Progetto di Metodi del Calcolo Scientifico

Tonelli Lidia Lucrezia (m. 813114)
Grassi Marco (m. 830694)
Giudice Gianluca (m. 829664)

University of Milano Bicocca

Giugno 2021

Sistema operativo e hardware

Sistema operativo (installazione da zero)

- Windows 10 Pro
- Ubuntu 20.04 LTS

Hardware

- CPU: Intel Core i7-8550U 4 x 1.8 - 4 GHz
- RAM: 32 GB, DDR4-2400

Ambienti di programmazione e librerie utilizzate

Abbiamo utilizzato 3 ambienti di programmazione; per Python abbiamo usato 3 librerie, di cui una necessaria per ottimizzare il calcolo su matrici molto grandi.

- Matlab R2021a
- GNU Octave 6.1.0
- Python 3.8.7
  - numpy 1.20.3
  - scipy 1.6.3
  - scikit-sparse 0.4.4

N.B.: Per entrambi i sistemi operativi é stata utilizzata la stessa versione di libreria in modo da avere risultati conmparabili.

Metriche di performance

Per misurare le performance sono state utilizzate le seguenti metriche:

- Tempo di calcolo della soluzione
- Picco memoria RAM utilizzata per risolvere il sistema
- Errore della soluzione:

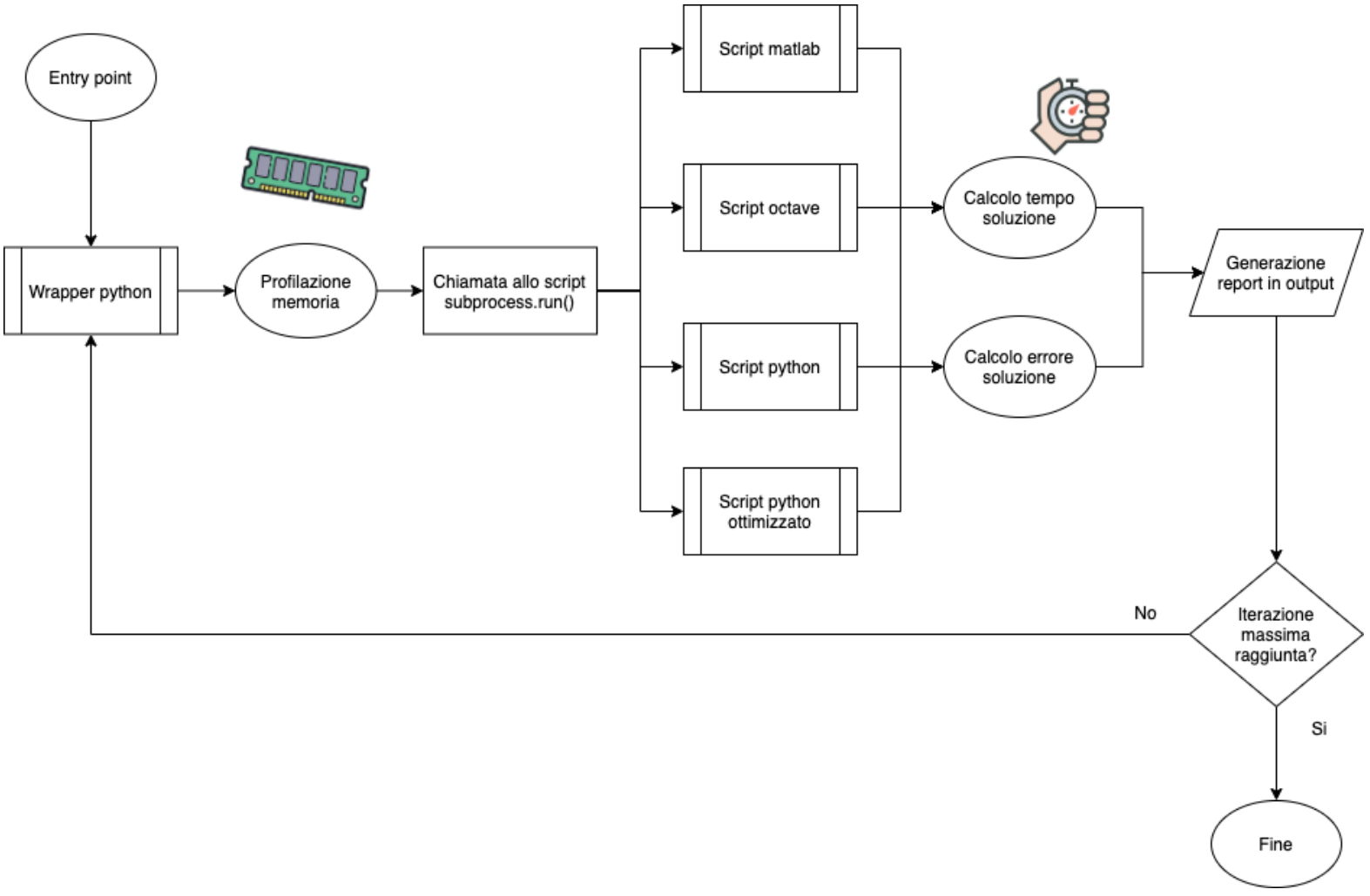
$$err\_rel = \frac{\|x - xe\|_2}{\|xe\|_2}$$

con $\|\cdot\|$ la norma euclidea dei vettori

Tecniche di misurazione

1. Creazione dei 4 script per il calcolo della soluzione di una matrice
   - INPUT: file matrice
   - OUTPUT: File di report
2. 10 iterazioni per il calcolo delle misure (in modo tale da avere risultati statisticamente validi)
   - Scirpt python con funzione di wrapper per profilare la memoria utilizzata.
     - 2.1 Run di ogni script per ogni matrice su ogni sistema operativo.
     - 2.2 Misurazione del tempo (si considera solo il tempo per il calcolo della soluzione)
     - 2.3 Misurazione dell'errore
3. Analisi dei report generati

Tecniche di misurazione



Matlab

Matlab é ben documentato alla pagina `it.mathworks.com/help/matlab`, con motore di ricerca ed esempi.

Memorizzazione di matrici sparse: data una matrice sparsa, Matlab salva solo gli elementi diversi da 0 in una lista, insieme al numero di colonna e riga.

Matlab

Risoluzione di $Ax = b$: Matlab ha a disposizione tanti risolutori di sistemi, il cui uso dipende dalla forma della matrice A; pertanto, quando viene eseguito il comando $A \setminus b$ si fanno una serie di controlli su A, i casi presi in considerazione sono: matrice quadrata, triangolare, triangolare permutata, Hermitiana, Hessenberg superiore, la matrice ha la diagonale tutta positiva o tutta negativa, la matrice é simmetrica e definita positiva. In tutti questi casi viene usato un solutore diverso, ad esempio se la matrice é simmetrica e definita positiva si applica l'algoritmo di Cholesky.

Octave

Octave é documentato in modo scomodo rispetto a Matlab alla pagina `https://octave.org/doc/v6.2.0`, che é un semplice manuale.

Memorizzazione di matrici sparse: usa il *compressed column format*, ovvero salva solo gli elementi diversi da 0 con il proprio numero di riga, e per ogni colonna salva il numero di elementi diversi da 0 per quella colonna; pertanto al posto di una tripletta per ogni elemento diverso da 0, si salva una coppia e in piú un numero per ogni colonna, ottimizzando la memorizzazione.

Octave

Risoluzione di $Ax = b$: si comporta allo stesso modo di Matlab, ovvero considera la forma della matrice A per scegliere il risolutore adatto. Anche Octave, come Matlab, prova ad applicare Cholesky e se non riesce significa che la matrice non é definita positiva e/o simmetrica.

Python (*numpy* + *scipy* + *scikit-sparse*)

Python è un linguaggio di programmazione general purpose. Abbiamo utilizzato 3 librerie apposite per il calcolo scientifico, ognuna con un compito diverso.

1. *numpy*: libreria utilizzata per leggere e gestire le matrice in memoria
2. *scipy*: libreria specifica per il calcolo scientifico. Utilizza metodi diretti per il calcolo della soluzione di un sistema. Il metodo standard per la risoluzione diretta di sistemi lineari non é ottimizzato come in Matlab o Octave, ovvero non divide i risolutori in base alla matrice. Supporta la fattorizzazione LU tuttavia non é abilitata di default; non supporta Cholesky su matrici sparse.
3. *scikit-sparse*: libreria che permette di applicare Cholsesky su matrici sparse; é implementata in c.

Python (*numpy* + *scipy* + *scikit-sparse*)

numpy e *scipy* é documentato peggio di Matlab ma meglio di Octave: ci sono molti esempi ed é disponibile il codice essendo open source, anche non essendo presente una buona spiegazione del workflow come in Matlab, si puó capire leggendo il codice; la documentazione si trova alla pagina `https://www.scipy.org/docs.html`.
sciki-sparse non ha una documentazione chiara, ma anche in questo caso, essendo open source, é possibile leggere direttamente il codice; la documentazione si trova alla pagina `https://scikit-sparse.readthedocs.io/en/latest/cholmod.html`.

Python (*numpy* + *scipy* + *scikit-sparse*)

Memorizzazione di matrici sparse: *numpy* salva le matrici sparse come Matlab.
Risoluzione di $Ax = b$: il risolutore assume che la soluzione x sia sparsa, perché puó capitare spesso; se invece x é densa, la costruzione risulta molto piú costosa. Utilizza sempre la fattorizzazione LU per risolvere il sistema, quindi il suo comportamento non é differente nel caso di matrici simmetriche e definite positive, perché non utilizza mai Cholesky; per questo motivo abbiamo introdotto la libreria *scikit-sparse*, che permette di applicare Cholesky a grandi matrici sparse, perció abbiamo potuto adattare l'algoritmo scritto in Python ai casi di matrici simmetriche e definite positive.

Nuovo workflow con *scikit-sparse*

Proprietá delle matrici

Matrice	Simmetrica	Def. pos.	Cand. Cholesky
Hook_1498	Y	Y	Y
G3_circuit	Y	Y	Y
nd24k	Y	Y	Y
boundle_adj	Y	Y	Y
ifiss_mat	N	N	N
TSC_OPF_1047	Y	N	N
ns3Da	N	N	N
GT01R	N	N	N

Durante l'esecuzione di ogni script si suppone di non essere a conoscenza di queste informazioni, cosí da rimanere il piú generali possibili.

Dimensione delle matrici

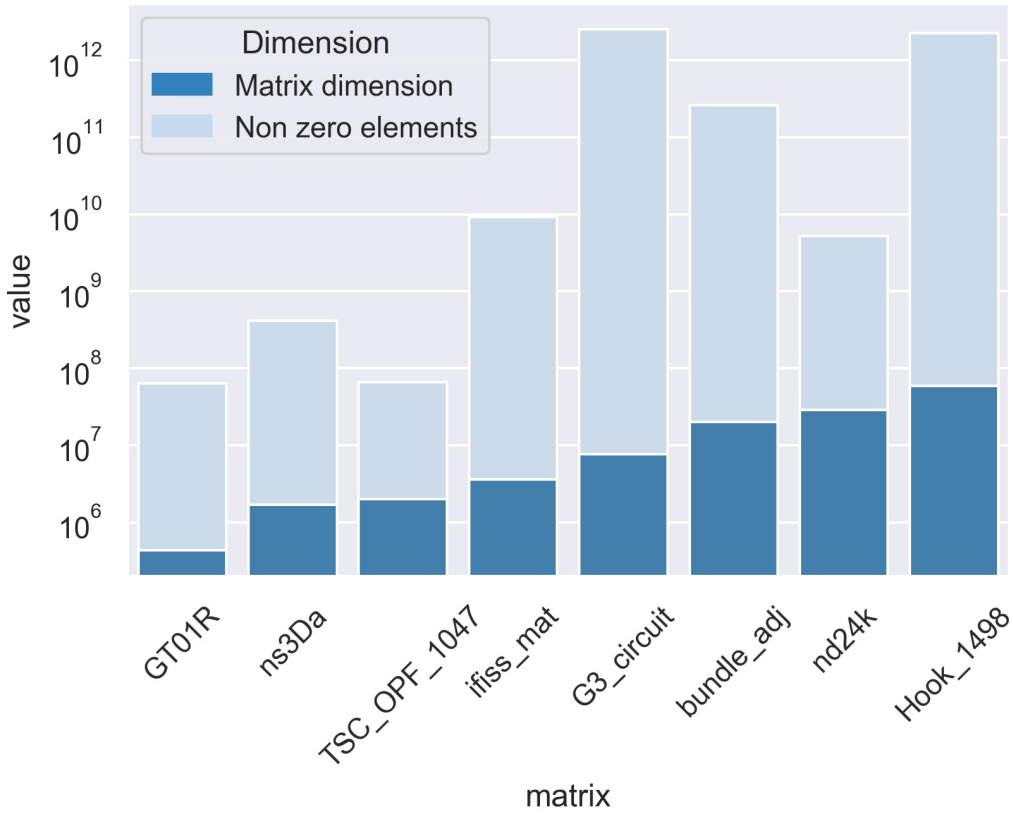


Figure: Confronto delle dimensioni e del numero di nnz delle matrici.

Grafici performance

Confronto tra Matlab, Octave, Python su Windows o Linux per i parametri velocità, precisione e occupazione di memoria.

Tempo di esecuzione

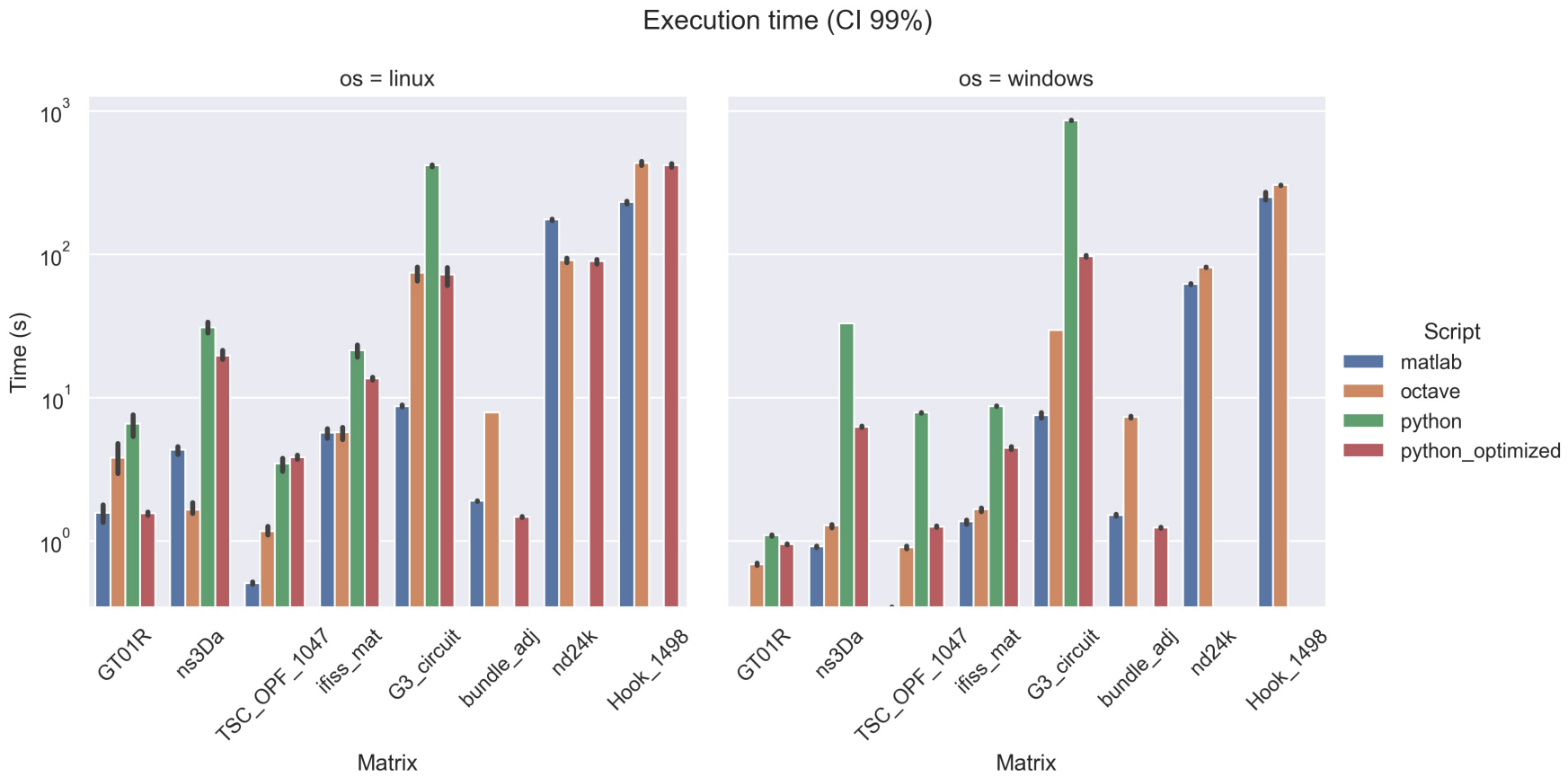


Figure: Confronto tempo di esecuzione.

Memoria utilizzata

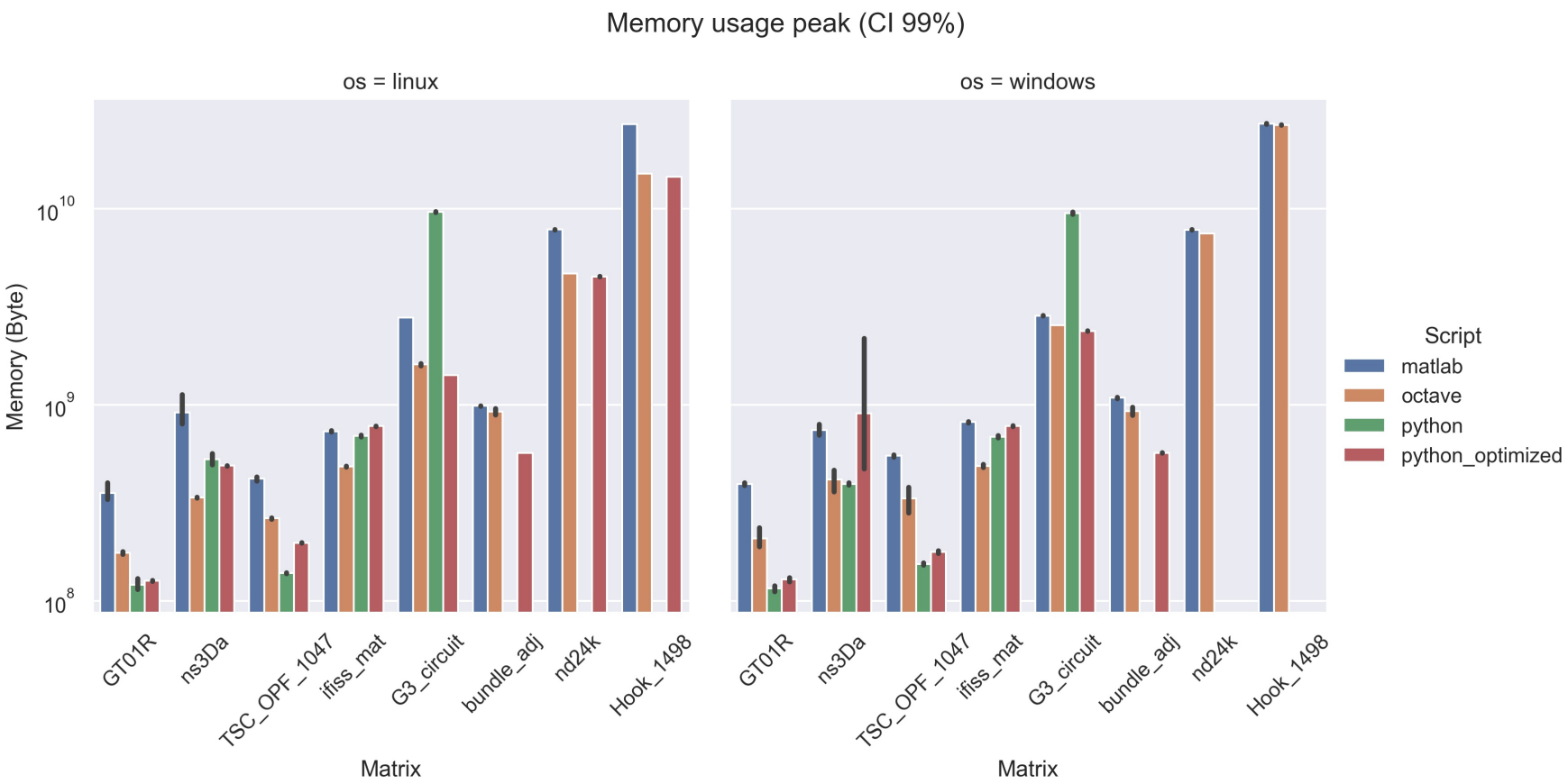


Figure: Picco di memoria utilizzata (in byte).

Tempo di esecuzione

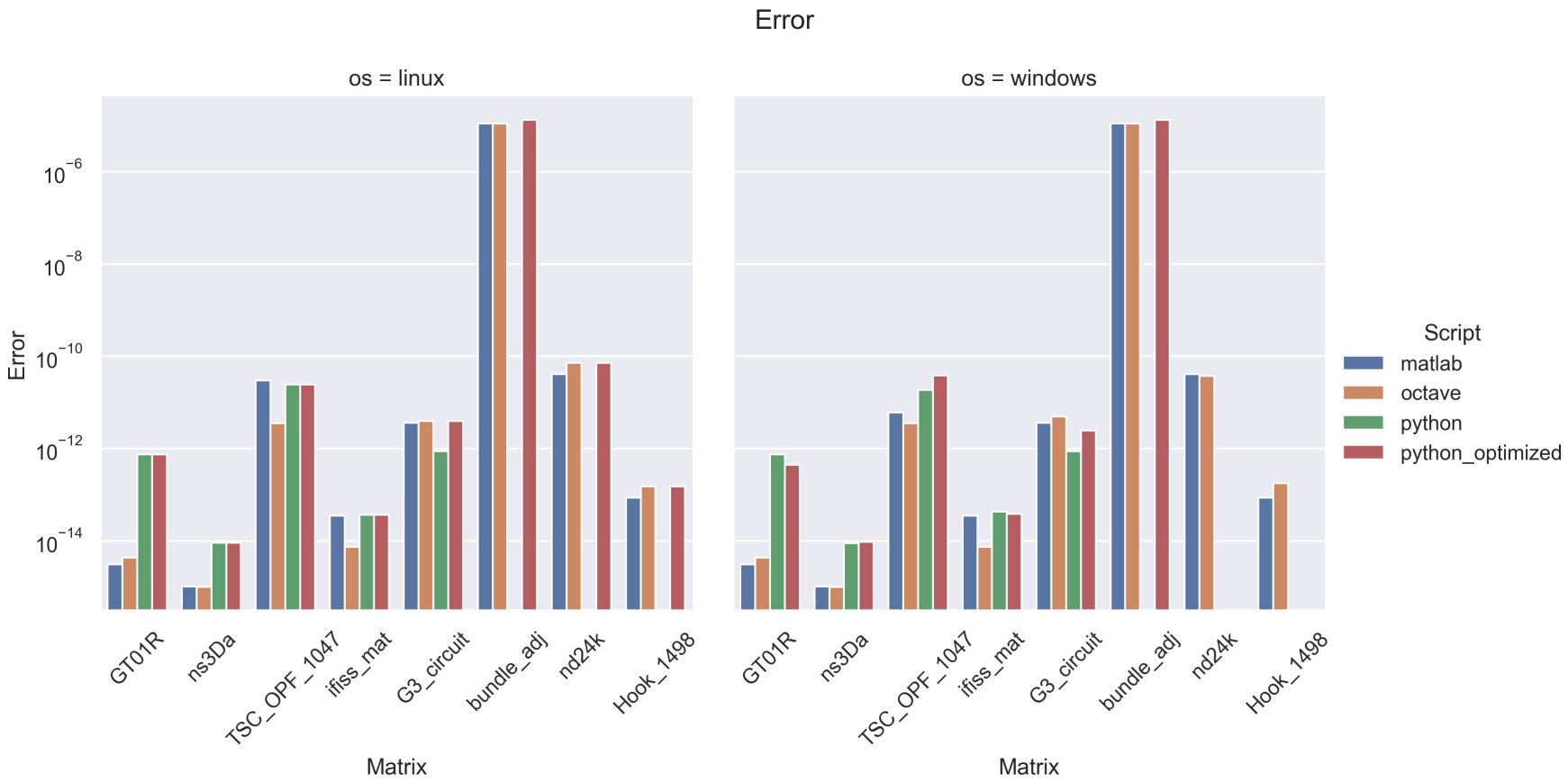


Figure: Confronto errore relativo soluzione.

Codice

Scipt per il calcolo delle soluzione utilizzando le diverse librerie.

Codice - Matlab

```
1  function [] = matlab_solver(matrix_file, result_file)
2
3      load(matrix_file);                % Carica matrice dato il path
4      A = Problem.A;                    % Accesso effettivo alla matrice
5
6      xe = ones(length(A(:,1)), 1);     % Ground truth soluzione
7      b = A * xe;                       % Calcolo termine noto data soluzione
8
9      tic;                              % Inizio timer
10     x = A \ b;                        % Calcolo della soluzione
11     time = toc;                       % Tmpo impiegato
12
13     % Statistice
14     er = norm(xe - x) / norm(xe);     % Errore relativo
15     m_size = numel(A);                % Dimensione matrice
16     m_nnz = nnz(A)                    % Elementi non zero matrice
17
18     fid = fopen(result_file, "a+"); % Apri file in append
19     fprintf(fid, "%d;%d;%d;%d", m_size, m_nnz, er, time);
20     fclose(fid);                      % Chiudi il file
21
22 end
```

Codice - Octave

```
1  args = argv();                    # Argomenti da linea di comando
2  matrix_file = args{1};            # Path matrice input
3  result_file = args{2};            # Path matrice output
4
5  load(matrix_file);                # Carica la matrice dato il path
6
7  A = Problem.A;                    # Accesso effettivo alla matrice
8  xe = ones(rows(A), 1);            # Ground truth soluzione
9  b = A * xe;                       # Calcolo termine noto data soluzione
10
11 tic;                              # Inizio timer
12 x = A \ b;                        # Calcolo della soluzione
13 time = toc;                       # Tempo impiegato
14
15 er = norm(xe - x) / norm(xe);     # Errore relativo
16 m_size = numel(A);                # Dimensione della matrice
17 m_nnz = nnz(A)                    # Elementi non zero
18
19 fid = fopen(result_file, "a+"); # Apri file
20 fprintf(fid, "%d;%d;%d;%d", m_size, m_nnz, er, time);
21 fclose(fid);                      # Chiudi file
```

Codice - Python

```python
matrix_file = sys.argv[1]            # Path matrice input
result_file = sys.argv[2]            # Path matrice output

struct = loadmat(matrix_file)

A = struct['Problem']['A'][0, 0]     # Accesso matrice
xe = np.ones(A.shape[0])             # Ground truth soluzione
b = A.dot(xe)                        # Termine noto dato xe

start_time = time.time()             # Start timer
x = spsolve(A, b)                    # Risoluzione sistema
elapsed = time.time() - start_time   # Tempo trascorso

er = norm(xe - x) / norm(xe)         # Errore relativo
m_size = A.shape[0] * A.shape[1]     # Dimensione matrice
nnz = A.nnz                          # Elementi non zero della matrice

# Scrittura risultati su file
with open(result_file, 'a+') as f:
    f.write(f'{m_size};{nnz};{er};{elapsed}')
```

Codice - Python ottimizzato

```python
matrix_file = sys.argv[1]           # Path matrice input
result_file = sys.argv[2]           # Path matrice output

struct = loadmat(matrix_file)

A = struct['Problem']['A'][0, 0]    # Accesso matrice
xe = np.ones(A.shape[0])            # Ground truth soluzione
b = A.dot(xe)                       # Termine noto dato xe

start_time = time.time()            # Start timer
try:
    # Calcolo soluzione con Cholesky
    factor = cholesky(A)            # Fattorizzazione con Cholesky
    x = factor(b)                   # Calcolo soluzione con fattorizzazione
except CholmodNotPositiveDefiniteError:
    # Eccezione: La matrice non e' definita positiva
    # Calcolo soluzione utilizzando fattorizzazione LU
    x = spsolve(A, b, use_umfpack=False)
elapsed = time.time() - start_time  # Tempo trascorso

er = norm(xe - x) / norm(xe)        # Errore relativo
m_size = A.shape[0] * A.shape[1]    # Dimensione matrice
nnz = A.nnz                         # Elementi non zero della matrice

# Scrittura risultati su file
with open(result_file, 'a+') as f:
    f.write(f'{m_size};{nnz};{er};{elapsed}')
```

Algoritmo usato

DCT2 fatta con numpy e FFT per confrontare presa da scipy.fft
array crescenti di dimensione 2^i

Algoritmo

```
1  def dct_personal(v):
2      n = len(v)
3      c = np.zeros(n)
4      for k in range(n):
5          if k == 0:
6              alpha = n
7          else:
8              alpha = n / 2
9          sum = 0
10         for i in range(n):
11             sum = sum + v[i] * math.cos(k * math.pi *
       ((2 * i + 1) / (2 * n)))
12         c[k] = (1 / math.sqrt(alpha)) * sum
13
14     return c
```

Algoritmo

```
1  def dct2_personal(a):
2      n = len(a)  # rows
3      m = len(a[0])  # columns
4      c = np.zeros((n, m))  # result matrix
5      # dct by rows
6      for i in range(n):
7          c[i] = dct_personal(a[i])  # i-th row
8      # dct by columns
9      for j in range(m):
10         c[:, j] = dct_personal(c[:, j])
```

Algoritmo

```
lib_times = []
my_times = []

for i in range(4, 12):
    N = 2 ** i
    a = np.random.rand(N, N)

    lib_ti = time.time()
    lib_dct = dct(dct(a, axis=1, norm="ortho"), axis=0,
    norm="ortho")
    lib_tf = time.time()
    lib_times.append(lib_tf - lib_ti)

    my_ti = time.time()
    my_dct = dct2_personal(a)
    my_tf = time.time()
    my_times.append(my_tf - my_ti)
```

Notizie su scipy.fft

`docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.dct.html`

For a single dimension array x, dct(x, norm='ortho') is equal to MATLAB dct(x). Ma é fft? Mi sa di no, provare fft

`docs.scipy.org/doc/scipy/reference/tutorial/fft.html`

Confronto tra Custom DCT2 e FFT

grafici - la dct é $O(N^3)$, la FFT é $O(N^2)$

Algoritmo

```python
import math
import numpy as np
from PIL import Image
from scipy.fft import dct
from scipy.fft import idct
```

```python
## fissa un range di valori possibili per n
def clamp(self, n, smallest, largest):
    return max(smallest, min(n, largest))

## forza n ad un valore intero tra 0 e 255
def fix_number(self, n):
    return self.clamp(round(n), 0, 255)
```

Algoritmo

```
1 def pseudo_jpeg(self, img_path, f, d):
2     img = Image.open(img_path)
3     img_mat = np.array(img)
4     c_mat = np.zeros_like(img_mat)
5     rows, columns = img_mat.shape
6
7     max_i = math.floor(rows / f)
8     max_j = math.floor(columns / f)
```

Algoritmo

```
    for i in range(max_i):
        for j in range(max_j):

            ## Applicazione DCT2
            ## Slice della matrice fxf
            block = img_mat[i * f: (i + 1) * f, j * f: (j + 1) * f]
            block = dct(dct(block, axis=1, norm="ortho"),axis=0, norm="ortho")
```

Algoritmo

```
1            ## Eliminazione elementi sotto diagonale
2            block_rows, block_columns = block.shape
3            for k in range(block_rows):
4                for l in range(block_columns):
5                    if (k + l) >= d:
6                        block[k, l] = 0
```

Algoritmo

```
            ## Applicazione IDCT2
            block = idct(idct(block, axis=1, norm="ortho"),axis=0, norm="ortho")

            ## fix dei numeri
            for k in range(block.shape[0]):
                for l in range(block.shape[1]):
                    block[k, l] = self.fix_number(block[k, l])

            c_mat[i * f: (i + 1) * f, j * f: (j + 1) * f] = block

    return c_mat
```

Esempi con le immagini proposte

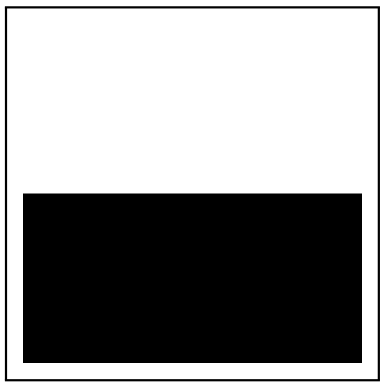


Figure: Originale 400x400

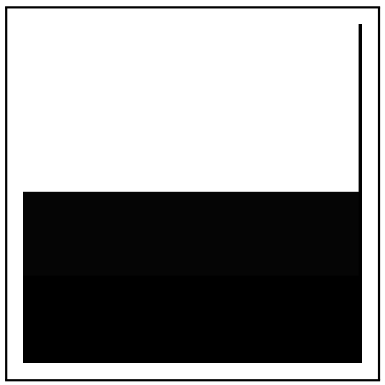


Figure: f = 99, d = 1

Esempi con le immagini proposte

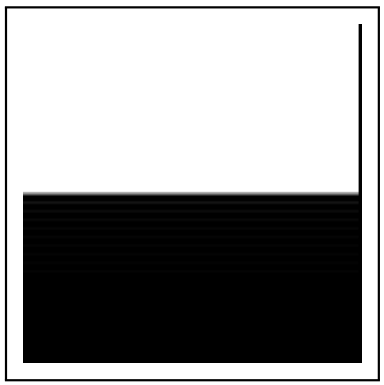


Figure: f = 99, d = 20

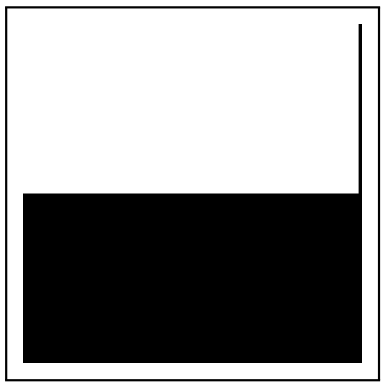


Figure: f = 99, d = 196

Sperimentazione

Momento della presentazione in cui facciamo girare il nostro programma

listato

Blocks of Highlighted Text

Block 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Block 2

Pellentesque sed tellus purus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vestibulum quis magna at risus dictum tempor eu vitae velit.

Block 3

Suspendisse tincidunt sagittis gravida. Curabitur condimentum, enim sed venenatis rutrum, ipsum neque consectetur orci, sed blandit justo nisi ac lacus.

Multiple Columns

Heading

1. Statement
2. Explanation
3. Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer lectus nisl, ultricies in feugiat rutrum, porttitor sit amet augue. Aliquam ut tortor mauris. Sed volutpat ante purus, quis accumsan dolor.

Theorem

Theorem (Mass–energy equivalence)

$E = mc^2$

Verbatim

Example (Theorem Slide Code)

```
\begin{frame}
\frametitle{Theorem}
\begin{theorem}[Mass--energy equivalence]
$E = mc^2$
\end{theorem}
\end{frame}
```

Figure

Uncomment the code on this slide to include your own image from the same directory as the template .TeX file.

Citation

An example of the `\cite` command to cite within the presentation:

This statement requires citation [Smith, 2012].

References

John Smith (2012)

Title of the publication

Journal Name 12(3), 45 – 678.

Grazie per l'attenzione