

UNIT 4

Solution of Linear Algebraic Equations

(10 Hours) — Numerical Methods (CAC252)

4.1 Review: Matrices and Existence of Solutions

A system of n linear equations in n unknowns is written in matrix form $\mathbf{A} \mathbf{x} = \mathbf{b}$, where $\mathbf{A} = [a_{ij}]$ is the $n \times n$ coefficient matrix, $\mathbf{x} = (x_1, \dots, x_n)^T$ the unknown vector and $\mathbf{b}$ the right-hand side. Solution methods are divided into **direct methods** (a finite number of operations gives the exact answer in exact arithmetic — Gauss, Gauss–Jordan, LU) and **iterative methods** (a sequence of approximations that converges to the answer — Jacobi, Gauss–Seidel).

4.1.1 Properties of Matrices

Term	Meaning
Identity matrix $\mathbf{I}$	Diagonal matrix with 1s on the diagonal; $\mathbf{A}\mathbf{I} = \mathbf{I}\mathbf{A} = \mathbf{A}$
Symmetric	$\mathbf{A}^T = \mathbf{A}$ ($a_{ij} = a_{ji}$)
Upper / lower triangular	All entries below / above the main diagonal are 0
Determinant $\det(\mathbf{A})$	$\det(\mathbf{AB}) = \det(\mathbf{A}) \cdot \det(\mathbf{B})$; $\det(\mathbf{A}^T) = \det(\mathbf{A})$; a row swap changes the sign
Singular / non-singular	$\mathbf{A}$ is non-singular iff $\det(\mathbf{A}) \neq 0$ iff $\mathbf{A}^{-1}$ exists
Inverse	$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$; $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$; $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$
Rank	Number of linearly independent rows (= columns) = order of the largest non-zero minor
Diagonally dominant	$ a_{ii} \geq \sum_{(j \neq i)} a_{ij} $ for every row (strictly, if $>$ for all rows)

If $\det(\mathbf{A}) \neq 0$, the system $\mathbf{Ax} = \mathbf{b}$ has exactly one solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ (Cramer's rule gives $x_i = \det(\mathbf{A}_i) / \det(\mathbf{A})$). Computing $\mathbf{A}^{-1}$ or determinants by cofactors costs far more than elimination, so elimination is used in practice.

4.2 Consistency of a Linear System

A system is **consistent** if it has at least one solution. Form the augmented matrix $[\mathbf{A} \mid \mathbf{b}]$. By the **Rouché–Capelli (rank) theorem**, for a system of m equations in n unknowns:

Condition on ranks	Conclusion
$\text{rank}(A) = \text{rank}([A \mid b]) = n$	Consistent — unique solution
$\text{rank}(A) = \text{rank}([A \mid b]) < n$	Consistent — infinitely many solutions ($n - \text{rank}$ free parameters)
$\text{rank}(A) < \text{rank}([A \mid b])$	Inconsistent — no solution

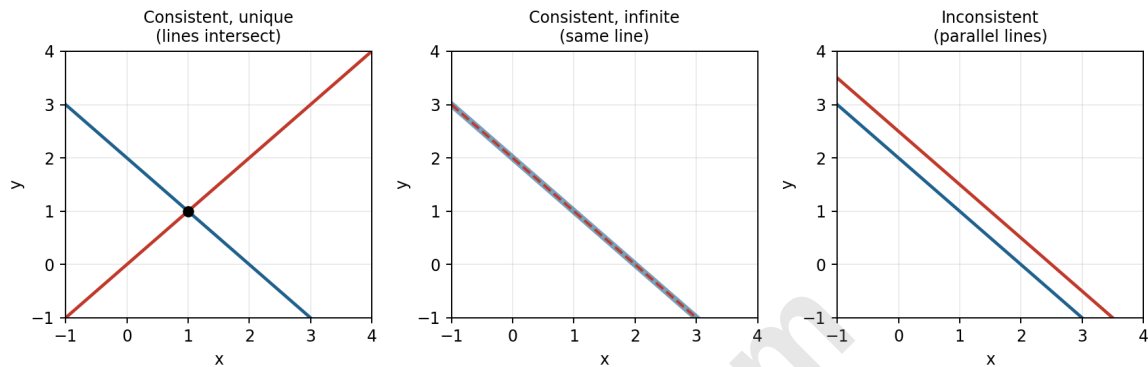


Figure 4.1 — Geometry of a 2×2 system: unique solution, infinitely many solutions, no solution.

Example 4.1

Test the consistency of each system (unknowns x, y, z or x, y):

- (a) $x + y = 2, x - y = 0$ (b) $x + y = 2, 2x + 2y = 4$ (c) $x + y = 2, 2x + 2y = 5$
 (d) $x + y + z = 3, x - y + z = 1, 2x + 2z = 4$ (e) same left sides as (d) but $2x + 2z = 5$

System	$\text{rank}(A)$	$\text{rank}([A \mid b])$	n	Conclusion
(a)	2	2	2	Unique solution
(b)	1	1	2	Infinitely many solutions
(c)	1	2	2	Inconsistent (no solution)
(d)	2	2	3	Infinitely many solutions
(e)	2	3	3	Inconsistent (no solution)

Note: In (d) and (e) the third row of A equals the sum of the first two, so $\text{rank}(A) = 2$. In (d) the right side obeys $3 + 1 = 4$ too, hence consistent; in (e) $3 + 1 \neq 5$, hence inconsistent.

4.3 Gaussian Elimination Method

Gaussian elimination reduces $[A \mid b]$ to an **upper triangular** system using elementary row operations (swap two rows, multiply a row by a non-zero constant, add a multiple of one row to another), and then solves by **back substitution**.

Algorithm

- **Forward elimination:** for $k = 1, \dots, n - 1$: (with partial pivoting, first swap in the row having the largest $|a_{ik}|$, $i \geq k$, to be the pivot); for each row $i > k$ compute the multiplier $m_{ik} = a_{ik}/a_{kk}$ and replace R_i by $R_i - m_{ik} \cdot R_k$.
- **Back substitution:** $x_n = b_n/a_{nn}$, and for $i = n - 1, \dots, 1$: $x_i = (b_i - \sum_{j > i} a_{ij}x_j) / a_{ii}$.

*Note: If a pivot a_{kk} is zero, the method breaks down unless rows are interchanged. Even a very small pivot amplifies round-off, which is why **partial pivoting** is used in practice. Elimination needs about $n^3/3$ multiplications.*

Example 4.2

Solve by Gaussian elimination: $2x + y + z = 5$, $4x - 6y = -2$, $-2x + 7y + 2z = 9$.

Stage	Row 1	Row 2	Row 3
Augmented matrix	2 1 1 5	4 -6 0 -2	-2 7 2 9
$R_2 \leftarrow R_2 - 2R_1$, $R_3 \leftarrow R_3 + R_1$	2 1 1 5	0 -8 -2 -12	0 8 3 14
$R_3 \leftarrow R_3 + R_2$	2 1 1 5	0 -8 -2 -12	0 0 1 2

Back substitution

From Row 3: $z = 2 / 1 = 2$

From Row 2: $-8y - 2(2) = -12 \Rightarrow -8y = -8 \Rightarrow y = 1$

From Row 1: $2x + 1 + 2 = 5 \Rightarrow x = 1$. **Solution: $(x, y, z) = (1, 1, 2)$**

Check in equation 3: $-2(1) + 7(1) + 2(2) = 9$ ✓

4.4 Gauss–Jordan Method

Gauss–Jordan continues the elimination until the coefficient matrix becomes the **identity** matrix. Each pivot row is divided by its pivot and the pivot column is cleared *above and below* the pivot, so the solution appears directly in the last column and no back substitution is needed. It needs about $n^3/2$ multiplications (more than Gauss).

Example 4.3

Solve the system of Example 4.2 by the Gauss–Jordan method.

Stage	Row 1	Row 2	Row 3
After Gaussian elimination	2 1 1 5	0 -8 -2 -12	0 0 1 2

Stage	Row 1	Row 2	Row 3
Make R_3 pivot 1 (already); clear column 3: $R_2 \leftarrow R_2 + 2R_3$, $R_1 \leftarrow R_1 - R_3$	2 1 0 3	0 -8 0 -8	0 0 1 2
$R_2 \leftarrow R_2/(-8)$; $R_1 \leftarrow R_1 - R_2$; $R_1 \leftarrow R_1/2$	1 0 0 1	0 1 0 1	0 0 1 2

Reading the last column: $x = 1$, $y = 1$, $z = 2$, the same as before.

4.4.1 Inverse of a Matrix by Gauss–Jordan Elimination

To find A^{-1} , form the augmented matrix $[A \mid I]$ and apply row operations until the left block becomes I . The right block is then A^{-1} . If a zero pivot cannot be removed by a row swap, A is singular and has no inverse.

Example 4.4

Find the inverse of $A = [[2, -1, 0], [-1, 2, -1], [0, -1, 2]]$ by Gauss–Jordan elimination.

Solution

Start with $[A \mid I]$. Normalise R_1 ($R_1 \leftarrow R_1/2$), then $R_2 \leftarrow R_2 + R_1$, normalise R_2 (pivot $3/2$), clear above and below, then normalise R_3 (pivot $4/3$) and clear above. The final right-hand block is: $A^{-1} = (1/4) \times [[3, 2, 1], [2, 4, 2], [1, 2, 3]]$ i.e.

	col 1	col 2	col 3
row 1	3/4	1/2	1/4
row 2	1/2	1	1/2
row 3	1/4	1/2	3/4

Check: $A \cdot A^{-1} = I$ (row 1 of A times column 1 of $A^{-1} = 2(3/4) - 1(1/2) + 0 = 1$ ✓; $\det(A) = 4$).

4.5 Method of Factorization (LU Decomposition)

If A can be written as $A = L U$ with L lower triangular and U upper triangular, then $Ax = b$ is solved in two easy steps:

- **Forward substitution:** solve $L y = b$ for y ;
- **Back substitution:** solve $U x = y$ for x .

Doolittle's method: L has 1s on its diagonal. **Crout's method:** U has 1s on its diagonal. (For a symmetric positive-definite matrix, $A = LL^T$ is the *Cholesky* factorisation.) The entries are

found by equating $A = LU$ term by term:

Doolittle: $u_{ij} = a_{ij} - \sum (k < i) l_{ik} u_{kj} \ (j \geq i)$, $l_{ji} = [a_{ji} - \sum (k < i) l_{jk} u_{ki}] / u_{ii} \ (j > i)$

The main advantage: once A is factorised, many right-hand sides b can be solved cheaply, and the factors also give $\det(A) = \prod u_{ii}$.

Example 4.5

Solve by Doolittle's LU method: $2x + 3y + z = 6$, $4x + 7y + 5z = 16$, $6x + 18y + 22z = 46$.

Solution

Row 1 of U = row 1 of $A = (2, 3, 1)$. Multipliers $l_{21} = 4/2 = 2$, $l_{31} = 6/2 = 3$.

$u_{22} = 7 - 2 \cdot 3 = 1$, $u_{23} = 5 - 2 \cdot 1 = 3$; $l_{32} = (18 - 3 \cdot 3)/1 = 9$; $u_{33} = 22 - 3 \cdot 1 - 9 \cdot 3 = -8$.

$L = [[1, 0, 0], [2, 1, 0], [3, 9, 1]]$, $U = [[2, 3, 1], [0, 1, 3], [0, 0, -8]]$

Forward: $Ly = b$ gives $y_1 = 6$, $y_2 = 16 - 2 \cdot 6 = 4$, $y_3 = 46 - 3 \cdot 6 - 9 \cdot 4 = -8$.

Back: $Ux = y$ gives $z = -8/(-8) = 1$, $y = 4 - 3(1) = 1$, $x = (6 - 3(1) - 1)/2 = 1$.

Solution: $x = y = z = 1$. Also $\det(A) = 2 \cdot 1 \cdot (-8) = -16$.

4.6 Iterative Methods

For large sparse systems, iterative methods are cheaper and self-correcting. Rewrite each equation for its diagonal unknown (this needs $a_{ii} \neq 0$) and repeat from an initial vector $x^{(0)}$ (usually 0).

4.6.1 Jacobi Iteration

$x_i^{(k+1)} = (b_i - \sum (j \neq i) a_{ij} x_j^{(k)}) / a_{ii}$ — every component of the new vector uses only *old* values.

4.6.2 Gauss–Seidel Iteration

$x_i^{(k+1)} = (b_i - \sum (j < i) a_{ij} x_j^{(k+1)} - \sum (j > i) a_{ij} x_j^{(k)}) / a_{ii}$ — the newest values are used as soon as they are computed, so it usually converges faster and needs less storage.

Convergence. Both methods converge for any starting vector if A is **strictly diagonally dominant** ($|a_{ii}| > \sum (j \neq i) |a_{ij}|$); this is sufficient but not necessary. The exact condition is that the spectral radius of the iteration matrix is less than 1. Rows may be re-ordered to obtain dominance. Stop when $\max |x_i^{(k+1)} - x_i^{(k)}| < \epsilon$.

Example 4.6

Solve by Jacobi and by Gauss–Seidel ($x^{(0)} = 0$):

$$10x + y + z = 12, \quad 2x + 10y + z = 13, \quad 2x + 2y + 10z = 14.$$

The matrix is strictly diagonally dominant ($10 > 1 + 1$, $10 > 2 + 1$, $10 > 2 + 2$). The iteration formulas are $x = (12 - y - z)/10$, $y = (13 - 2x - z)/10$, $z = (14 - 2x - 2y)/10$.

k	x	y	z
0	0.00000	0.00000	0.00000
1	1.20000	1.30000	1.40000
2	0.93000	0.92000	0.90000
3	1.01800	1.02400	1.03000
4	0.99460	0.99340	0.99160
5	1.00150	1.00192	1.00240
6	0.99957	0.99946	0.99932

Jacobi iterates (above). For **Gauss–Seidel**, x is updated first and the new x is used at once in the y -equation, then the new x and y in the z -equation:

k	x	y	z
0	0.00000	0.00000	0.00000
1	1.20000	1.06000	0.94800
2	0.99920	1.00536	0.99909
3	0.99956	1.00018	1.00005
4	0.99998	1.00000	1.00000
5	1.00000	1.00000	1.00000
6	1.00000	1.00000	1.00000

The exact solution is (1, 1, 1). After 6 iterations the error norm of Gauss–Seidel is 3.1×10^{-8} , against 9.7×10^{-4} for Jacobi.

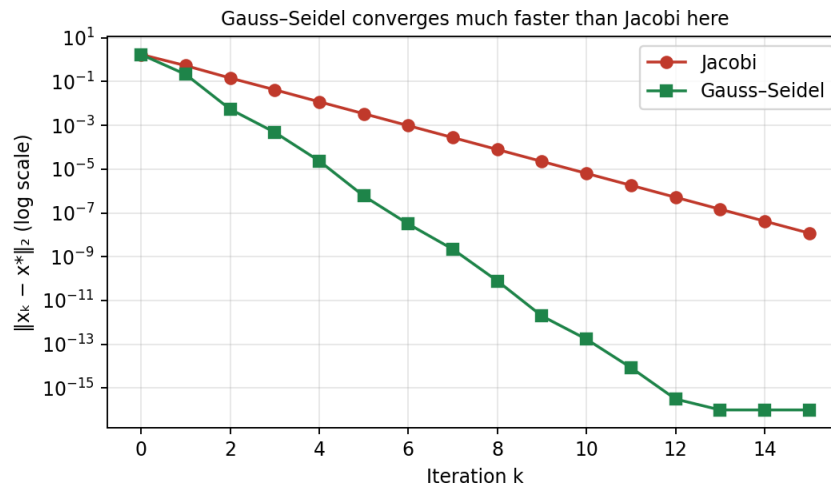


Figure 4.2 — Error norm versus iteration for Jacobi and Gauss–Seidel on Example 4.6.

Feature	Direct (Gauss, LU)	Iterative (Jacobi, GS)
Result	Exact (up to round-off) after a fixed number of steps	Approximation improved every step
Cost	$\approx n^3/3$ operations	$\approx n^2$ per iteration (n per iteration if sparse)
Best for	Small / dense systems, several right-hand sides (LU)	Large sparse systems, diagonally dominant
Risk	Zero or tiny pivots; needs pivoting	May diverge if not diagonally dominant

4.7 Power Method for the Dominant Eigenvalue

An **eigenvalue** λ and **eigenvector** v of A satisfy $A v = \lambda v$ ($v \neq 0$). The power method finds the dominant eigenvalue (largest in magnitude) and its eigenvector, assuming $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$.

Algorithm

- Choose a starting vector $v^{(0)}$ with a non-zero component along the dominant eigenvector (often all 1s).
- Compute $w = A v^{(k)}$. Let $\lambda^{(k+1)}$ be the component of w with the largest magnitude.
- Normalise: $v^{(k+1)} = w / \lambda^{(k+1)}$ (so the largest component of v is 1).
- Repeat until $\lambda^{(k)}$ changes by less than ϵ . Then $\lambda_1 \approx \lambda^{(k)}$ and $v^{(k)}$ is the eigenvector.

The error decreases like $|\lambda_2/\lambda_1|^k$, so convergence is slow when the two largest eigenvalues are close. The smallest eigenvalue is found by applying the method to A^{-1} (inverse power method), and other eigenvalues by shifting $A - sI$.

Example 4.7

Find the dominant eigenvalue of $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ with $\mathbf{v}^{(0)} = (1, 1, 1)^T$.
The exact eigenvalues are $2 \pm \sqrt{2}$ and 2, so $\lambda_1 = 2 + \sqrt{2} = 3.414214$.

k	$\mathbf{v}^{(k-1)}$ (start)	$\mathbf{w} = A \mathbf{v}$	$\lambda^{(k)}$
1	(1.0000, 1.0000, 1.0000)	(1.0000, 0.0000, 1.0000)	1.00000
2	(1.0000, 0.0000, 1.0000)	(2.0000, -2.0000, 2.0000)	2.00000
3	(1.0000, -1.0000, 1.0000)	(3.0000, -4.0000, 3.0000)	-4.00000
4	(-0.7500, 1.0000, -0.7500)	(-2.5000, 3.5000, -2.5000)	3.50000
5	(-0.7143, 1.0000, -0.7143)	(-2.4286, 3.4286, -2.4286)	3.42857
6	(-0.7083, 1.0000, -0.7083)	(-2.4167, 3.4167, -2.4167)	3.41667
7	(-0.7073, 1.0000, -0.7073)	(-2.4146, 3.4146, -2.4146)	3.41463
8	(-0.7071, 1.0000, -0.7071)	(-2.4143, 3.4143, -2.4143)	3.41429

After 8 iterations $\lambda \approx 3.4143$ and after 10, $\lambda \approx 3.4142$, approaching the exact value 3.4142. The limiting eigenvector is proportional to $(1, -\sqrt{2}, 1) \approx (-0.7071, 1, -0.7071)$ after normalising by the largest component.

Key Takeaways — Unit 4

- ✓ $A\mathbf{x} = \mathbf{b}$ has a unique solution iff $\det(A) \neq 0$; general test: compare $\text{rank}(A)$ with $\text{rank}([A \mid \mathbf{b}])$.
- ✓ Gaussian elimination: forward elimination to upper triangular form, then back substitution (use partial pivoting).
- ✓ Gauss–Jordan: reduce to identity; also gives A^{-1} from $[A \mid I]$.
- ✓ LU factorisation $A = LU$ (Doolittle: unit L; Crout: unit U): solve $L\mathbf{y} = \mathbf{b}$ then $U\mathbf{x} = \mathbf{y}$; $\det(A) = \prod u_{ii}$.
- ✓ Jacobi uses only old values; Gauss–Seidel uses the newest values and converges faster; strict diagonal dominance $\Rightarrow$ convergence.
- ✓ Power method: $\mathbf{v} \leftarrow A \mathbf{v} / (\text{largest component})$; the scale factor converges to the dominant eigenvalue at rate $|\lambda_2/\lambda_1|$.

— End of Unit 4 —