LAB 2

Esercizio 1

L'esercizio ci chiede di calcolare la **norma** ∞ di quattro matrici.

Partiamo dalle prime 2 che sono presenti nel punto **a**.

a)

$$A1 = \begin{pmatrix} 3 & 1 & -1 & 0 \\ 0 & 7 & -3 & 0 \\ 0 & -3 & 9 & -2 \\ 0 & 0 & 4 & -10 \end{pmatrix} \qquad A2 = \begin{pmatrix} 2 & 4 & -2 & 0 \\ 1 & 3 & 0 & 1 \\ 3 & -1 & 1 & 2 \\ 0 & -1 & 2 & 1 \end{pmatrix}$$

La **norma infinito** di una matrice A è definita come il massimo della somma dei valori assoluti degli elementi di ogni riga della matrice.
La norma infinito è data da:

$$\|A_\infty\| = max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$

- a_{ij} è l'elemento della matrice nella i-esima riga e j-esima colonna
- $\sum_{j=1}^{n} |a_{ij}|$ Calcola la somma dei valori assoluti degli elementi della i-esima riga.
- Il massimo viene prese tra tutte le righe della matrice

Quindi, nella matrice A1 la norma ∞ si calcola in questo modo :

- **Prima riga**: |3|+|1|+|−1|+|0| = 3+1+1+0 = 3+1+1+0=5
- **Seconda riga**: |0|+|7|+|−3|+|0| = 0+7+3+0 = 10
- **Terza riga**: |0|+|−3|+|9|+|−2| = 0+3+9+2 = 14
- **Quarta riga**: |0|+|0|+|4|+|−10| = 0+0+4+10 = 14

Ora, prendiamo il massimo tra i risultati delle somme:

$$\|A_\infty\| = max\ (5,10,14,14)=14$$

Quindi, la **norma infinito** della matrice è **14**.

Per la matrice 2 usiamo gli stessi passaggi e come risultato otteniamo **8.**

Spiegazione codice

La nostra funzione **normaInfinito** ci serve per poter eseguire i calcoli spiegati precedentemente, la matrice è rappresentata come un **vector<vector<double>>,** dove
ogni **vector<double>** rappresenta una riga della matrice. Il ciclo esterno percorre ogni riga della matrice, mentre il ciclo interno somma i valori assoluti degli elementi della riga. Per ottenere la norma infinito, il codice tiene traccia della somma più grande tra tutte le righe, aggiornando una variabile **max_sum** ogni volta che viene trovata una somma maggiore di quella precedentemente registrata. Una volta che tutte le righe sono state elaborate, la funzione restituisce il valore massimo trovato, che rappresenta la norma infinito della matrice.

NOTA
Usiamo un **doppio vector** per rappresentare una matrice perché, in C++, un singolo vector può essere utilizzato per rappresentare un array unidimensionale (una riga), mentre
un **doppio vector** permette di creare una struttura bidimensionale.

b)

Il **punto b** ci chiede invece di calcolare la **norma ∞ della matrice Pascal** 10 x 10.

La matrice di Pascal *P* è una matrice quadrata in cui gli elementi sono determinati dai coefficienti binomiali, secondo questa formula:

$$P[i,j] = \binom{i+j-2}{i-1}$$

Il coefficiente binomiale si calcola con:

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Spiegazione codice

Abbiamo deciso di utilizzare una funzione iterativa che ci calcoli direttamente il coefficiente binomiale invece di una funzione che calcola direttamente i fattoriali in quanto abbiamo notato che con l'utilizzo di una **funzione fact** si presentano diversi errori di approssimazioni da parte del computer, inoltre, calcolando il coefficiente binomiale non c'è bisogno di memorizzare fattoriali molto grandi, evitando quindi sprechi di risorse e non abbiamo neanche il problema riguardo alla crescita esponenziale dei numeri nei fattoriali.
La funzione **coeffBinomiale** calcola il valore del coefficiente binomiale. Se k è maggiore di n, la funzione restituisce 0, poiché non è possibile scegliere più elementi di quelli disponibili. Se k è 0 o uguale a n, la funzione restituisce 1.
La funzione **generaMatricePascal** crea la matrice di Pascal di dimensioni n×n. La matrice viene rappresentata da un **vector<vector<double>>,** dove ogni elemento è inizializzato a 0.
Creiamo un vettore di n righe, ognuna delle quali è un vettore di n colonne, inizializzato a zero.
è presente un ciclo annidato: il ciclo esterno itera sulle righe e il ciclo interno itera sulle colonne.
Infine viene calcolato il valore che va inserito nella posizione **[i][j]** della matrice di Pascal utilizzando la chiamata della funzione **coeffBinomiale.**

c)

Ci viene richiesto di calcolare la norma infinito di una matrice tridiagonale di dimensione n x n. La formula per calcolare n è definita in questo modo:

$$\mathrm{n} = 10 \cdot (d_1 + 1) + d_0$$

Nel nostro caso d_0 vale 4 mentre d_1 vale 2 .

Spiegazione codice

la funzione inizia creando una matrice di dimensioni **size×size** inizializzata a zero utilizzando il tipo **vector<vector<double>>,** che è una struttura di dati dinamica adatta per gestire matrici di dimensione variabile.
La matrice viene poi riempita tramite due cicli annidati: il primo ciclo itera sulle righe della matrice, mentre il secondo ciclo itera sulle colonne. Per ogni elemento della matrice, viene controllato se la posizione è sulla diagonale principale. Se la posizione si trova immediatamente sopra o sotto la diagonale principale il valore della cella viene impostato a -1. Tutti gli altri valori vengono mantenuti a zero.
Alla fine, la funzione restituisce la matrice tridiagonale generata

Output ottenuti

```
**********************************************

Norma infinito della prima matrice: 14
Norma infinito della seconda matrice: 8

Norma infinito della matrice di Pascal: 92378

Norma infinito della matrice Triangolare: 4
**********************************************
```

Esercizio 2

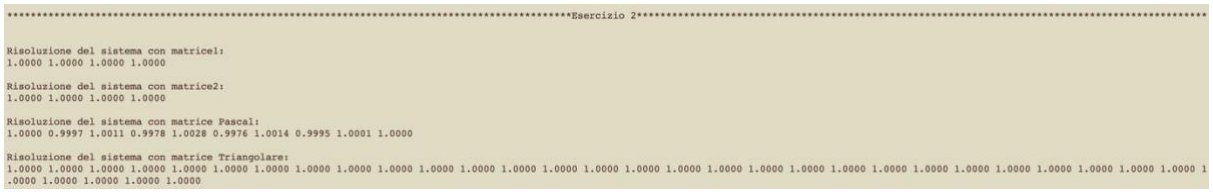
In questo esercizio, ci viene chiesto di implementare un programma che segua questi passaggi per quattro matrici predefinite:

1. **Costruzione della matrice A**: La matrice A deve essere costruita senza chiedere all'utente di inserire direttamente gli elementi. Le matrici sono già definite nel punto precedente, quindi bisogna generarle automaticamente nel programma.
2. **Calcolo del termine noto:** Data la soluzione $\overline{x} = (1,1,\ldots,1)^T$ bisogna calcolare il termine noto b per il sistema lineare, dato dal prodotto $b = A \cdot \overline{x}$
3. **Risoluzione del sistema**: Dobbiamo risolvere il sistema lineare usando l'algoritmo di eliminazione gaussiana in precisione singola .

Spiegazione codice

La funzione **eliminazioneGaussiana** utilizza due cicli nidificati per applicare il metodo di eliminazione. In pratica, scorre lungo la diagonale principale e, allo stesso tempo, si occupa di eliminare le variabili nelle righe sotto la diagonale. Durante questo processo, aggiorniamo sia il termine **noto b** che la matrice A per trasformare A in una matrice a forma di **triangolo superiore.** Una volta ottenuta questa forma, si trova la soluzione usando la retro-sostituzione, cioè risolvendo le equazioni dalla fine all'inizio. Con l'aggiunta del pivoting parziale, facciamo in modo di usare sempre gli elementi più grandi per stabilizzare il calcolo e ridurre gli errori.

Output ottenuti



Esercizio 3

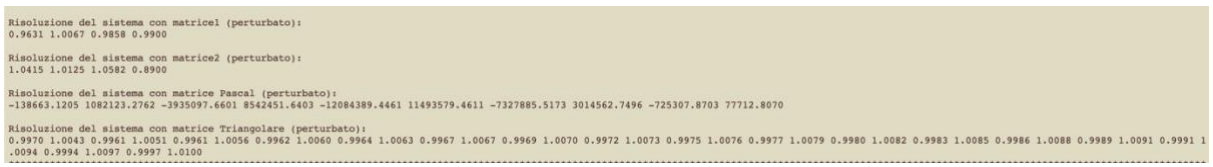
In questo esercizio ci viene richiesto di verificare la sensibilità del sistema lineare alle perturbazioni del termine noto b. A tal fine, per ogni matrice del punto precedente, consideriamo un termine noto perturbato b + δb, dove il vettore di perturbazioni δb è definito come:

$$\delta b = \|\boldsymbol{b}\|_{\infty} \cdot (-0.01,\ 0.01\ , -0.01, \ldots, 0.01)^{T}$$

Spiegazione codice

Per calcolare **δb**, abbiamo prima calcolato la norma infinito del vettore b utilizzando la funzione **normaInfinitoVettore**. Successivamente, abbiamo creato un vettore **δb** in cui ciascun elemento alterna tra -0.01 e 0.01, moltiplicato per la **norma infinito di b**. In questo modo, la perturbazione applicata è proporzionale alla grandezza del termine noto, il che permette di valutare l'effetto delle piccole variazioni.
La funzione **aggiungiPerturbazione** è stata utilizzata per generare il vettore perturbato **b + δb.** Questo nuovo termine noto è stato poi utilizzato per risolvere nuovamente il sistema lineare, ottenendo una nuova soluzione $\tilde{\mathbf{x}}$**.**

Output ottenuti



Differenza esercizio 2 ed esercizio 3

Confrontando le soluzioni $\mathbf{x}$ (senza perturbazione) e $\tilde{\mathbf{x}}$ (con perturbazione), abbiamo osservato che la soluzione $\tilde{\mathbf{x}}$ può differire significativamente da $\mathbf{x}$, soprattutto pe:

- **Matrice di Pascal:** Questa matrice è chiaramente mal condizionata, come evidenziato dai numeri estremamente grandi nella soluzione perturbata $\tilde{\mathbf{x}}$. Questo risultato dimostra come anche una piccola perturbazione nel termine noto possa portare a errori enormi nella soluzione.

- **Matrice triangolare:** Al contrario della matrice di Pascal, questa matrice mostra un comportamento ben condizionato. Anche con l'aggiunta di perturbazioni al termine noto, la soluzione perturbata $\tilde{\mathbf{x}}$ rimane molto vicina alla soluzione originale $\mathbf{x}$. Questo evidenzia come il condizionamento della matrice sia basso, rendendola numericamente stabile e meno sensibile alle variazioni nei dati di ingresso.