

Laboratorio di Programmazione e Calcolo  
Canale 2  
Appunti del corso

Simone Cacace e Giuseppe Visconti

Dipartimento di Matematica  
Sapienza Università di Roma

Anno Accademico 2025–2026

## Laboratorio – Sistemi lineari

### Esercizi

#### E0. Confronto tra i metodi di Jacobi e Gauss–Seidel con matrici casuali

Generare e risolvere numericamente un sistema lineare  $Ax = b$  di dimensione  $n = 100$  la matrice  $A$  è tridiagonale con entrate casuali, con la diagonale principale costruita in modo da garantire dominanza diagonale. Ad esempio:

$$a_{ii} = 5 + \text{rand}() \% 5, \quad a_{i,i-1} = -(\text{rand}() \% 3), \quad a_{i,i+1} = -(\text{rand}() \% 3).$$

In questo modo  $A$  è diagonale dominante e i metodi iterativi risultano ben comportati.

Il vettore dei termini noti è, invece,

$$b_i = \text{rand}() \% 10.$$

Scrivere un programma che:

- implementa le funzioni che eseguono le iterazione dei metodi di Jacobi e Gauss–Seidel, secondo le formule viste a lezione. Prevedere che le funzioni restituiscano anche l’array dei residui ad ogni iterazione e l’iterazione di uscita;
- genera, nel main, la matrice  $A$  e il vettore dei termini noti  $b$ ;
- inizializza  $x^{(0)}$  come vettore nullo;
- calcola la soluzione con il metodo di Jacobi e con il metodo di Gauss–Seidel;
- salva su due file di testo `jacobi.txt` e `gaussseidel.txt` due colonne con il numero di iterazioni e i valori del residuo delle due sequenze, utili per creare un grafico esterno con Gnuplot.

Analizzare i risultati:

- verificare quale metodo converge più velocemente e discutere perché Gauss–Seidel tende a richiedere meno iterazioni rispetto a Jacobi;
- esplorare cosa accade ripetendo l’esperimento senza garantire la dominanza diagonale.

*Suggerimento.* Per ottenere risultati confrontabili tra esecuzioni diverse, inizializzare il generatore di numeri casuali con `srand(0)`.

**Opzionale.** Verificare come il rapporto tra numero di iterazioni che impiega Jacobi a convergere e il numero di iterazioni che impiega Gauss–Seidel a convergere varia al variare di  $n$ , dimensione del sistema.

#### E1. Matrici di Hilbert

Studiamo il comportamento numerico dell’algoritmo di eliminazione di Gauss applicato a una classe di matrici note per essere particolarmente problematiche dal punto di vista del calcolo numerico: le *matrici di Hilbert*.

Per ogni  $n \in \mathbb{N}$ , si definisce la matrice di Hilbert  $H_n \in \mathbb{R}^{n \times n}$  come

$$(H_n)_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, \dots, n.$$

Tali matrici sono simmetriche e definite positive, dunque invertibili per ogni  $n$ .

Considerare il sistema lineare

$$H_n x = b,$$

dove il vettore  $b \in \mathbb{R}^n$  ha tutte le componenti uguali a 1, cioè

$$b = (1, 1, \dots, 1)^T.$$

Scrivere un programma che

- implementa la costruzione della matrice di Hilbert  $H_n$ ;
- risolvere il sistema lineare  $H_n x = b$  mediante il metodo di eliminazione di Gauss, usando la funzione scritta a lezione e scrivendo una funzione che risolve il sistema triangolare superiore;
- ripetere il calcolo per diversi valori di  $n$  crescenti;
- valuta a posteriori la qualità della soluzione calcolando il residuo  $r = H_n x - b$  e osservando come l'errore numerico cresce all'aumentare di  $n$ .

**Spiegazione matematica.** Sebbene le matrici di Hilbert siano teoricamente ben poste (simmetriche e invertibili), esse sono *fortemente mal condizionate*. Ciò significa che piccoli errori di arrotondamento, inevitabili nei calcoli in aritmetica floating point, vengono amplificati durante l'eliminazione di Gauss, producendo soluzioni numericamente poco affidabili già per valori moderati di  $n$ .

Questo esercizio mette in evidenza la differenza tra invertibilità teorica e stabilità numerica, mostrando i limiti dei metodi diretti quando applicati a matrici mal condizionate e motivando l'uso di tecniche numeriche più robuste o di un'analisi preventiva del problema.