rightarrow \boxed{x_1 = 3,\ x_2 = 1,\ x_3 = 2}$

Complexity of the LU factorization

Q: why use LU factorization? A: Efficiency (if A has LU factorization).

Note that with "Classical" Gaussian elimination, the reduction requires $\mathcal{O}\left(\frac{2}{3}n^3\right)$ total operations.

Consider now the case of a set of k problems/systems, all with the same coefficient matrix: A.

$$\left\{A\vec{x}=\vec{b}_1,\ A\vec{x}=\vec{b}_2,\ \ldots,\ A\vec{x}=\vec{b}_k\right\}.$$

With Classical Gaussian elimination, solving this entire set of systems of equations requires $\boxed{\mathcal{O}\left(\frac{2}{3}kn^3\right)}$ operations.

However, solving the same set of systems is drastically more efficient with LU factorization. Observe that with LU factorization, the $\vec{b}$ vector doesn't enter into the algorithm until after the factorization: $A=LU$ is found. Consequently, for each new $\vec{b}$ vector, we simply perform the two-step back-sub procedure using triangular matrices; this backsub requires only $\mathcal{O}(2n^2)$ ops.

In total then, the LU factorization method requires:

$\boxed{\mathcal{O}\left(\frac{2}{3}n^3+2kn^2\right)}$ ops. to solve the k systems!!

(vs. $\mathcal{O}\left(\frac{2}{3}kn^3\right)$ for Gaussian elimination)

Notice That even when $k=1$ (i.e. Re case of one system), no extra computations are required by LU.

BTW, such systems/sets of systems are commonly encountered, among oRer fields, in structul egineering & Re study of "dynamical systems."

Ex. Assme it Takes one second To factor Re 300×300 matrix $A = LU$. How many problems: $A\vec{x}_1 = \vec{b}_1$, $A\vec{x}_2 = \vec{b}_2$, ..., $A\vec{x}_k = \vec{b}_k$ can be solved in Re next second?

The Two back-subs require a Total of $2n^2$ ops.

So, $\dfrac{\frac{2n^3}{3}}{2n^2} \approx 1\text{sec}$ $= \dfrac{n}{3} = \dfrac{(n=300)}{100}$ systems in one second.

Note That LU works for many matrices (but not all - see 2.

NOT all matrices admit of an LU factorizati!

Ex. $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

If $A = LU$, Then $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}}_{\text{"L"}} \underbrace{\begin{bmatrix} b & c \\ 0 & d \end{bmatrix}}_{\text{"U"}} = \begin{bmatrix} b & c \\ ab & ac+d \end{bmatrix}$

implying: $b=0$, but $ab=1$, a contradiction. Thus, A has no LU factorization.