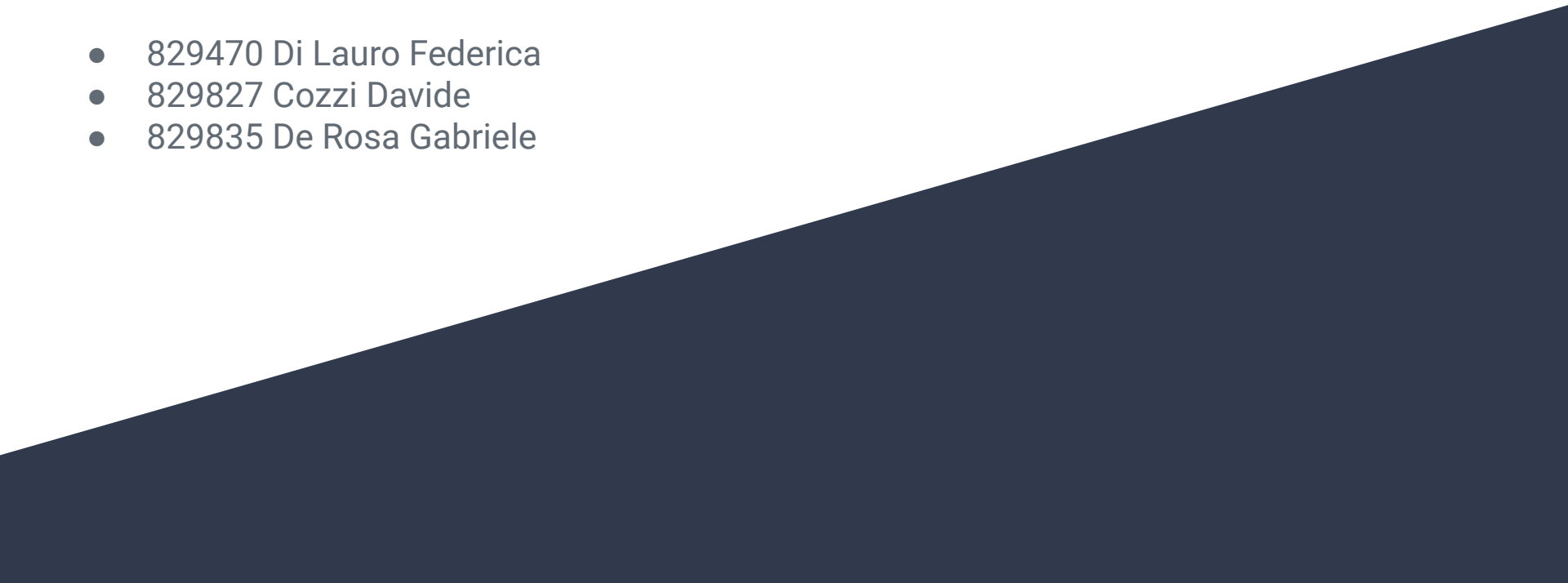


Linear Solver Comparison

Metodi del Calcolo Scientifico 2020/2021 - *Progetto1*

- 829470 Di Lauro Federica
 - 829827 Cozzi Davide
 - 829835 De Rosa Gabriele
- 
- A large, dark blue, curved shape that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

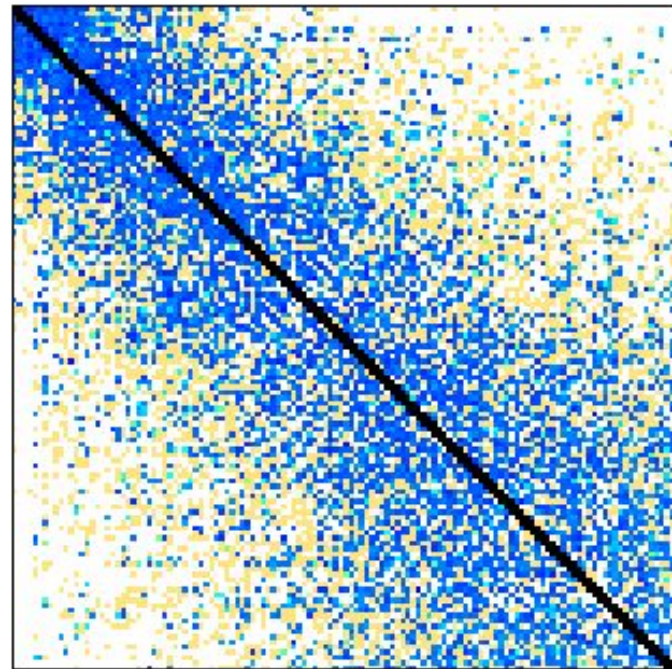
Introduzione

Obiettivo: studiare librerie e tool per la risoluzione di sistemi lineari confrontando le differenze tra quelli open-source e quelli proprietari.

Dataset: un set di matrici disponibili su [SuiteSparse Matrix Collection](#).

Codice:

- Jupyter Notebook per analisi dati: [Colab](#)
- Dati Benchmark: [Google Drive](#)
- Codice: [Google Drive](#)



Dataset

Matrice	Simmetrica	Definita Positiva
Hook 1498 **	✓	✓
G3 circuit	✓	✓
nd24k	✓	✓
bundle adj	✓	✓
ifiss mat	✗	✗
TSC OPF 1047	✓	✗
ns3Da	✗	✗
GT01R	✗	✗

** matrice non utilizzata per i confronti, approfondimento in seguito

Hardware

Thinkpad 480:

- Intel **i5** 8250U
- **16 GB** RAM

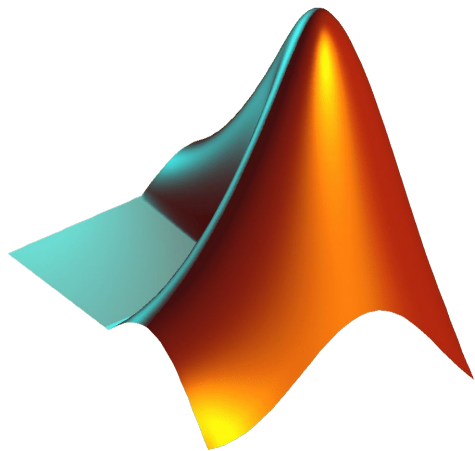
Sistemi operativi*:

- Linux (PopOs 18.04)
- Windows 10

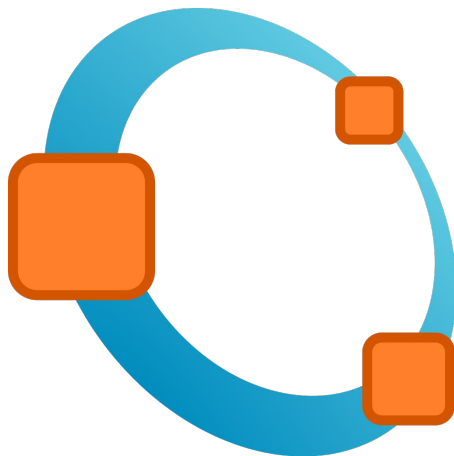
* entrambi i sistemi operativi sono stati installati su SSD



Linguaggi e ambienti utilizzati



MATLAB 2019b



Octave 6.2.0



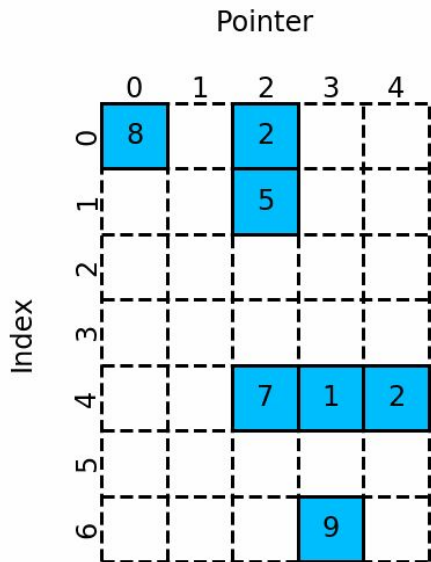
Python3, Scipy

Rappresentazione in memoria delle matrici

Tutti e 3 i linguaggi scelti memorizzano le matrici in formato CSC (Compressed Sparse Column)

Fonti:

- [MATLAB](#)
- [Octave](#)
- [Python](#)



© Matt Edling

CSC

Index Pointers

0	1	1	4	6	7
---	---	---	---	---	---

Indices

0	0	1	4	4	6	4
---	---	---	---	---	---	---

Data

8	2	5	7	1	9	2
---	---	---	---	---	---	---

MATLAB

MATLAB usa una release non recente di SuiteSparse (anche se non per tutti gli algoritmi), e usa Fortran BLAS (Basic Linear Algebra Subprograms).

Si ha il riconoscimento automatico della matrice in analisi con conseguente scelta dell'algoritmo, come visibile sulla [documentazione ufficiale](#).

L'analisi dell'algoritmo usato e dei dati di benchmark è resa possibile tramite l'uso della funzione *spumoni*. Tale strumento fornisce informazioni di debug quali l'utilizzo di memoria, l'algoritmo utilizzato, il numero di FLOPS e varie informazioni sulla struttura della matrice.

Le considerazioni fatte valgono sia su Windows che su Linux.

Octave

Octave usa una release più recente di SuiteSparse rispetto a MATLAB, e usa Fortran BLAS.

Si ha il riconoscimento automatico della matrice in analisi con conseguente scelta dell'algoritmo.

L'analisi dell'algoritmo usato e dei dati di benchmark è resa possibile tramite l'uso della funzione *spumoni*, come per MATLAB.

Le considerazioni fatte valgono sia su Windows che su Linux.

Python

È stata utilizzata la libreria **scipy**.

Linux: scipy chiama la libreria **scikit-sparse**, un wrapper per la libreria SuiteSparse. Anche in questo caso viene usato Fortran BLAS.

Di default la funzione per la risoluzione di sistemi usa per ogni matrice il risolutore UMFPACK (decomposizione LU). Per ottimizzare la risoluzione è stato scritto uno script che riconosce il tipo di matrice e nel caso sia definita positiva e simmetrica utilizza CHOLMOD (decomposizione Cholesky).

Windows: non è stato possibile utilizzare scikit-sparse, per problemi di installazione sul sistema operativo. Anche provare a compilare la libreria da sorgente non ha dato esito positivo. In alternativa scipy usa la libreria SuperLU, che contiene soltanto metodi per la decomposizione LU.

Codice dei risolutori

```
from scipy.sparse.linalg import spsolve
from sksparse.cholmod import cholesky

def sparse_solve(A,b):
    chol = True
    start = time.perf_counter()
    # Cholesky solo Linux
    try:
        x = cholesky(A)(b)
    except Exception as e:
        chol = False
        x = spsolve(A, b)
    finally:
        stop = time.perf_counter()
        fntime = stop - start
        erel = norm(x - xe) / norm(xe)
    return [chol, fntime, erel]
```

Python

```
function x = my_solve(A,b)
    x = A\b;
end

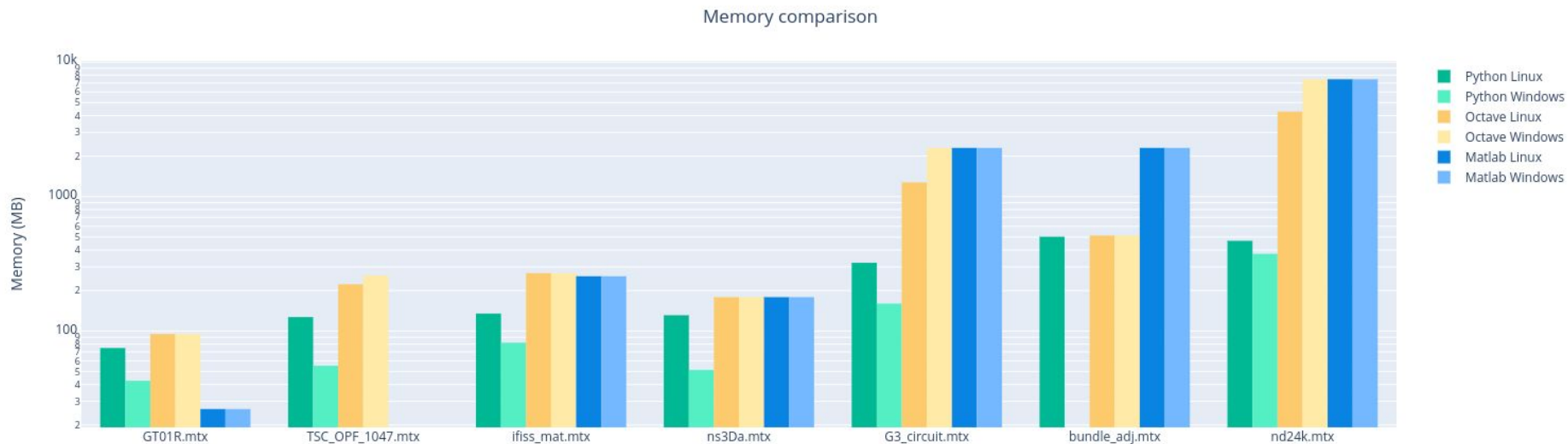
spparms('spumoni', 2);
tic;
x = my_solve(A,b);
tempo = toc;
erel = norm(x-xe) / norm(xe);
```

MATLAB/Octave

Algoritmi utilizzati

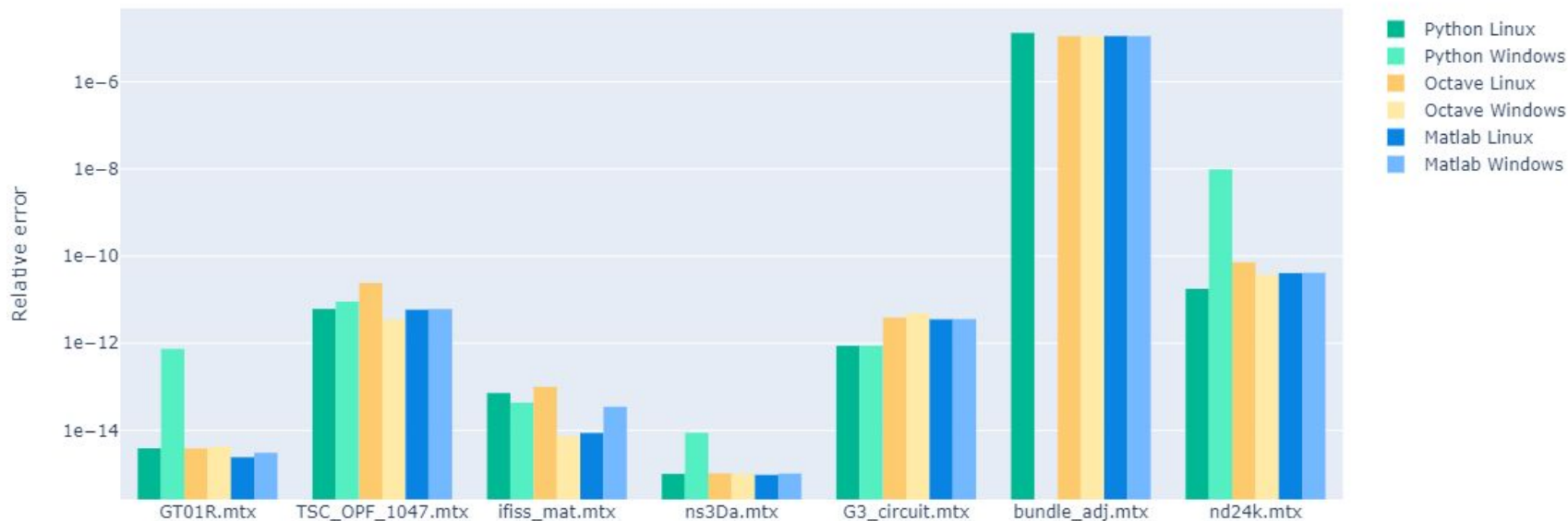
Matrice	MATLAB	Octave	Python	
			Linux	Windows
G3 circuit	CHOLMOD	CHOLMOD	CHOLMOD	SUPERLU
nd24k	CHOLMOD	CHOLMOD	CHOLMOD	SUPERLU
bundle adj	CHOLMOD	CHOLMOD	CHOLMOD	SUPERLU
ifiss mat	UMFPACK	UMFPACK	UMFPACK	SUPERLU
TSC OPF 1047	MA57	UMFPACK	UMFPACK	SUPERLU
ns3Da	UMFPACK	UMFPACK	UMFPACK	SUPERLU
GT01R	UMFPACK	UMFPACK	UMFPACK	SUPERLU

Confronto: MEMORIA



Confronto: ERRORE RELATIVO

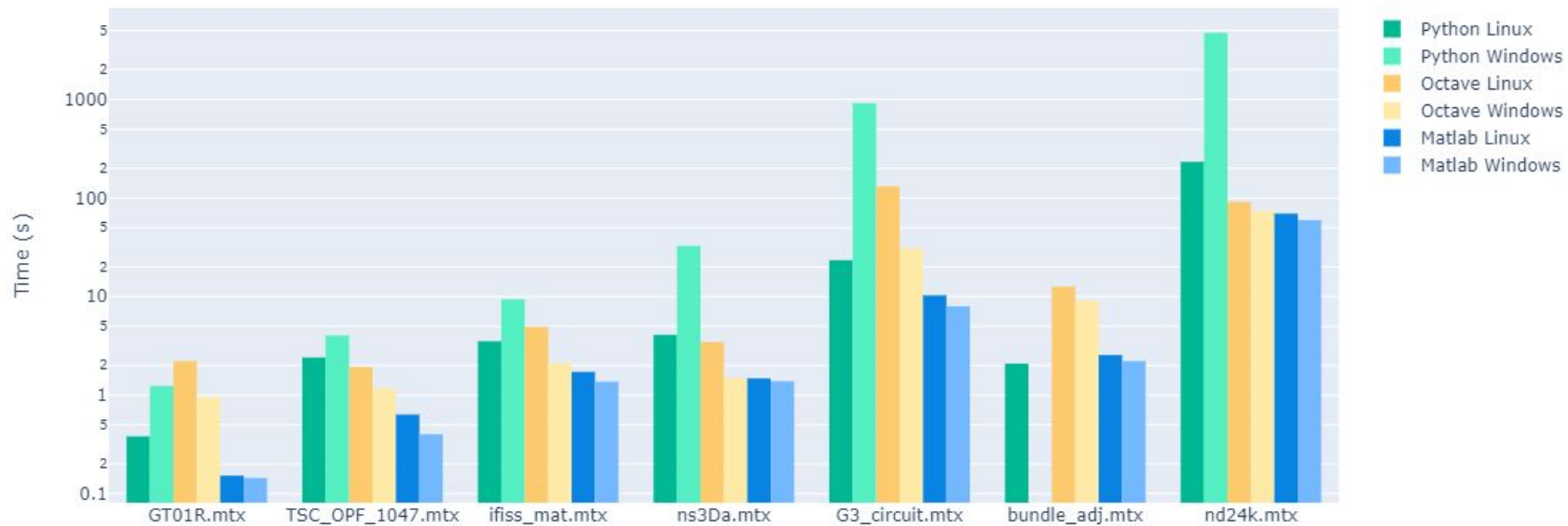
Relative error comparison



*numero di condizionamento di bundle_adj: $6.0244e+15$

Confronto: TEMPO

Time comparison



Problematiche Benchmark

Per l'analisi della memoria in MATLAB e in Octave è stato utilizzato l'output di *spumoni*, in quanto i profiler dei due ambienti non si sono rivelati affidabili.

Riguardo il confronto della memoria utilizzata non risulta disponibile il dato per la matrice `TSC OPF 1047`, in MATLAB in quanto è stato automaticamente scelto l'algoritmo **MA57** che nell'output di debug offerto da *spumoni* non presenta tale informazione.

È risultato impossibile risolvere le matrici `bundle_adj` utilizzando Python su Windows per mancanza di RAM.

In merito alla matrice `Hook_1498` si segnala che, con l'hardware a disposizione, non si è riuscito a risolvere il sistema in nessuno degli ambienti disponibili, tranne con MATLAB su Windows. In questo ambiente MATLAB è stato in grado di fare un'analisi preliminare dell'utilizzo di memoria usando Cholesky, e, dato che la RAM disponibile non era sufficiente, ha usato l'algoritmo MA57 con i seguenti risultati:

- **tempo:** 2.402426e+02 secondi **errore relativo:** 6.497802e-14

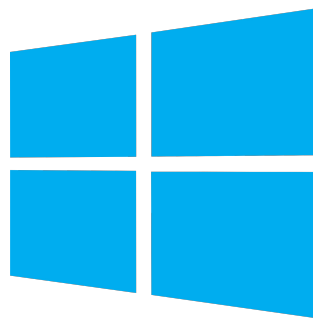
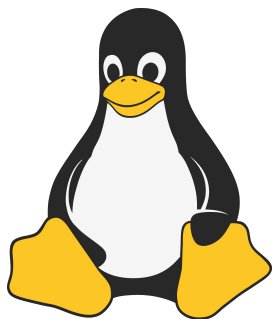
Scelta del linguaggio

MATLAB	Python	Octave
<ul style="list-style-type: none">● Ottime performance● Selezione automatica del metodo di risoluzione● Poco documentato● Software commerciale e non open-source	<ul style="list-style-type: none">● Discrete performance● Buona documentazione● Open-source e costantemente mantenuto dalla community● Risultati molto scarsi su Windows● Scelta del metodo da utilizzare da effettuare manualmente	<ul style="list-style-type: none">● Open-source● Selezione automatica del metodo di risoluzione● Poco documentato● Scarse performance (specialmente nell'import della matrice)

Scelta del sistema operativo

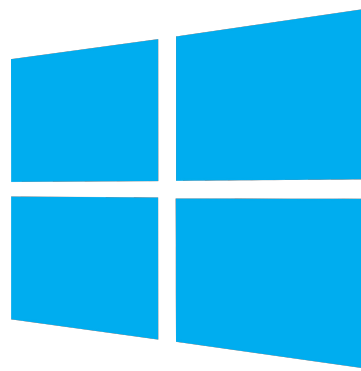
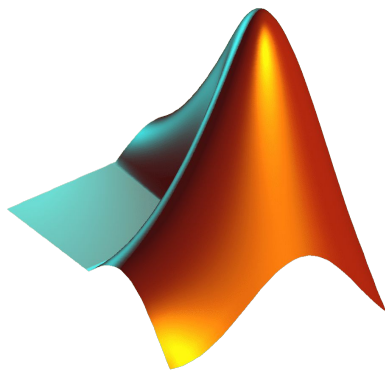
La scelta del sistema operativo potrebbe dipendere dal linguaggio/ambiente desiderato:

- per Python la scelta di GNU/Linux è obbligata, in quanto la libreria SuperLU utilizzata su Windows non è affatto performante.
- per MATLAB la scelta dovrebbe ricadere su Windows, sia per i tempi di risoluzione migliore, sia per un'apparente migliore scelta dell'algoritmo di risoluzione.
- per Octave i tempi risultano migliori su Windows.



Scelta finale

In conclusione, per le considerazioni fatte su tempi e algoritmi di risoluzione, la scelta ricade su MATLAB con Windows.



Ulteriori considerazioni

Sarebbe interessante confrontare le performance utilizzando una versione più ottimizzata di BLAS, come MKL (un esempio di benchmark è presente al seguente [link](#)).

Abbiamo quindi effettuato un test per tentare l'utilizzo **scikit-intel** per Python, ovvero una versione di scikit che si appoggia ad MKL. Purtroppo però si sono verificati dei problemi con l'utilizzo di tale strumento nella risoluzione delle matrici.

Per l'import delle matrici è stato utilizzato il **formato .mtx** (Matrix Market), utilizzando la funzione `mmread` fornita dagli sviluppatori di [Matrix Market](#).

Un secondo test è stato fatto utilizzando il formato `.mat`, il quale però portava delle problematiche in fase di importazione sui linguaggi Python e Octave dovuti a compatibilità tra le versioni.