

Algorytmy macierzowe

Laboratorium 3

Sprawozdanie

Algorytmy rekurencyjnej kompresji i dekompresji macierzy

Adam Naumiec



Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki
Grudzień MMXXIII

Algorytmy macierzowe

Laboratorium 3

Adam Naumiec

Grudzień 2023

Spis treści

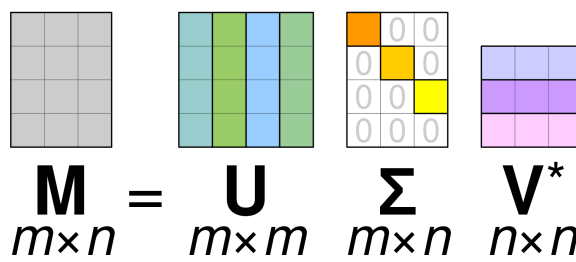
0	Abstrakt	3
1	Streszczenie wykładu	4
2	Zadania zrealizowane w ramach laboratorium	5
2.1	Zadania	5
2.2	Raporty	5
3	Pseudokody algorytmów i fragmenty kodu	6
3.1	Pseudokod: Tworzenie drzewa - <i>CreateTree</i>	6
3.2	Pseudokod: Kompresja macierzy - <i>CompressMatrix</i>	7
3.3	Fragmenty kodu programu realizującego zadanie	7
3.3.1	Klasa <i>MatrixTree</i> - <i>MatrixTree</i>	7
3.4	Metoda kompresująca - <i>compress</i>	8
3.4.1	Metoda dekompresująca - <i>decompress</i>	8
3.4.2	Metoda obliczająca rank macierzy - <i>get_rank</i>	8
3.4.3	Funkcja rysująca drzewo macierzy - <i>draw_tree</i>	8
4	Testy wydajnościowe i czas działania oraz pomiar błędu	10
4.1	Czasy obliczeń	11
4.2	Porównanie wyniku - obliczenie sumy różnicy kwadratów de- konstrukcji macierzy gęstej z macierzy skompresowanej	12
5	Wartości osobliwe σ dla macierzy	13

6	H-macierze dla różnych parametrów	13
6.1	$b = 1, \delta = \sigma_2$	14
6.1.1	99% zer	14
6.1.2	98% zer	14
6.1.3	95% zer	14
6.1.4	90% zer	14
6.1.5	80% zer	15
6.2	$b = 1, \delta = \sigma_{2^{k-1}}$	15
6.2.1	99% zer	15
6.2.2	98% zer	15
6.2.3	95% zer	15
6.2.4	90% zer	16
6.2.5	80% zer	16
6.3	$b = 1, \delta = \sigma_{2^k}$	16
6.3.1	99% zer	16
6.3.2	98% zer	16
6.3.3	95% zer	17
6.3.4	90% zer	17
6.3.5	80% zer	17
6.4	$b = 4, \delta = \sigma_2$	17
6.4.1	99% zer	17
6.4.2	98% zer	18
6.4.3	95% zer	18
6.4.4	90% zer	18
6.4.5	80% zer	18
6.5	$b = 4, \delta = \sigma_{2^{k-1}}$	19
6.5.1	99% zer	19
6.5.2	98% zer	19
6.5.3	95% zer	19
6.5.4	90% zer	19
6.5.5	80% zer	20
6.6	$b = 4, \delta = \sigma_{2^k}$	20
6.6.1	99% zer	20
6.6.2	98% zer	20
6.6.3	95% zer	20
6.6.4	90% zer	21
6.6.5	80% zer	21
6.7	Macierz, którą powiesiłbym w salonie	22

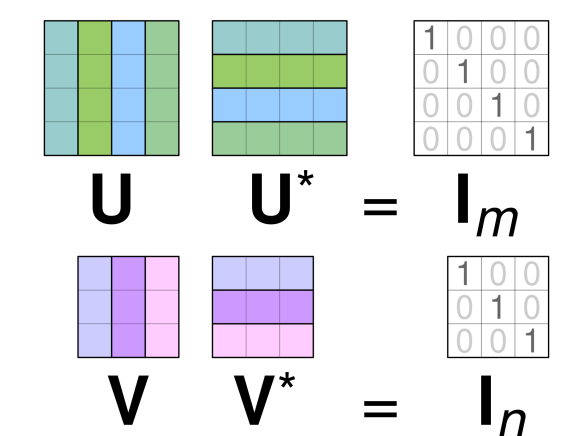
7 Wnioski	23
7.1 Refleksje płynące z laboratorium	23
7.2 Wykorzystanie kompresji macierzy	23
7.3 Ogólny wniosek	24
Literatura	25

0 Abstrakt

NINIEJSZY dokument jest sprawozdaniem z wykonania Laboratorium 3 z przedmiotu Algorytmy macierzowe prowadzonego przez Pana prof. dr hab. Macieja Paszyńskiego w roku akademickim 2023/2024 na piątym semestrze studiów pierwszego stopnia na kierunku Informatyka prowadzonego na Akademii Górniczo-Hutniczej im. Stanisława Staszica w Krakowie na Wydziale Informatyki.



$$\begin{matrix}
 \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & = & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} \\
 \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\
 m \times n & & m \times m & m \times n & n \times n
 \end{matrix}$$



$$\begin{matrix}
 \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & = & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} \\
 \mathbf{U} & \mathbf{U}^* & = & \mathbf{I}_m
 \end{matrix}$$

$$\begin{matrix}
 \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} & = & \begin{matrix} \text{4x4} \\ \text{Matrix} \end{matrix} \\
 \mathbf{V} & \mathbf{V}^* & = & \mathbf{I}_n
 \end{matrix}$$

Rysunek 1: Wizualizacja mnożenia macierzy w rozkładzie wartości osobliwych (2)

1 Streszczenie wykładu

Tzeci wykład skupiony był wokół kompresji i dekompresji macierzy. Poruszane były tematy:

- idea hierarchicznej (bezstratnej) kompresji macierzy
- warunek dopuszczalności
- rekurencyjna kompresja macierzy za pomocą algorytmu SVD
- przykłady hierarchicznej kompresji macierzy
- arytmetyka macierzy skompresowanych za pomocą SVD

Zaprezentowane zostało także wykorzystanie i znaczenie tych algorytmów w praktyce oraz historia wykorzystania, postrzegania i zastosowania macierzy na przestrzeni wieków, a także matematyczne, informatyczne i algorytmiczne podstawy operacji na macierzach.

2 Zadania zrealizowane w ramach laboratorium

W ramach laboratorium zrealizowano zadanie polegające na implementacji, wykonaniu testów wydajnościowych i przygotowaniu sprawozdania.

2.1 Zadania

1. Proszę wybrać ulubiony język programowania.
2. Proszę wygenerować macierze losowe z wartościami z przedziału $(0,1)$ które będą posiadać określony procent wartości niezerowych
3. Proszę napisać rekurencyjną kompresję macierzy z wykorzystaniem częściowego SVD (20 punktów) dla wybranych parametrów δ = najmniejsza wartość osobliwa (wyrzucamy mniejsze) b = maksymalny rank (liczba wartości osobliwych)
4. Proszę zaimplementować rysowacz (10 punktów)

2.2 Raporty

1. Proszę opisać pseudo-kod swojego rekurencyjnego algorytmu.
2. Proszę umieścić wybrane najbardziej istotne fragmenty kodu.
3. Proszę wybrać rozmiar macierzy $2^k \times 2^k$ dla dużego k .
4. Proszę wylosować 5 macierzy dla 1 procent wartości niezerowych, 2, 5, 10 i 20 procent (czyli 99 procent zer, 98, 95, 90, 80 procent).
5. Dla każdej macierzy proszę podać czas kompresji oraz
 - Proszę uruchomić SVD dla całej macierzy i znaleźć wartości osobliwe tej macierzy $\sigma_1, \dots, \sigma_{2^k}$ i narysować je na wykresie
 - Dla $b = 1$ oraz $\delta = \sigma_1$ narysować H-macierz.
 - Dla $b = 1$ oraz $\delta = \sigma_{2^k}$ narysować H-macierz.
 - Dla $b = 1$ oraz $\delta = \sigma_{2^{k-1}}$ narysować H-macierz.
 - Dla $b = 4$ oraz $\delta = \sigma_1$ narysować H-macierz.
 - Dla $b = 4$ oraz $\delta = \sigma_{2^k}$ narysować H-macierz.

- Dla $b = 4$ oraz $\delta = \sigma_{2^k-1}$ narysować H-macierz.
- Proszę napisać dekonstrukcję macierzy gęstej z macierzy skompresowanej i porównać z wynikiem licząc sumę różnicy kwadratów

$$\|A - B\|^2 = \sum_{i,j} (a_{ij} - b_{ij})^2$$

3 Pseudokody algorytmów i fragmenty kodu

PRZYGOTOWANO pseudokody testowanych algorytmów oraz przedstawiono wybrane najważniejsze fragmenty kodu zaimplementowanych algorytmów w języku programowania wysokiego poziomu.

3.1 Pseudokod: Tworzenie drzewa - *CreateTree*

Pseudokod tworzenia drzewa macierzy skompresowanej i wywoływania rekurencyjnej kompresji z wykorzystaniem częściowego SVD

Algorithm 1 CreateTree Function

```

1: procedure CREATETREE( $t_{\min}, t_{\max}, s_{\min}, s_{\max}, r, \epsilon$ )
2:   Require:  $1 \leq t_{\min} \leq t_{\max} \leq n, 1 \leq s_{\min} \leq s_{\max} \leq m$  where  $n \times m$  is
   the size of the matrix block,  $r, \epsilon$  are compression and threshold
3:    $[U, D, V] \leftarrow \text{truncatedSVD}(A(t_{\min} : t_{\max}, s_{\min} : s_{\max}), r + 1)$ 
4:   if  $D(r + 1, r + 1) < \epsilon$  then
5:      $v \leftarrow \text{CompressMatrix}(t_{\min}, t_{\max}, s_{\min}, s_{\max}, U, D, V, r)$ 
6:   else
7:     create new node  $v$ 
8:     Append( $v, \text{CreateTree}(t_{\min}, t_{\text{newmax}}, s_{\min}, s_{\text{newmax}})$ )
9:     Append( $v, \text{CreateTree}(t_{\min}, t_{\text{newmax}}, s_{\text{newmax}} + 1, s_{\max})$ )
10:    Append( $v, \text{CreateTree}(t_{\text{newmax}} + 1, t_{\max}, s_{\min}, s_{\text{newmax}})$ )
11:    Append( $v, \text{CreateTree}(t_{\text{newmax}} + 1, t_{\max}, s_{\text{newmax}} + 1, s_{\max})$ )
12:   end if
13:   RETURN  $v$ 
14: end procedure

```

3.2 Pseudokod: Kompresja macierzy - *CompressMatrix*

Pseudokod kompresji macierzy - podejście rekurencyjne z wykorzystaniem częściowego SVD

Algorithm 2 CompressMatrix Function

```
function COMPRESSMATRIX( $t_{\min}, t_{\max}, s_{\min}, s_{\max}, U, D, V, r$ )
2:   Require:  $t_{\min}, t_{\max}, s_{\min}, s_{\max}$  - range of indexes of the block,
    $[U, D, V] \leftarrow \text{truncatedSVD}(A(t_{\min} : t_{\max}, s_{\min} : s_{\max}, r + 1))$ 
4:   if block ( $t_{\min}, t_{\max}, s_{\min}, s_{\max}$ ) consists of zeros then
       create new node  $v$ 
6:        $v.\text{rank} \leftarrow 0$ 
        $v.\text{size} \leftarrow \text{size}(t_{\min}, t_{\max}, s_{\min}, s_{\max})$ 
8:       return  $v$ 
   end if
10:   $\sigma \leftarrow \text{diag}(D)$ 
    $\text{rank} \leftarrow r$ 
12:  create new node  $v$ 
    $v.\text{rank} \leftarrow \text{rank}$ 
14:   $v.\text{singularvalues} \leftarrow \sigma(1 : \text{rank})$ 
    $v.U \leftarrow U(*, 1 : \text{rank})$ 
16:   $v.V \leftarrow D(1 : \text{rank}, 1 * \text{rank}) \cdot V(1 : \text{rank}, *)$ 
    $v.\text{sons} \leftarrow \emptyset$ 
18:   $v.\text{size} \leftarrow \text{size}(t_{\min}, t_{\max}, s_{\min}, s_{\max})$ 
   return  $v$ 
20: end function
```

3.3 Fragmenty kodu programu realizującego zadanie

3.3.1 Klasa MatrixTree - *MatrixTree*

```
class MatrixTree(object):
    def __init__(self, matrix, row_min, row_max, col_min, col_max):
        self.matrix = matrix
        self.row_min = row_min
        self.row_max = row_max
        self.col_min = col_min
        self.col_max = col_max
        self.rank = None
        self.u = None
        self.s = None
        self.v = None
        self.leaf = False
        self.children = []
```


3.4 Metoda kompresująca - *compress*

```
def compress(self, r: int, eps: float) -> None:
    M = self.matrix[self.row_min:self.row_max, self.col_min:self.col_max]
    U, Sigma, V = randomized_svd(M, n_components=r + 1, random_state=0)
    if self.row_min + r == self.row_max or Sigma[r] <= eps:
        self.leaf = True
        if not M.any():
            self.rank = 0
        else:
            self.rank = len(Sigma)
            self.u = U
            self.s = Sigma
            self.v = V
    else:
        self.children = []
        new_row_max = (self.row_min + self.row_max) // 2
        new_col_max = (self.col_min + self.col_max) // 2
        self.children.append(MatrixTree(self.matrix, self.row_min, new_row_max, self.col_min, new_col_max))
        self.children.append(MatrixTree(self.matrix, self.row_min, new_row_max, new_col_max, self.col_max))
        self.children.append(MatrixTree(self.matrix, new_row_max, self.row_max, self.col_min, new_col_max))
        self.children.append(MatrixTree(self.matrix, new_row_max, self.row_max, new_col_max, self.col_max))
        for child in self.children:
            child.compress(r, eps)
```

3.4.1 Metoda dekompresująca - *decompress*

```
def decompress(self, matrix: np.ndarray) -> None:
    if self.leaf:
        if self.rank:
            sigma = np.zeros((self.rank, self.rank))
            np.fill_diagonal(sigma, self.s)
            M = self.u @ sigma @ self.v
            matrix[self.row_min:self.row_max, self.col_min:self.col_max] = M
        else:
            M = self.matrix[self.row_min:self.row_max, self.col_min:self.col_max]
            matrix[self.row_min:self.row_max, self.col_min:self.col_max] = M
    else:
        for child in self.children:
            child.decompress(matrix)
```

3.4.2 Metoda obliczająca rank macierzy - *get_rank*

```
def get_rank(self) -> int:
    if self.leaf:
        return self.rank
    else:
        return sum([child.get_rank() for child in self.children])
```

3.4.3 Funkcja rysująca drzewo macierzy - *draw_tree*

```
def draw_tree(root: "MatrixTree", title: str="") -> None:
    image = np.ones(root.matrix.shape) * 255
    Q = deque()
    Q.append(root)
    while Q:
        v = Q.pop()
        if v.leaf:
            image[v.row_min:v.row_max, v.col_min:v.col_min + v.rank] = np.zeros((v.row_max - v.row_min, v.rank))
```

```

        image[v.row_min:v.row_min + v.rank, v.col_min:v.col_max] = np.zeros((v.rank, v.col_max - v.col_min))
        image[v.row_min, v.col_min:v.col_max] = np.zeros((1, v.col_max - v.col_min))
        image[v.row_max - 1, v.col_min:v.col_max] = np.zeros((1, v.col_max - v.col_min))
        image[v.row_min:v.row_max, v.col_min] = np.zeros(v.row_max - v.row_min)
        image[v.row_min:v.row_max, v.col_max - 1] = np.zeros(v.row_max - v.row_min)
    else:
        for child in v.children:
            Q.append(child)
plt.imshow(image, cmap="gist_gray", vmin=0, vmax=255)
plt.title(title)
plt.xticks([])
plt.yticks([])
plt.show()

```

4 Testy wydajnościowe i czas działania oraz pomiar błędu

PRZYGOTOWANO testy wydajnościowe omawianych algorytmów w celu prezentacji czasu obliczeń . Czynność ta pozwoliła na praktyczne przetestowanie algorytmów oraz konformantacji ich teoretycznych złożoności obliczeniowych wraz z danymi empirycznymi oraz sprawdzenie poprawności impelentacji algorytmów.

4.1 Czasy obliczeń

Dokonano pomiaru czasu kompresji macierzy dla różnych parametrów, wyniki przedstawiono w tabeli.

Tabela 1: Czasy obliczeń dla różnych parametrów

#	Niezerowe	b	σ	Czas
1	1%	1	σ_1	0.021960
2	1%	1	$\sigma_{2^{k-1}}$	0.577161
3	1%	1	σ_{2^k}	20.231883
4	1%	4	σ_1	0.028359
5	1%	4	$\sigma_{2^{k-1}}$	0.717139
6	1%	4	σ_{2^k}	6.833700
7	2%	1	σ_1	0.023493
8	2%	1	$\sigma_{2^{k-1}}$	0.544299
9	2%	1	σ_{2^k}	29.033727
10	2%	4	σ_1	0.025757
11	2%	4	$\sigma_{2^{k-1}}$	0.258863
12	2%	4	σ_{2^k}	8.499308
13	5%	1	σ_1	0.022849
14	5%	1	$\sigma_{2^{k-1}}$	0.221370
15	5%	1	σ_{2^k}	64.627048
16	5%	4	σ_1	0.023469
17	5%	4	$\sigma_{2^{k-1}}$	0.238199
18	5%	4	σ_{2^k}	15.988675
19	10%	1	σ_1	0.021120
20	10%	1	$\sigma_{2^{k-1}}$	0.217040
21	10%	1	σ_{2^k}	114.064481
22	10%	4	σ_1	0.024118
23	10%	4	$\sigma_{2^{k-1}}$	0.257509
24	10%	4	σ_{2^k}	26.809606
25	20%	1	σ_1	0.020499
26	20%	1	$\sigma_{2^{k-1}}$	0.198258
27	20%	1	σ_{2^k}	199.693175
28	20%	4	σ_1	0.022919
29	20%	4	$\sigma_{2^{k-1}}$	0.234090
30	20%	4	σ_{2^k}	48.766644

4.2 Porównanie wyniku - obliczenie sumy różnicy kwadratów dekonstrukcji macierzy gęstej z macierzy skompresowanej

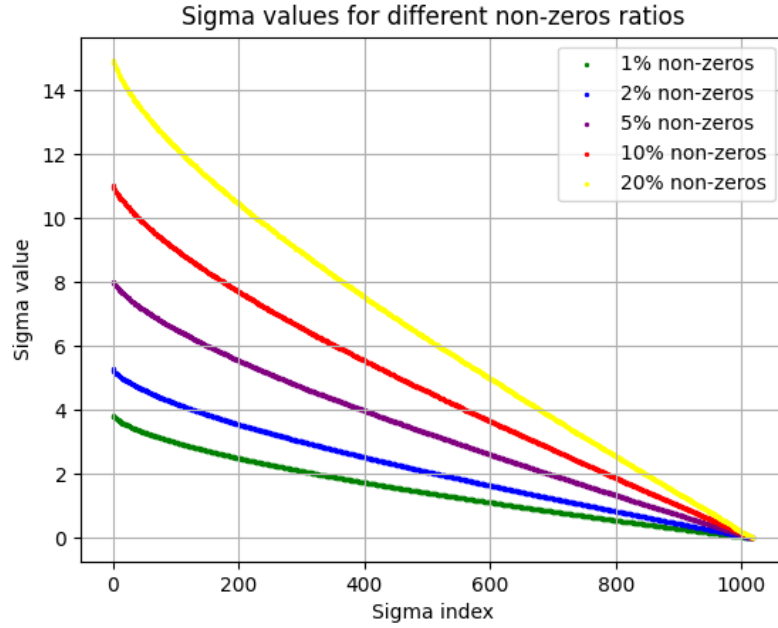
Dokonano pomiaru sumy różnicy kwadratów dekonstrukcji macierzy gęstej z macierzy skompresowanej używając wzoru: $\|A - B\|^2 = \sum_{i,j} (a_{ij} - b_{ij})^2$.

Tabela 2: Błąd - suma różnicy kwadratów - dla różnych parametrów

#	Niezerowe	b	σ	Błąd
1	1%	1	σ_1	5.875743×10^1
2	1%	1	$\sigma_{2^{k-1}}$	5.102409×10^1
3	1%	1	σ_{2^k}	8.500174×10^{-15}
4	1%	4	σ_1	5.838015×10^1
5	1%	4	$\sigma_{2^{k-1}}$	4.558788×10^1
6	1%	4	σ_{2^k}	4.074786×10^{-14}
7	2%	1	σ_1	8.286845×10^1
8	2%	1	$\sigma_{2^{k-1}}$	7.606209×10^1
9	2%	1	σ_{2^k}	1.244579×10^{-14}
10	2%	4	σ_1	8.237188×10^1
11	2%	4	$\sigma_{2^{k-1}}$	7.485865×10^1
12	2%	4	σ_{2^k}	6.550157×10^{-14}
13	5%	1	σ_1	1.298218×10^2
14	5%	1	$\sigma_{2^{k-1}}$	1.264752×10^2
15	5%	1	σ_{2^k}	2.026586×10^{-14}
16	5%	4	σ_1	1.290964×10^2
17	5%	4	$\sigma_{2^{k-1}}$	1.200415×10^2
18	5%	4	σ_{2^k}	1.131958×10^{-13}
19	10%	1	σ_1	1.792990×10^2
20	10%	1	$\sigma_{2^{k-1}}$	1.752112×10^2
21	10%	1	σ_{2^k}	2.877998×10^{-14}
22	10%	4	σ_1	1.783093×10^2
23	10%	4	$\sigma_{2^{k-1}}$	1.669733×10^2
24	10%	4	σ_{2^k}	2.318141×10^{-3}
25	20%	1	σ_1	2.430412×10^2
26	20%	1	$\sigma_{2^{k-1}}$	2.379345×10^2
27	20%	1	σ_{2^k}	4.155036×10^{-14}
28	20%	4	σ_1	2.416912×10^2
29	20%	4	$\sigma_{2^{k-1}}$	2.273164×10^2
30	20%	4	σ_{2^k}	8.446529×10^{-4}

5 Wartości osobliwe σ dla macierzy

DOKONANO pomiaru wartości osobliwych macierzy $\sigma_1, \dots, \sigma_{2^k}$ i przedstawiono je na wykresie.



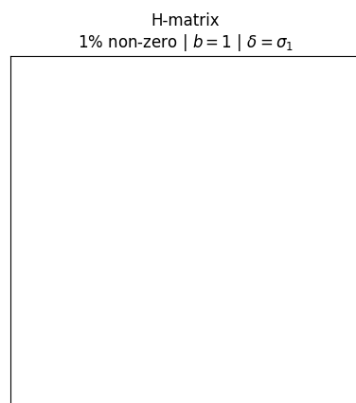
Rysunek 2: Wartości osobliwe σ

6 H-macierze dla różnych parametrów

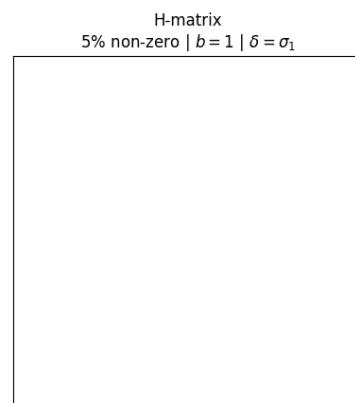
Z pomocą zaimplementowanego rysowacza przedstawiono H-macierze dla różnych wartości parametrów. Macierze były rozmairu $2^k \times 2^k$, gdzie $k = 10$.

6.1 $b = 1, \delta = \sigma_2$

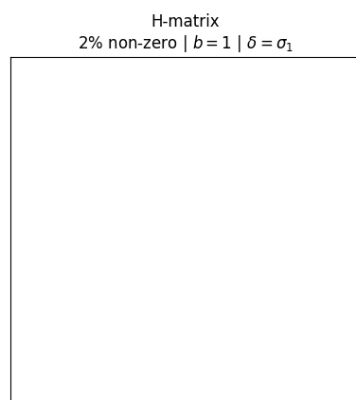
6.1.1 99% zer



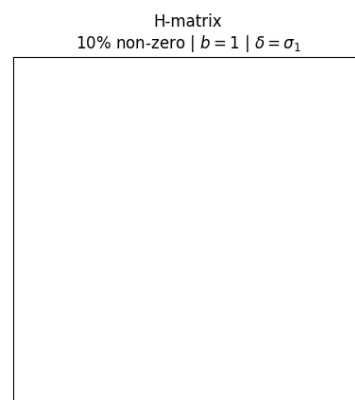
6.1.3 95% zer



6.1.2 98% zer

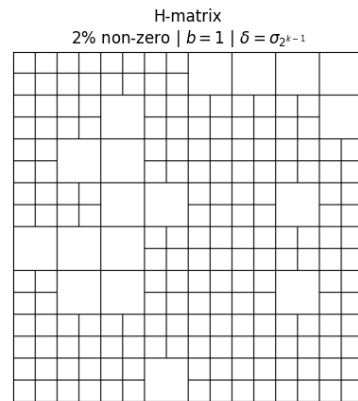
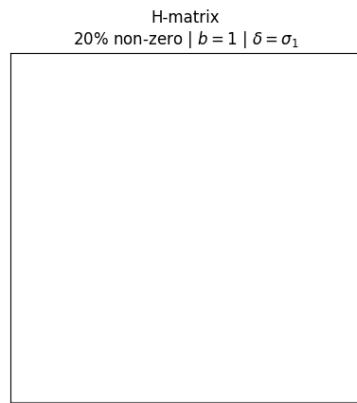


6.1.4 90% zer



6.1.5 80% zer

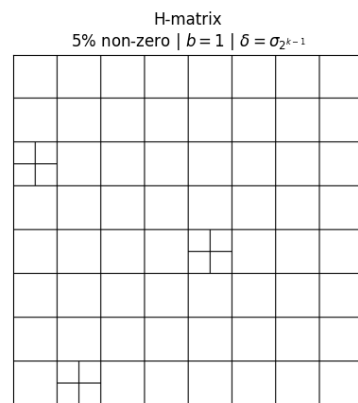
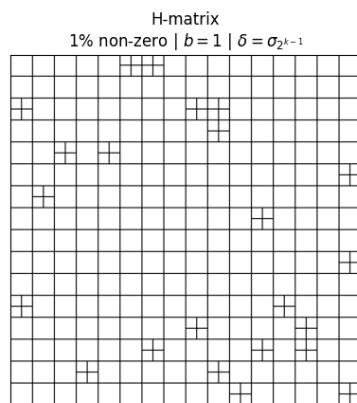
6.2.2 98% zer



6.2 $b = 1, \delta = \sigma_{2^{k-1}}$

6.2.3 95% zer

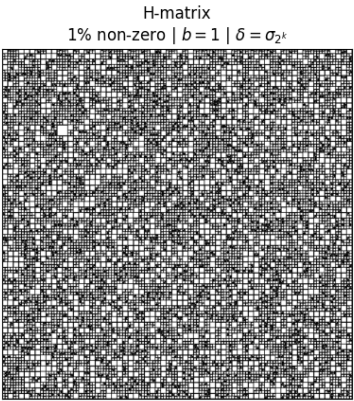
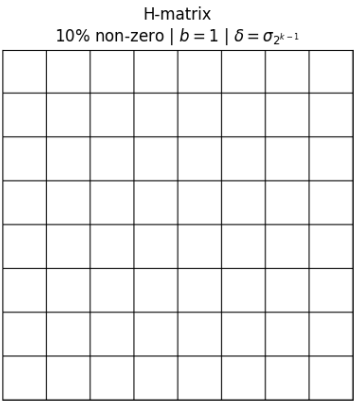
6.2.1 99% zer



6.2.4 90% zer

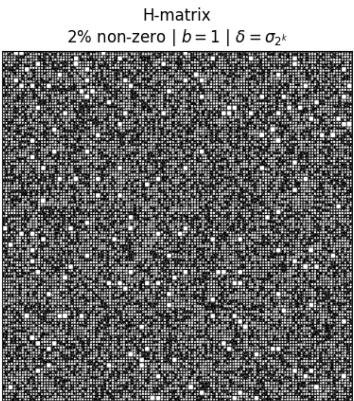
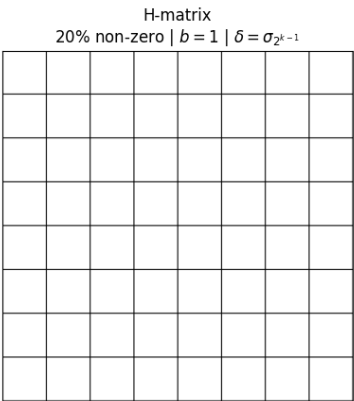
6.3 $b = 1, \delta = \sigma_{2^k}$

6.3.1 99% zer



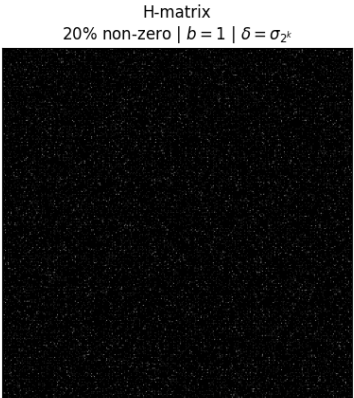
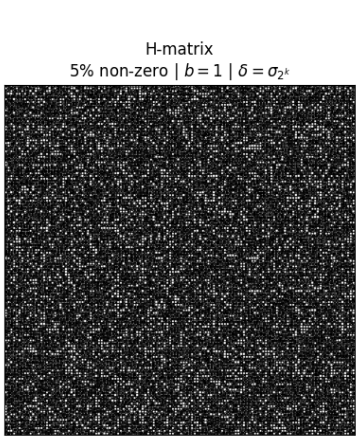
6.2.5 80% zer

6.3.2 98% zer



6.3.3 95% zer

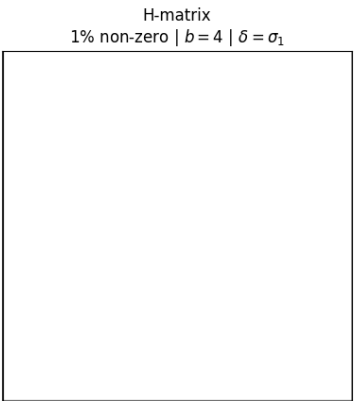
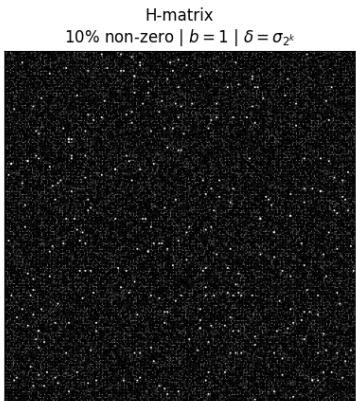
6.3.5 80% zer



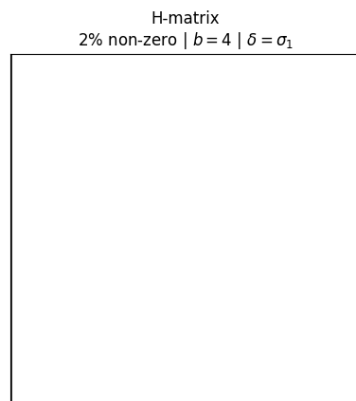
6.4 $b = 4, \delta = \sigma_2$

6.3.4 90% zer

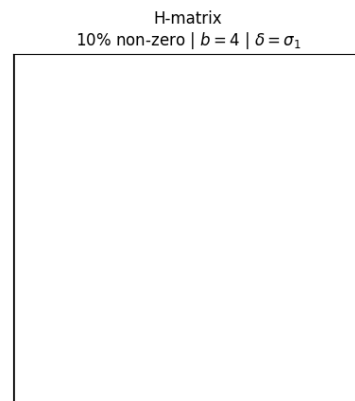
6.4.1 99% zer



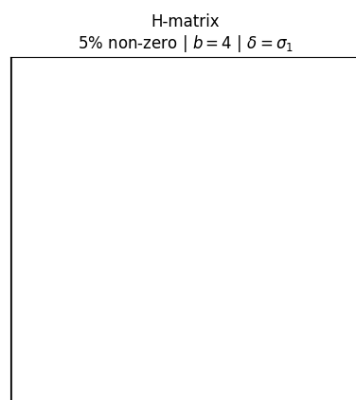
6.4.2 98% zer



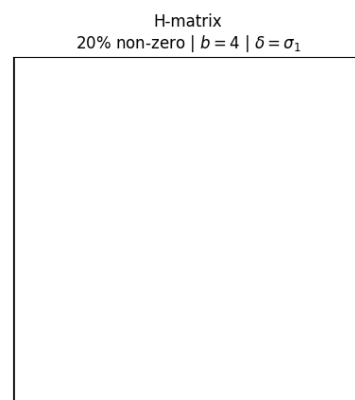
6.4.4 90% zer



6.4.3 95% zer



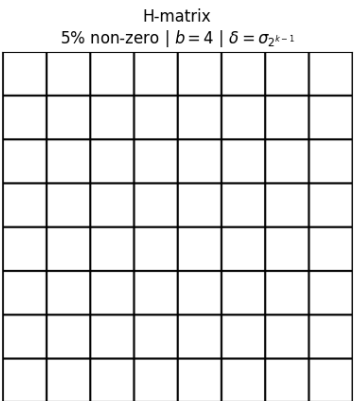
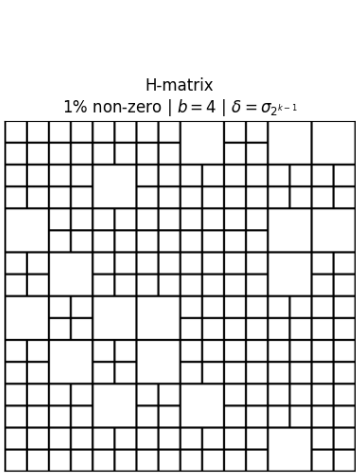
6.4.5 80% zer



6.5 $b = 4, \delta = \sigma_{2^{k-1}}$

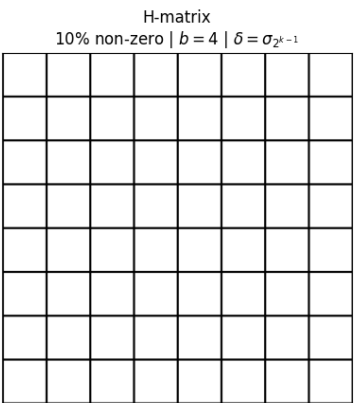
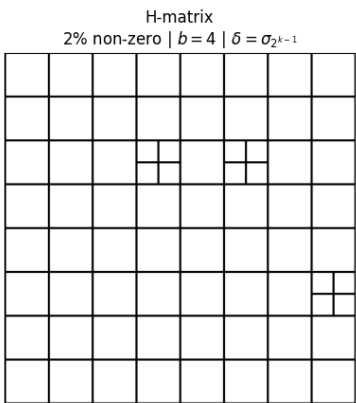
6.5.3 95% zer

6.5.1 99% zer



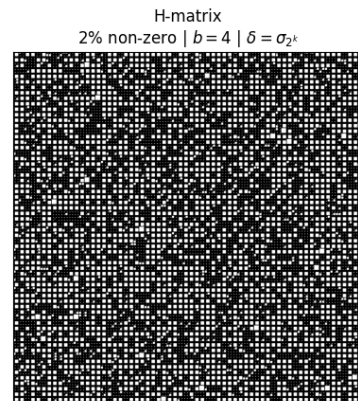
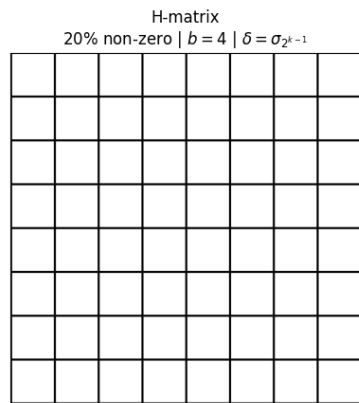
6.5.2 98% zer

6.5.4 90% zer



6.5.5 80% zer

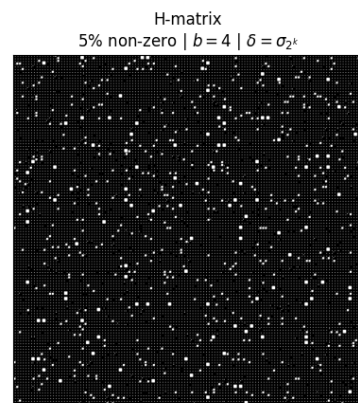
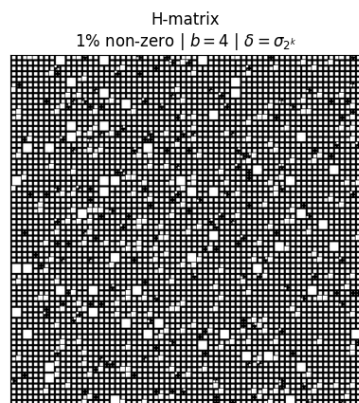
6.6.2 98% zer



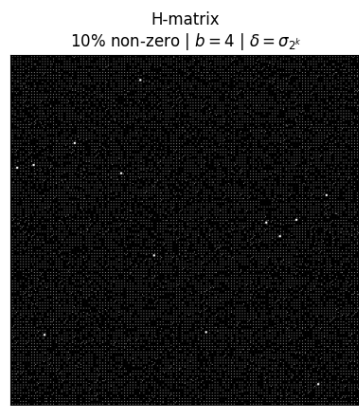
6.6 $b = 4, \delta = \sigma_{2^k}$

6.6.3 95% zer

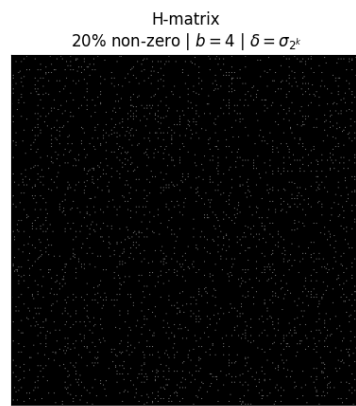
6.6.1 99% zer



6.6.4 90% zer



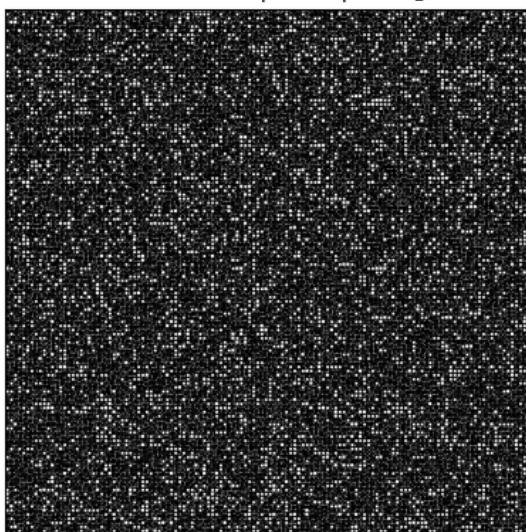
6.6.5 80% zer



6.7 Macierz, którą powiesiłbym w salonie

Ta macierz wydaje się najbardziej majestatyczna, intrygująca i tajemnicza...

H-matrix
5% non-zero | $b = 1$ | $\delta = \sigma_2^k$



7 Wnioski

LABORATORIUM wykazało istotność zagadnienia kompresji i dekompresji macierzy oraz parametryzacji tych operacji oraz wykorzystania tej wiedzy w praktyce.

7.1 Refleksje płynące z laboratirum

1. Generowanie macierzy losowych:

- Parametr sparsity wpływa na stopień zagęszczenia macierzy poprzez kontrolowanie procentowej ilości wartości niezerowych.
- Generowanie macierzy losowych o kontrolowanej sparsity jest istotne w kontekście symulowania rzeczywistych danych, które często posiadają struktury rzadkie.

2. Częściowy rozkład wartości osobliwych (SVD):

- SVD jest używane do rozkładu macierzy na iloczyn trzech macierzy: U, Σ , i V .
- Częściowy SVD pozwala na przybliżenie macierzy oryginalnej za pomocą ograniczonej liczby wartości osobliwych, co może być przydatne w redukcji wymiarów danych.

3. Rekurencyjna kompresja macierzy:

- Rekurencyjna kompresja macierzy przy użyciu częściowego SVD umożliwia eliminację małych wartości osobliwych, co prowadzi do skutecznej kompresji dla macierzy rzadkich.
- Wybór progu kompresji (threshold) wpływa na ilość wartości osobliwych uwzględnionych w kompresji. Niższy próg może prowadzić do większej kompresji kosztem dokładności.

7.2 Wykorzystanie kompresji macierzy

- Kompresja obrazów: Hierarchiczna kompresja może być stosowana w kontekście kompresji obrazów, gdzie szczegółowe informacje o teksturze są zachowywane na różnych poziomach hierarchii.
- Analiza danych: W analizie danych, hierarchiczna kompresja może pomóc w redukcji wymiarów danych hierarchicznie, zaczynając od ogólnych cech.

7.3 Ogólny wniosek

W praktyce, eksperymentowanie z różnymi wartościami parametrów, takimi jak sparsity, k w częściowym SVD, oraz próg kompresji, pozwala na dostosowanie metody do konkretnych danych i celów analizy. Częściowe SVD i rekurencyjna kompresja są używane w dziedzinach takich jak analiza obrazów, redukcja wymiarów danych oraz kompresja danych, gdzie istnieje potrzeba zachowania istotnych cech danych przy jednoczesnym zmniejszeniu ich rozmiaru.

Literatura

- [1] Wykłady i laboratoria prowadzone przez Pana prof. dr hab. Macieja Paszyńskiego
- [2] Wikipedia: SVD
- [3] Wikipedia: macierze ortogonalne
- [4] Wikipedia: matrix sqrt
- [5] Wikipedia: wskaźnik uwarunkowania
