Metodi del Calcolo Scientifico
Relazione Progetto 1

A.A. 2019-2020, Appello 04/06/2020

Componenti gruppo:

- Christian Bernasconi 816423
- Gabriele Ferrario 817518
- Riccardo Pozzi 807857
- Marco Ripamonti 806785

Link al repository: `https://gitlab.com/m.ripamonti/mcs_projects`

Indice

1 Introduzione

In questa relazione verrà presentato un confronto tra Matlab e ambienti gratuiti ed open source nei sistemi operativi di Windows e Linux. Tale confronto sarà fatto sia a livello di usabilità, sia in termini di performance ottenute dall'implementazione del metodo di Cholesky per la risoluzione di sistemi lineari per matrici sparse, simmetriche e definite positive.

Al capitolo 2 sono illustrati ambienti e librerie che sono stati presi in considerazione in una fase di analisi preliminare, corredati delle motivazioni che hanno portato alla scelta di Octave come ambiente open source. Proseguendo al capitolo 3 è presentata una sintesi delle funzioni implementate, seguita dalla descrizione delle fasi con cui è stata affrontata l'analisi. Il capitolo 4 verte sul confronto dell'usabilità nei diversi ambienti e sull'analisi tramite grafici dei risultati di performance ottenuti. Infine, nel capitolo 5 sono esposte le considerazioni finali in merito ai risultati del confronto.

2 Ambienti e librerie

Inizialmente sono stati considerati diversi ambienti e librerie; di seguito vengono riportate le loro caratteristiche. Nella sezione 2.6 viene discussa la scelta della libreria adottata in questa analisi.

2.1 Matlab

Matlab, da Matrix Laboratory, è un ambiente utilizzato principalmente per il calcolo numerico e l'analisi statistica. Le prime versioni di Matlab risalgono agli anni 80 quando Cleve Moler, professore di Algebra Lineare e Analisi Numerica presso L'Università del New Mexico, sente l'esigenza di fornire ai suoi studenti un facile accesso agli strumenti matematici forniti dalle attuali librerie e linguaggi di programmazione [1]. Matlab è infatti basato su un linguaggio matrix-based che accetta gran parte delle naturali espressioni matematiche [2]. L'attuale versione fornisce un ambiente desktop ottimizzato per l'analisi interattiva consentendo per esempio di visualizzare funzioni e matrici.
In più non si limita ad un'implementazione desktop ma rende possibile scalare le analisi su cluster, GPU e cloud.
Matlab è distribuito sotto licenza proprietaria, a pagamento, da "The MathWorks, Inc." [3].

2.2 GNU Octave

GNU Octave è un linguaggio di programmazione di alto livello rivolto principalmente all'analisi numerica. La sua sintassi è largamente compatibile con quella di Matlab rendendo possibile la scrittura di codice supportato da entrambi. Lo sviluppo di GNU Octave inizia intorno al 1988 per esigenze simili a quelle che hanno portato alla nascita di Matlab: James B. Rawlings dall'Università del Wisconsin-Madison e John G. Ekerdt dall'Università del Texas ritenevano che gli studenti perdessero troppo tempo nel sistemare programmi Fortran piuttosto che dedicarlo all'ingegneria chimica [4].
Anche Octave fornisce un ambiente desktop interattivo in grado di visualizzare semplicemente funzioni o matrici.
GNU Octave è distribuito gratuitamente e sotto licenza libera, più precisamente sotto la GNU General Public License (GPL) [5].

2.3 FreeMat

FreeMat è un software libero e installabile gratuitamente, distribuito sotto la licenza GNU General Public License (GPL) [5], per analisi computazionale e numerica. Anch'esso largamente compatibile con Matlab, nasce con l'obiettivo di offrire funzionalità avanzate come il semplice utilizzo di librerie scritte in altri linguaggi di programmazione, lo sviluppo di algoritmi paralleli e distribuiti e la visualizzazione 3D.
Tuttavia il progetto appare non attivamente mantenuto: secondo il sito ufficiale l'ultima release di FreeMat risale al 2013 [6].

2.4 Scilab

Scilab è un software libero e distribuito gratuitamente sotto la licenza GNU General Public License (GPL) [5]. Nato negli anni 90 per conto di Inria (Institut national de recherche en sciences et technologies du numérique) e Scilab Group, fornisce strumenti per operazioni matematiche e di data analysis, così come per visualizzazioni di grafici 2D e 3D ed include funzionalità di simulazione nei campi di signal processing, meccanica e termodinamica [7]. Ha una sintassi simile a quella di Matlab, ma alcuni studi lo riportano come meno performante di matlab e octave [8].

2.5 Python - SciPy/NumPy

Python è un linguaggio di programmazione ad alto livello, interpretato e orientato agli oggetti [9] nato negli anni 90 [10] e caratterizzato da una sintassi considerata semplice e di facile apprendimento, da una vasta community e da una grande varietà di pacchetti che ne permettono l'utilizzo in svariati campi.
È gratuito e distribuito sotto licenze open-source GPL-compatibili [10].
SciPy è un "ecosistema basato su python rivolto alla matematica, alle scienze e all'ingegneria" [11]. Tra i suoi "core packages" troviamo NumPy: il pacchetto rivolto al calcolo scientifico che fornisce ad esempio supporto per array N-dimensionali e funzioni relative all'algebra lineare, trasformazioni di Fourier e generazione di numeri casuali [12]. L'utilizzo di entrambi è gratuito e sono distribuiti sotto licenza opensource (BSD 3-Clause) [13] [14].

2.6 Valutazione ambienti e librerie open source

Con lo scopo di confrontare un ambiente/libreria a pagamento con licenza proprietaria come Matlab con un software gratuito ed open source, sono stati valutati tutti gli ambienti e le librerie sopracitate.

FreeMat è stato scartato perché non attivamente mantenuto. L'ecosistema Python+SciPy/NumPy, pur vantando di una vasta community, non fornisce alcune funzioni presenti in Matlab (e Octave) utilizzate durante il progetto; ad esempio la permutazione di matrice utilizzata per ottimizzare la decomposizione di Cholesky, come mostrato nella sezione 3.2. Scilab si è dimostrato un progetto valido, tuttavia la preferenza è ricaduta su GNU Octave una volta a conoscenza di studi che ne ritengono le prestazioni inferiori e meno consistenti rispetto a Matlab e Octave [8].

3 Sviluppo

In questa sezione verrà mostrato il procedimento adottato per la fase sperimentale di questo lavoro.
Verranno prima descritte brevemente le principali funzioni implementate, poi illustrate le varie fasi dello sviluppo che hanno portato alla realizzazione del progetto. In particolare il lavoro è stato svolto e completato a partire da Matlab, dopodiché si è passati allo studio di Octave con cui si è verificata la riproducibilità di quanto fatto precedentemente. Dato che per entrambi gli ambienti si è cercato di mantenere la medesima struttura del codice, le funzioni e le fasi presentate in seguito possono essere generalizzate rispetto ai due softwares.

3.1 Funzioni implementate

Il codice sorgente di questo progetto è disponibile all'indirizzo `https://gitlab.com/m.ripamonti/mcs_projects/-/tree/master/proj1/src` .
Le funzioni principali su cui si basa il lavoro sono le seguenti:

- [A, x, b] = readMatrix(filename, usesymamd): prende in input il nome *filename* del file della matrice e un flag *usesymamd* che specifica se effettuare una permutazione sulla matrice; viene restituita in A la matrice e, per comodità, in x il vettore di 1 delle incognite e in b il vettore del termine noto (calcolato a parte dalle x).

- [err] = relError(xEs, xApp): prende in input la soluzione esatta *xEs* e la soluzione approssimata *xApp* e restituisce in *err* l'errore relativo calcolato.

- [xApp] = solveWithCholesky(A, b): prende in input una matrice A e un vettore dei termini noti b, effettua la decomposizione di Cholesky e risolve il sistema; restituisce in *xApp* la soluzione approssimata.

- [err, time] = matrixAnalyzer(filename, usesymamd): prende in input il nome *filename* del file della matrice e il flag *usesymamd* che specifica se effettuare una permutazione sulla matrice e, appoggiandosi ai metodi precedentemente illustrati, legge la matrice, risolve il sistema e calcola l'errore relativo; vengono restituiti in *err* l'errore relativo e in *time* il tempo impiegato nella risoluzione del sistema.

- singlePlotter(table, filterField, filterValue, compareField, compareValue): prende in input una tabella *table* nel formato csv[1] che viene filtrata in base ai

[1] `https://gitlab.com/m.ripamonti/mcs_projects/-/blob/master/proj1/reports.csv`

campi *filterField* e *compareField* con i rispettivi valori *filterValue* e *compareValue* e disegna un grafico sulla base dei campi scelti; nel grafico l'asse delle X rappresenta le dimensioni della matrice e l'asse Y, in scala logaritmica, rappresenta tempo, errore e memoria.

- comparePlotter(table, filterField, filterValue, compareField, compareValue1, compareValue2): prende in input una tabella *table* nel formato csv che viene filtrata in base ai campi *filterField* e *compareField* con i rispettivi valori *filterValue*, *compareValue1*, *compareValue2* e disegna un grafico sulla base dei campi scelti per il confronto; nel grafico l'asse delle X rappresenta le dimensioni della matrice e l'asse Y, in scala logaritmica, rappresenta tempo, errore e memoria.

- memoryPlotter(matrixName): prende in input il nome *matrixName* di una matrice e traccia un grafico della variazione di memoria utilizzata durante la risoluzione del sistema in funzione del tempo.

- [memory] = memoryReadDelta(filename): prende in input il nome di un file *filename* da cui legge i dati sull'utilizzo di memoria durante la risoluzione del sistema e ne calcola l'escursione togliendo la memoria base usata dal processo; i nuovi valori della memoria vengono restituiti in *memory*

Le funzioni implementate sono state documentate ed è possibile consultarle tramite il comando *help* sia in Matlab che in Octave.

3.2 Fase 1: Lettura matrici e permutazione

Per prima cosa è stata implementata la lettura delle matrici attraverso il metodo *readMatrix*. La lettura avviene mantenendo la natura sparsa delle matrici, le quali vengono caricate nel workspace da file *.mat*. Dato lo scopo del progetto, per comodità sono stati direttamente integrati nel metodo di lettura anche la generazione dei vettori delle incognite e del termine noto che saranno poi utilizzati per la risoluzione del sistema e per il calcolo dell'errore relativo.
Come è stato anticipato durante la descrizione delle funzioni, è stato necessario introdurre la possibilità di richiedere una permutazione sulle matrici dopo la loro lettura. Questo è dovuto al fatto che il metodo di Cholesky nell'effettuare la decomposizione di alcune matrici sparse potrebbe incorrere nel problema del fill-in. Per ovviare a questo problema, esistono diverse strategie di ordinamento che permettono di ridurre il numero di elementi diversi da zero derivanti dalla decomposizione. Tra le possibilità di permutazione è stata scelta quella basata

sull'algoritmo di *approximate minimum degree (amd)*[2][3]. In particolare, dato che tutte le matrici da analizzare sono simmetriche e definite positive, è stata utilizzata l'apposita versione *symamd*[4][5]. Come si vedrà nella sezione 4.2.2, ciò porterà a notevoli benefici nel calcolo della decomposizione di Cholesky.

3.3 Fase 2: Procedura di risoluzione di un sistema

Per effettuare la risoluzione di un sistema sfruttando la decomposizione di Cholesky è stato implementato il metodo *solveWithCholesky*. Per prima cosa viene effettuata la decomposizione tramite il metodo *chol*[6][7]. Per entrambi gli ambienti questo metodo è in grado di controllare che la matrice passata sia definita positiva e restituisce di default una matrice triangolare superiore. L'unica differenza sta nel fatto che Matlab è in grado di gestire il caso in cui la matrice non sia simmetrica, dando la possibilità di restituire un errore. Octave, invece, procede ugualmente considerando la parte triangolare inferiore come trasposta della superiore o viceversa. Dopo aver effettuato la decomposizione e aver ottenuto la matrice R tale che soddisfi $A = R^T R$, dove A è la matrice originale, si può procedere alla risoluzione del sistema iniziale eseguendo in sequenza i seguenti due passi:

1. Risoluzione del sistema $R^T y = b$ mediante il metodo di sostituzione in avanti.
2. Risoluzione del sistema $Rx = y$ mediante il metodo di sostituzione all'indietro e ricavare la soluzione x approssimata.

Entrambi i sistemi vengono risolti mediante l'operatore '\'[8] che riconoscendo rispettivamente le caratteristiche delle due matrici, la prima triangolare inferiore e la seconda triangolare superiore, applica rispettivamente il metodo di sostituzione in avanti e il metodo di sostituzione all'indietro [15].

3.4 Fase 3: Raccolta dati relativi alle performance

Per agevolare la raccolta dei dati sulle performance per ogni matrice, il metodo implementato è quello di *matrixAnalyzer*. Questo metodo era stato inizialmente

[2] `https://it.mathworks.com/help/matlab/ref/amd.html`
[3] `https://octave.sourceforge.io/octave/function/amd.html`
[4] `https://it.mathworks.com/help/matlab/ref/symamd.html`
[5] `https://octave.sourceforge.io/octave/function/symamd.html`
[6] `https://it.mathworks.com/help/matlab/ref/chol.html`
[7] `https://octave.sourceforge.io/octave/function/chol.html`
[8] `https://it.mathworks.com/help/matlab/ref/mldivide.html#bt42oms_head`

predisposto in Matlab per calcolare errore relativo, tempo impiegato per la risoluzione e picco di memoria. In Octave, però, non esiste alcuna funzione che permetta di monitorare l'utilizzo della RAM. È stato quindi necessario intraprendere un'altra strada per mantenere coerenza rispetto al metodo di rilevazione di tale misura di performance. La soluzione adottata è stata quella di utilizzare degli scripts[9] in modo da tracciare i valori della memoria occupata dai processi di Matlab/Octave ad intervalli di tempo regolari durante la risoluzione del sistema.
I passaggi con cui sono stati raccolti i dati sono i seguenti per ogni matrice, ambiente e sistema operativo:

1. avvio dello script di monitoraggio memoria
2. esecuzione metodo *matrixAnalyzer*
3. inserimento risultati ottenuti in un csv[10]

3.5 Fase 4: Visualizzazione risultati

Per quanto riguarda la parte di analisi, sono stati implementati i metodi *singlePlotter* e *comparePlotter* per produrre i grafici richiesti per la relazione. Questi metodi hanno permesso di automatizzare la lettura da csv per disegnare in modo efficace i grafici sulle performance date le varie combinazioni software/OS.
Inoltre, è stato reputato interessante visualizzare dei grafici aggiuntivi contenenti lo storico delle variazioni di memoria nel tempo per ogni matrice a seconda delle diverse configurazioni. A questo scopo è stata implementata la funzione *memoryPlotter*.
I grafici prodotti in questa fase saranno discussi in dettaglio alla sezione 4.3.

[9] `https://gitlab.com/m.ripamonti/mcs_projects/-/tree/master/proj1/scripts`
[10] `https://gitlab.com/m.ripamonti/mcs_projects/-/blob/master/proj1/reports.csv`

4 Confronto ambienti

In questa sezione verranno discusse le differenze riscontrate durante l'implementazione e l'analisi dei diversi ambienti. Si discuteranno gli aspetti riguardanti l'usabilità nella sezione 4.1 e gli aspetti prettamente legati alle performance nella sezione 4.2.

4.1 Usabilità

4.1.1 Matlab

Documentazione Matlab è tutt'oggi un software molto utilizzato sia professionalmente, sia in ambito accademico. Il software è mantenuto e anche la sua documentazione resta aggiornata e precisa. La documentazione, oltre che essere consultabile sul web, è facilmente reperibile all'interno dell'ambiente con l'utilizzo della direttiva *help* seguita dalla funzione per cui si vuole richiedere la documentazione.

Facilità d'uso Il software è installabile recandosi sul sito `https://it.mathworks.com/downloads/`. Essendo un software a pagamento è richiesta una licenza per poter usufruire dei suoi servizi. Il suo utilizzo si è dimostrato semplice e abbastanza intuitivo avendo anche il supporto di un'ottima documentazione.

Community Un altro aspetto che è stato valutato è stato quello della community. La community di Matlab si dimostra essere molto attiva e partecipante. Mathworks offre un servizio simile a quello offerto da `https://stackoverflow.com/` limitato però esclusivamente al software di Matlab. È possibile, infatti, porre domande riguardo ad un problema riscontrato durante lo sviluppo di un proprio script, oppure riguardo al corretto utilizzo delle funzioni offerte dal software. Gli utenti, quindi, sono resi partecipi dando la possibilità di rispondere a queste domande aiutando e risolvendo problemi riscontrati da altre persone. Questo strumento permette, inoltre, di condividere funzioni custom, sviluppate da utenti, che possono essere scaricate e integrate all'interno del proprio lavoro.

Problemi riscontrati Durante l'utilizzo di questo software non sono stati riscontrati particolari problemi.

4.1.2 Octave

Documentazione Anche GNU Octave è un software tutt'oggi mantenuto. Esso offre una documentazione ricca e aggiornata offrendo esempi e modalità d'uso

delle sue funzioni. Pur dichiarandosi strettamente compatibile con Matlab molte funzioni sembrano non essere però ancora implementate[11]. Come Matlab, Octave offre la possibilità di consultare la documentazione online e direttamente all'interno dell'ambiente con la stessa direttiva *help* utilizzata in Matlab.

Facilità d'uso Il software è completamente open source e non ha bisogno di alcuna licenza a pagamento. Può essere scaricato gratuitamente dal sito `https://www.gnu.org/software/octave/download.html`. L'installazione richiede un tempo minore rispetto a Matlab necessitando di uno spazio su disco inferiore (Da circa 5GB per Matlab a circa 2GB per Octave). Il suo utilizzo risulta essere semplice offrendo, per quanto già detto in precedenza, la stessa sintassi di Matlab.

Community Octave, a differenza di Matlab, non offre alcuna possibilità di interagire tra utenti e problematiche riscontrate durante lo sviluppo di un proprio script. Questo risulta in una maggiore difficoltà di ricerca di un problema, tanto da finire a consultare i forum di Mathworks. Pur essendo in gran parte compatibile, ci sono comunque alcune differenze tra i due ambienti.

Problemi riscontrati Durante la realizzazione di questa analisi è stato necessario leggere un file csv contenente diverse tipologie di dato. Le funzioni offerte da Octave non offrono però questa possibilità costringendo nel fare l'operazione in Matlab. Inoltre, come già accennato nella sezione 3.4, in Octave non esiste la possibilità di monitorare l'impiego di memoria RAM utilizzato per una data funzione contrariamente a come avviene in Matlab. È stata infine riscontrata un'imprecisione nell'errore ricevuto in seguito al fallimento della decomposizione di Cholesky (Figura 1).

[11] funzioni come *memory* e *readtable* restituiscono *"function is not yet implemented in Octave"*

```
warning: warning -2, at line 146 in file ../Core/cholmod_memory.c:
    out of memory
warning: called from
    solveWithCholesky at line 2 column 9
    matrixAnalyzer at line 5 column 16
error: chol: input matrix must be positive definite
error: called from
    solveWithCholesky at line 2 column 9
    matrixAnalyzer at line 5 column 16
```

Figura 1: Octave - imprecisione errore

Come riportato nel primo warning si incorre in un problema di *out of memory*, tuttavia Octave segnala come errore *chol: input matrix must be positive definite*. Ciò è inesatto in quanto la matrice data in input è definita positiva e l'errore è effettivamente causato dall'*out of memory*.

4.2 Analisi risultati ottenuti durante la fase operativa

La fase sperimentale è avvenuta su una macchina avente le seguenti specifiche:

- **CPU**: Intel Core i5 8250u
- **RAM**: 8GB
- **Sistema Linux**: Arch Linux / Kernel: 5.6.3-arch1-1
- **Sistema Windows**: Windows 10
- **Versione Matlab**: 9.7.0.1319299 (R2019b)
- **Versione Octave**: 5.2.0

Di seguito vengono riportati i dati e i grafici per valutare velocità, errore ed occupazione di memoria considerando ogni ambiente e ogni sistema operativo. Tutti i grafici sono caratterizzati da due assi: asse X rappresentante la dimensione della matrice e asse Y (in scala logaritmica) rappresentante l'errore, il tempo e il picco di occupazione di memoria che si sono verificati durante la risoluzione dei sistemi lineari ottenuti tramite la rispettive matrici.

I risultati che sono stati riportati fanno riferimento alle seguenti matrici: apache2, cfd1, cfd2, ex15, G3_circuit, Flan_1565, parabolic_fem e shallow_water1 e StocF-1465 (in rosso sono state riportate le matrici la cui decomposizione non è stata possibile per problemi di memoria causati dal fill-in).

4.2.1 Tabella riassuntiva

name	os	sw	perm	err	time	mem	size
apache2	linux	matlab	false	-1	-1	-1	82807336
apache2	linux	matlab	true	3,81E-07	295.191	6,49E+13	82807336
apache2	linux	octave	false	-1	-1	-1	82807336
apache2	linux	octave	true	3,4479E-11	481.47	6858836000	82807336
apache2	windows	matlab	false	-1	-1	-1	82807336
apache2	windows	matlab	true	3,81E-07	351.746	5,13E+13	82807336
apache2	windows	octave	false	-1	-1	-1	82807336
apache2	windows	octave	true	4,307E-11	77.811	5,39E+13	82807336
cfd1	linux	matlab	false	3,14E-09	172.532	3,35E+13	29774536
cfd1	linux	matlab	true	1,15E-09	57.312	2,02E+13	29774536
cfd1	linux	octave	false	1,36E-08	169.14	3889228000	29774536
cfd1	linux	octave	true	2,27E-09	43.227	1700028000	29774536
cfd1	windows	matlab	false	3,48E-09	133.504	3,04E+13	29774536
cfd1	windows	matlab	true	1,24E-09	47.819	1,64E+13	29774536
cfd1	windows	octave	false	1,62E-09	20.633	3,82E+13	29774536
cfd1	windows	octave	true	2,61E-10	99.243	1,67E+13	29774536
cfd2	linux	matlab	false	8,60E-09	235.094	5,65E+13	50354024
cfd2	linux	matlab	true	4,20E-09	95.382	3,22E+13	50354024
cfd2	linux	octave	false	9,68E-09	306.04	6774068000	50354024
cfd2	linux	octave	true	6,51E-09	160.21	3666280000	50354024
cfd2	windows	matlab	false	9,16E-09	278.257	5,19E+13	50354024
cfd2	windows	matlab	true	3,88E-09	120.449	3,22E+13	50354024
cfd2	windows	octave	false	6,17E-09	45.510	5,12E+13	50354024
cfd2	windows	octave	true	4,07E-09	19.298	3,63E+13	50354024
ex15	linux	matlab	false	8,57E-03	0.1837	913804000	1633696
ex15	linux	matlab	true	7,97E-03	0.0337	916172000	1633696
ex15	linux	octave	false	8,2138E-07	0.30051	203712000	1633696
ex15	linux	octave	true	8,4804E-07	0.048232	204096000	1633696
ex15	windows	matlab	false	8,57E-03	0.1853	730296320	1633696
ex15	windows	matlab	true	7,86E-03	0.0393	782589952	1633696
ex15	windows	octave	false	9,0188E-07	39.201	70803456	1633696
ex15	windows	octave	true	8,8238E-07	0.22685	65843200	1633696
Flan_1565	linux	matlab	false	-1	-1	-1	1839164312
Flan_1565	linux	matlab	true	-1	-1	-1	1839164312
Flan_1565	linux	octave	false	-1	-1	-1	1839164312
Flan_1565	linux	octave	true	-1	-1	-1	1839164312
Flan_1565	windows	matlab	false	-1	-1	-1	1839164312

Flan_1565	windows	matlab	true	-1	-1	-1	1839164312
Flan_1565	windows	octave	false	-1	-1	-1	1839164312
Flan_1565	windows	octave	true	-1	-1	-1	1839164312
G3_circuit	linux	matlab	false	-1	-1	-1	135257048
G3_circuit	linux	matlab	true	4,12E-08	307.832	6,71E+13	135257048
G3_circuit	linux	octave	false	-1	-1	-1	135257048
G3_circuit	linux	octave	true	3,26E-08	310.96	6805784000	135257048
G3_circuit	windows	matlab	false	-1	-1	-1	135257048
G3_circuit	windows	matlab	true	4,16E-08	554.263	4,47E+13	135257048
G3_circuit	windows	octave	false	-1	-1	-1	135257048
G3_circuit	windows	octave	true	5,43E-08	107.64	5,63E+13	135257048
parabolic_fem	linux	matlab	false	-1	-1	-1	63000608
parabolic_fem	linux	matlab	true	1,15E-08	32.170	1,66E+13	63000608
parabolic_fem	linux	octave	false	-1	-1	-1	63000608
parabolic_fem	linux	octave	true	9,68E-09	17.643	1717712000	63000608
parabolic_fem	windows	matlab	false	-1	-1	-1	63000608
parabolic_fem	windows	matlab	true	1,08E-08	51.456	1,84E+13	63000608
parabolic_fem	windows	octave	false	-1	-1	-1	63000608
parabolic_fem	windows	octave	true	1,29E-08	14.348	1,18E+13	63000608
shallow_water1	linux	matlab	false	3,20E-12	103.291	1,52E+13	5898248
shallow_water1	linux	matlab	true	2,70E-12	0.3007	912056000	5898248
shallow_water1	linux	octave	false	3,20E-12	44.299	1100452000	5898248
shallow_water1	linux	octave	true	2,71E-12	21.405	189148000	5898248
shallow_water1	windows	matlab	false	3,20E-12	136.709	1,14E+13	5898248
shallow_water1	windows	matlab	true	2,70E-12	0.3066	552861696	5898248
shallow_water1	windows	octave	false	3,09E-12	27.158	1,21E+13	5898248
shallow_water1	windows	octave	true	2,66E-12	14.395	84443136	5898248
StocF-1465	linux	matlab	false	-1	-1	-1	347807328
StocF-1465	linux	matlab	true	-1	-1	-1	347807328
StocF-1465	linux	octave	false	-1	-1	-1	347807328
StocF-1465	linux	octave	true	-1	-1	-1	347807328
StocF-1465	windows	matlab	false	-1	-1	-1	347807328
StocF-1465	windows	matlab	true	-1	-1	-1	347807328
StocF-1465	windows	octave	false	-1	-1	-1	347807328
StocF-1465	windows	octave	true	-1	-1	-1	347807328

4.2.2 Symamd vs NoSymamd

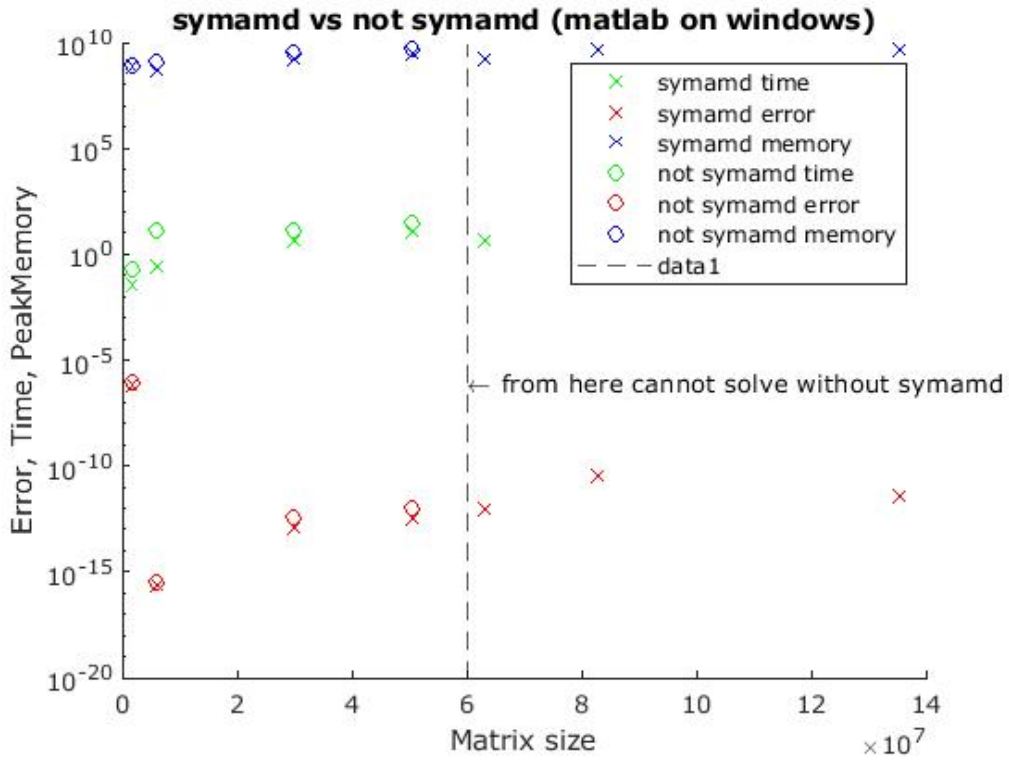


Figura 2: symamd vs not symamd

Nel grafico è riportato il confronto delle performance ottenute con e senza la permutazione basata su *symamd*. Analizzando il grafico si nota che la funzione *symamd* migliora l'efficienza della decomposizione, rendendola anche possibile su matrici che altrimenti causerebbero fill-in.

4.2.3 Matlab

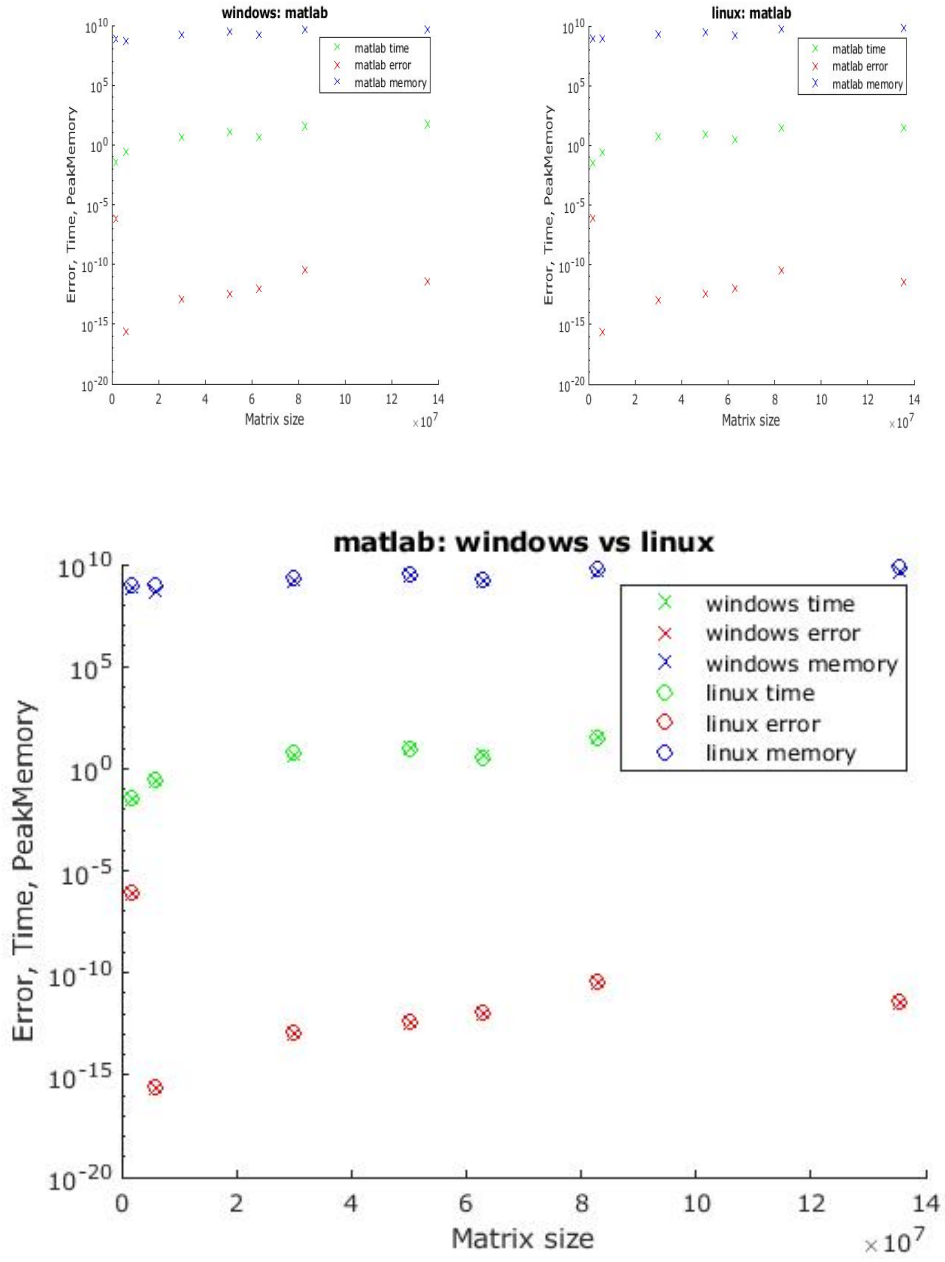
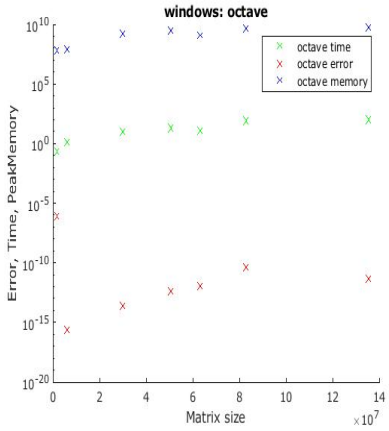


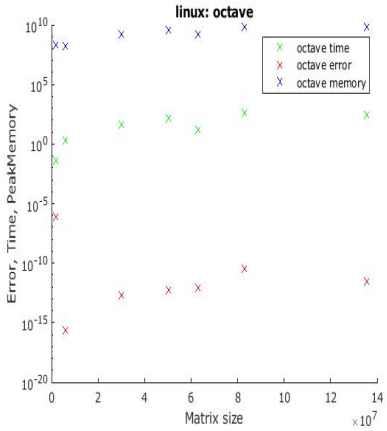
Figura 3: Risultati ottenuti con Matlab

Prendendo in considerazione il grafico in figura 3 sulle performance di Matlab, si può notare un andamento di tutte le misure che va di pari passo per entrambi i sistemi operativi.
Per quanto riguarda l'errore si può notare un outsider. La matrice di dimensione inferiore (ex15) presenta il valore peggiore, mentre per le altre matrici l'errore tende a crescere all'aumentare delle dimensioni. Questo caso particolare si ripresenta anche nelle altre combinazioni di ambiente e sistema operativo.

Si può inoltre osservare che tra i due sistemi operativi Windows è leggermente migliore nell'utilizzo di memoria.

4.2.4 Octave





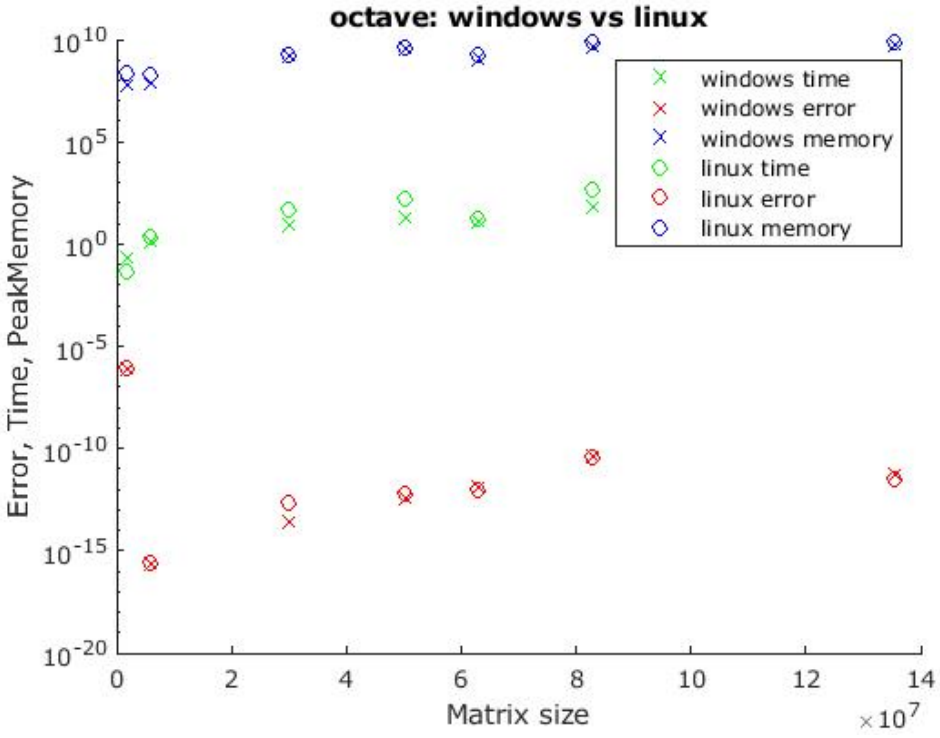


Figura 4: Risultati ottenuti con Octave

Nel grafico sulle performance di Octave in figura 4, si possono notare differenze molto più marcate tra Linux e Windows. Per quanto riguarda l'errore, l'andamento è ambiguo e non significativo per un confronto. Per la memoria e soprattutto per il tempo, invece, è possibile notare che Windows si dimostra superiore.

4.2.5 Linux

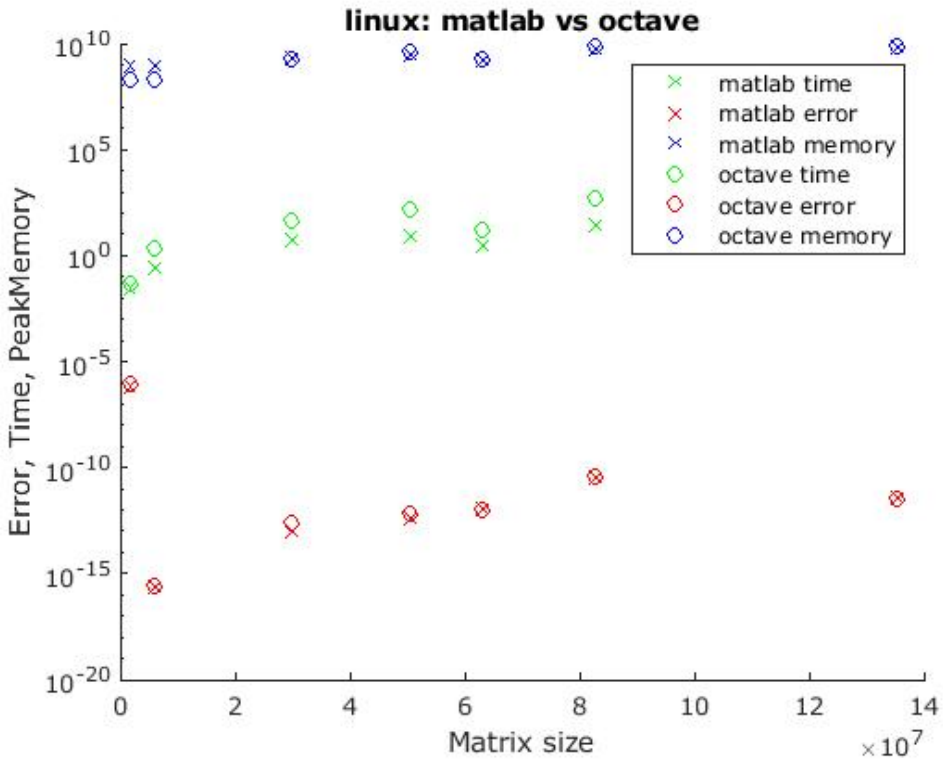


Figura 5: Risultati ottenuti in Linux

Analizzando il grafico in figura 5 sui due ambienti in Linux si notano performance migliori di Matlab per quanto concerne l'errore e soprattutto il tempo. Riguardo la memoria, si può osservare come Octave abbia richiesto meno memoria per le matrici più piccole.

4.2.6 Windows

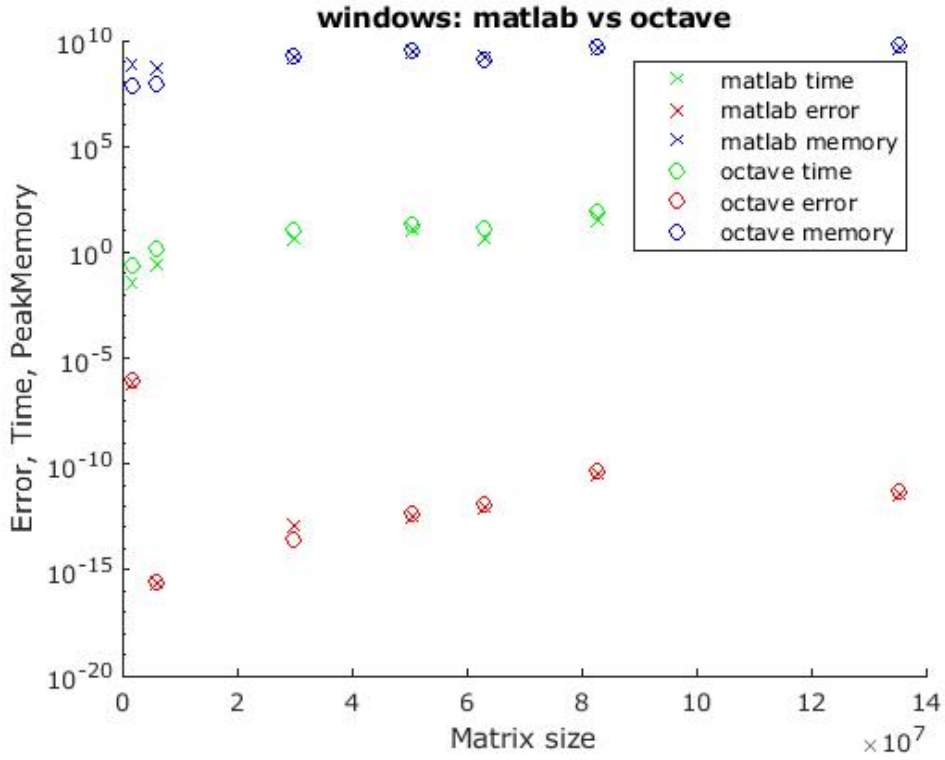


Figura 6: Risultati ottenuti in Windows

Il grafico in figura 6 che confronta i due ambienti in Windows permette di osservare che Matlab ha performance nettamente superiori in termini di tempo. Per le misure di memoria ed errore non si notano particolari differenze, fatta eccezione di alcune matrici più piccole.

4.3 Grafici aggiuntivi

Di seguito sono riportati degli ulteriori grafici che permettono di visualizzare la variazione di memoria durante la risoluzione di ciascuno dei sistemi lineari nelle diverse combinazioni ambiente/OS. A differenza dei picchi di memoria analizzati nella sezione 4.2, ai valori è stata sottratta la quantità iniziale di memoria utilizzata dal processo in esecuzione perché ritenuta ininfluente per questo tipo di confronto.

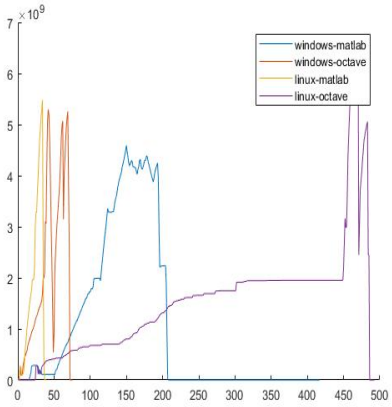


Figura 7: apache2

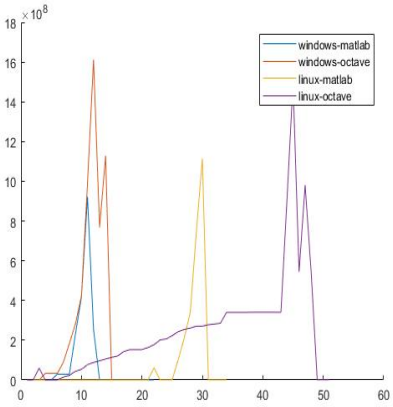


Figura 8: cfd1

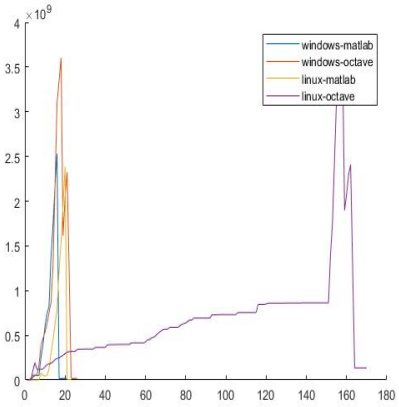


Figura 9: cfd2

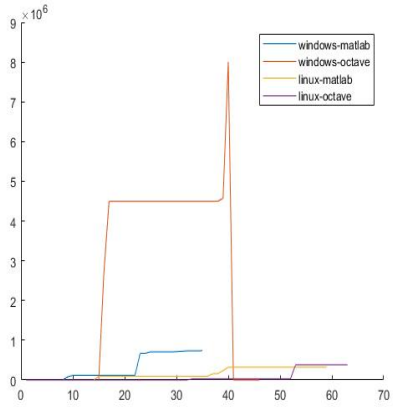


Figura 10: ex15

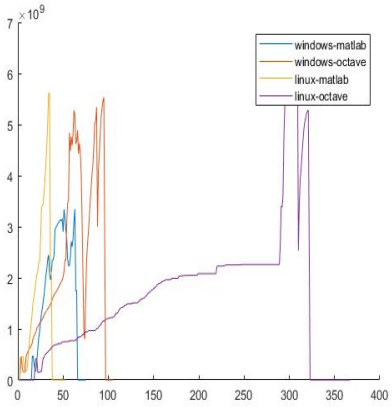


Figura 11: G2_circuit

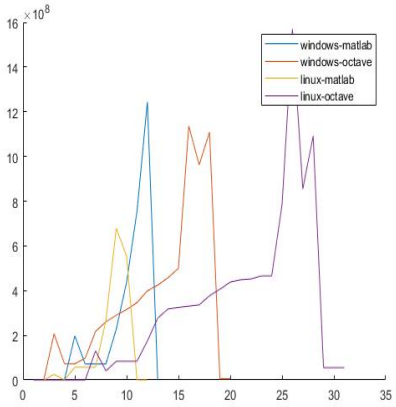


Figura 12: parabolic_fem

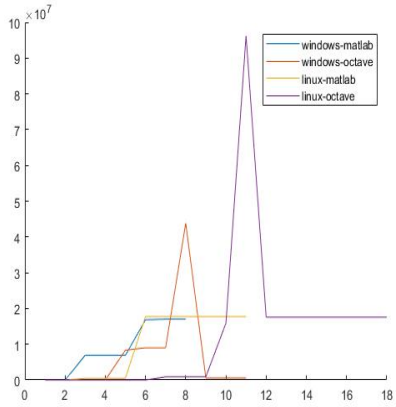


Figura 13: shallow_water1

5 Conclusioni

Le impressioni maturate durante questo lavoro in termini di usabilità favoriscono Matlab perché, come già detto in precedenza, è stata trovata una community più ampia e attiva e una buona documentazione, permettendo uno sviluppo senza particolari difficoltà. Al contrario Octave, pur mantenendo un'alta usabilità, è risultato meno completo non offrendo alcune funzionalità presenti in Matlab. Sia su Linux che su Windows l'usabilità dei due softwares resta la medesima.

Per quanto riguarda le performance si è complessivamente osservata una superiorità da parte di Matlab, sebbene Octave resti un'alternativa più che valida. Dai risultati ottenuti si può inoltre dire che la combinazione migliore risulta essere quella di Windows-Matlab, mentre la peggiore quella di Linux-Octave. In ogni caso, ognuna delle combinazioni analizzate permette di arrivare alla risoluzione di sistemi lineari dopo la decomposizione di Cholesky senza troppe difficoltà. Essendo i risultati ottenuti generalmente buoni, fare una scelta operativa dipende quindi strettamente dalle politiche aziendali.

In conclusione, avendo la possibilità di scegliere senza vincoli economici, sicuramente la scelta migliore risulta essere quella di Matlab a prescindere dal sistema operativo che si intende utilizzare. Se invece fosse necessario l'utilizzo di Octave sarebbe preferibile il suo utilizzo su una macchina Windows.

Riferimenti bibliografici

[1] M. Cleve Moler, "A brief history of matlab." https://it.mathworks.com/company/newsletters/articles/a-brief-history-of-matlab.html, 2018.

[2] The MathWorks Inc., "What is matlab?." https://it.mathworks.com/discovery/what-is-matlab.html.

[3] The MathWorks Inc., "Matlab." https://it.mathworks.com/products/matlab.html.

[4] John W. Eaton, "About." https://www.gnu.org/software/octave/about.html.

[5] Free Software Foundation Inc., "About." https://www.gnu.org/licenses/gpl-3.0.html, 2016.

[6] Free Software Foundation Inc., "About." https://www.gnu.org/licenses/gpl-3.0.html, 2016.

[7] ESI Group 2020, "www.scilab.org." https://www.scilab.org/about.

[8] E. S. de Almeida; Antonio C. Medeiros; Alejandro C. Frery, "How good are matlab, octave and scilab for computational modelling?," *Laboratório de Computação Científica e Análise Numérica (LaCCAN) Centro de Pesquisas em Matemática Computacional (CPMAT) Universidade Federal de Alagoas BR 104 Norte Km 97, 57072-970 Maceió, AL, Brazil*, 2012.

[9] Python Software Foundation, "What is python? executive summary." https://www.python.org/doc/essays/blurb/.

[10] Python Software Foundation, "History and license." https://docs.python.org/3/license.html.

[11] SciPy developers, "Scipy." https://www.scipy.org/.

[12] NumPy developers, "Numpy." https://numpy.org/.

[13] SciPy developers, "Scipy license." https://www.scipy.org/scipylib/license.html.

[14] NumPy developers, "Numpy license." https://numpy.org/license.html.

[15] A. Quarteroni, F. Saleri, and P. Gervasio, *Scientific computing with MATLAB and Octave*, vol. 2, pp. 154–155. Springer, 2006.