



# Progetto 1

## Metodi del Calcolo Scientifico

Algebra lineare numerica

Sistemi lineari con matrici sparse simmetriche e definite positive

AA 2024/2025

*di*

⌚ Andrea Falbo

a.falbo7@campus.unimib.it

⌚ Ruben Tenderini

r.tenderini@campus.unimib.it

*Università degli Studi di Milano-Bicocca*

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Obiettivo . . . . .	1
1.2	Contesto applicativo . . . . .	1
1.3	Repository . . . . .	1
<b>2</b>	<b>Fondamenti Teorici</b>	<b>2</b>
2.1	Matrici sparse . . . . .	2
2.1.1	Definizione . . . . .	2
2.1.2	Tecniche di memorizzazione compatta . . . . .	2
2.1.3	Fill-In . . . . .	2
2.2	Metodo di Cholesky . . . . .	2
2.2.1	Applicabilità a matrici SPD . . . . .	2
2.2.2	Confronto con Gauss . . . . .	3
2.3	SuiteSparse Matrix Collection . . . . .	3
<b>3</b>	<b>Ambienti di Programmazione e Librerie</b>	<b>4</b>
3.1	MATLAB . . . . .	4
3.2	SciPy . . . . .	4
3.3	BLAS . . . . .	4
<b>4</b>	<b>Metodologia Sperimentale</b>	<b>5</b>
4.1	Configurazione dell'ambiente di test . . . . .	5
4.2	Matrici di test . . . . .	5
4.3	Protocollo di test . . . . .	6
4.3.1	Strumenti di misurazione . . . . .	6
<b>5</b>	<b>Risultati Sperimentali</b>	<b>8</b>
5.1	Risultati MATLAB . . . . .	8
5.1.1	Run su Windows . . . . .	8
5.1.2	Run su Linux . . . . .	10
5.2	Risultati SciPy . . . . .	12
5.2.1	Run su Windows . . . . .	12
5.2.2	Run su Linux . . . . .	14
<b>6</b>	<b>Analisi Comparativa</b>	<b>16</b>
6.1	Confronto prestazioni temporali . . . . .	16
6.2	Confronto accuratezza . . . . .	17
6.3	Confronto occupazione memoria . . . . .	17
<b>7</b>	<b>Conclusione e Sviluppi Futuri</b>	<b>19</b>
7.1	Sintesi dei risultati . . . . .	19
7.2	Raccomandazioni operative . . . . .	19
7.3	Limitazioni dello studio . . . . .	19

7.4 Sviluppi futuri . . . . .	20
<b>A Codice MATLAB</b>	<b>21</b>
A.1 runner.m . . . . .	21
A.2 run_experiment.m . . . . .	22
A.3 utils . . . . .	23
A.3.1 matrix_sort.m . . . . .	23
A.3.2 profile_memory.m . . . . .	24
A.3.3 system_info.m . . . . .	25
<b>B Codice SciPy</b>	<b>26</b>
B.1 main.py . . . . .	26
B.2 utils . . . . .	26
B.2.1 runner.py . . . . .	26
B.2.2 functions.py . . . . .	29
B.2.3 CSVLogger.py . . . . .	29
<b>C Dati completi</b>	<b>31</b>
C.1 MATLAB . . . . .	31
C.1.1 Windows . . . . .	31
C.1.2 Linux . . . . .	32
C.2 SciPy . . . . .	33
C.2.1 Windows . . . . .	33
C.2.2 Linux . . . . .	34

# Capitolo 1

## Introduzione

### 1.1 Obiettivo

Lo scopo del seguente progetto è di studiare l'implementazione in **ambienti di programmazione** open source del **metodo di Cholesky** per la risoluzione sistemi lineari per *matrici sparse, simmetriche e definite positive* (SPD), e di confrontarli con l'implementazione di **MATLAB**. Nel nostro caso specifico, la libreria open source selezionata è **SciPy**.

Il confronto si basa su tre criteri principali: il **tempo di esecuzione** necessario per risolvere i sistemi, l'**accuratezza** delle soluzioni ottenute e l'**uso della memoria** durante il calcolo.

### 1.2 Contesto applicativo

Le *matrici sparse* sono estremamente comuni nel calcolo scientifico e in numerose applicazioni ingegneristiche e scientifiche. Si definiscono sparse quelle matrici caratterizzate da pochi elementi diversi da zero distribuiti in modo tale che la memorizzazione e il calcolo possano essere ottimizzati.

Per la risoluzione di sistemi lineari con matrici SPD, il **metodo di Cholesky** rappresenta una tecnica particolarmente **efficace** e **stabile**, in grado di sfruttare la struttura della matrice per ridurre il costo computazionale e l'uso della memoria rispetto a metodi più generici come l'eliminazione di Gauss.

### 1.3 Repository

Per completezza e trasparenza, l'intero set di dati, i codici sorgente utilizzati per la generazione dei benchmark, e gli script per l'analisi e la visualizzazione dei risultati sono disponibili pubblicamente nella seguente repository GitHub:

<https://github.com/LilQuacky/choleski-sparse-benchmark>

La repository contiene:

- Codice Python e MATLAB per l'esecuzione dei test.
- I dati CSV con i risultati raccolti.
- Script per la generazione dei grafici e delle tabelle presenti in questo elaborato.

# Capitolo 2

## Fondamenti Teorici

### 2.1 Matrici sparse

#### 2.1.1 Definizione

Come già anticipato nella Sezione 1.2, le *matrici sparse* sono matrici in cui la maggior parte degli elementi è uguale a zero. Questo tipo di matrici è frequente in numerose applicazioni del calcolo scientifico e ingegneristico, dove le dimensioni possono essere estremamente grandi, ma solo pochi elementi sono significativi. La presenza di molti zeri consente di ottimizzare sia la **memorizzazione** sia i **calcoli**, evitando di allocare spazio o effettuare operazioni su elementi nulli.

#### 2.1.2 Tecniche di memorizzazione compatta

Per gestire efficacemente le matrici sparse si utilizzano **formati di memorizzazione compatta** che salvano esclusivamente gli elementi diversi da zero e le loro posizioni nella matrice. Tra i formati più comuni troviamo il *Compressed Sparse Row* (CSR), il *Compressed Sparse Column* (CSC) e il *Coordinate List* (COO).

#### 2.1.3 Fill-In

L'eliminazione di Gauss applicata a matrici sparse può generare nuovi elementi non nulli nella matrice triangolare risultante, fenomeno noto come **fill-in**. Questo comporta un aumento significativo della memoria necessaria e del costo computazionale, annullando i vantaggi delle matrici sparse. Per questo motivo, senza opportune strategie, il metodo di Gauss diventa spesso *inapplicabile* per matrici di grandi dimensioni sparse.

### 2.2 Metodo di Cholesky

#### 2.2.1 Applicabilità a matrici SPD

Il **metodo di Cholesky** è un algoritmo diretto per la risoluzione di sistemi lineari con *matrici SPD*. Tale tecnica fattorizza la matrice  $A$  nella forma  $A = LL^T$  (dove  $L$  è una matrice triangolare inferiore) sfruttandola simmetria e le proprietà di positività per garantire stabilità numerica e un'efficiente risoluzione del sistema.

Per contenere il problema del fill-in anche nel metodo di Cholesky, è fondamentale applicare **permutazioni preliminari** alle righe e colonne della matrice.

### 2.2.2 Confronto con Gauss

Il metodo di Cholesky, rispetto all'eliminazione di Gauss, risulta più efficiente, ma può essere utilizzato solo quando la matrice del sistema è SPD. In questi casi, offre prestazioni superiori sia in termini di velocità che di uso della memoria, poiché sfrutta la simmetria della matrice. Tuttavia, il suo principale svantaggio è proprio la limitata applicabilità: se la matrice non è SPD, il metodo di Cholesky non può essere usato.

## 2.3 SuiteSparse Matrix Collection

La **SuiteSparse Matrix Collection** è una raccolta di matrici sparse derivanti da problemi reali in diversi campi, quali ingegneria strutturale, fluidodinamica, elettromagnetismo, e grafi. Questa collezione è ampiamente utilizzata come benchmark per lo sviluppo e la valutazione di algoritmi numerici su matrici sparse.

Maggiori dettagli sulle matrici utilizzate per il seguente confronto sono forniti alla Sezione 4.2.

# Capitolo 3

## Ambienti di Programmazione e Librerie

### 3.1 MATLAB

MATLAB è un linguaggio di programmazione ad alto livello e un ambiente interattivo sviluppato da MathWorks. È utilizzato principalmente per il calcolo numerico, l'analisi dei dati, la simulazione e la visualizzazione grafica.

### 3.2 SciPy

SciPy è una libreria open-source per Python che fornisce strumenti avanzati per il calcolo scientifico, come l'algebra lineare, l'ottimizzazione, l'integrazione numerica e l'elaborazione dei segnali. Scikit-sparse è una libreria complementare specializzata nella gestione efficiente di matrici sparse, e fornisce interfacce a librerie esterne ad alte prestazioni, tra cui CHOLMOD una libreria efficiente basata su C per la fattorizzazione di Cholesky. L'integrazione tra scikit-sparse e CHOLMOD consente di risolvere sistemi lineari di grandi dimensioni in modo molto più efficiente rispetto alle routine standard di SciPy.

### 3.3 BLAS

BLAS (Basic Linear Algebra Subprograms) è una libreria di basso livello ottimizzata per operazioni fondamentali di algebra lineare, come prodotti tra vettori e matrici.

Nel caso di **MATLAB**, l'utilizzo di BLAS è gestito in modo trasparente dal sistema, che rileva automaticamente le capacità multicore del processore e le sfrutta senza interventi esterni.

Nel caso di **SciPy**, invece, è stato necessario importare la libreria **OpenBLAS** e configurare l'ambiente affinché abilitasse il supporto multicore.

# Capitolo 4

## Metodologia Sperimentale

### 4.1 Configurazione dell'ambiente di test

I test sono stati eseguiti su una macchina che ospita come sistema operativo Windows 11 e con le seguenti caratteristiche hardware:

- **Processore:** Intel Core i7-10700F, 8 core, 16 thread
- **Memoria RAM:** 32 GB DDR4 3600 MHz

Tale dispositivo ha ospitato il sistema operativo Linux tramite una macchina virtuale con distribuzione Ubuntu 24.04.2 e configurata con i limiti permessi dal software della VM, ovverosia:

- **RAM allocata:** 24 GB
- **Core allocati:** 8

Le versioni del software utilizzato sono:

- **MATLAB R2024.2.0**
- **Python 3.11.11** con librerie SciPy 1.15.2 e scikit-sparse 0.4.14

### 4.2 Matrici di test

Per la valutazione delle prestazioni sono state utilizzate matrici sparse SPD tratte dalla **SuiteSparse Matrix Collection**. Di seguito una descrizione dettagliata:

- **Flan 1565:** matrice derivata da un problema strutturale tridimensionale che discretizza una flangia d'acciaio con elementi finiti esaedrici.
- **StocF-1465:** matrice proveniente da un problema di flusso in mezzi porosi con griglia tetraedrica, rappresentante un acquifero stocastico.
- **cfld2:** matrice simmetrica rappresentante una matrice di pressione derivata da un problema di CFD (Computational Fluid Dynamics).
- **cfld1:** analoga a *cfld2*, questa matrice è anch'essa simmetrica e proviene da un problema CFD.
- **G3\_circuit:** matrice derivante da simulazione circuitale complessa.
- **parabolic\_fem:** matrice da problema di equazioni paraboliche alle derivate parziali, con diffusione costante.

- **apache2**: matrice strutturale simmetrica da simulazioni di strutture ingegneristiche complesse.
- **shallow\_water1**: matrice relativa alle equazioni delle acque basse.
- **ex15**: matrice derivante da un problema di CFD, utilizzata per la simulazione del flusso nei fluidi compressibili.

Le caratteristiche dimensionali delle matrici sono riassunte nella seguente tabella:

Name	Group	Num Rows	Nonzeros	Pattern Entries
Flan_1565	Janna	1,564,794	114,165,372	117,406,044
StocF-1465	Janna	1,465,137	21,005,389	21,005,389
cf2	Rothberg	123,440	3,085,406	3,087,898
cf1	Rothberg	70,656	1,825,580	1,828,364
G3_circuit	AMD	1,585,478	7,660,826	7,660,826
parabolic_fem	Wissgott	525,825	3,674,625	3,674,625
apache2	GHS_psdef	715,176	4,817,870	4,817,870
shallow_water1	MaxPlanck	81,920	327,680	327,680
ex15	FIDAP	6,867	98,671	98,671

**Tabella 4.1:** Dettagli dimensionali delle matrici selezionate dalla SuiteSparse Matrix Collection

## 4.3 Protocollo di test

In ciascuno dei due ambienti di programmazione, abbiamo utilizzato un solutore diretto per matrici sparse SPD risolvendo il sistema lineare  $Ax = b$  dove il termine  $b$  è scelto in modo che la soluzione esatta sia il vettore  $x_e = [1 \ 1 \ \cdots \ 1]$ , cioè  $b = Ax_e$ .

Le metriche valutate sono:

- **Tempo di calcolo**: misurato come il tempo totale impiegato per la fattorizzazione e la risoluzione del sistema;
- **Errore relativo**: definito come

$$\text{errore relativo} = \frac{\|x - x_e\|_2}{\|x_e\|_2}$$

dove  $x$  è la soluzione calcolata;

- **Occupazione di memoria**: valutata come incremento approssimativo della memoria occupata dal programma tra il caricamento della matrice e il termine della risoluzione.

### 4.3.1 Strumenti di misurazione

Le misurazioni delle prestazioni, in particolare il tempo di esecuzione, l'occupazione di memoria e l'errore relativo, sono state eseguite direttamente all'interno del codice, sia per l'ambiente **MATLAB** che per **SciPy**:

- **Tempo di esecuzione:** misurato usando il modulo `time` in Python (funzione `time.perf_counter()`) e la funzione nativa `tic/toc` in MATLAB.
- **Memoria utilizzata:** stimata in Python tramite la libreria `psutil`, usando `memory_info().rss` per misurare la memoria residente. In MATLAB è stato utilizzato l'output dello strumento di `profiling` integrato.
- **Errore relativo:** calcolato come norma euclidea del vettore differenza tra la soluzione calcolata  $x$  e quella esatta  $x_e$ , normalizzata su  $x_e$ .

I risultati sono stati esportati in file CSV attraverso procedure automatizzate in entrambi gli ambienti. In Python è stato utilizzato il modulo `csv`, mentre in MATLAB è stata utilizzata la funzione nativa `writetable`.

# Capitolo 5

## Risultati Sperimentali

In questo capitolo vengono presentati i risultati sperimentali ottenuti dall'esecuzione su MATLAB e SciPy nella risoluzione di sistemi lineari con matrici sparse SPD. Le prove sono state condotte a parità di macchina sia su Windows che su Linux, anche se quest'ultimo aveva 8 GB di RAM in meno rispetto a Windows, come già descritto nella Sezione 4.1.

Per ogni combinazione di ambiente software e sistema operativo, sono riportati i grafici con scala delle ordinate logaritmica per permettere un confronto tra la **dimensione delle matrici** e:

- il **tempo** necessario per calcolare la soluzione;
- l'**errore relativo** tra soluzione calcolata e soluzione esatta;
- la **memoria** necessaria per la fase di decomposizione.

Per ogni combinazione di sistema operativo e ambiente software, è stato eseguito il test **5 volte** consecutivamente. In questo capitolo verranno mostrate, per ogni metrica, le 5 esecuzioni.

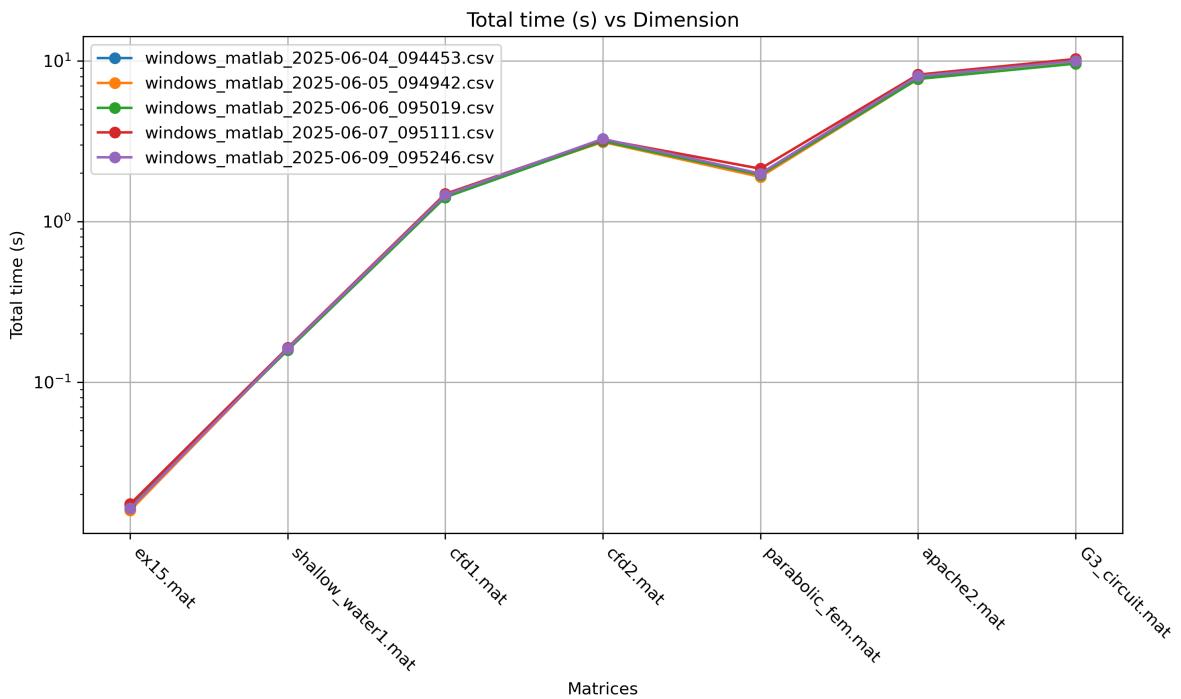
Successivamente sono state calcolate le **medie** dei valori ottenuti per tempo di esecuzione, errore relativo e memoria utilizzata per ridurre il rumore sperimentale dovuto a fluttuazioni transitorie del sistema operativo o del carico computazionale. Tali medie verranno mostrate, invece, nel capitolo successivo.

### 5.1 Risultati MATLAB

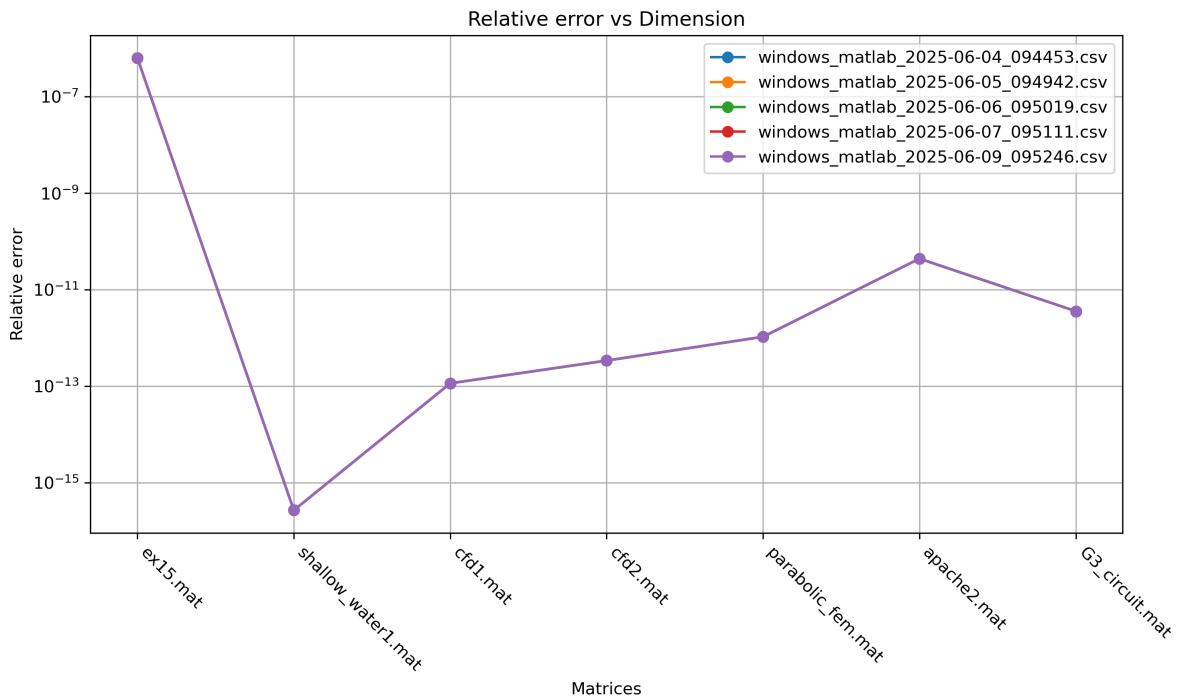
#### 5.1.1 Run su Windows

I risultati ottenuti con MATLAB su Windows mostrano come il solutore integrato offra un comportamento corretto e prevedibile: tempi di esecuzione, occupazione di memoria ed errore numerico crescono in modo proporzionale con l'aumentare della dimensione delle matrici.

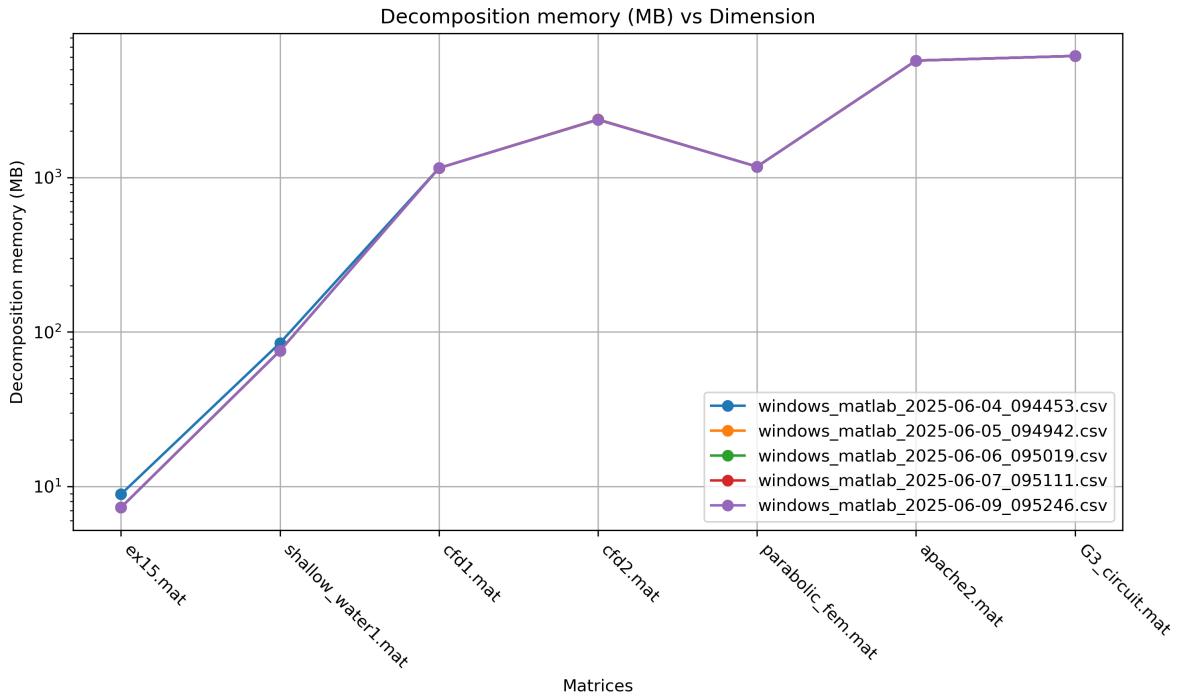
Tuttavia, è importante segnalare che le due matrici più complesse analizzate **Flan 1565** e **StocF-1465** non sono state risolte da MATLAB, in quanto non è riuscito a completare il procedimento entro il tempo limite previsto dal sistema stesso.



**Figura 5.1:** Tempo di risoluzione delle run eseguite in ambiente Windows MATLAB



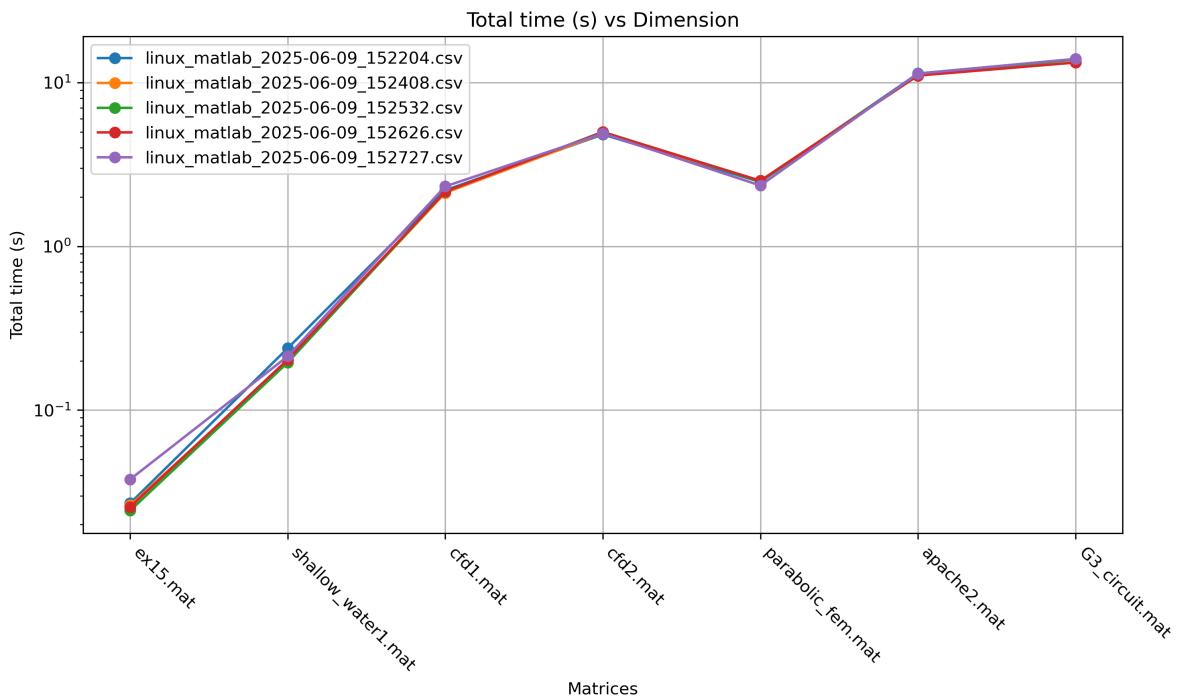
**Figura 5.2:** Errore relativo delle run eseguite in ambiente Windows MATLAB



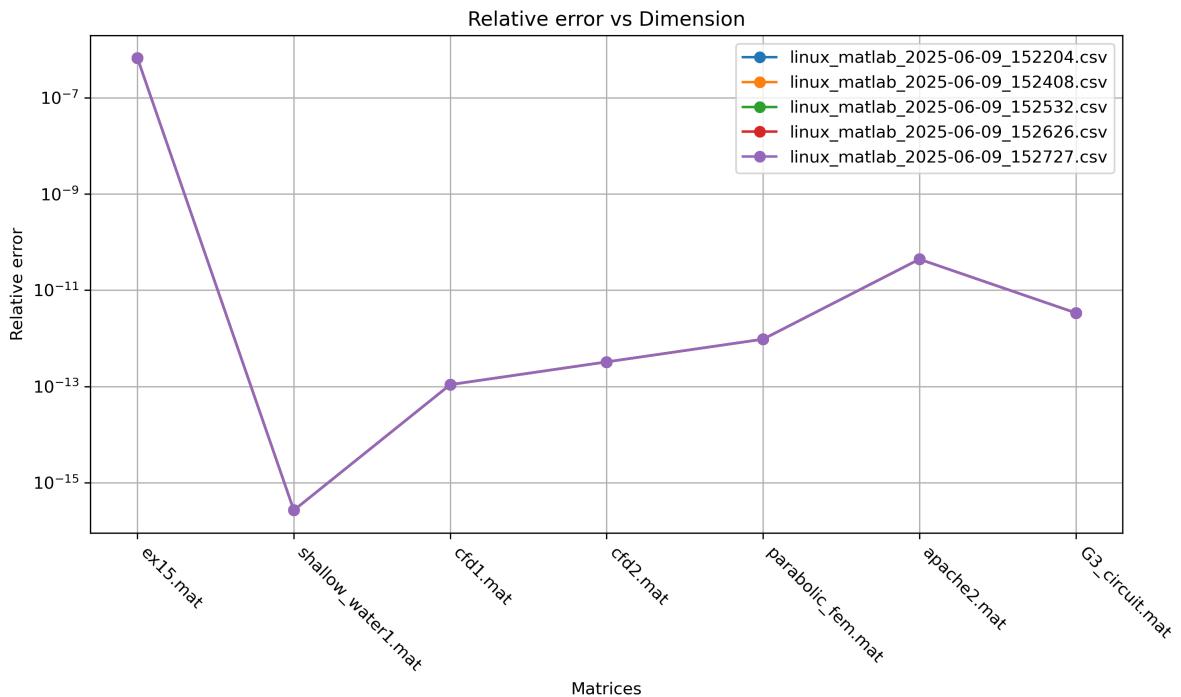
**Figura 5.3:** Memoria usata nella fase di decomposizione delle run eseguite in ambiente Windows MATLAB

### 5.1.2 Run su Linux

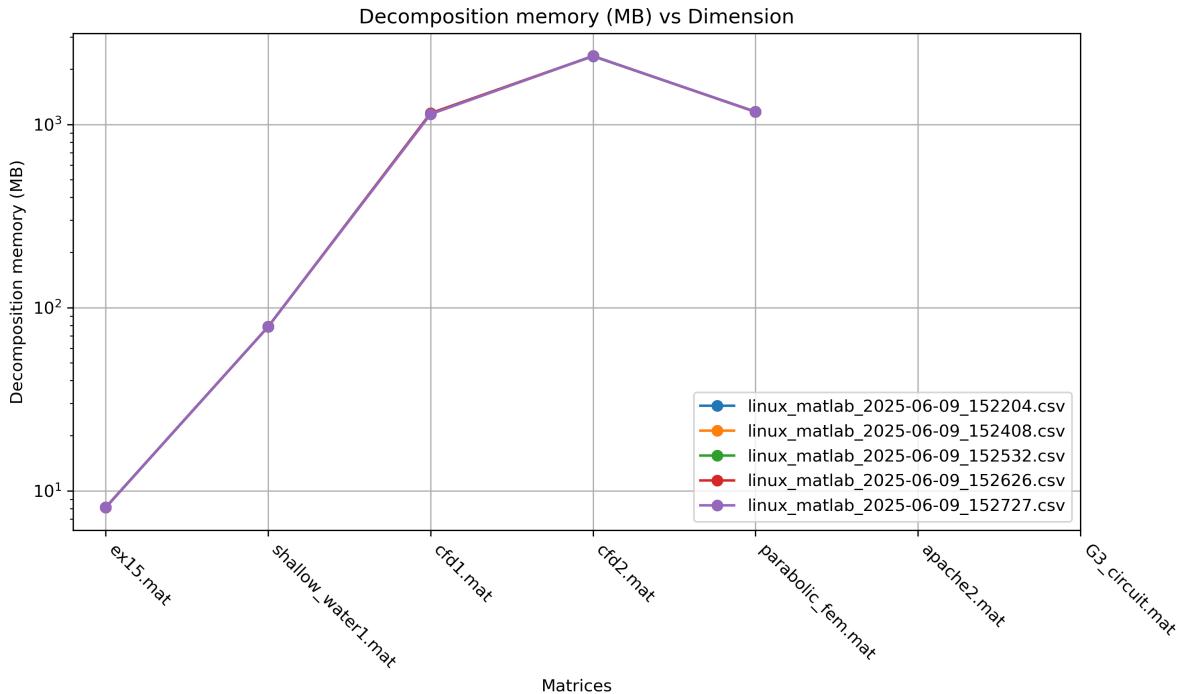
Su Linux, MATLAB presenta performance comparabili a quelle osservate su Windows, confermando le limitazioni mostrate per le matrici di grande dimensione. Tuttavia, per **G3\_circuit** e **apache2**, la funzione per il calcolo della memoria utilizzata ha restituito un valore di overflow, probabilmente a causa degli 8 GB di RAM in meno disponibili sul sistema Linux rispetto a Windows.



**Figura 5.4:** Tempo di risoluzione delle run eseguite in ambiente Linux MATLAB



**Figura 5.5:** Errore relativo delle run eseguite in ambiente Linux MATLAB



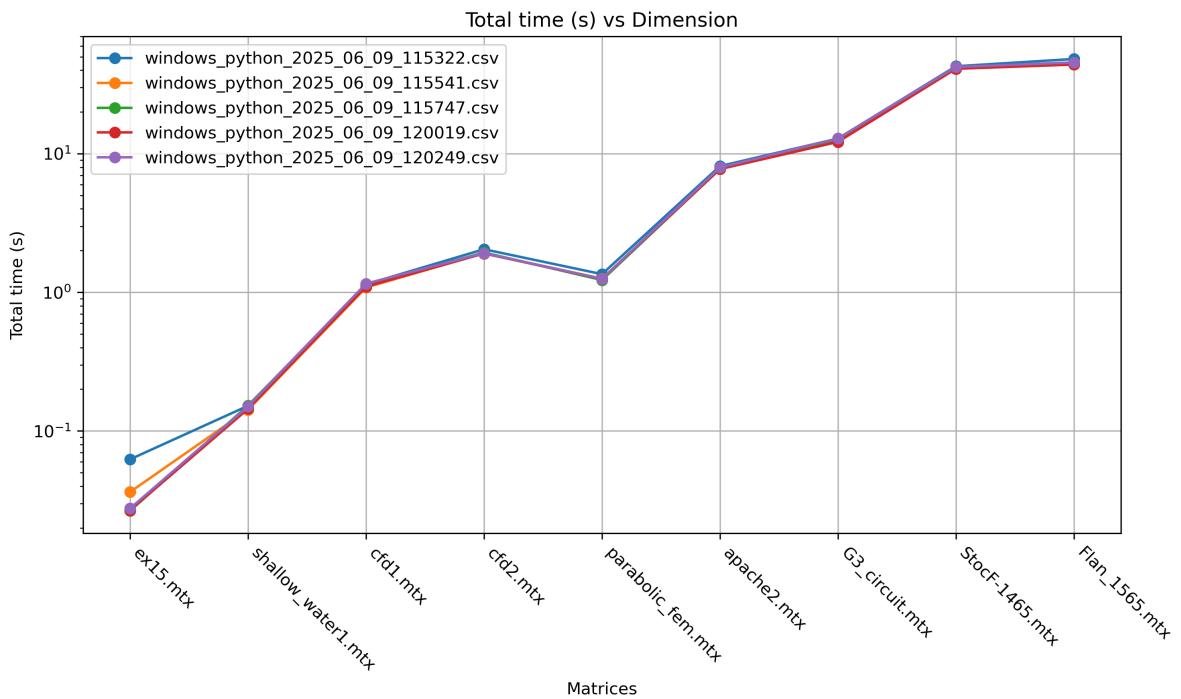
**Figura 5.6:** Memoria usata nella fase di decomposizione delle run eseguite in ambiente Linux MATLAB

## 5.2 Risultati SciPy

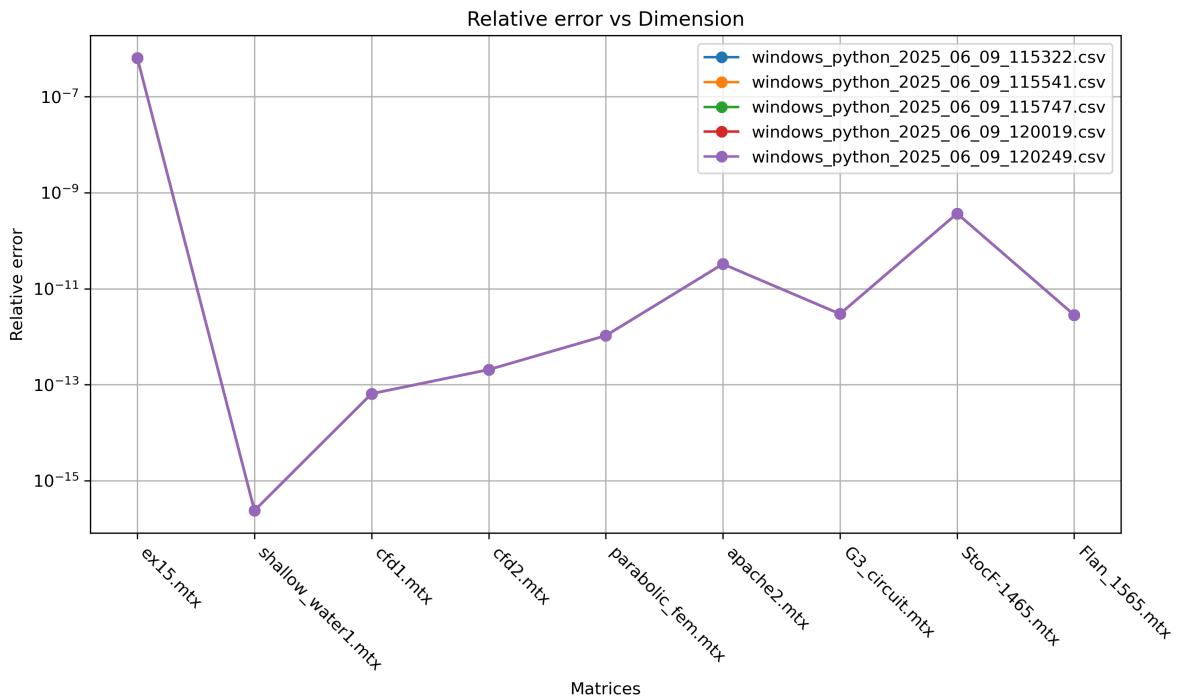
### 5.2.1 Run su Windows

Utilizzando SciPy su Windows, si evidenziano risultati prevedibili e corretti, così come avveniva su MATLAB: tempi di esecuzione, occupazione di memoria ed errore relativo crescono in modo proporzionale con l'aumentare della dimensione delle matrici.

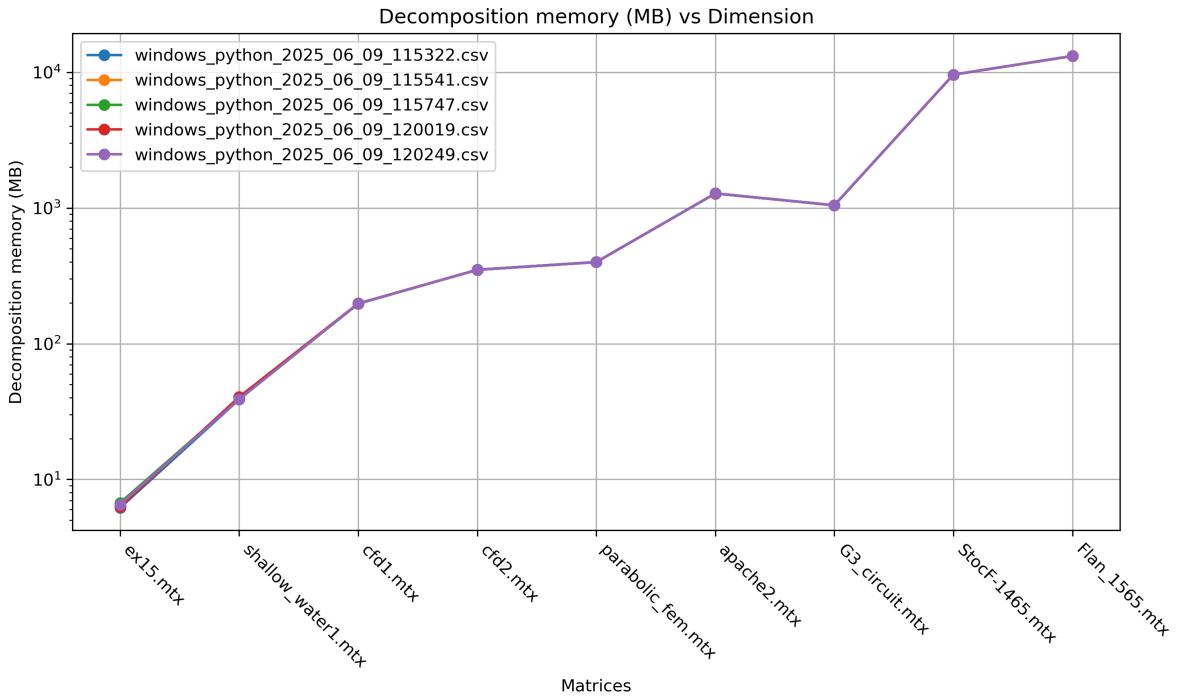
La principale differenza è data dalla capacità del solutore open-source di completare le due matrici più pesanti: **Flan 1565** e **StocF-1465**.



**Figura 5.7:** Tempo di risoluzione delle run eseguite in ambiente Windows SciPy



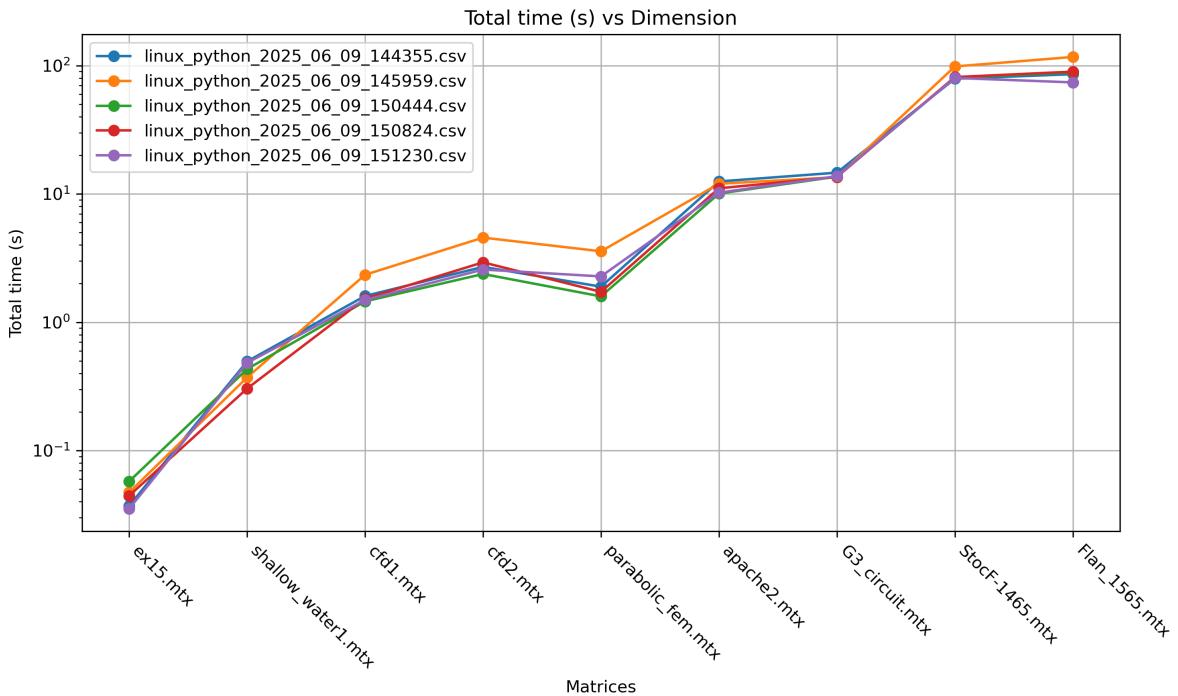
**Figura 5.8:** Errore relativo delle run eseguite in ambiente Windows SciPy



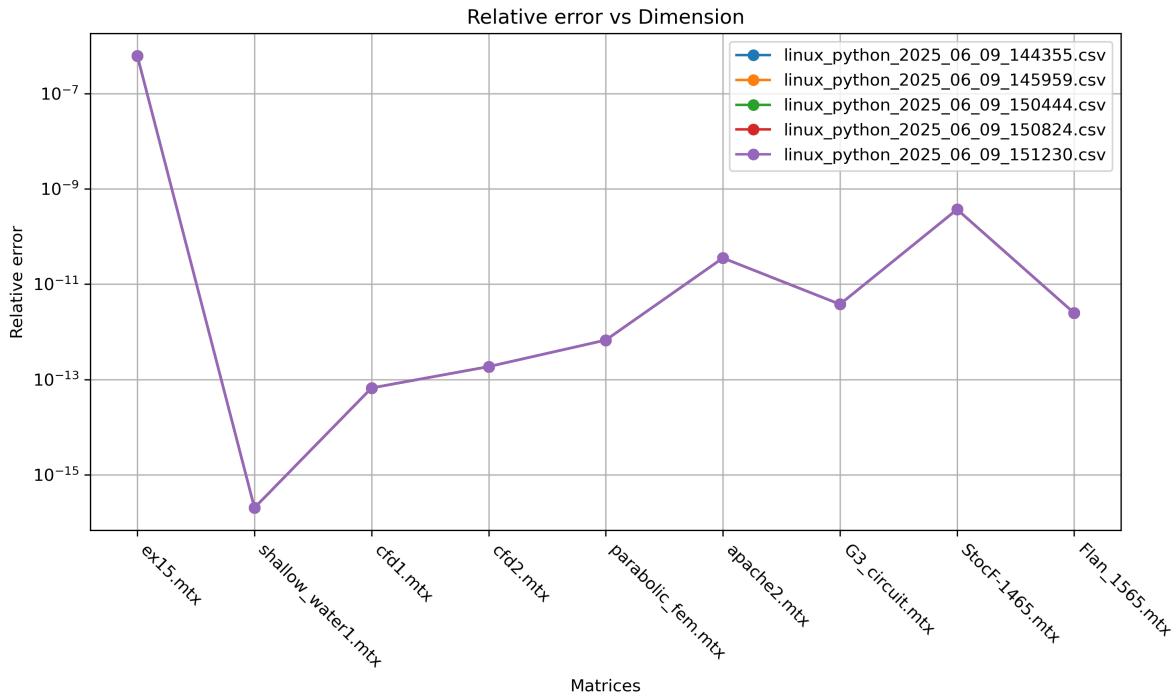
**Figura 5.9:** Memoria usata nella fase di decomposizione delle run eseguite in ambiente Windows SciPy

### 5.2.2 Run su Linux

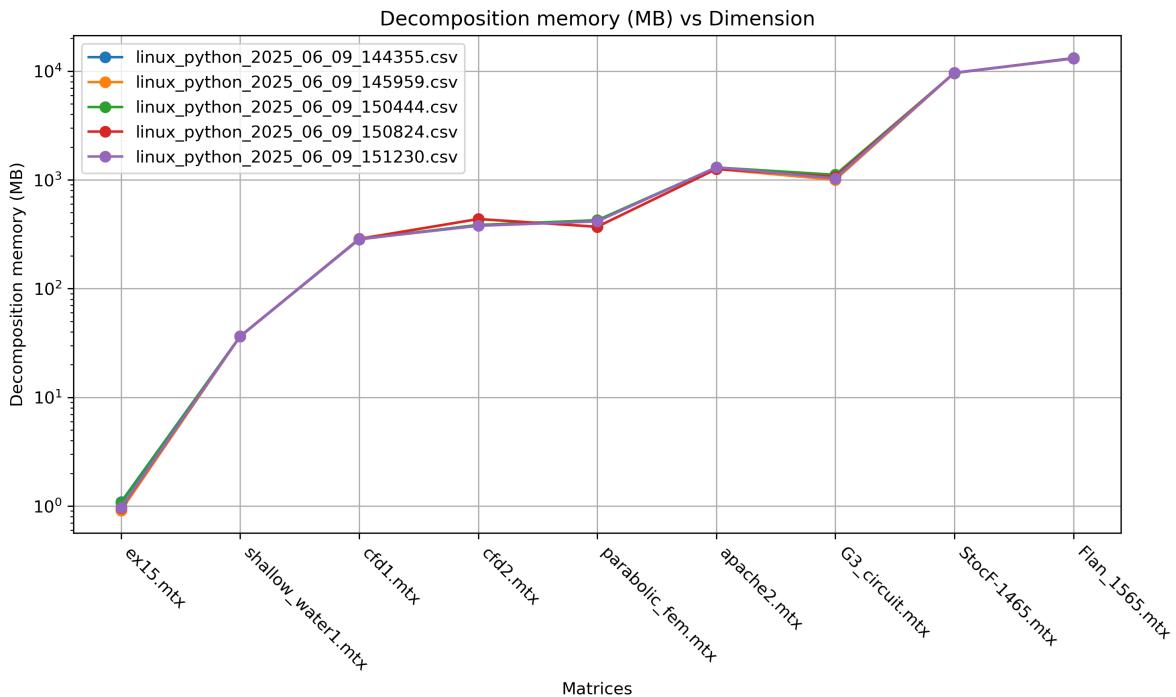
Anche su Linux, la libreria open source riesce a risolvere tutte le matrici proposte con risultati nella norma. Non sono stati identificati problemi legati alla riduzione della memoria RAM allocata, a differenza di quanto è avvenuto con MATLAB su Linux.



**Figura 5.10:** Tempo di risoluzione delle run eseguite in ambiente Linux SciPy



**Figura 5.11:** Errore relativo delle run eseguite in ambiente Linux SciPy



**Figura 5.12:** Memoria usata nella fase di decomposizione delle run eseguite in ambiente Linux SciPy

# Capitolo 6

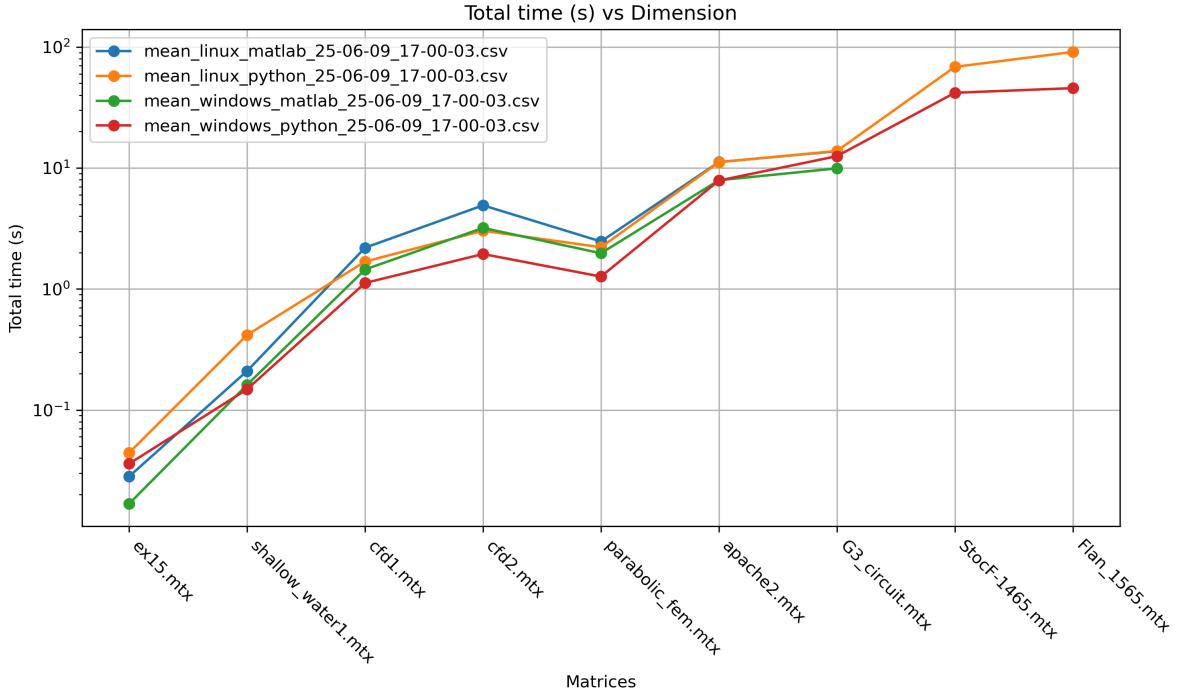
## Analisi Comparativa

In questo capitolo vengono confrontati i risultati ottenuti con MATLAB e SciPy per la risoluzione di sistemi lineari con matrici sparse SPD. I grafici riportati in ciascuna sezione mostrano, in funzione della dimensione della matrice, le performance dei due ambienti su entrambi i sistemi operativi analizzati.

### 6.1 Confronto prestazioni temporali

Dall'analisi dei tempi di esecuzione si osserva che le prestazioni tra i vari ambienti (MATLAB e SciPy) sono pressoché equivalenti, a parità di sistema operativo.

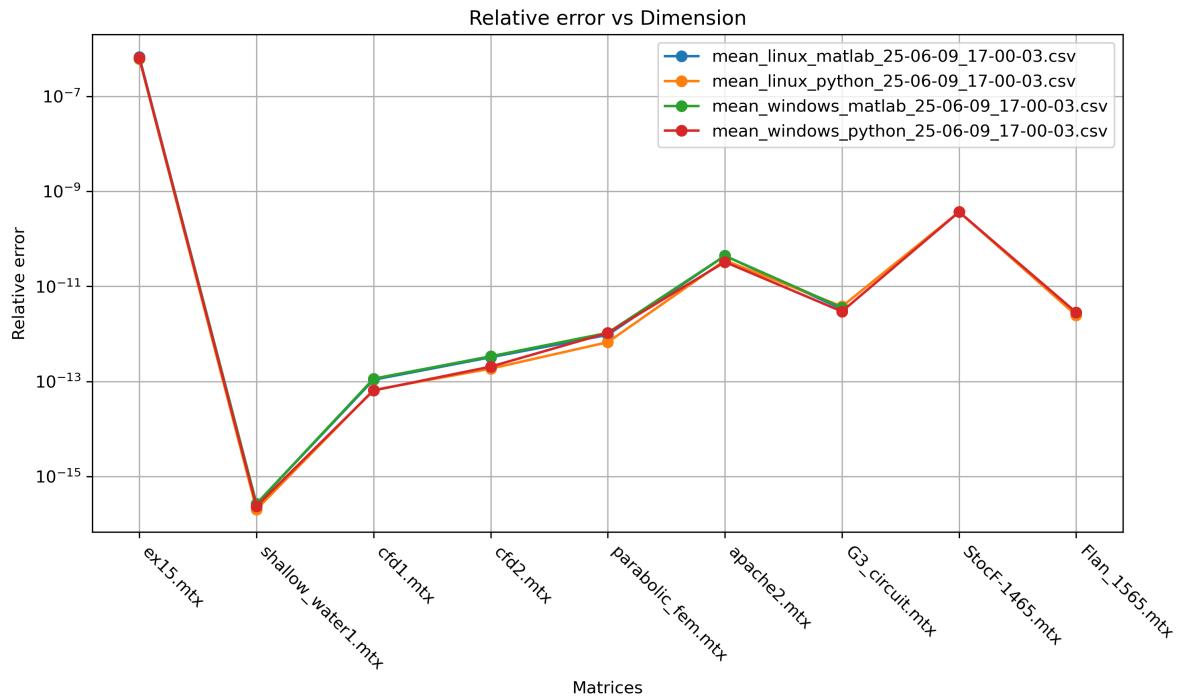
A parità di ambiente software, invece, le due esecuzioni su Windows offrono rispettivamente tempi minori rispetto alle controparti di Linux. Questo effetto è presumibilmente generato dalla configurazione virtualizzata di quest'ultimo, che potrebbe introdurre un overhead nella gestione delle risorse hardware.



**Figura 6.1:** Tempo medio di risoluzione in funzione della dimensione del sistema, per ciascuna combinazione di sistema operativo e ambiente software.

## 6.2 Confronto accuratezza

Tutti i solutori testati presentano un errore relativo numerico contenuto e sostanzialmente comparabile. In generale, SciPy ha mostrato una leggera superiorità rispetto a MATLAB in termini di accuratezza, in particolare sui test condotti con le matrici **cf2d**, **cf1d** e **apache2** dove l'errore risulta visibilmente più basso. Questo conferma che il solutore open source è in grado di offrire prestazioni numeriche pienamente equiparabili, e in alcuni casi leggermente superiori, rispetto alla soluzione proprietaria di MATLAB.

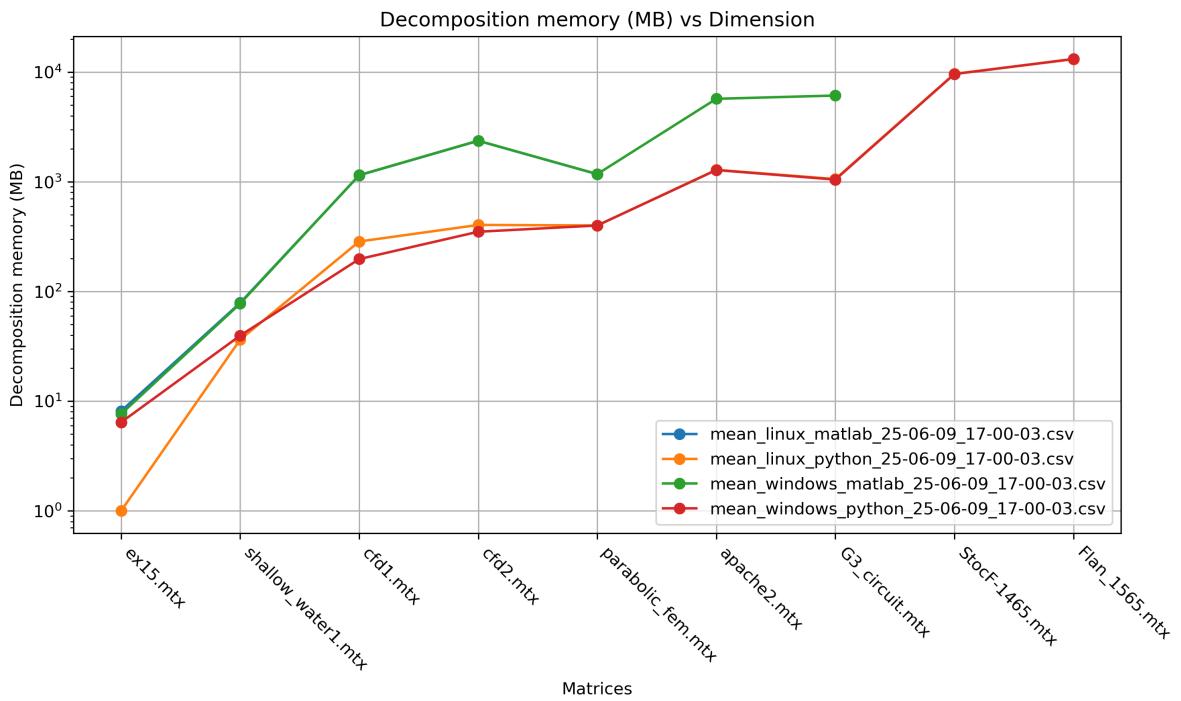


**Figura 6.2:** Errore relativo medio in funzione della dimensione del sistema, per ciascuna combinazione di sistema operativo e ambiente software.

## 6.3 Confronto occupazione memoria

Anche in termini di uso della memoria, SciPy si è dimostrato in generale più efficiente di MATLAB, con un'occupazione inferiore soprattutto per matrici di grandi dimensioni.

Tuttavia, si osserva che, a parità di ambiente software, le differenze tra Windows e Linux in termini di utilizzo della memoria sono contenute, suggerendo che il principale collo di bottiglia è l'overhead introdotto dall'esecuzione su macchina virtuale.



**Figura 6.3:** Memoria media utilizzata durante la risoluzione, per ciascuna combinazione di sistema operativo e ambiente software.

# Capitolo 7

## Conclusione e Sviluppi Futuri

### 7.1 Sintesi dei risultati

L’analisi comparativa tra MATLAB e SciPy ha evidenziato diversi elementi di interesse. In termini di **prestazioni temporali**, SciPy ha mostrato un comportamento più efficiente, soprattutto all’aumentare della dimensione delle matrici, grazie alla maggiore scalabilità del solutore. Ciò è particolarmente evidente per le matrici più complesse come **Flan 1565** e **StocF-1465**, che MATLAB non è stato in grado di risolvere entro i limiti temporali, mentre SciPy ha completato correttamente l’elaborazione.

Per quanto riguarda l’**accuratezza numerica**, tutti i solutori hanno prodotto risultati stabili e comparabili. SciPy ha mostrato un leggero vantaggio in termini di errore relativo, in particolare su matrici come **cf2d**, **cf1d** e **apache2**, suggerendo che anche dal punto di vista della precisione l’approccio open source sia assolutamente affidabile.

Dal punto di vista del **consumo di memoria**, SciPy si è dimostrato sistematicamente più efficiente di MATLAB, indipendentemente dalla dimensione delle matrici e dal sistema operativo.

### 7.2 Raccomandazioni operative

Alla luce dei risultati ottenuti, la libreria open source SciPy, in combinazione con scikitsparse/CHOLMOD, ha dimostrato prestazioni superiori a MATLAB sia in termini di tempo di calcolo che di utilizzo della memoria, mantenendo un’accuratezza comparabile o leggermente migliore. Per questo motivo, risulta essere la soluzione consigliata nella maggior parte degli scenari applicativi, soprattutto quando si lavora con matrici di grandi dimensioni o si ricerca una maggiore scalabilità.

Il confronto tra **Windows** e **Linux** non ha evidenziato differenze strutturali significative a parità di ambiente software, fatta eccezione per un leggero vantaggio di Windows nei test svolti, che però potrebbe essere attribuito alla natura virtualizzata dell’ambiente Linux.

Pertanto, la scelta tra i due sistemi operativi rimane **aperta** e dovrebbe essere valutata in funzione del contesto specifico, della disponibilità di risorse hardware native e delle preferenze dell’utente o del team di sviluppo.

### 7.3 Limitazioni dello studio

L’analisi presentata è soggetta ad alcune limitazioni:

- È stato utilizzato un numero limitato di matrici sparse, seppur significative, tratte dalla SuiteSparse Matrix Collection.
- I test sono stati condotti su un'unica configurazione hardware.
- Il sistema operativo Linux è stato eseguito tramite macchina virtuale.

## 7.4 Sviluppi futuri

Il presente studio può essere ampliato e approfondito in diverse direzioni:

- Valutare le prestazioni su diverse architetture hardware e sistemi operativi.
- Considerare l'impiego di diversi linguaggi di programmazione e librerie, oltre a SciPy.
- Integrare lo studio con ulteriori metriche di valutazione, quali tempi di pre-processing, robustezza numerica su sistemi mal condizionati, e consumo energetico.
- Estendere l'analisi a differenti algoritmi risolutivi, non limitandosi alla decomposizione di Cholesky.

# Appendice A

## Codice MATLAB

### A.1 runner.m

```
1 function runner()
2 % runner - Executes experiments on matrices sorted by size and saves
3 % results
4 %
5 % This function scans the matrices folder, sorts the matrix files by
6 % size,
7 % runs the experiment on each matrix, and saves the results to a CSV
8 % file
9 % in the runs folder.
10 %
11 % No inputs.
12 % No outputs.
13
14 addpath(fullfile(pwd, 'utils'));
15 osName = system_info();
16
17 matrixFolder = 'matrices';
18 resultsFolder = 'runs';
19
20 if ~exist(resultsFolder, 'dir')
21     mkdir(resultsFolder);
22 end
23
24 timestamp = datestr(now, 'yyyy-mm-dd_HHMMSS');
25 csvFileName = sprintf('%s_matlab_%s.csv', osName, timestamp);
26 outputCSV = fullfile(resultsFolder, csvFileName);
27
28 matFiles = dir(fullfile(matrixFolder, '*.mat'));
29 fprintf('Found %d matrix files.\n', length(matFiles));
30
31 fileList = fullfile(matrixFolder, {matFiles.name});
32
33 % Sort matrix files by size
34 sortedFiles = matrix_sort(fileList);
35
36 results = repmat(struct(), length(sortedFiles), 1); % Temporary
            prealloc
37
38 for k = 1:length(sortedFiles)
39     matPath = sortedFiles{k};
```

```

37     fprintf('Processing %s...\n', matPath);
38
39     if usejava('jvm')
40         java.lang.System.gc();
41     end
42
43     result = run_experiment(matPath);
44
45     if k == 1
46         % Initialize struct array with the same fields as the
47         % first result
48         results = repmat(result, length(sortedFiles), 1);
49     end
50
51     results(k) = result;
52 end
53
54 fprintf('All experiments completed.\n');
55
56 resultsTable = struct2table(results);
57 writetable(resultsTable, outputCSV);
58 fprintf('Results saved to %s\n', outputCSV);
end

```

## A.2 run\_experiment.m

```

1 function runner()
2 % runner - Executes experiments on matrices sorted by size and saves
3 % results
4 %
5 % This function scans the matrices folder, sorts the matrix files by
6 % size,
7 % runs the experiment on each matrix, and saves the results to a CSV
8 % file
9 % in the runs folder.
10 %
11 % No inputs.
12 % No outputs.
13
14 addpath(fullfile(pwd, 'utils'));
15 osName = system_info();
16
17 matrixFolder = 'matrices';
18 resultsFolder = 'runs';
19
20 if ~exist(resultsFolder, 'dir')
21     mkdir(resultsFolder);
22 end
23
24 timestamp = datestr(now, 'yyyy-mm-dd_HHMMSS');

```

```

22 csvFileName = sprintf('%s_matlab_%s.csv', osName, timestamp);
23 outputCSV = fullfile(resultsFolder, csvFileName);
24
25 matFiles = dir(fullfile(matrixFolder, '*.mat'));
26 fprintf('Found %d matrix files.\n', length(matFiles));
27
28 fileList = fullfile(matrixFolder, {matFiles.name});
29
30 % Sort matrix files by size
31 sortedFiles = matrix_sort(fileList);
32
33 results = repmat(struct(), length(sortedFiles), 1); % Temporary
34 % prealloc
35
36 for k = 1:length(sortedFiles)
37     matPath = sortedFiles{k};
38     fprintf('Processing %s...\n', matPath);
39
40     if usejava('jvm')
41         java.lang.System.gc();
42     end
43
44     result = run_experiment(matPath);
45
46     if k == 1
47         % Initialize struct array with the same fields as the
48         % first result
49         results = repmat(result, length(sortedFiles), 1);
50     end
51
52     results(k) = result;
53 end
54
55 fprintf('All experiments completed.\n');
56
57 resultsTable = struct2table(results);
58 writetable(resultsTable, outputCSV);
59 fprintf('Results saved to %s\n', outputCSV);
60
61 end

```

## A.3 utils

### A.3.1 matrix\_sort.m

```

1 function sortedFiles = matrix_sort(fileList)
2 % matrix_sort - Sorts .mat files by the number of non-zero elements
3 % in Problem.A
4 %
5 % INPUT:
6 %   fileList - Cell array of file paths (strings)

```

```

6 %
7 % OUTPUT:
8 %     sortedFiles - Cell array sorted by increasing number of non-zero
9 %     entries in Problem.A
10
11 n = numel(fileList);
12 nnzCounts = zeros(n, 1);
13
14 for i = 1:n
15     try
16         S = load(fileList{i});
17         if isfield(S, 'Problem') && isfield(S.Problem, 'A')
18             A = S.Problem.A;
19             if ~issparse(A)
20                 A = sparse(A);
21             end
22             A = double(A);
23             nnzCounts(i) = nnz(A);
24         else
25             error('Matrix A not found in file %s.', fileList{i})
26             ;
27         end
28     catch ME
29         warning('Error reading %s: %s', fileList{i}, ME.message)
30         ;
31         nnzCounts(i) = inf; % Mette in fondo i file
32         problematici
33     end
34 end
35
36 [~, idx] = sort(nnzCounts, 'ascend');
37 sortedFiles = fileList(idx);
38
39 end

```

### A.3.2 profile\_memory.m

```

1 function memUsedMB = profile_memory(profileInfo)
2 % profile_memory - Calculates memory usage in MB from profile info
3 %
4 % INPUT:
5 %     profileInfo - Output structure from MATLAB profiler (profile('
6 %         info'))
7 %
8 % OUTPUT:
9 %     memUsedMB - Memory used in megabytes
10
11 memAllocated = 0;
12 memFreed = 0;
13 for i = 1:length(profileInfo.FunctionTable)
14     memAllocated = memAllocated + profileInfo.FunctionTable(i).
15         TotalMemAllocated;

```

```

14     memFreed = memFreed + profileInfo.FunctionTable(i).
15         TotalMemFreed;
16 end
17 memUsedMB = max(0, (memAllocated - memFreed) / 1e6);

```

### A.3.3 system\_info.m

```

1 function osName = system_info()
2 % system_info - Detects the operating system
3 %
4 % OUTPUT:
5 %     osName - Operating system name as a string
6
7 if ispc
8     osName = "windows";
9 elseif ismac
10    osName = "macOS";
11 elseif isunix
12    osName = "linux";
13 else
14    osName = "unknownOS";
15 end
16 end

```

# Appendice B

## Codice SciPy

### B.1 main.py

```
1 from python.utils.Runner import Runner  
2  
3 runner = Runner("matrices/", "runs/")  
4 runner.run()
```

### B.2 utils

#### B.2.1 runner.py

```
1 import os  
2 import csv  
3 import time  
4 import psutil  
5 import platform  
6 import numpy as np  
7 import scipy.io  
8 import scipy.sparse as sp  
9 from datetime import datetime  
10 from sksparse.cholmod import cholesky  
11 import gc  
12  
13 from python.utils.CSVLogger import CSVLogger  
14 from python.utils.functions import measure_memory_mb,  
     get_peak_memory_mb, get_nnz  
15  
16  
17 class Runner:  
18     """  
19         Class to run Cholesky decomposition on a set of .mtx matrices.  
20         In particular, given the matrix, it solves the system Ax = b  
21             using Cholesky decomposition, with b such that the  
22             exact solution xe is [1 1 1 1...]. A log file is created with  
23             times, relative error and memory usage for each  
24             matrix.  
25             """  
26  
27     def __init__(self, matrices_path: str = './', logs_path: str = "  
        .//") -> None:  
            """
```

```

27     Runner constructor.
28     :param matrices_path: path to the matrices folder
29     :param logs_path: path to save the logs to
30     """
31
32     self.path = matrices_path
33     self.logger = CSVLogger(platform.system().lower() + "_python"
34                             , logs_path)
35
36     def run(self) -> None:
37         """
38         Method to start the Runner
39         """
40
41         matrix_paths = [
42             os.path.join(self.path, f)
43             for f in os.listdir(self.path)
44             if f.endswith(".mtx")
45         ]
46
47         matrix_files = sorted(matrix_paths, key=get_nnz)
48
49         if not matrix_files:
50             print("No matrices found in specified path")
51             return
52
53         for fname in matrix_files:
54             self.process_matrix(fname)
55
56         print(f"\nLog file: {self.logger.log_file}\n")
57
58     def process_matrix(self, matrix_path: str) -> None:
59         """
60         Method to execute Cholesky decomposition on a single matrix.
61         :param matrix_path: Path to the matrix file
62         """
63
64         gc.collect()
65         matrix_name = os.path.basename(matrix_path)
66
67         try:
68             print(f"\nLoading matrix: {matrix_name}")
69
70             t0 = time.perf_counter()
71             mem0 = measure_memory_mb()
72             A = scipy.io.mmread(matrix_path)
73             mem1 = measure_memory_mb()
74             t1 = time.perf_counter()
75             load_time = t1 - t0
76             load_mem = mem1 - mem0
77
78             n, m = A.shape
79             nnz = A.nnz
80             xe = np.ones(n)

```

```

77
78     A_csr = A.tocsr()
79     A_csc = A.tocsc()
80     b = A_csr @ xe
81
82     print(f"Decomposing matrix: {matrix_name}")
83     gc.collect()
84     mem2 = measure_memory_mb()
85     t2 = time.perf_counter()
86     factor = cholesky(A_csc)
87     t3 = time.perf_counter()
88     mem3 = measure_memory_mb()
89     decomp_time = t3 - t2
90     decomp_mem = mem3 - mem2
91     decomp_peak = get_peak_memory_mb()
92
93     print(f"Solving matrix: {matrix_name}")
94     gc.collect()
95     mem4 = measure_memory_mb()
96     t4 = time.perf_counter()
97     x = factor(b)
98     t5 = time.perf_counter()
99     mem5 = measure_memory_mb()
100    solve_time = t5 - t4
101    solve_mem = mem5 - mem4
102    solve_peak = get_peak_memory_mb()
103
104    rel_err = np.linalg.norm(x - xe, ord=2) / np.linalg.norm
105        (xe, ord=2)
106
106    row = {
107        "os": platform.system(),
108        "timestamp": datetime.now().isoformat(timespec="seconds"),
109        "matrixName": matrix_name,
110        "rows": n,
111        "cols": m,
112        "nonZeros": nnz,
113        "loadTime": f"{load_time:.6f}",
114        "loadMem": f"{load_mem:.2f}",
115        "decompTime": f"{decomp_time:.6f}",
116        "decompMem": f"{decomp_mem:.2f}",
117        "decompPeakMem": f"{decomp_peak:.2f}",
118        "solveTime": f"{solve_time:.6f}",
119        "solveMem": f"{solve_mem:.2f}",
120        "solvePeakMem": f"{solve_peak:.2f}",
121        "relativeError": f"{rel_err:.2e}"
122    }
123
124    print(f"Completed: {matrix_name} | total time: {load_time + decomp_time + solve_time:.4f}s")

```

```

125         self.logger.write_row(row)
126
127     except Exception as e:
128         print(f"Error in {matrix_name}: {e}")

```

### B.2.2 functions.py

```

1 import os
2 import psutil
3 import scipy
4
5
6 def measure_memory_mb() -> float:
7     """
8         Function to measure current memory usage.
9         :return: Current memory usage in MB
10    """
11    return psutil.Process(os.getpid()).memory_info().rss / (1024 * 1024)
12
13
14 def get_peak_memory_mb() -> float:
15     """
16         Function to get peak memory usage
17         :return: Peak memory usage in MB
18    """
19    return psutil.Process(os.getpid()).memory_info().peak_wset / (1024 * 1024) if hasattr(
20        psutil.Process(), "memory_info()", "peak_wset") else 0
21
22
23 def get_nnz(path: str) -> int:
24     """
25         Function to get the number of non zeros given a .mtx matrix.
26         :param path: Path to the .mtx matrix
27         :return: Number of non zeros
28     """
29
30     try:
31         A = scipy.io.mmread(path)
32         return A.nnz
33     except Exception as e:
34         print(f"Error reading {path}: {e}")
35         return int(float('inf'))

```

### B.2.3 CSVLogger.py

```

1 import csv
2 import os
3 from datetime import datetime
4

```

```

5
6 class CSVLogger:
7     """
8         Class to simplify logging the runs data to .csv files.
9     """
10    def __init__(self, filename: str = None, path: str = "") -> None:
11        :
12        """
13            CSVLogger constructor.
14            :param filename: Log file name
15            :param path: Path to save the file to
16        """
17        self._log_file = self._create_log_file_path(filename, path)
18
19    @property
20    def log_file(self) -> str:
21        return self._log_file
22
23    def _create_log_file_path(self, filename: str, path: str) -> str:
24        :
25        """
26            Method to create the logs file path.
27            :param filename: Name of the log file
28            :param path: Path to save the file to
29            :return: Full path file
30        """
31
32        if filename is None:
33            filename = 'log_chol'
34
35        filename += " " + datetime.now().strftime("%Y_%m_%d_%H%M%S")
36        + ".csv"
37        log_file = os.path.join(path, filename)
38
39        os.makedirs(path, exist_ok=True)
40        return log_file
41
42    def write_row(self, data: dict) -> None:
43        """
44            Method to write a row to the logs file. If the file is empty
45            , it writes the header first
46            :param data: dict of data to write
47        """
48
49        with open(self._log_file, 'a', newline='') as f:
50            writer = csv.DictWriter(f, fieldnames=data.keys())
51            if os.stat(self._log_file).st_size == 0:
52                writer.writeheader()
53            writer.writerow(data)

```

# Appendice C

## Dati completi

### C.1 MATLAB

#### C.1.1 Windows

##### Prima Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
windows	2025-06-04 09:44:54	ex15.mat	6867	6867	98671	0.0046722	0	0.0114862	8.904704	0.0010892	0	6.31831199483162e-07
windows	2025-06-04 09:44:55	shallow_water1.mat	81920	81920	327680	0.0168961	10.518528	0.1360298	84.852736	0.0090362	0	2.67234700436789e-16
windows	2025-06-04 09:44:55	cfd1.mat	70656	70656	1825580	0.0851155	58.540032	1.257849	1150.353408	0.0893756	0.16384	1.14331658728209e-13
windows	2025-06-04 09:44:57	cfd2.mat	123440	123440	3085406	0.1364587	99.065856	2.8504202	2374.885376	0.1987482	0	3.38304638867385e-13
windows	2025-06-04 09:45:01	parabolic_fem.mat	525825	525825	3674625	0.1472911	134.709248	1.6566071	1175.330816	0.1064307	33.767424	1.0537057085094e-12
windows	2025-06-04 09:45:04	apache2.mat	715176	715176	4817870	0.1139176	165.953536	7.4140363	5707.759616	0.4198549	45.8752	4.39187829690422e-11
windows	2025-06-04 09:45:12	G3_circuit.mat	1585478	1585478	7660826	0.1863291	271.065088	9.4153424	6113.828864	0.5140173	101.687296	3.57545524230986e-12

**Tabella C.1:** Prima Esecuzione del solutore MATLAB su Windows.

##### Seconda Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
windows	2025-06-05 09:49:43	ex15.mat	6867	6867	98671	0.0046763	0	0.0100793	7.315456	0.0010816	0	6.31831199483162e-07
windows	2025-06-05 09:49:43	shallow_water1.mat	81920	81920	327680	0.0170156	11.173888	0.1364587	75.603968	0.0080177	0	2.67234700436789e-16
windows	2025-06-05 09:49:44	cfd1.mat	70656	70656	1825580	0.0850661	58.540032	1.2846941	1150.369792	0.0859209	0	1.14331658728209e-13
windows	2025-06-05 09:49:46	cfd2.mat	123440	123440	3085406	0.1383025	98.934784	2.8240806	2377.203712	0.162311	0.065536	3.38304638867385e-13
windows	2025-06-05 09:49:50	parabolic_fem.mat	525825	525825	3674625	0.1504145	134.709248	1.6380787	1175.404544	0.1045809	33.685504	1.0537057085094e-12
windows	2025-06-05 09:49:52	apache2.mat	715176	715176	4817870	0.1101428	166.076416	7.1769879	5707.350016	0.4241114	45.858816	4.39187829690422e-11
windows	2025-06-05 09:50:00	G3_circuit.mat	1585478	1585478	7660826	0.1853383	271.056896	8.9185973	6113.886208	0.5331562	101.769216	3.57545524230986e-12

**Tabella C.2:** Seconda Esecuzione del solutore MATLAB su Windows.

##### Terza Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
windows	2025-06-05 09:49:43	ex15.mat	6867	6867	98671	0.0046763	0	0.0100793	7.315456	0.0010816	0	6.31831199483162e-07
windows	2025-06-05 09:49:43	shallow_water1.mat	81920	81920	327680	0.0170156	11.173888	0.1364587	75.603968	0.0080177	0	2.67234700436789e-16
windows	2025-06-05 09:49:44	cfd1.mat	70656	70656	1825580	0.0850661	58.540032	1.2846941	1150.369792	0.0859209	0	1.14331658728209e-13
windows	2025-06-05 09:49:46	cfd2.mat	123440	123440	3085406	0.1383025	98.934784	2.8240806	2377.203712	0.162311	0.065536	3.38304638867385e-13
windows	2025-06-05 09:49:50	parabolic_fem.mat	525825	525825	3674625	0.1504145	134.709248	1.6380787	1175.404544	0.1045809	33.685504	1.0537057085094e-12
windows	2025-06-05 09:49:52	apache2.mat	715176	715176	4817870	0.1101428	166.076416	7.1769879	5707.350016	0.4241114	45.858816	4.39187829690422e-11
windows	2025-06-05 09:50:00	G3_circuit.mat	1585478	1585478	7660826	0.1853383	271.056896	8.9185973	6113.886208	0.5331562	101.769216	3.57545524230986e-12

**Tabella C.3:** Terza Esecuzione del solutore MATLAB su Windows.

##### Quarta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
windows	2025-06-06 09:50:19	ex15.mat	6867	6867	98671	0.0047201	0	0.0111457	7.307264	0.0012574	0	6.31831199483162e-07
windows	2025-06-06 09:50:20	shallow_water1.mat	81920	81920	327680	0.0168673	10.518528	0.1330678	75.595776	0.0079568	0	2.67234700436789e-16
windows	2025-06-06 09:50:21	cfd1.mat	70656	70656	1825580	0.0849775	58.540032	1.2453428	1150.353408	0.0828368	0	1.14331658728209e-13
windows	2025-06-06 09:50:23	cfd2.mat	123440	123440	3085406	0.1368696	98.934784	2.8612495	2365.186048	0.1625621	0	3.38304638867385e-13
windows	2025-06-06 09:50:26	parabolic_fem.mat	525825	525825	3674625	0.1470856	134.709248	1.6974042	1175.298048	0.1035533	33.75104	1.0537057085094e-12
windows	2025-06-06 09:50:29	apache2.mat	715176	715176	4817870	0.1095348	165.945344	7.1541036	5707.014144	0.4441345	45.867008	4.39187829690422e-11
windows	2025-06-06 09:50:37	G3_circuit.mat	1585478	1585478	7660826	0.1849285	271.056896	8.9397825	6113.8944	0.4811858	101.679104	3.57545524230986e-12

**Tabella C.4:** Quarta Esecuzione del solutore MATLAB su Windows.

## Quinta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
windows	2025-06-09 09:52:46	ex15.mat	6867	6867	98671	0.004821	0	0.0103921	7.315456	0.0010925	0	6.31831199483162e-07
windows	2025-06-09 09:52:47	shallow_water1.mat	81920	81920	327680	0.0171087	10.518528	0.1356912	75.595776	0.0088123	0	2.67234700436789e-16
windows	2025-06-09 09:52:47	cfd1.mat	70656	70656	1825580	0.0852836	58.540032	1.2911524	1156.345216	0.0834224	0	1.143316588728209e-13
windows	2025-06-09 09:52:49	cfd2.mat	123440	123440	3085406	0.1384756	98.934784	2.9603462	2365.186048	0.1671839	0	3.38304638867385e-13
windows	2025-06-09 09:52:53	parabolic_fem.mat	525825	525825	3674625	0.148003	134.709248	1.7300158	1175.298048	0.1069791	33.75104	1.0537057085094e-12
windows	2025-06-09 09:52:56	apache2.mat	715176	715176	4817870	0.1145625	165.945344	7.4656096	5707.07968	0.4564873	45.867008	4.3918782969042e-11
windows	2025-06-09 09:53:05	G3_circuit.mat	1585478	1585478	7660826	0.1898467	271.056896	9.2983688	6114.131968	0.4967554	101.679104	3.57545524230986e-12

Tabella C.5: Quinta Esecuzione del solutore MATLAB su Windows.

## C.1.2 Linux

### Prima Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
linux	2025-06-09 15:22:05	ex15.mat	6867	6867	98671	0.007334	3.809072	0.014931	8.099904	0.00475	1.09245	6.64962774879362e-07
linux	2025-06-09 15:22:06	shallow_water1.mat	81920	81920	327680	0.016177	12.426544	0.208629	78.692704	0.013902	5.871776	2.66998119777909e-16
linux	2025-06-09 15:22:07	cfd1.mat	70656	70656	1825580	0.082238	61.397408	0.203049	1150.975048	0.09317	5.150336	1.08828690726715e-13
linux	2025-06-09 15:22:10	cfd2.mat	123440	123440	3085406	0.138045	103.386688	4.507659	2356.974848	0.171596	8.52912	3.23128582090735e-13
linux	2025-06-09 15:22:15	parabolic_fem.mat	525825	525825	3674625	0.142238	135.062016	2.213262	1175.726272	0.115187	34.312704	9.5623884588073e-13
linux	2025-06-09 15:22:18	apache2.mat	715176	715176	4817870	0.100509	166.249312	10.826224	NaN	0.419866	46.41568	4.39883197853885e-11
linux	2025-06-09 15:22:31	G3_circuit.mat	1585478	1585478	7660826	0.191453	271.156864	13.244894	NaN	0.479336	102.113056	3.36713698458665e-12

Tabella C.6: Prima Esecuzione del solutore MATLAB su Linux.

### Seconda Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
linux	2025-06-09 15:24:09	ex15.mat	6867	6867	98671	0.006435	3.865344	0.013827	8.11376	0.005964	1.065408	6.64962774879362e-07
linux	2025-06-09 15:24:10	shallow_water1.mat	81920	81920	327680	0.017001	12.42944	0.165734	78.673472	0.016157	5.879488	2.66998119777909e-16
linux	2025-06-09 15:24:11	cfd1.mat	70656	70656	1825580	0.080758	61.400256	1.942468	1138.3152	0.097251	5.155552	1.08828690726715e-13
linux	2025-06-09 15:22:15	cfd2.mat	123440	123440	3085406	0.132157	103.39024	4.576931	2362.415968	0.178674	8.536704	3.23128582090735e-13
linux	2025-06-09 15:24:13	parabolic_fem.mat	525825	525825	3674625	0.13893	135.043936	2.252029	1170.95296	0.12098	34.291616	9.56238845880873e-13
linux	2025-06-09 15:24:19	apache2.mat	715176	715176	4817870	0.104845	166.247872	10.602671	NaN	0.422216	46.408224	4.39883197853885e-11
linux	2025-06-09 15:24:22	G3_circuit.mat	1585478	1585478	7660826	0.191544	271.151392	13.110628	NaN	0.492174	102.119264	3.36713698458665e-12

Tabella C.7: Seconda Esecuzione del solutore MATLAB su Linux.

### Terza Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
linux	2025-06-09 15:24:09	ex15.mat	6867	6867	98671	0.006435	3.865344	0.012658	8.125792	0.005692	1.078752	6.64962774879362e-07
linux	2025-06-09 15:24:10	shallow_water1.mat	81920	81920	327680	0.017001	12.42944	0.165734	78.673472	0.016157	5.879488	2.66998119777909e-16
linux	2025-06-09 15:24:11	cfd1.mat	70656	70656	1825580	0.0805307	61.392352	1.981179	1150.983968	0.090566	5.161568	1.08828690726715e-13
linux	2025-06-09 15:22:15	cfd2.mat	123440	123440	3085406	0.127867	103.396704	4.686681	2362.431904	0.184224	8.535552	3.23128582090735e-13
linux	2025-06-09 15:24:19	parabolic_fem.mat	525825	525825	3674625	0.143878	135.056992	2.235045	1173.709824	0.113819	34.290336	9.56238845880873e-13
linux	2025-06-09 15:24:46	apache2.mat	715176	715176	4817870	0.100755	166.246468	10.627143	NaN	0.427721	46.411552	4.39883197853885e-11
linux	2025-06-09 15:25:58	G3_circuit.mat	1585478	1585478	7660826	0.198485	271.153024	13.125758	NaN	0.476139	102.108032	3.36713698458665e-12

Tabella C.8: Terza Esecuzione del solutore MATLAB su Linux.

### Quarta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
linux	2025-06-09 15:26:27	ex15.mat	6867	6867	98671	0.006132	3.886544	0.014103	8.122816	0.005382	1.076192	6.64962774879362e-07
linux	2025-06-09 15:26:28	shallow_water1.mat	81920	81920	327680	0.017365	12.425792	0.169628	78.668224	0.016366	5.878752	2.66998119777909e-16
linux	2025-06-09 15:26:29	cfd1.mat	70656	70656	1825580	0.082942	61.39984	1.980094	1150.983884	0.095923	5.15856	1.08828690726715e-13
linux	2025-06-09 15:26:31	cfd2.mat	123440	123440	3085406	0.127559	103.389696	4.656047	2363.762048	0.1917	8.53712	3.23128582090735e-13
linux	2025-06-09 15:26:37	parabolic_fem.mat	525825	525825	3674625	0.13825	135.053888	2.25948	1170.889568	0.11703	34.290208	9.56238845880873e-13
linux	2025-06-09 15:26:40	apache2.mat	715176	715176	4817870	0.106016	166.242208	10.52072	NaN	0.396783	46.408096	4.39883197853885e-11
linux	2025-06-09 15:26:52	G3_circuit.mat	1585478	1585478	7660826	0.185086	271.151648	12.605999	NaN	0.454141	102.107872	3.36713698458665e-12

Tabella C.9: Quarta Esecuzione del solutore MATLAB su Linux.

## Quinta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Solve (s)	Solve (MB)	Err. Rel.
linux	2025-06-09 15:27:28	ex15.mtx	6867	6867	98671	0.007038	3.891552	0.024043	8.121392	0.006699	1.076528	6.64962774879362e-07
linux	2025-06-09 15:27:29	shallow_water1.mtx	81920	81920	327680	0.024051	12.427296	0.173518	78.67632	0.015564	5.87904	2.66998119777909e-16
linux	2025-06-09 15:27:30	cfd1.mtx	70656	70656	1825580	0.078239	61.399296	2.143088	1138.529792	0.100242	5.150528	1.08828690726715e-13
linux	2025-06-09 15:27:33	cfd2.mtx	123440	123440	3085406	0.136384	103.39328	4.577916	2365.235136	0.167278	8.536576	3.23128582090735e-13
linux	2025-06-09 15:27:38	parabolic_fem.mtx	525825	525825	3674625	0.130558	135.044096	2.093292	1173.64928	0.126315	34.289824	9.56238845880873e-13
linux	2025-06-09 15:27:41	apache2.mtx	715176	715176	4817870	0.102516	159.822336	10.808628	NaN	0.414485	46.414752	4.39883197853885e-11
linux	2025-06-09 15:27:53	G3_circuit.mtx	1585478	1585478	7660826	0.193725	271.152288	13.248669	NaN	0.484282	102.109728	3.36713698458665e-12

Tabella C.10: Quinta Esecuzione del solutore MATLAB su Linux.

## C.2 SciPy

### C.2.1 Windows

#### Prima Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Windows	2025-06-09T11:53:27	ex15.mtx	6867	6867	98671	0.009813	2.88	0.050845	6.18	3247.25	0.001794	0.10	3247.25	6.39e-07
Windows	2025-06-09T11:53:28	shallow_water1.mtx	81920	81920	327680	0.024626	4.79	0.119126	38.83	3247.25	0.008200	1.25	3247.25	2.38e-16
Windows	2025-06-09T11:53:28	cfd1.mtx	70656	70656	1828364	0.069179	28.21	1.051938	197.26	3247.25	0.020990	1.11	3247.25	6.47e-14
Windows	2025-06-09T11:53:30	cfd2.mtx	123440	123440	3087898	0.118831	47.75	1.887363	350.75	3247.25	0.038865	1.41	3247.25	2.05e-13
Windows	2025-06-09T11:53:32	parabolic_fem.mtx	525825	525825	3674625	0.204427	56.66	1.058888	398.99	3247.25	0.089183	4.02	3247.25	1.05e-12
Windows	2025-06-09T11:53:34	apache2.mtx	715176	715176	4817870	0.173388	56.66	7.791980	1279.26	3247.25	0.168471	5.46	3247.25	3.26e-11
Windows	2025-06-09T11:53:53	G3_circuit.mtx	1585478	1585478	7660826	0.281362	117.49	12.357471	1046.79	3247.25	0.223817	12.11	3247.25	2.97e-12
Windows	2025-06-09T11:54:37	StoF+1465.mtx	1465137	1465137	21005389	0.373220	320.63	41.262889	9600.32	10873.62	0.729917	11.20	10873.62	3.68e-10
Windows	2025-06-09T11:55:29	Fran_1565.mtx	1564794	1564794	117406044	3.723152	1792.03	43.162704	13171.36	18690.59	1.341035	11.94	18690.59	2.83e-12

Tabella C.11: Prima Esecuzione del solutore SciPy su Windows.

#### Seconda Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Windows	2025-06-09T11:55:46	ex15.mtx	6867	6867	98671	0.007927	2.93	0.027363	6.32	3248.00	0.001019	0.10	3248.00	6.39e-07
Windows	2025-06-09T11:55:47	shallow_water1.mtx	81920	81920	327680	0.025487	4.65	0.107233	40.36	3248.00	0.009522	0.01	3248.00	2.38e-16
Windows	2025-06-09T11:55:47	cfd1.mtx	70656	70656	1828364	0.066234	29.40	0.999725	196.80	3248.00	0.17655	1.11	3248.00	6.47e-14
Windows	2025-06-09T11:55:49	cfd2.mtx	123440	123440	3087898	0.110000	47.71	1.771511	350.11	3248.00	0.034572	0.48	3248.00	2.05e-13
Windows	2025-06-09T11:55:51	parabolic_fem.mtx	525825	525825	3674625	0.172746	56.66	1.013930	399.34	3248.00	0.069382	4.02	3248.00	1.05e-12
Windows	2025-06-09T11:55:59	apache2.mtx	715176	715176	4817870	0.167163	74.09	7.441067	1279.46	3248.00	0.160647	5.46	3248.00	3.26e-11
Windows	2025-06-09T11:56:11	G3_circuit.mtx	1585478	1585478	7660826	0.279499	117.50	11.860212	1046.99	3248.00	0.221827	12.10	3248.00	2.97e-12
Windows	2025-06-09T11:56:53	StoF+1465.mtx	1465137	1465137	21005389	0.390197	320.63	39.887304	9598.33	10871.97	0.890042	11.18	10871.97	3.68e-10
Windows	2025-06-09T11:57:42	Fran_1565.mtx	1564794	1564794	117406044	2.807622	1792.07	41.264303	13171.40	18690.06	1.248037	11.94	18690.06	2.83e-12

Tabella C.12: Seconda Esecuzione del solutore SciPy su Windows.

#### Terza Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Windows	2025-06-09T11:57:52	ex15.mtx	6867	6867	98671	0.008141	1.45	0.017521	6.73	3247.42	0.000896	0.10	3247.42	6.39e-07
Windows	2025-06-09T11:57:52	shallow_water1.mtx	81920	81920	327680	0.025214	4.81	0.118830	39.26	3247.42	0.008613	0.95	3247.42	2.38e-16
Windows	2025-06-09T11:57:53	cfd1.mtx	70656	70656	1828364	0.071849	29.39	1.028624	196.85	3247.42	0.21707	1.11	3247.42	6.47e-14
Windows	2025-06-09T11:57:55	cfd2.mtx	123440	123440	3087898	0.114067	47.70	1.790966	350.23	3247.42	0.032413	1.43	3247.42	2.05e-13
Windows	2025-06-09T11:57:57	parabolic_fem.mtx	525825	525825	3674625	0.171787	56.64	0.987459	399.52	3247.42	0.066464	4.02	3247.42	1.05e-12
Windows	2025-06-09T11:58:09	apache2.mtx	715176	715176	4817870	0.172117	74.09	7.433208	1280.38	3247.42	0.154505	5.46	3247.42	3.26e-11
Windows	2025-06-09T11:58:17	G3_circuit.mtx	1585478	1585478	7660826	0.259547	117.48	11.848115	1045.36	3247.42	0.222196	12.10	3247.42	2.97e-12
Windows	2025-06-09T11:58:59	StoF+1465.mtx	1465137	1465137	21005389	0.397872	320.67	40.444430	9601.44	10874.37	0.726992	11.19	10874.37	3.68e-10
Windows	2025-06-09T11:59:49	Fran_1565.mtx	1564794	1564794	117406044	3.819201	1792.04	40.518623	13173.27	18693.02	0.958877	11.94	18693.02	2.83e-12

Tabella C.13: Terza Esecuzione del solutore SciPy su Windows.

#### Quarta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Windows	2025-06-09T12:00:23	ex15.mtx	6867	6867	98671	0.008535	2.92	0.017718	6.28	3247.26	0.000577	0.10	3247.26	6.39e-07
Windows	2025-06-09T12:00:23	shallow_water1.mtx	81920	81920	327680	0.022609	4.80	0.112262	40.50	3247.26	0.009582	0.01	3247.26	2.38e-16
Windows	2025-06-09T12:00:25	cfd1.mtx	70656	70656	1828364	0.065358	28.26	1.044413	197.91	3247.26	0.020149	1.11	3247.26	6.47e-14
Windows	2025-06-09T12:00:27	cfd2.mtx	123440	123440	3087898	0.110631	47.71	1.762161	350.09	3247.26	0.032836	1.89	3247.26	2.05e-13
Windows	2025-06-09T12:00:28	parabolic_fem.mtx	525825	525825	3674625	0.174229	56.64	1.002044	399.16	3247.26	0.071195	4.02	3247.26	1.05e-12
Windows	2025-06-09T12:00:36	apache2.mtx	715176	715176	4817870	0.173757	74.09	7.411600	1278.27	3247.26	0.159835	5.46	3247.26	3.26e-11
Windows	2025-06-09T12:00:48	G3_circuit.mtx	1585478	1585478	7660826	0.269057	117.49	11.674797	1047.18	3247.26	0.229823	12.10	3247.26	2.97e-12
Windows	2025-06-09T12:01:38	StoF+1465.mtx	1465137	1465137	21005389	0.604478	320.57	39.631541	9601.13	10874.48	0.742430	11.18	10874.48	3.68e-10
Windows	2025-06-09T12:02:18	Fran_1565.mtx	1564794	1564794	117406044	2.848542	1792.04	39.734983	13172.40	18692.20	1.381894	11.94	18692.20	2.83e-12

Tabella C.14: Quarta Esecuzione del solutore SciPy su Windows.

## Quinta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Windows	2025-06-09T12:02:53	ex15.mtx	6867	98671	0.008117	1.11	0.018886	6.55	0.000611	0.10	3248.00	3248.00	6.39e-07	
Windows	2025-06-09T12:02:53	shallow_water1.mtx	81920	327680	0.023596	5.55	0.116721	38.92	3248.00	0.009957	1.26	3248.00	2.38e-16	
Windows	2025-06-09T12:02:55	cfl1.mtx	70656	70656	1828364	0.070634	29.44	1.066295	197.54	3248.00	0.018761	0.83	3248.00	6.47e-14
Windows	2025-06-09T12:02:57	cfl2.mtx	123440	123440	3087898	0.113670	47.74	1.770795	349.82	3248.00	0.032796	1.90	3248.00	2.05e-13
Windows	2025-06-09T12:02:58	parabolic_fem.mtx	525825	525825	3674625	0.173578	56.69	1.005192	399.55	3248.00	0.071228	4.02	3248.00	1.05e-12
Windows	2025-06-09T12:03:06	apache2.mtx	715176	715176	4817870	0.177822	74.11	7.604512	1280.08	3248.00	0.165835	5.46	3248.00	3.26e-11
Windows	2025-06-09T12:03:19	G3_circuit.mtx	1585478	1585478	7660826	0.286891	117.15	12.356354	1045.87	3248.00	0.221287	12.10	3248.00	2.97e-12
Windows	2025-06-09T12:04:02	StocF-1465.mtx	1465137	1465137	21005389	0.739521	320.64	40.630436	9600.05	10872.78	0.849694	11.20	10872.78	3.68e-10
Windows	2025-06-09T12:04:52	Fian_1565.mtx	1564794	1564794	117406044	3.724024	1792.05	40.581305	13178.59	18698.10	1.258327	11.94	18698.10	2.83e-12

Tabella C.15: Quinta Esecuzione del solutore SciPy su Windows.

## C.2.2 Linux

### Prima Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Linux	2025-06-09T14:44:14	ex15.mtx	6867	98671	0.019254	0.00	0.015832	1.09	0.00	0.001989	0.12	0.00	0.00	6.12e-07
Linux	2025-06-09T14:44:15	shallow_water1.mtx	81920	81920	327680	0.087332	0.00	0.399232	36.25	0.00	0.008505	0.00	0.00	2.03e-16
Linux	2025-06-09T14:44:17	cfl1.mtx	70656	70656	1828364	0.093770	27.84	1.482899	283.31	0.00	0.024727	0.00	0.00	6.63e-14
Linux	2025-06-09T14:44:19	cfl2.mtx	123440	123440	3087898	0.110318	0.00	2.534067	384.88	0.00	0.055807	0.00	0.00	1.86e-13
Linux	2025-06-09T14:44:22	parabolic_fem.mtx	525825	525825	3674625	0.072432	0.00	1.744224	417.88	0.00	0.076080	0.00	0.00	6.70e-13
Linux	2025-06-09T14:44:24	apache2.mtx	715176	715176	4817870	0.110489	36.73	12.187079	1300.40	0.00	0.205110	0.00	0.00	3.53e-11
Linux	2025-06-09T14:44:24	G3_circuit.mtx	1585478	1585478	7660826	0.210368	116.64	14.157161	1083.68	0.00	0.271022	0.00	0.00	3.78e-12
Linux	2025-06-09T14:44:49	StocF-1465.mtx	1465137	1465137	21005389	0.398919	320.82	77.460696	9662.57	0.00	1.168283	0.00	0.00	3.69e-10
Linux	2025-06-09T14:46:38	Fian_1565.mtx	1564794	1564794	117406044	1.846464	1791.71	82.247547	13064.47	0.00	1.602685	0.00	0.00	2.47e-12

Tabella C.16: Prima Esecuzione del solutore SciPy su Linux.

### Seconda Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Linux	2025-06-09T14:44:14	ex15.mtx	6867	98671	0.019254	0.00	0.022964	0.91	0.00	0.008134	0.15	0.00	0.00	6.12e-07
Linux	2025-06-09T14:44:15	shallow_water1.mtx	81920	81920	327680	0.057874	0.00	0.276730	36.38	0.00	0.036854	0.00	0.00	2.03e-16
Linux	2025-06-09T14:44:17	cfl1.mtx	70656	70656	1828364	0.116302	20.89	2.143115	287.79	0.00	0.082949	0.00	0.00	6.63e-14
Linux	2025-06-09T14:44:19	cfl2.mtx	123440	123440	3087898	0.092961	0.00	4.229052	435.50	0.00	0.251723	0.00	0.00	1.86e-13
Linux	2025-06-09T14:44:22	parabolic_fem.mtx	525825	525825	3674625	0.072432	0.12	3.130309	370.38	0.00	0.169644	0.00	0.00	6.70e-13
Linux	2025-06-09T14:44:24	apache2.mtx	715176	715176	4817870	0.102465	0.00	11.717721	1259.02	0.00	0.172464	0.00	0.00	3.53e-11
Linux	2025-06-09T14:44:49	G3_circuit.mtx	1585478	1585478	7660826	0.143079	58.43	13.157672	999.86	0.00	0.194962	0.00	0.00	3.78e-12
Linux	2025-06-09T15:05:59	StocF-1465.mtx	1465137	1465137	21005389	0.336680	320.95	97.185387	9623.43	0.00	0.913634	0.00	0.00	3.69e-10
Linux	2025-06-09T15:05:38	Fian_1565.mtx	1564794	1564794	117406044	1.570632	1791.62	112.983636	13071.81	0.00	2.195988	0.00	0.00	2.47e-12

Tabella C.17: Seconda Esecuzione del solutore SciPy su Linux.

### Terza Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Linux	2025-06-09T15:04:21	ex15.mtx	6867	98671	0.016064	0.00	0.022964	0.91	0.00	0.008134	0.15	0.00	0.00	6.12e-07
Linux	2025-06-09T15:04:22	shallow_water1.mtx	81920	81920	327680	0.072049	0.00	0.341281	36.50	0.00	0.018102	0.00	0.00	2.03e-16
Linux	2025-06-09T15:04:24	cfl1.mtx	70656	70656	1828364	0.102709	20.88	1.311771	285.41	0.00	0.039626	0.00	0.00	6.63e-14
Linux	2025-06-09T15:04:52	cfl2.mtx	123440	123440	3087898	0.062963	0.12	2.279838	384.84	0.00	0.034326	0.00	0.00	1.86e-13
Linux	2025-06-09T15:04:53	parabolic_fem.mtx	525825	525825	3674625	0.090772	0.12	1.429157	426.00	0.00	0.070594	0.00	0.00	6.70e-13
Linux	2025-06-09T15:05:04	apache2.mtx	715176	715176	4817870	0.096026	36.74	9.801321	1300.29	0.00	0.162320	0.00	0.00	3.53e-11
Linux	2025-06-09T15:05:18	G3_circuit.mtx	1585478	1585478	7660826	0.177692	116.61	13.263719	1113.86	0.00	0.197787	0.00	0.00	3.78e-12
Linux	2025-06-09T15:06:39	StocF-1465.mtx	1465137	1465137	21005389	0.370884	320.90	79.421822	9602.45	0.00	0.709816	0.00	0.00	3.69e-10
Linux	2025-06-09T15:08:10	Fian_1565.mtx	1564794	1564794	117406044	1.490175	1791.62	85.912928	13190.59	0.00	1.098830	0.00	0.00	2.47e-12

Tabella C.18: Terza Esecuzione del solutore SciPy su Linux.

### Quarta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Linux	2025-06-09T15:08:26	ex15.mtx	6867	98671	0.024857	0.00	0.018900	0.96	0.00	0.000700	0.18	0.00	0.00	6.12e-07
Linux	2025-06-09T15:08:26	shallow_water1.mtx	81920	81920	327680	0.032498	0.00	0.263396	36.38	0.00	0.008200	0.00	0.00	2.03e-16
Linux	2025-06-09T15:08:28	cfl1.mtx	70656	70656	1828364	0.140430	20.69	1.355399	285.43	0.00	0.030382	0.00	0.00	6.63e-14
Linux	2025-06-09T15:08:31	cfl2.mtx	123440	123440	3087898	0.087864	0.00	2.732290	436.75	0.00	0.108306	0.00	0.00	1.86e-13
Linux	2025-06-09T15:08:33	parabolic_fem.mtx	525825	525825	3674625	0.095460	0.00	1.535192	370.38	0.00	0.094912	0.00	0.00	6.70e-13
Linux	2025-06-09T15:08:45	apache2.mtx	715176	715176	4817870	0.114163	0.00	10.786327	1257.55	0.00	0.155479	0.00	0.00	3.53e-11
Linux	2025-06-09T15:08:59	G3_circuit.mtx	1585478	1585478	7660826	0.157796	139.93	13.216488	1065.68	0.00	0.190122	0.00	0.00	3.78e-12
Linux	2025-06-09T15:10:21	StocF-1465.mtx	1465137	1465137	21005389	0.385478	320.59	80.233706	9604.21	0.00	0.736834	0.00	0.00	3.69e-10
Linux	2025-06-09T15:11:53	Fian_1565.mtx	1564794	1564794	117406044	1.474483	1791.69	87.105278	13153.99	0.00	1.100440	0.00	0.00	2.47e-12

Tabella C.19: Quarta Esecuzione del solutore SciPy su Linux.

## Quinta Esecuzione

OS	TimeStamp	Matrice	Rows	Cols	NNZ	Load (s)	Load (MB)	Dec (s)	Dec (MB)	Dec Peak (MB)	Solve (s)	Solve (MB)	Solve Peak (MB)	Err. Rel.
Linux	2025-06-09T15:12:33	ex15.mtx	6867	6867	98671	0.017039	0.12	0.016364	0.97	0.00	0.001785	0.18	0.00	6.12e-07
Linux	2025-06-09T15:12:34	shallow_water1.mtx	81920	81920	327680	0.078105	0.12	0.388687	36.25	0.00	0.015277	0.00	0.00	2.03e-16
Linux	2025-06-09T15:12:35	cfl1.mtx	70656	70656	1828364	0.088571	20.82	1.370185	285.12	0.00	0.033243	0.00	0.00	6.63e-14
Linux	2025-06-09T15:12:38	cfl2.mtx	123440	123440	3087898	0.089711	0.88	2.446128	377.72	0.00	0.037378	0.00	0.00	1.86e-13
Linux	2025-06-09T15:12:41	parabolic_fem.mtx	525825	525825	3674625	0.101382	0.00	2.102679	414.75	0.00	0.072121	0.00	0.00	6.70e-13
Linux	2025-06-09T15:12:51	apache2.mtx	715176	715176	4817870	0.084947	0.00	9.932738	1306.71	0.00	0.211070	0.00	0.00	3.53e-11
Linux	2025-06-09T15:13:05	G3_circuit.mtx	1585478	1585478	7660826	0.148365	58.32	13.391710	1030.33	0.00	0.200749	0.00	0.00	3.78e-12
Linux	2025-06-09T15:14:26	StocF-1465.mtx	1465137	1465137	21005389	0.387309	320.70	78.553183	9638.75	0.00	1.148863	0.00	0.00	3.69e-10
Linux	2025-06-09T15:15:42	Fian_1565.mtx	1564794	1564794	117406044	1.612939	1791.80	71.379784	13070.25	0.00	0.955325	0.00	0.00	2.47e-12

Tabella C.20: Quinta Esecuzione del solutore SciPy su Linux.