Università degli Studi di Milano Bicocca

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Progetto 1-bis

Progetto del corso di Metodi del Calcolo Scientifico

Davide Costantini 856114

David Gargaro 845738

Anno Accademico 2023-2024

Sommario

Progetto 1-bis

L'obiettivo del progetto è implementare una libreria C++ che sia in grado di risolvere un sistema lineare utilizzando i metodi di Jacobi, di Gauß-Seidel, del Gradiente e del Gradiente Coniugato.

Il codice del progetto è disponibile in questa repository:
https://github.com/Deivmercer/Metodi-del-Calcolo-Scientifico-1-bis

I Metodi Iterativi

Supponiamo di avere un sistema lineare $Ax = b$, dove $A \in R^{n \times n}$, $b \in R^n$ è il termine noto e $x \in R^n$ è il vettore delle incognite. I metodi iterativi si differenziano dai metodi diretti in quanto non restituiscono immediatamente la soluzione del sistema lineare, ma costruiscono iterativamente, partendo da un vettore iniziale $x^{(0)} \in R^n$, una sequenza di vettori $x^{(k)} \in R^n$ che si avvicinano sempre di più alla soluzione esatta $\tilde{x} \in R^n$.

Questo tipo di metodi sono particolarmente efficaci quando la matrice A è molto grande e sparsa. In questa situazione, i metodi diretti soffrono del problema di Fill In, ovvero la decomposizione della matrice perde la sua natura sparsa, aumentando notevolmente il numero delle entrate non nulla e causando problemi di memoria. Al contrario, le procedure iterative mantengono la natura sparsa della matrice.

Il numero di iterazioni necessarie a raggiungere la soluzione più vicina alla soluzione esatta non è noto a priori. Come condizione di arresto della procedura è possibile impostare una soglia di tolleranza e verificare se il residuo scalato è inferiore di questo valore:

$$\frac{\|Ax^{(k)} - b\|}{\|b\|} < tol$$

Inoltre, è possibile utilizzare un numero massimo di iterazioni. Per lo scopo del progetto, è stato impostato a 20000.

I metodi iterativi si distinguono in stazionari e non stazionari. In generale, in un metodo iterativo, la soluzione all'iterazione $k + 1$ viene calcolata come $x^{(k+1)} = x^{(k)} + \alpha P^{-1} r^{(k)}$; nei metodi iterativi stazionari il coefficiente α è costante tra le varie iterazioni e dipende dall'algoritmo utilizzato, mentre nei metodi iterativi non stazionari il coefficiente viene calcolato ad ogni iterazione e si avrà dunque una sequenza di coefficienti α_k.

Metodi Iterativi Stazionari

Il Metodo di Jacobi

Il metodo di Jacobi è una tecnica iterativa per risolvere sistemi lineari di equazioni della forma $Ax = b$, dove A è una matrice quadrata di dimensione $n \times n$, x è il vettore incognito di dimensione n, e b è il vettore dei termini noti di dimensione n.

Partendo da $x_n = \frac{1}{a_{n,n}}\left(b_n - \sum_{k \neq n} a_{n,k} x_k\right)$ si può notare un aspetto importante, ovvero per ogni equazione che si salta l'indice dell'entrata corrispondente.

Partendo dall'equazione precedente si possono definire le matrici P e N della matrice A per il metodo di Jacobi. Consideriamo una generica matrice $A \in R^{n \times n}$, allora:

> P: è la diagonale di A
>
> N: è la matrice che si ottiene mettendo a 0 le entrate sulla diagonale di A e considerando l'opposto di tutte le altre entrate

Si vede facilmente che le matrici P e N soddisfano la decomposizione

$$x = P^{-1}Nx + P^{-1}b.$$

Se $A \in R^{n \times n}$ è una matrice a dominanza diagonale stretta per righe allora il metodo di Jacobi converge. Da notare che la condizione precedente è una condizione necessaria ma non sufficiente. Ossia potrebbero esserci matrici non a dominanza diagonale stretta per cui il metodo di Jacobi converge.

Il calcolo del residuo segue la formula $r^{(k)} = b - Ax^{(k)}$, che richiede un prodotto tra matrice e vettore, dal costo $O(n^2)$, e la somma di due vettori, dal costo $O(n)$. In totale costerà $O(n^2)$.

Il calcolo della soluzione all'iterazione $k + 1$ avviene come segue:

$$x^{(k+1)} = x^{(k)} - P^{-1}r^{(k)}.$$

Il calcolo di $P^{-1}r^{(k)}$ è un prodotto tra una matrice ed un vettore, che normalmente costerebbe $O(n^2)$. In questo caso però sappiamo che P^{-1} è una matrice diagonale; dunque, possiamo ottimizzare il prodotto considerando solo le entrate sulla diagonale della matrice, ottenendo un costo $O(n)$. Infine, la differenza tra il vettore risultante e $x^{(k)}$ costa $O(n)$.

Considerati questi contributi, il costo computazionale del metodo di Jacobi è

$$O(n^2) + O(n) + O(n) = O(n^2).$$

Il Metodo di Gauß-Seidel

Il metodo di Gauß-Seidel è una variante del metodo di Jacobi. Osservando la formula per calcolare l'entrata k-esima del vettore x, possiamo notare che nel termine di destra ci sono tutti valori che dipendono dall'iterata precedente. L'idea su cu si basa il metodo di Jacobi è di sfruttare le entrate del vettore x già calcolate durante l'iterazione attuale.

Considerata la matrice $A \in R^{nxn}$, definiamo la scomposizione nelle matrici P e N come segue:

$$P = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}$$

$$N = \begin{bmatrix} 0 & -a_{1,2} & \cdots & -a_{1,n} \\ 0 & 0 & \cdots & -a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

La matrice P è facilmente invertibile. La soluzione x all'iterazione $k + 1$ viene calcolata come segue:

$$x^{(k+1)} = x^{(k)} + P^{-1}r^{(k)}.$$

Il residuo $r^{(k)}$ può essere calcolato tramite la seguente equazione:

$$r^{(k)} = b - Ax^{(k)}.$$

Poiché calcolare la matrice inversa sarebbe molto costoso, lo riconduciamo alla risoluzione del seguente sistema lineare:

$$y = P^{-1}r^{(k)}.$$

Sapendo che la matrice P è triangolare inferiore, possiamo risolvere questo sistema in modo rapido applicando il metodo della sostituzione in avanti.

Il metodo di Gauß-Seidel converge se la matrice $A \in R^{nxn}$ è a dominanza diagonale stretta per righe. Poiché la condizione è sufficiente ma non necessaria, è possibile che il metodo converga anche con matrici che non rispettano questa condizione.

Consideriamo la formula per l'aggiornamento del vettore vista sopra.
Il calcolo del residuo è composto dal prodotto fra una matrice ed un vettore, che costa $O(n^2)$, e la somma di due vettori, che costa $O(n)$. Dunque, richiederà $O(n^2)$ operazioni.
Il prodotto $P^{-1}r^{(k)}$ viene interpretato come risoluzione del sistema lineare $y = P^{-1}r^{(k)}$. Dato che la matrice P è triangolare inferiore, questa operazione costa $O(n^2)$.
Infine, la somma tra $x^{(k)}$ e $P^{-1}r^{(k)}$ richiederà $O(n)$ operazioni.
Dunque, una iterata costerà

$$O(n^2) + O(n^2) + O(n) = O(n^2).$$

Metodi Iterativi Non Stazionari

Il Metodo del Gradiente

In questo metodo consideriamo la funzione $\phi(y) = \frac{1}{2} y^t A y - b^t y$ che, se A è simmetrica e definita positiva, avrà forma quadratica nella variabile vettoriale $y \in R^n$, ossia sarà l'analogo multidimensionale di una parabola con concavità verso l'alto. Trovare la soluzione del sistema lineare $Ax = b$ equivarrà a trovare l'unico punto di minimo della funzione $\phi(y)$. Per trovare questo punto di minimo calcoliamo il gradiente $\nabla\phi(y)$, e troveremo così il vettore $y \in R^n$ tale per cui $Ay - b = 0$,che equivale a trovare la soluzione del sistema di partenza.

La sequenza di soluzioni x^k può essere vista come il cammino percorso verso il punto di minimo della funzione ϕ. La soluzione all'iterazione $k + 1$ viene calcolata come. $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$, dove d^k è la direzione verso cui mi muovo e α_k rappresenta l'intensità del movimento. Poiché il gradiente valutato in un punto restituisce la direzione di massima crescita, per raggiungere il minimo sceglieremo la direzione opposta al gradiente. Tale direzione coincide con il residuo all'iterazione k, ovvero:

$$r^{(k)} = b - Ax^{(k)}.$$

Dunque, la formula per l'aggiornamento della variabile $x^{(k)}$ è:

$$x^{(k+1)} = x^{(k)} + \alpha_k r^{(k)}$$

Infine, il coefficiente α_k verrà calcolato come segue:

$$\alpha_k = \frac{\left(r^{(k)}\right)^t r^{(k)}}{\left(r^{(k)}\right)^t A r^{(k)}}.$$

Il metodo del gradiente converge, per ogni valore del dato iniziale $x^{(0)}$, se la matrice $A \in R^{nxn}$ è simmetrica e definita positiva. Infatti, se A non fosse simmetrica e definita positiva, la funzione ϕ non sarebbe quadratica e l'algoritmo non potrebbe essere applicato. Poiché calcolare tutti gli autovalori di una matrice grande richiederebbe molto tempo, per lo scopo del progetto facciamo questo controllo applicando la decomposizione di Cholesky. Infatti, se la matrice è definita positiva, allora è possibile effettuare la decomposizione di Cholesky.

La complessità del metodo del gradiente è $O\left(n^2\right)$.
Il calcolo di $r^{(k)}$ costa quanto un prodotto matrice-vettore, ovvero $O\left(n^2\right)$.
Nel calcolo di α_k, $Ar^{(k)}$ costerà $O\left(n^2\right)$, il numeratore costerà $O(n)$ come il denominatore dove abbiamo già calcolato $Ar^{(k)}$ e infine abbiamo il calcolo della divisione. Dunque, α_k

costerà $O(n) + O(n^2) + O(n) + 1 = O(n^2)$.
Il prodotto fra lo scalare α_k ed il vettore $r^{(k)}$ costerà $O(n)$.

Infine, la somma fra il vettore $x^{(k)}$ e $\alpha_k r^{(k)}$ richiederà $O(n)$ operazioni.

Sommando questi contributi otteniamo $O(n^2) + O(n^2) + O(n) + O(n) = O(n^2)$.

Il Metodo del Gradiente Coniugato

Questo metodo può essere visto come un miglioramento del metodo del gradiente dato che pone rimedio all'effetto di "convergenza a zig-zag"

Consideriamo una matrice $A \in R^{n \times n}$ simmetrica e definita positiva e il funzionale ϕ definito nel metodo del gradiente.

Un vettore $x^{(k)} \in R^n$ è ottimale rispetto alla direzione $d \in R^n$ se

$$\phi(x^{(k)}) \leq \phi(x^{(k)} + \lambda d), \ \forall \lambda \in R.$$

Il vettore $d \in R^n$ modifica il vettore di partenza $x^{(k)}$ tramite tutte le sue entrate e il coefficiente λ. Dire che il vettore $x^{(k)}$ è ottimale rispetto alla direzione d comporta che qualsiasi modifica fatta proporzionale a d non permette di avvicinarsi al minimo di ϕ. Infatti, nella definizione precedente il valore $\phi(x^{(k)})$ è minore o uguale al valore che ϕ assume se modificassimo $x^{(k)}$ nella direzione d. L'idea su cui si basa la procedura iterativa del gradiente coniugato è che, una volta trovato un vettore ottimale $x^{(k)}$ rispetto a una direzione d, non lo si dovrà più modificare lungo questa direzione.
Nel metodo del gradiente questo non è verificato per tutte le iterazioni. Infatti, l'effetto della "convergenza a zig-zag" è dovuto al fatto che a un certo punto otteniamo un vettore ottimale rispetto a una direzione e poi viene modificato di nuovo lungo la direzione d.

Un vettore $x^{(k)}$ è ottimale rispetto a una direzione $d \in R^n$ se $d \cdot r^{(k)} = 0$.

Sia $A \in R^{n \times n}$ una matrice simmetrica e definita positiva, il metodo del gradiente coniugato converge in al più n iterazioni.

Il metodo del gradiente coniugato cura la convergenza del metodo del gradiente preservando l'ottimalità dei vettori della successione $x^{(k)}$ rispetto alle direzioni precedenti. Questo fatto porta a un notevole vantaggio in termini di iterazioni necessarie alla convergenza.

I coefficienti α_k vengono calcolati come segue:

$$\alpha_k = \frac{\left(d^{(k)}\right)^t r^{(k)}}{\left(d^{(k)}\right)^t A d^{(k)}}.$$

Questo calcolo corrisponde al calcolo degli α_k del metodo del gradiente, che abbiamo visto costare $2O(n) + O\left(n^2\right) + 1 = O\left(n^2\right)$.

Il calcolo dell'iterata successiva avviene con la formula

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)},$$

che richiede un prodotto scalare ed una somma di vettori, per un costo totale di $O(n)$.

Il calcolo del residuo avviene come nel metodo del gradiente, e costa $O\left(n^2\right)$.

Il calcolo dei coefficienti β_k sfrutta la seguente formula:

$$\beta_k = \frac{\left(d^{(k)}\right)^t A r^{(k+1)}}{\left(d^{(k)}\right)^t A d^{(k)}}.$$

Il calcolo del numeratore e del denominatore richiede rispettivamente $O\left(n^2\right)$ e $O(n)$ operazioni; perciò, la complessità del calcolo dei coefficienti β_k è $O\left(n^2\right)$.

Infine, la nuova direzione viene calcolata come

$$d^{(k+1)} = r^{(k+1)} - \beta_k d^{(k)}.$$

L'equazione è simile a quella del calcolo dell'iterata successiva, che costa $O(n)$.

Dunque, sommando tutti questi contributi, la complessità totale del metodo del gradiente coniugato dunque è:

$$O\left(n^2\right) + O(n) + O\left(n^2\right) + O\left(n^2\right) + O(n) = O\left(n^2\right).$$

Struttura del progetto

La libreria implementata è composta principalmente da una classe astratta `IterativeSolver`, che definisce i seguenti metodi:

- `solve`: prende in input il sistema lineare $Ax = b$ e la soglia di tolleranza. La matrice ed i vettori di cui è composto il sistema contengono valori float. Si occupa di chiamare il metodo `getNextXk` implementato dalla sottoclasse e di verificare il criterio di arresto.

- `getNextXk`: è un metodo virtuale puro, che prende in input la matrice A, il vettore b ed il vettore $x^{(k)}$. Deve essere obbligatoriamente implementato dalle classi figlie, e si occupa di calcolare la soluzione $x^{(k+1)}$. Non restituisce nulla.
- `checkConvergence`: è un metodo virtuale puro, che prende in input la matrice A. Deve essere obbligatoriamente implementato dalle classi figlie, e si occupa di verificare il criterio di convergenza del metodo implementato dalla classe. Restituisce un valore `bool`.

Inoltre, implementa i metodi getter per le seguenti proprietà `protected`:

- `x`: è un vettore che rappresenta la soluzione all'iterazione corrente.
- `residual`: è un vettore che rappresenta il residuo all'iterazione corrente.
- `n_iteration`: è un `int` che rappresenta il numero di iterazione corrente.

Infine, definisce la costante `MAX_ITERATIONS`, un valore int che rappresenta il numero massimo di iterazioni raggiungibile. È impostato a 20000.

Sono presenti altre quattro classi, una per ogni metodo richiesto dalla traccia del progetto.

La classe `GaussSeidel` implementa, oltre ai due metodi virtuali, i metodi `getLowerMatrix` e `getUpperMatrix`, che restituiscono la decomposizione della matrice A rispettivamente nella matrice triangolare inferiore e superiore definita dal metodo.

La classe `GradientConjugate` implementa, oltre ai due metodi virtuali, il metodo getter per la proprietà privata `direction`, un vettore che rappresenta la direzione all'iterazione corrente.

Le classi `Jacobi` e `GradientMethod` implementano solamente i metodi virtuali richiesti dalla classe padre.

Sono inoltre presenti i seguenti namespace:

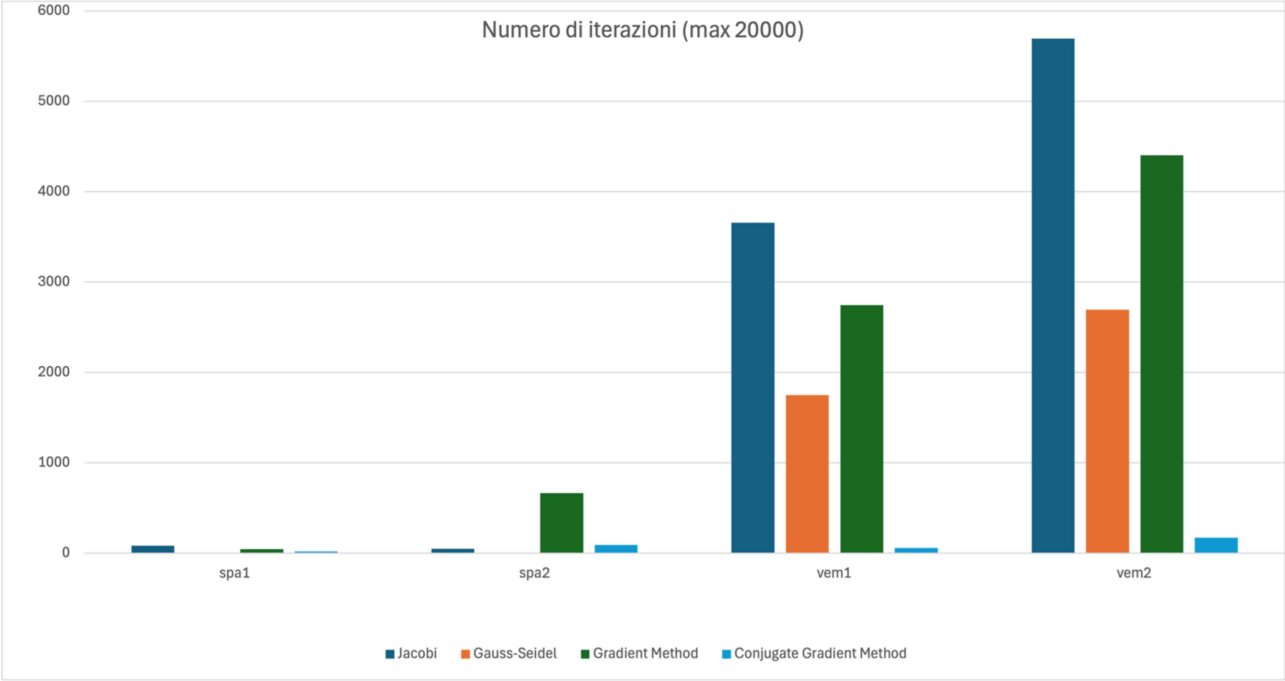
- `ForwardSubstitution`: implementa il metodo della sostituzione all'indietro.
- `CholeskyDecomposition`: implementa la decomposizione di Cholesky.
- `Utils`: implementa varie funzioni di utilità che vengono utilizzate dai metodi risolutivi.

Infine, l'header `types` definisce alcuni tipi di dati utilizzati dalla libreria.

Analisi dei Risultati

Numero di iterazioni (massimo 20000):

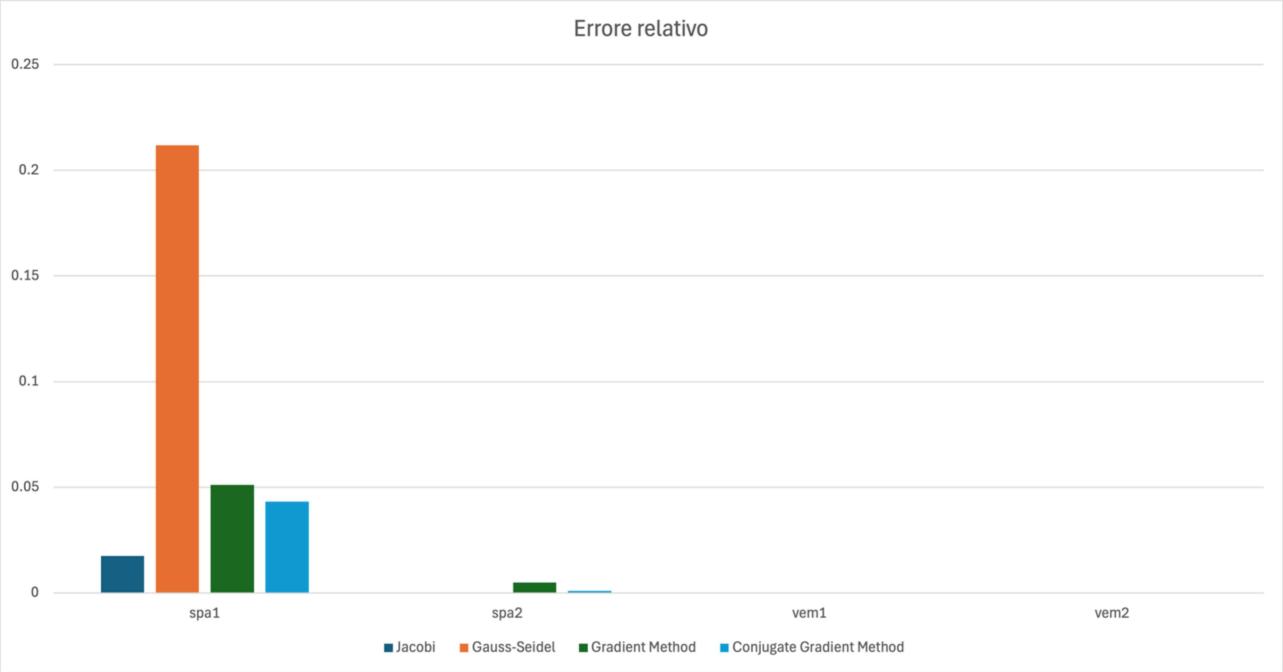
	Jacobi	Gauss-Seidel	Metodo del Gradiente	Metodo del Gradiente Coniugato
spa1	82	5	43	17
spa2	47	7	663	90
vem1	3655	1748	2745	58
vem2	5696	2694	4403	170



Guardando il numero di iterazioni possiamo notare come per tutti i metodi tranne il gradiente coniugato le matrici spa vengono risolte molto più rapidamente rispetto alle matrici vem. Inoltre, possiamo notare il netto miglioramento che il metodo del gradiente coniugato porta rispetto al metodo del gradiente.

Errore relativo:

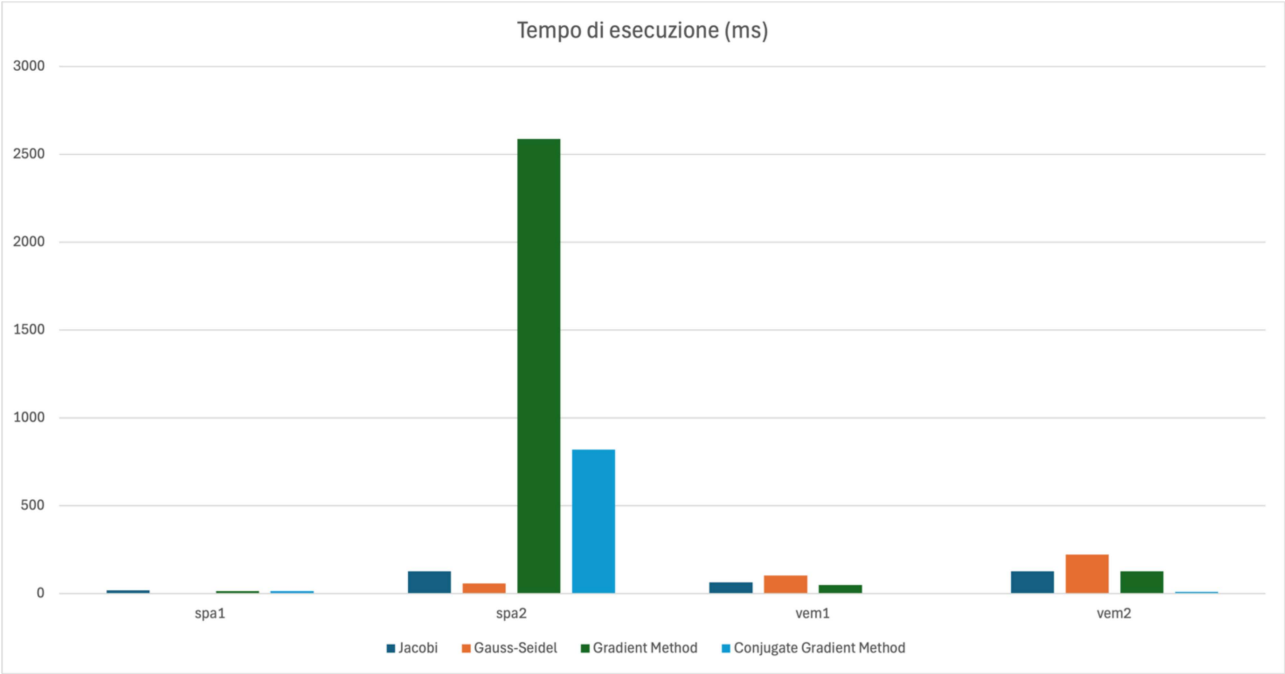
	Jacobi	Gauss-Seidel	Metodo del Gradiente	Metodo del Gradiente Coniugato
spa1	0,0175799	0,211808	0,0510309	0,0431601
spa2	0,0001551	0,000192227	0,00494196	0,00104067
vem1	8,04E-08	8,44E-08	8,39E-08	2,12E-08
vem2	0	0	0	0



Eccetto per il picco di errore nella matrice spa1 con il metodo di Gauss-Seidel, possiamo dire che in generale gli errori relativi, a parità di tolleranza, sono comparabili.

Tempo di esecuzione (in millisecondi):

	Jacobi	Gauss-Seidel	Metodo del Gradiente	Metodo del Gradiente Coniugato
spa1	17,8235	3,83929	14,4706	13,6069
spa2	125,682	57,9014	2587,86	819,306
vem1	63,5653	103,194	49,0114	3,94746
vem2	126,631	221,972	126,574	10,0857



Anche dai tempi di esecuzione è possibile notare il netto miglioramento portato dal metodo del gradiente coniugato rispetto al metodo del gradiente.