UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Dipartimento di Informatica Sistemistica e Comunicazione

Corso di Laurea Magistrale in Informatica

Metodi del Calcolo Scientifico

—

Progetto 1

736195 Pavlo Lysytsya
756610 Stefano Secci
766267 Luca Simonetta

Anno Accademico 2015-2016

Indice

Capitolo 1

Introduzione

Lo scopo di questo progetto è di valutare l'implementazione in ambienti di programmazione differenti degli algoritmi di risoluzione diretta di sistemi lineari per matrici sparse. La scelta degli ambienti, da parte del team, si è focalizzata su:

- Matlab.
- Python.
- Scilab.

Le matrici che sono state considerate per la realizzazione del progetto sono quelle del gruppo FEMLAB (tranne la matrice FEMLAB/waveguide3D (l'ultima della serie) che è a coefficienti complessi):

- Ns3Da.
- Problem1.
- Poisson2D.
- Poisson3Da.
- Poisson3Db.
- Sme3Da.
- Sme3Db.
- Sme3Dc.

Un sistema lineare si può rappresentare in forma matriciale come $A \cdot x = b$ dove:

- A è la matrice dei coefficienti del sistema ed è rappresentata da una delle matrici del gruppo FEMLAB prima elencate.
- x è il vettore colonna delle incognite.
- b è il vettore colonna dei termini noti; b è scelto in modo che la soluzione esatta sia il vettore $xe = [1\ 1\ 1\ 1\ 1\ 1\ ...]$ avente tutte le componenti uguali a 1: in altre parole, $b = A \cdot xe$.

I punti chiave del progetto sono:

- Tempo necessario per calcolare la soluzione x, dove $x = A \setminus b$.
- Errore relativo tra la soluzione calcolata x e la soluzione esatta xe, definito come $errore\ relativo = \frac{||x-xe||}{||xe||}$.
- Memoria necessaria per risolvere il sistema.

1.1 Matrici sparse

Le matrici sparse sono una classe di matrici con la caratteristica di contenere un significativo numero di elementi uguali a zero. Spesso il numero di elementi diversi da zero su ogni riga è un numero piccolo (per esempio dell'ordine di 10^1) indipendente dalla dimensione della matrice, che può essere anche dell'ordine di 10^8.

Le matrici sparse si possono memorizzare in modo compatto, tenendo solo conto degli elementi diversi da zero; per esempio, è sufficiente per ogni elemento diverso da zero memorizzare solo la sua posizione ij e il suo valore a_{ij}, ignorando gli elementi uguali a zero. Ciò consente di ridurre il tempo di calcolo eliminando operazioni su elementi nulli.

Capitolo 2

Ambienti di sviluppo

Nel presente capitolo verranno introdotti gli ambienti di sviluppo utilizzati nello svolgimento del progetto.

2.1 Matlab

Versione Matlab: R2016a (0.0.0.341360) 64 bit.
Sito: http://it.mathworks.com/products/matlab/

Descrizione: *Matlab* è un linguaggio di programmazione ad alto livello[1]; la piattaforma interattiva permette di effettuare task più rapidi rispetto ai tradizionali linguaggi di programmazione come *C*, *C++*, e *Fortran*. Steve Bangert e Jack Little, assieme a Cleve Moler, riconobbero in questo software il suo pieno potenziale. *Matlab*, come lo conosciamo oggi, è un ottimo ambiente per il calcolo scientifico.

[1]Un **linguaggio di programmazione ad alto livello** è un linguaggio di programmazione caratterizzato da una significativa astrazione dai dettagli del funzionamento di un calcolatore e dalle caratteristiche del linguaggio macchina. I linguaggi di programmazione ad alto livello sono progettati per essere facilmente comprensibili dagli esseri umani. Per **linguaggio di programmazione a basso livello** si intende un linguaggio di programmazione che coincide con il linguaggio macchina, fornendo poca o nessuna astrazione dai dettagli del funzionamento fisico del calcolatore. Si può dire che i linguaggi di programmazione di basso livello sono orientati "alla macchina" (ovvero il loro scopo è di essere direttamente eseguibili dal processore, o di poter essere tradotti facilmente in programmi eseguibili dal processore), mentre i linguaggi ad alto livello sono orientati "al programmatore" (il loro scopo è quello di essere facilmente utilizzabili dai programmatori umani).

2.2 Scilab

Versione Scilab: 5.5.2.
Sito: http://www.scilab.org/

Descrizione: *Scilab* è un software open source per il calcolo numerico distribuito sotto licenza CeCill[2], che venne creato dall'INRIA (French National Research Institution) nel 1990. Esso è compatibile con Linux, Mac OS X e Windows.

Scilab è un linguaggio di alto livello che può essere usato in una ampia gamma di applicazioni tra le quali l'analisi e la visualizzazione dei dati, la statistica, l'elaborazione di immagini, i sistemi dinamici, l'ottimizzazione. *Scilab* è spesso confrontato con *Matlab*: la sintassi dei due software è infatti molto simile, ma i due programmi, i loro applicativi ed i plug-in non sono completamente compatibili, anche se esiste un convertitore nel pacchetto di *Scilab*, che opera le conversioni *Matlab* $\rightarrow$ *Scilab*[3]. Esistono altri prodotti simili a *Scilab* (e.g. *Octave*, *Maxima*); tuttavia *Scilab* si differenzia da questi ultimi per la maggiore maturità in quanto lo sviluppo e la manutenzione del software sono gestite e controllate dal *Scilab Enterprises*.

Scilab è dotato dell'*ATOMS* (AuTomatic mOdules Management for Scilab); è una repository che consente di scaricare pacchetti ("Toolboxes") non compresi di default, per creare applicazioni complesse; nel nostro caso, è stato scaricato il pacchetto matrixMarket. Esso fornisce funzioni per gestire *Matrix Market files* (file formato mtx).

[2]CeCILL (da CEA CNRS INRIA Logiciel Libre) è una licenza per software libero adatta sia al diritto internazionale che a quello francese

[3]Si rimanda al seguente link per la conversione: https://help.scilab.org/docs/5.5.2/en_US/mfile2sci.html

2.3 PyCharm

Versione PyCharm: PyCharm Community Edition 2016.1.4.
Sito: https://www.jetbrains.com/pycharm/

Descrizione: PyCharm è un ambiente di sviluppo utilizzato per programmare in Python. Esistono due versioni di PyCharm:

- Community Edition (utilizzata da noi), rilasciata con licenza libera; è meno estesa della versione Professional.
- Professional Edition, rilasciata con licenza proprietaria.

Python è un linguaggio di programmazione ad alto livello. Per poter realizzare il progetto in questo linguaggio, è stata adoperata la libreria *Scipy*; è una libreria open source di algoritmi e strumenti matematici per il linguaggio di programmazione *Python*. Il suo sviluppo è portato avanti da una vasta comunità di sviluppatori. Contiene moduli per l'ottimizzazione, per l'algebra lineare, l'integrazione, funzioni speciali, elaborazione di segnali ed immagini e altri strumenti comuni nelle scienze e nell'ingegneria. I moduli utilizzati sono:

- **scipy.sparse**: modulo riguardante matrici sparse e algoritmi correlati.
- **scipy.linalg**: adatta per trattare ed elaborare al meglio tutti i tipi di matrici. In particolare, per le matrici sparse, sono integrati tutti i metodi di risoluzione e di rappresentazione.
- **scipy.io**: adatta per importare ed esportare file di diversi formati.

Sul sito https://www.scipy.org/scipylib/index.html è fornito ampio supporto ed una vasta documentazione da parte degli sviluppatori. Con l'installazione di PyCharm, non abbiamo dovuto importare e installare alcuna libreria aggiuntiva, tra cui anche scipy.

Capitolo 3

Risultati delle elaborazioni

I risultati proposti di seguito riguardano i confronti tra i tre ambienti utilizzati (ambienti diversi, stessa macchina), sui punti chiave citati inizialmente.

Per rendere apprezzabile i risultati delle elaborazioni sui sistemi lineari, sono stati effettuati i confronti tra i risultati ottenuti dalle macchine (macchine diverse, stesso ambiente), dei componenti del gruppo.

I test sono stati effettuati su un *ASUS X550C* e su un *APPLE MacBook Pro Retina Late 2012*.

Le caratteristiche dei dispositivi sono le seguenti:

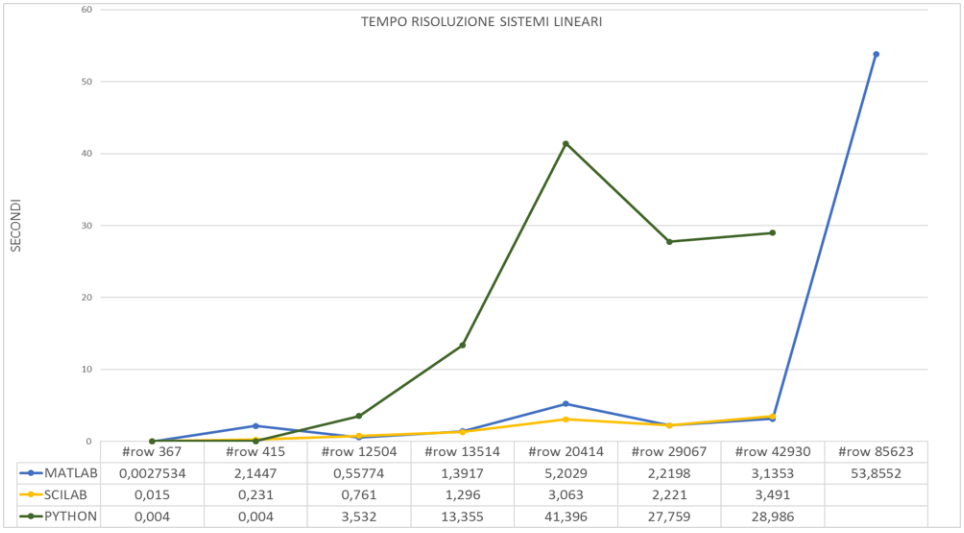
Caratteristiche tecniche	ASUS X550C	APPLE MacBook Pro Retina
Sistema operativo	Windows 10	OS X El Capitan
Processore	Intel Core i3	Intel Core i7
Clock	1.80 GHz	2.7 GHz
RAM	4 GB	16 GB

3.1 Confronto tra ambienti sulla stessa macchina

Verranno ora confrontate le due macchine utilizzate in ambito di ambienti di sviluppo utilizzati.

3.1.1 Tempo di risoluzione dei sistemi lineari

I grafici proposti di seguito riguardano i tempi di risoluzione dei sistemi lineari, misurati in secondi, eseguiti sulle due macchine nei tre ambienti proposti.

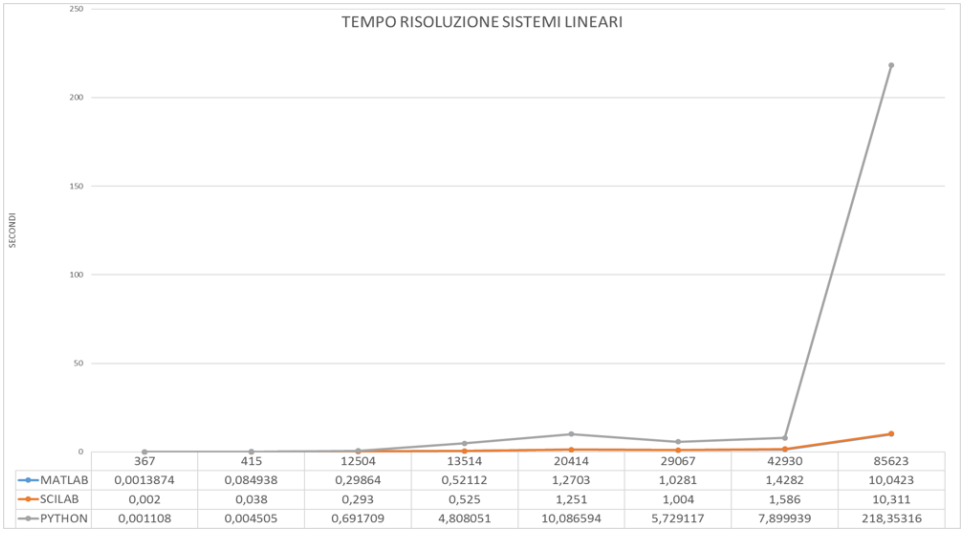


	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
MATLAB	0,0027534	2,1447	0,55774	1,3917	5,2029	2,2198	3,1353	53,8552
SCILAB	0,015	0,231	0,761	1,296	3,063	2,221	3,491	
PYTHON	0,004	0,004	3,532	13,355	41,396	27,759	28,986	

Figura 3.1: Tempo di risoluzione dei sistemi lineari su Asus X550C.

Per quanto riguarda la prima macchina (figura 3.1) *Scilab* si è mostrato più performante, ma non è riuscito a risolvere l'ultimo sistema lineare con una matrice di 85623 righe (stesso risultato per *Python*). *Matlab* ha avuto tempi leggermente più alti, ma è riuscito a risolvere tutti i sistemi lineari. *Python* ha avuto tempi di risoluzione peggiori rispetto agli altri due ambienti.

Sul Mac (figura 3.2) i tempi di risoluzione dei sistemi lineari in *Matlab* e *Scilab* sono coincidenti. *Python* mantiene tempi contenuti rispetto all'Asus X550C, ma comunque alti rispetto agli altri due ambienti. In tutti e 3 gli ambienti, i sistemi lineari vengono tutti risolti. Di seguito il grafico.



	367	415	12504	13514	20414	29067	42930	85623
MATLAB	0,0013874	0,084938	0,29864	0,52112	1,2703	1,0281	1,4282	10,0423
SCILAB	0,002	0,038	0,293	0,525	1,251	1,004	1,586	10,311
PYTHON	0,001108	0,004505	0,691709	4,808051	10,086594	5,729117	7,899939	218,35316

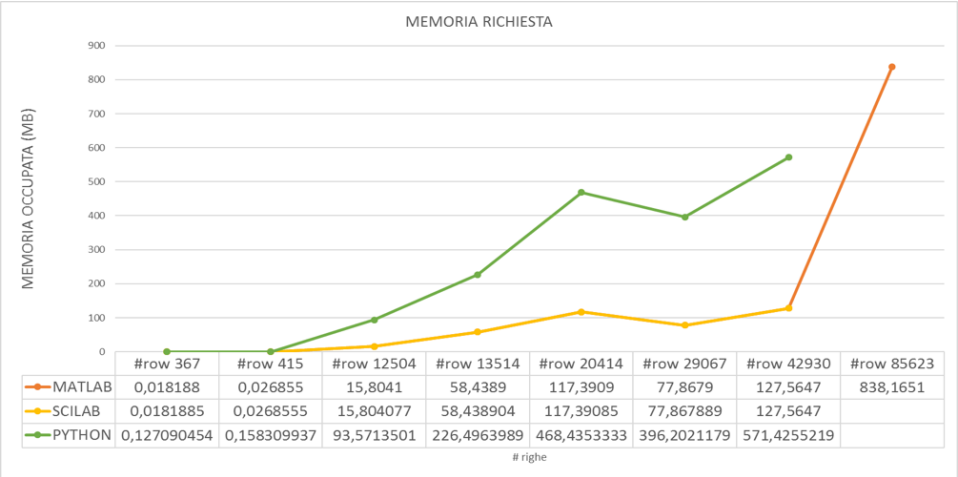
Figura 3.2: Tempo di risoluzione dei sistemi lineari su MacBook Pro Retina Late 2012.

3.1.2 Occupazione memoria

Il calcolo dell'occupazione della memoria è stato effettuato applicando la fattorizzazione LU alla matrice A letta dal file *.mat* per *Matlab* ed *.mtx* negli altri ambienti. Sono stati calcolati i numeri di elementi diversi da zero della matrice A e della matrice L ottenuta precedentemente.

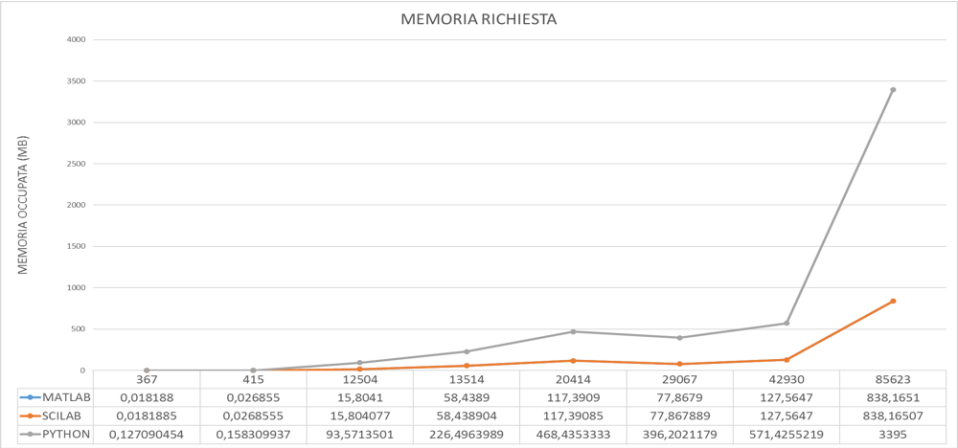
La differenza tra i numeri di non zeri dei due elementi è un metodo per il calcolo dell'aumenta della memoria nella risoluzione del sistema lineare. Per giungere ai MB occupati, si moltiplica la differenza tra i numeri di non zeri per 16 (4 byte per la colonna matrice, 4 byte per la riga matrice e 8 byte necessari per salvare la variabile *double*). Infine si converte i valori ottenuti in MB.

Per quanto riguarda l'Asus (figura 3.3) *Scilab* e *Matlab* occupano la stessa quantità di memoria. Per *Scilab*, non riuscendo a risolvere il sistema lineare per la matrice con *Poisson3Db*, non è stato possibile calcolare la quantità di memoria occupata. *Python*, rispetto agli altri due ambienti, ha dimostrato un utilizzo maggiore della memoria. Anche per esso, non essendo stato in grado di risolvere il sistema lineare, non è stato possibile registrare la quantità di memoria occupata.



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
MATLAB	0,018188	0,026855	15,8041	58,4389	117,3909	77,8679	127,5647	838,1651
SCILAB	0,0181885	0,0268555	15,804077	58,438904	117,39085	77,867889	127,5647	
PYTHON	0,127090454	0,158309937	93,5713501	226,4963989	468,4353333	396,2021179	571,4255219	

Figura 3.3: Occupazione memoria su Asus X550C.



	367	415	12504	13514	20414	29067	42930	85623
MATLAB	0,018188	0,026855	15,8041	58,4389	117,3909	77,8679	127,5647	838,1651
SCILAB	0,0181885	0,0268555	15,804077	58,438904	117,39085	77,867889	127,5647	838,16507
PYTHON	0,127090454	0,158309937	93,5713501	226,4963989	468,4353333	396,2021179	571,4255219	3395

Figura 3.4: Occupazione memoria su MacBook Pro Retina Late 2012.

La memoria richiesta dagli ambenti su Mac (figura 3.4), è lo stesso riscontrato su ASUS X550C. La matrice di 85623 righe, tuttavia, seppur riesce ad essere computata, richiede un utilizzo di memoria considerevole.

3.1.3 Errore relativo

Per una maggior leggibilità dei dati è stato apportato un cambio di scala e si è optato per una scala logaritmica in base 10.

Dati originali e dati in scala			
Numero righe/colonne matrice	Matlab	Scilab	Python
367 (Poisson2D)	**3,84E-16** -15,415737	**6,48E-16** -15,188424	**4,92E-16** 15,308034
415 (Problem1)	**7,27E-01** -0,117634	**1,41E-09** -8,85078	**1,35026042** 0,130333
12504 (Sme3Da)	**1,72E-12** -11,76447	**5,43E-13** -12,2652	**1,1353E-11** -10,944889
13514 (Poisson3Da)	**2,22E-15** -14,653647	**4,92E-15** -14,308034	**1,89E-14** -13,723538
20414 (Ns3Da)	**9,28E-16** -15,032452	**7,12E-16** -15,14752	**8,99E-15** -14,04624
29067 (Sme3Db)	**3,89E-12** -11,410084	**3,41E-12** -11,50307	**1,504E-11** -10,822766
42930 (Sme3Dc)	**3,48E-12** -11,415668	**1,13E-12** -11,946921	**6,70E-12** -11,173925
85623 (Poisson3Db)	**2,69E-15** -14,570247	**1,368E-12** -11,863913	**1,41E-13** -12,85078

Tabella 3.1: La notazione scientifica è evidenziata in grassetto.

Di seguito i grafici circa l'errore relativo sulle due macchine.

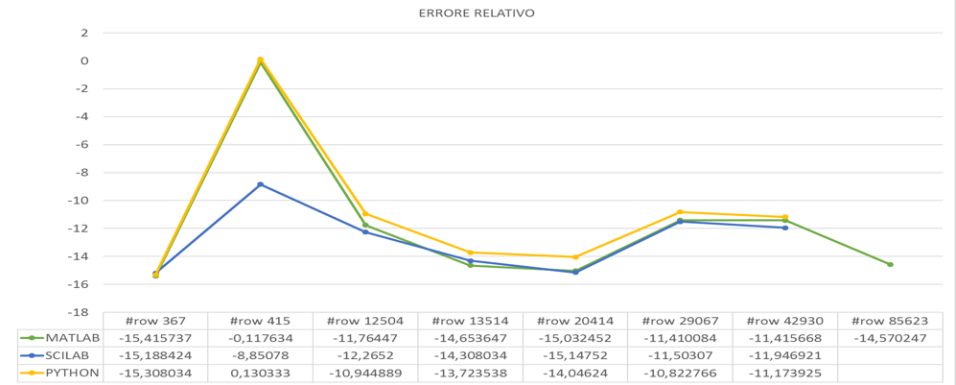
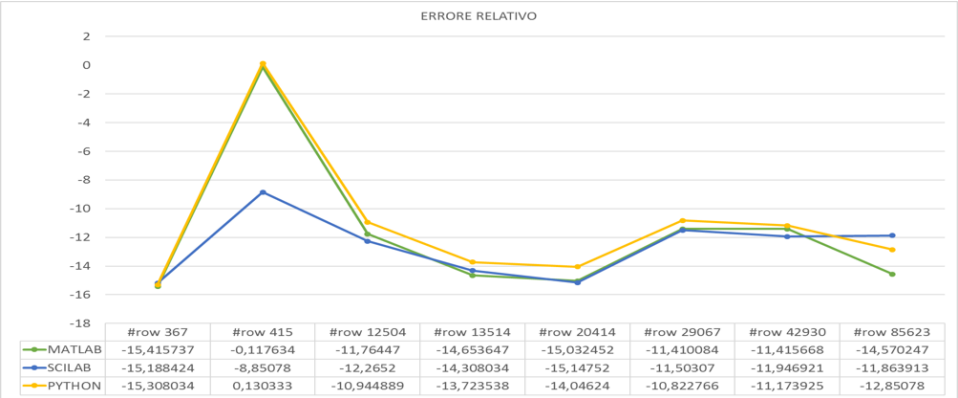


Figura 3.5: Errore relativo su Asus X550C.

Scilab ha dimostrato di aver un errore relativo inferiore rispetto agli altri due ambienti. Sull'ultima matrice non è stato possibile calcolarlo per mancata risoluzione (stessa cosa per *Python*).



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
MATLAB	-15,415737	-0,117634	-11,76447	-14,653647	-15,032452	-11,410084	-11,415668	-14,570247
SCILAB	-15,188424	-8,85078	-12,2652	-14,308034	-15,14752	-11,50307	-11,946921	-11,863913
PYTHON	-15,308034	0,130333	-10,944889	-13,723538	-14,04624	-10,822766	-11,173925	-12,85078

Figura 3.6: Errore relativo su MacBook Pro Retina Late 2012.

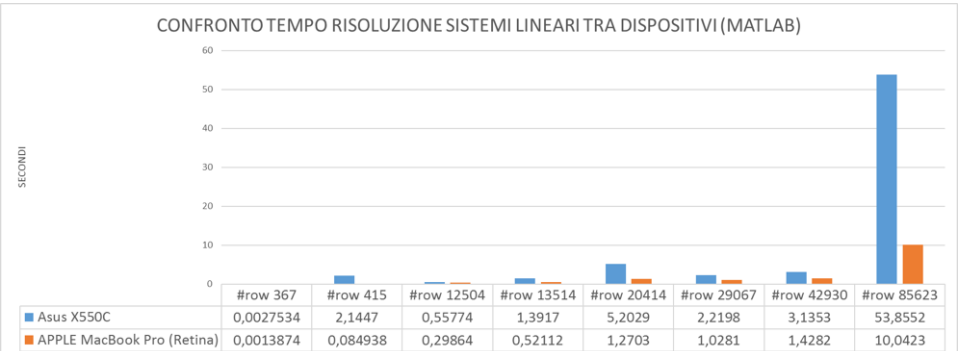
I dati rilevati su Mac dei tre ambienti sono gli stessi rilevati su ASUS X550C.

3.2 Confronto tra ambienti su macchine diverse

Viene ora riproposta la stessa logica adottata nella sezione precedente confrontando, tuttavia, gli ambienti di sviluppo, e quindi dei rispettivi linguaggi di programmazione, sulle due macchine.

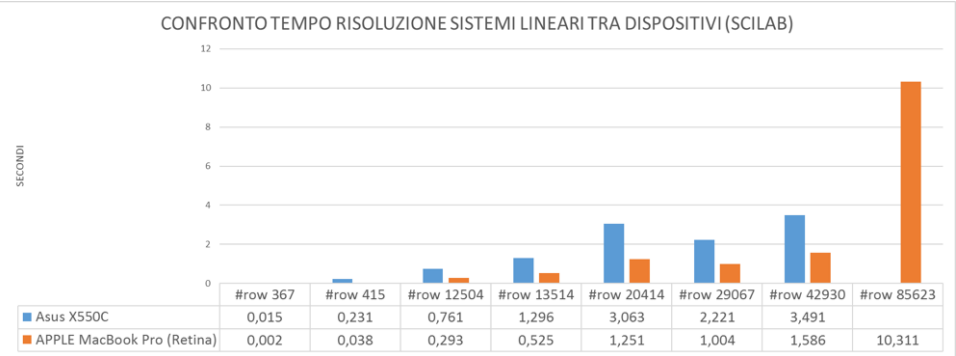
3.2.1 Tempo di risoluzione dei sistemi lineari

Dai grafici si intuisce che il Mac, per tutti e tre gli ambienti, ha fornito risultati più ottimistici rispetto all'Asus. Ciò dipende dall'hardware installato sulla macchina. Il divario dei tempi tra le macchine per i differenti ambienti è notevole.



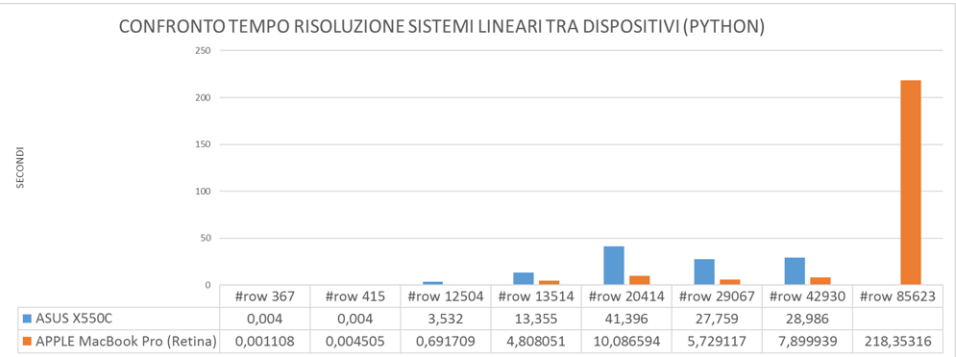
	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
Asus X550C	0,0027534	2,1447	0,55774	1,3917	5,2029	2,2198	3,1353	53,8552
APPLE MacBook Pro (Retina)	0,0013874	0,084938	0,29864	0,52112	1,2703	1,0281	1,4282	10,0423

Figura 3.7: Tempo di risoluzione in *Matlab* sulle due macchine.



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
Asus X550C	0,015	0,231	0,761	1,296	3,063	2,221	3,491	
APPLE MacBook Pro (Retina)	0,002	0,038	0,293	0,525	1,251	1,004	1,586	10,311

Figura 3.8: Tempo di risoluzione in *Scilab* sulle due macchine.

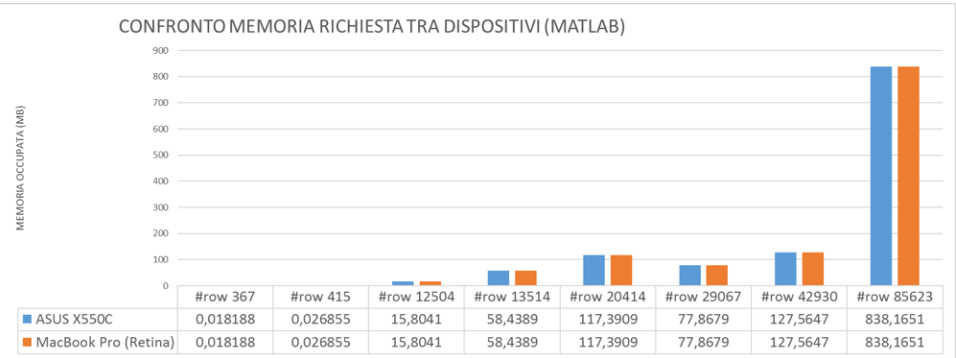


	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	0,004	0,004	3,532	13,355	41,396	27,759	28,986	
APPLE MacBook Pro (Retina)	0,001108	0,004505	0,691709	4,808051	10,086594	5,729117	7,899939	218,35316

Figura 3.9: Tempo di risoluzione in *Python* sulle due macchine.

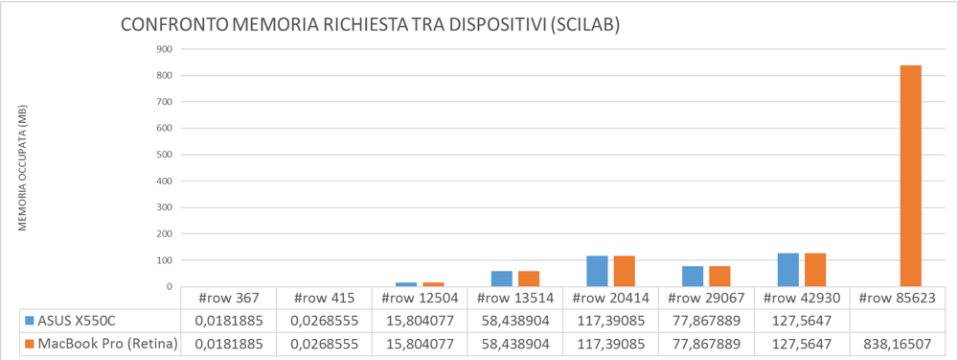
3.2.2 Occupazione memoria

La memoria occupata su entrambe le macchine da parte degli ambienti, è risultata uguale.



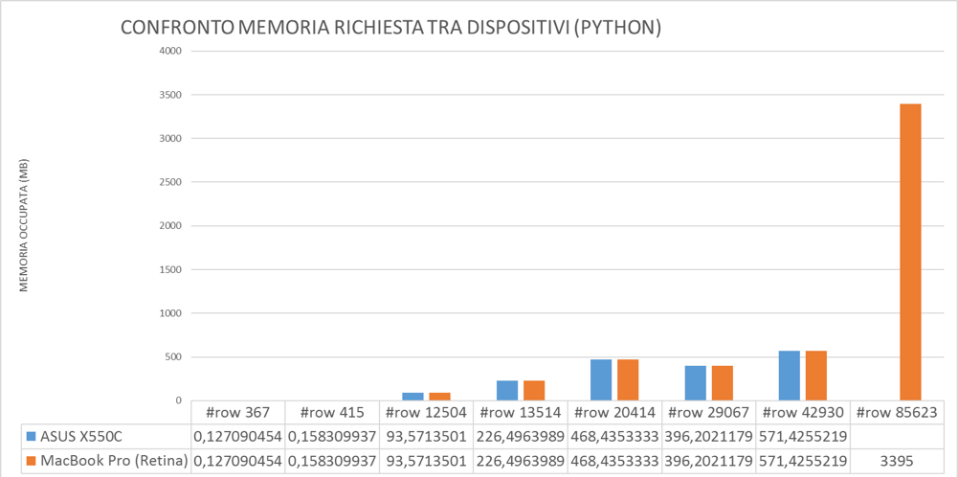
	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	0,018188	0,026855	15,8041	58,4389	117,3909	77,8679	127,5647	838,1651
MacBook Pro (Retina)	0,018188	0,026855	15,8041	58,4389	117,3909	77,8679	127,5647	838,1651

Figura 3.10: Memoria occupata in *Matlab* sulle due macchine.



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	0,0181885	0,0268555	15,804077	58,438904	117,39085	77,867889	127,5647	
MacBook Pro (Retina)	0,0181885	0,0268555	15,804077	58,438904	117,39085	77,867889	127,5647	838,16507

Figura 3.11: Memoria occupata in *Scilab* sulle due macchine.

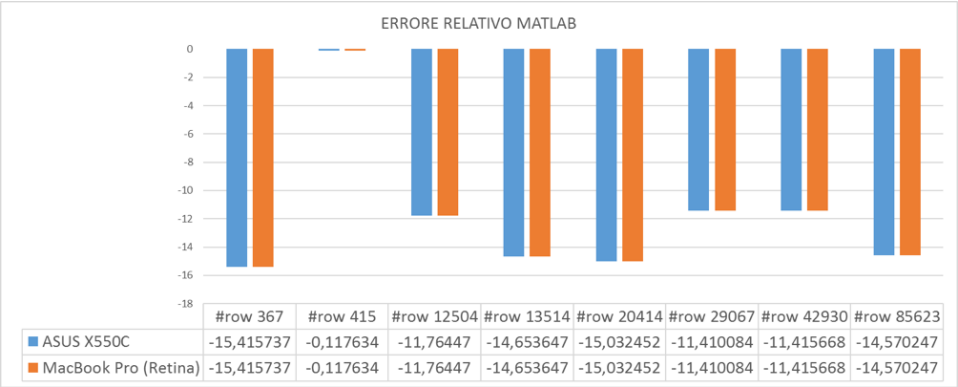


	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	0,127090454	0,158309937	93,5713501	226,4963989	468,4353333	396,2021179	571,4255219	
MacBook Pro (Retina)	0,127090454	0,158309937	93,5713501	226,4963989	468,4353333	396,2021179	571,4255219	3395

Figura 3.12: Memoria occupata in *Python* sulle due macchine.

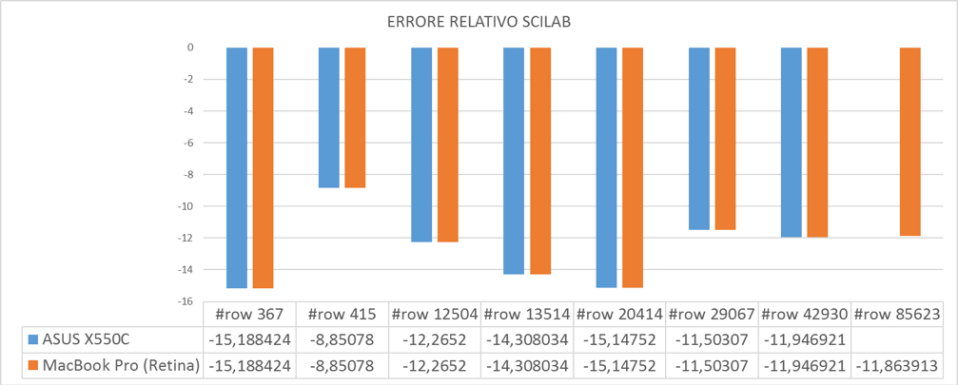
3.2.3 Errore relativo

L'errore relativo su entrambe le macchine da parte degli ambienti, è risultato uguale. Ciò può essere osservato dai grafici che seguono.



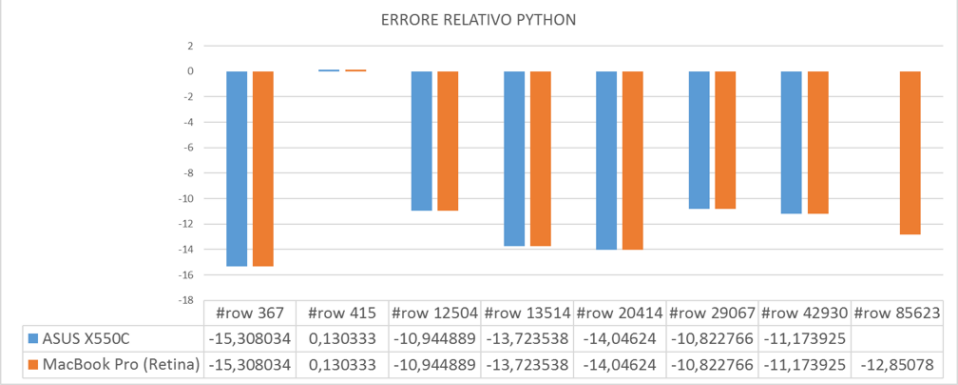
	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	-15,415737	-0,117634	-11,76447	-14,653647	-15,032452	-11,410084	-11,415668	-14,570247
MacBook Pro (Retina)	-15,415737	-0,117634	-11,76447	-14,653647	-15,032452	-11,410084	-11,415668	-14,570247

Figura 3.13: Errore relativo in *Matlab* sulle due macchine.



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	-15,188424	-8,85078	-12,2652	-14,308034	-15,14752	-11,50307	-11,946921	
MacBook Pro (Retina)	-15,188424	-8,85078	-12,2652	-14,308034	-15,14752	-11,50307	-11,946921	-11,863913

Figura 3.14: Errore relativo in *Scilab* sulle due macchine.



	#row 367	#row 415	#row 12504	#row 13514	#row 20414	#row 29067	#row 42930	#row 85623
ASUS X550C	-15,308034	0,130333	-10,944889	-13,723538	-14,04624	-10,822766	-11,173925	
MacBook Pro (Retina)	-15,308034	0,130333	-10,944889	-13,723538	-14,04624	-10,822766	-11,173925	-12,85078

Figura 3.15: Errore relativo in *Python* sulle due macchine.

Capitolo 4

Conclusioni

Durante i test sono state riscontrate marcate differenze, sia a livello di prestazioni di macchina che a livello di prestazioni di ambienti utilizzati. Dai grafici sopra riportati, confrontando gli ambienti testati su una stessa macchina, analizziamo le prestazioni relative a *Matlab*, *Scilab* e *Python* in termini di tempo, occupazione memoria, errore relativo.

4.1 Tempo di risoluzione dei sistemi lineari

- **Ambienti diversi, stessa macchina**: nel caso di matrici di piccole dimensioni (367x367, 415x415), non ci sono state differenze rilevanti tra gli ambienti, in termini di tempo per la risoluzione del sistema lineare. Nel caso di matrici di dimensioni considerevoli, le differenze di performance tra gli ambienti diventano più marcate. Analizzando i grafici del tempo di risoluzione, notiamo che *Matlab* e *Scilab* offrono risultati coincidenti; il tempo di risoluzione è simile (nel caso dei test effettuati su ASUS X550C) ed uguale (nel caso dei test effettuati su MacBook Pro Late 2012) per ogni matrice. In *Python* è stato riscontrato un tempo di risoluzione elevato dei sistemi lineari: 218 secondi di Python contro i 10 secondi di *Matlab* e *Scilab* per la risoluzione del sistema lineare della matrice con 85623 righe. In questo contesto, *Matlab* e *Scilab* superano *Python*, affermando la loro ottima rapidità nel calcolo scientifico.

- **Stesso ambiente, macchine diverse**: il tempo di risoluzione dei sistemi lineari in uno stesso ambiente cambia notevolmente da macchina

a macchina: ciò dipende dal processore che è installato sulla macchina, dal numero di core, dalla velocità di clock. Esempio con *Matlab*: sul MacBook Pro Late 2012, avendo un processore Intel Core i7 a 2.7 GHz il sistema lineare della matrice 85623x85623 viene risolto in circa 10 secondi, contro i 53 secondi dell'ASUS X550C, dotato di un processore Intel Core i3 a 1.80 GHz. Il distaccamento tra le due macchine, si fa più evidente in *Python*: su Mac abbiamo un tempo di risoluzione che arriva a essere circa $\frac{1}{4}$ di quello richiesto da Asus.

4.2 Occupazione memoria

```
 Dimensione matrice :85623x85623
    memoria_occupata=((((nnz(L) - nnz(A))*16)/1024)/1024);
                                !--error 17
superata la dimensione dello stack!
Usare la funzione stacksize per aumentarla.
Memoria usata per le variabili: 176130358
Memoria intermedia necessaria: 86000222
Memoria totale disponibile: 223739903
at line      35 of exec file called by :
Scilab\SistemiLineari.sce', -1
```

Figura 4.1: Errore di memoria sulla macchina Asus X550C.

- **Ambienti diversi, stessa macchina**: dai risultati si riscontra che *Matlab* e *Scilab*, usufruiscono dello stesso spazio di memoria. All'aumentare delle dimensioni della matrice coinvolta, lo spazio di memoria occupato aumenta in maniera contenuta. Durante la risoluzione del sistema che ha coinvolto la matrice di dimensione 85623 righe, lo spazio di memoria occupata è arrivata fino a raggiungere 838 MB. In *Python* è stato notato un utilizzo notevole di memoria, fino a un massimo di 3 GB per la risoluzione della matrice di 85623 righe. Tale differenza potrebbe essere dovuta dal fatto che la libreria *scipy* utilizzata in *Python* implementi in modo meno efficiente il metodo di fattorizzazione LU. Sul MacBook Pro è stato possibile risolvere tutti i sistemi lineari in tutti e tre gli ambienti, mentre sull'ASUS X550C, per *Scilab* e *Python*, non è stato possibile risolvere la matrice di 85623 righe (Poisson3Db),

a causa dello spazio eccessivo di memoria richiesto. In *Scilab* è stato restituito un messaggio di errore che avvisava che non c'era memoria sufficiente per eseguire i calcoli richiesti, portando ad una mancata restituzione dei dati (si riporta nell'immagine 4.1 il messaggio di errore generato). MacBook Pro è dotato di 16 GB di RAM, mentre l'ASUS di 4 GB.

- **Stesso ambiente, macchine diverse**: indipendentemente dalla macchina utilizzata, la memoria richiesta da ogni ambiente su entrambe le macchine è la stessa.

4.3 Errore relativo

Matlab	Scilab	Python
9,08E-02	**1,77E-10**	1,69E-01

Dal grafico dell'errore relativo medio tra i tre ambienti (figura 4.2), il migliore risulta essere *Scilab*. Il suo errore relativo risulta essere di gran lunga inferiore rispetto a *Python* e *Matlab*. Un ambiente in due macchine diverse, osservando dai grafici, ha lo stesso grado di errore.

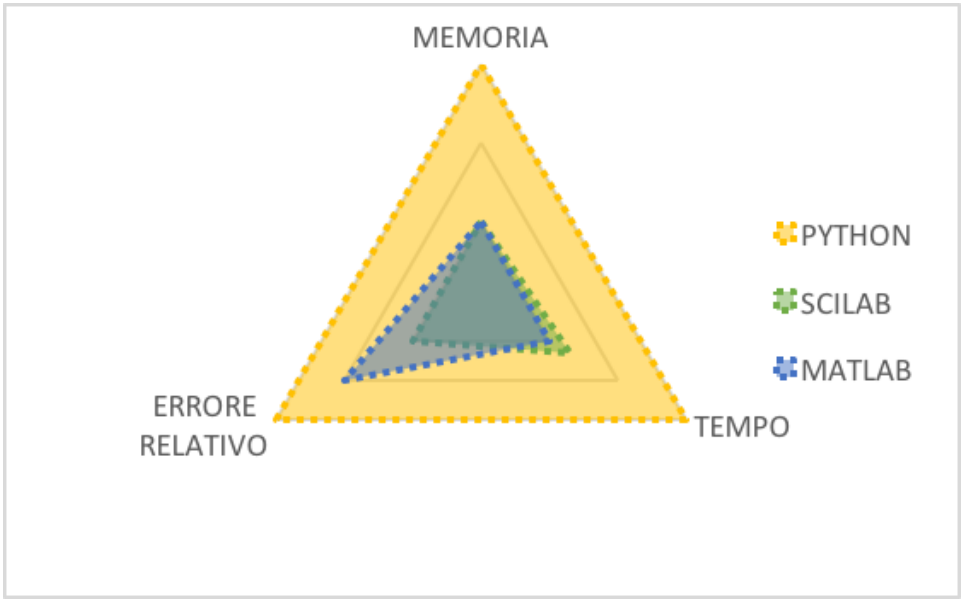


Figura 4.2: Errore di memoria sulla macchina Asus X550C.

4.4 Commento circa i risultati ottenuti

Concludendo, *Matlab* rimane il linguaggio di programmazione per calcolo scientifico migliore, in quanto ha permesso la risoluzione in entrambe le macchine di tutti i sistemi lineari proposti per l'attività. Le sue ottime performance in termini di memoria e tempo di risoluzione sono identiche a *Scilab*, anche se l'errore relativo risulta essere un po' più alto del suo rivale OpenSource *Scilab*.

Si sottolinea che la mancata scelta di *Java* per il progetto è stata dovuta alla scarsa performance delle librerie OpenSource trovate in rete come *la4j* (errore restituito: *Out of Memory*, ovvero eccessivo utilizzo della memoria). Per testare queste librerie, la dimensione della *Java Virtual Machine* è stata portata fino a 6 GB. Per evitare che si verificasse di nuovo l'eccezione della mancanza di memoria. Ciò non ha portato alcun beneficio.

Appendice A

Listato del codice *Matlab*

```
clear all
clc
percorso_matrici =  %DEFINITO DALL'UTENTE
lista_file_matrici= {'problem1.mat','poisson2D.mat','poisson3Da.mat','ns3Da.mat',
    'sme3Da.mat','sme3Db.mat','sme3Dc.mat','poisson3DB.mat'};
for i = 1 : length(lista_file_matrici)
    disp([num2str(i),') START']);
    %Carico file matrice
    mat = load(strcat(percorso_matrici,lista_file_matrici{i}));

    %CREAZIONE MATRICE A, VETTORE B, VETTORE XE
    %Instanzio la matrice A
    A = mat.Problem.A;
    disp(['Nome matrice: ',lista_file_matrici{i}]);
    disp(['Dimensione matrice :',num2str(size(A,1)),'x',num2str(size(A,2))]);
    %Creazione vettore colonna di soli 1
    xe = ones(length(A), 1);
    %Calcolo vettore B
    b = A * xe;

    %RISOLUZIONE SISTEMA E CALCOLO TEMPO RISOLUZIONE
    %Start conteggio del tempo
    tic;
    %Risoluzione sistema lineare
    x = A \ b;
    %Stop conteggio del tempo
    time = toc;

    %ERRORE RELATIVO E MEMORIA OCCUPATA
    %Calcolo l'errore relativo
    errore_relativo = norm(x - xe)/norm(xe);
    %Fattorizzazione LU della matrice A
    [L,U,P,Q,R] = lu(A);
    %Calcolo la memoria come numeri di non zeri che vengono creati
    %occupa 16 byte e viene diviso per 1024^2 per avere l'occupazione in MB
    memoria_occupata = ((((nnz(L) - nnz(A))*16)/1024)/1024);

    %RIASSUNTO: TEMPO IMPIEGATO, MEMORIA OCCUPATA, ERRORE RELATIVO
    disp(['Tempo necessario per eseguire il calcolo (secondi): ',num2str(time)]);
    disp(['Errore relativo: ',num2str(errore_relativo)]);
    disp(['Memoria occupata durante risoluzione sistema: ',num2str(
        memoria_occupata)]);
end
```

Appendice B

Listato del codice *Scilab*

```
clear
clc
stacksize('max');
percorso_matrici =  //DEFINITO DALL'UTENTE
lista_file_matrici= list("problem1.mtx","poisson2D.mtx","poisson3Da.mtx","ns3Da.
    mtx","sme3Da.mtx","sme3Db.mtx","sme3Dc.mtx","poisson3DB.mtx");
for i =1:size(lista_file_matrici);
    disp(string(i)+") START");
    //Carico file matrice
    mmread(percorso_matrici+lista_file_matrici(i));

    //CREAZIONE MATRICE A, VETTORE B, VETTORE XE
    //Carico file matrice
    A = ans;
    disp("Nome matrice: "+lista_file_matrici(i));
    disp("Dimensione matrice :"+string(size(A,1))+'x'+string(size(A,2)));
    //Creazione vettore colonna di soli 1
    xe = ones(length(A),1);
    //Calcolo vettore B
    b=A*xe;

    //RISOLUZIONE SISTEMA E CALCOLO TEMPO RISOLUZIONE
    //Start conteggio del tempo
    tic();
    //Risoluzione sistema lineare
    x = umfpack(A," \ ",b);
    //Stop conteggio del tempo
    time = toc();

    //ERRORE RELATIVO E MEMORIA OCCUPATA
    //Calcolo l'errore relativo
    errore_relativo = norm(x-xe)/norm(xe);
    //Fattorizzazione LU della matrice A
    LU_puntatore= umf_lufact(A);
    //Ottenimento matrice L
    [L,U,p,q,Rd] = umf_luget(LU_puntatore);
    //Calcolo la memoria come numeri di non zeri che vengono creati
    //occupa 16 byte e viene diviso per 1024^2 per avere l'occupazione in MB
    memoria_occupata=((((nnz(L) - nnz(A))*16)/1024)/1024);

    //RIASSUNTO: TEMPO IMPIEGATO, MEMORIA OCCUPATA, ERRORE RELATIVO
    disp("Tempo necessario per eseguire il calcolo (secondi): " + string(time));
    disp("Errore relativo: " + string(errore_relativo));
    disp("Memoria occupata durante risoluzione sistema: " + string(
        memoria_occupata) + "MB");
end
```

Appendice C

Listato del codice *Python*

```
1  from scipy.sparse import csc_matrix #Formato matrice di conversione
2  from scipy.io import mmread         #Funzione lettura file
3  import scipy as scipy               #Utilizzo del vettore scipy
4  from scipy.sparse import linalg     #Libreria algebra lineare
5  from scipy.linalg import norm       #Funzione norma euclidea
6  from datetime import datetime       #Time stamp
7
8  percorso_matrici= #DEFINITO DALL'UTENTE
9  for lista_file_matrici in ['problem1.mtx','poisson2D.mtx','poisson3Da.mtx','ns3Da
       .mtx','sme3Da.mtx','sme3Db.mtx','sme3Dc.mtx','poisson3Db.mtx']:
10         print("START")
11     #Carico file matrice ed effettuo conversione da file mtx a matrice sparsa csc
           (compressed sparse column)
12         A = mmread(percorso_matrici + lista_file_matrici).tocsc()
13         print ("Nome matrice: "+lista_file_matrici)
14         print ("Dimensione matrice :"+str(A.shape[0])+'x'+str(A.shape[1]))
15
16         #CREAZIONE MATRICE A, VETTORE B, VETTORE XE
17         #Creazione vettore colonna di soli 1
18         xe = scipy.ones(A.shape[0])
19         #Calcolo vettore B
20         b= A*xe
21         #Inizializzazione vettore soluzione
22         x = scipy.empty(A.shape[0])
23
24         #RISOLUZIONE SISTEMA E CALCOLO TEMPO RISOLUZIONE
25         #Start conteggio del tempo
26         inizio = datetime.now()
27         #Risoluzione sistema lineare
28         x=scipy.sparse.linalg.spsolve(A,b,use_umfpack=True)
29         #Stop conteggio del tempo
30         fine=datetime.now() - inizio
31
32         #ERRORE RELATIVO E MEMORIA OCCUPATA
33         #Calcolo l'errore relativo
34         errore_relativo= norm(x-xe)/norm(xe)
35         #Fattorizzazione LU della matrice A
36         LU=linalg.splu(A)
37         #Calcolo la memoria come numeri di non zeri che vengono creati
38         memoria_occupata=float(float((LU.nnz-A.nnz)*16)/1024)/1024
39         #Occupa 16 byte e viene diviso per 1024^2 per avere l'occupazione in MB
40
41         print ("Tempo necessario per eseguire il calcolo (secondi): " + str(fine)
               )
42         print ("Errore relativo: "+str(errore_relativo))
43         print ("Memoria occupata durante risoluzione sistema: "+str(
               memoria_occupata)+" MB")
```