Algorytmy Macierzowe

Sprawozdanie II

Grupa wtorek 13:00b

Michał Kuszewski i Michał Nożkiewicz

12 grudnia 2023

1 Opis zadania i użyte narzędzia

Naszym zadaniem było zaimplementowanie korzystającej z częściowego SVD hierarchicznej kompresji macierzy i wizualizacja tej macierzy po kompresji.
Do realizacji zadania użyliśmy języka Python. Korzystaliśmy z bibliotek numpy, matplotlib, pandas, scipy i Pillow.

2 Pseudokod algorytmu kompresji

Algorytm 1: Matrix Compression

Data: Matrix A, σ, r

$\epsilon := 1 \times 10^{-10}$
$n =$ A.size
if $n = 1$ **then**
    return createLeaf($U = A$, $V = [1]$)
else
    if $n <= r$ **then**
        $r := n - 1$
    $U, S, V := truncated\_svd(A, r + 1)$
    if $|S_{r+1,r+1}| <= \sigma + \epsilon$ **then**
        return createLeaf($U = U_{1:r,1:r} * diag(S_{1:r,1:r})$, $V = V_{1:r,1:r}$)
    $n := n$ **div** 2
    $Node1 := compress(A_{1:n,1:n}, \sigma, r)$
    $Node2 := compress(A_{1:n,n+1:2n}, \sigma, r)$
    $Node3 := compress(A_{n+1:2n,1:n}, \sigma, r)$
    $Node4 := compress(A_{n+1:2n,n+1:2n}, \sigma, r)$
    return createInternalNode(Node1, Node2, Node3, Node4)

3 Ważne fragmenty kodu

3.1 Funkcja tworząca drzewo

```
def compress(matrix, min_value, max_rank, length):
    eps = 1e-10
    if length == 1:
        return Leaf(U=matrix, V=np.array([1])) if abs(matrix[0, 0]) > eps else Leaf(zeros=True)
    else:
        if length <= max_rank + 1:
            max_rank = length - 1
        U, s, V = truncated_svd(matrix, k=max_rank + 1)
        if np.abs(s[-1]) < min_value + eps:
            s_values = s[np.abs(s) >= min_value + eps]
            k = s_values.shape[0]
            if k == 0:
                return Leaf(zeros=True)
            return Leaf(U=U[:, :k] @ np.diag(s_values), V=V[:k])

        length //= 2
        node = InternalNode(
            left_up=compress(matrix[:length, :length], min_value, max_rank, length),
            right_up=compress(matrix[:length, length:], min_value, max_rank, length),
            left_low=compress(matrix[length:, :length], min_value, max_rank, length),
            right_low=compress(matrix[length:, length:], min_value, max_rank,  length)
        )
        return node
```

Rysunek 1: Funkcja compress

3.2 Klasa reprezentująca liść drzewa

```
class Leaf(Node):
    def __init__(self, U=None, V=None, zeros=False):
        self.U = U
        self.V = V
        self.zeros = zeros

    def eval(self, length):
        if self.zeros:
            return 0
        return self.U @ self.V

    def draw(self, image_matrix, left, up, sizes, depth=0):
        if not self.zeros:
            length = sizes[depth]

            k = self.V.shape[0]
            image_matrix[up:up+length, left:left+k] = 0
            image_matrix[up:up+k, left:left+length] = 0
```

Rysunek 2: Klasa Leaf

3.3 Klasa reprezentująca węzeł wewnętrzny drzewa

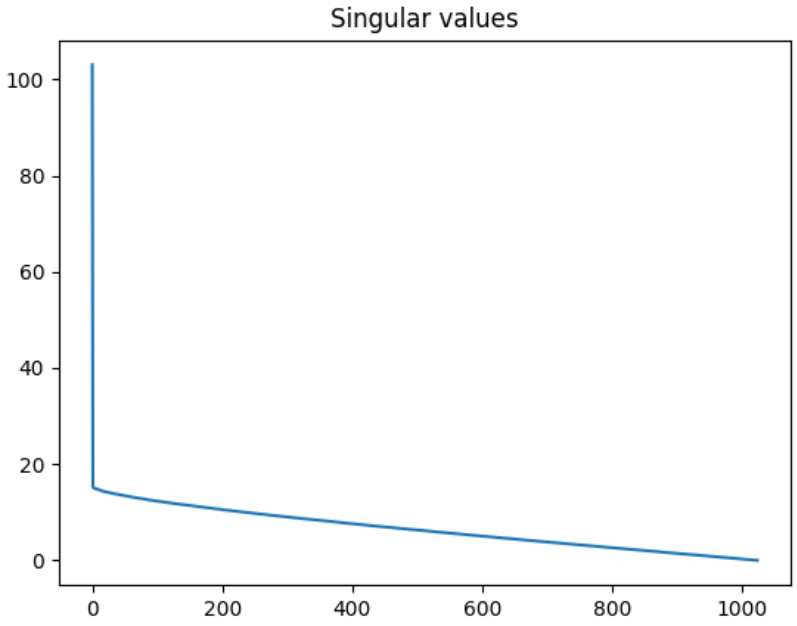
```
class InternalNode(Node):
    def __init__(self, left_up, right_up, left_low, right_low):
        self.left_up = left_up
        self.right_up = right_up
        self.left_low = left_low
        self.right_low = right_low

    def eval(self, length):
        matrix = np.zeros((length, length))
        length //= 2
        matrix[:length, :length] = self.left_up.eval(length)
        matrix[:length, length:] = self.right_up.eval(length)
        matrix[length:, :length] = self.left_low.eval(length)
        matrix[length:, length:] = self.right_low.eval(length)
        return matrix

    def draw(self, im_mat, l, u, sizes, depth=0):
        k = sizes[depth] // 2

        im_mat[u: u + sizes[depth], l + k] = 0
        im_mat[u + k, l: l + sizes[depth]] = 0

        self.left_up.draw(im_mat, l, u, sizes, depth + 1)
        self.right_up.draw(im_mat, l+k+1, u, sizes, depth + 1)
        self.left_low.draw(im_mat, l, u+k+1, sizes, depth + 1)
        self.right_low.draw(im_mat, l+k+1, u+k+1, sizes, depth + 1)
```

Rysunek 3: Klasa InternalNode

4 Testy algorytmu kompresji

Do testów użyliśmy macierzy o wymiarach 1024x1024. Przeprowadziliśmy testy dla macierzy o odpowiednio 80, 90, 95, 98 oraz 99 procent wartości zerowych.

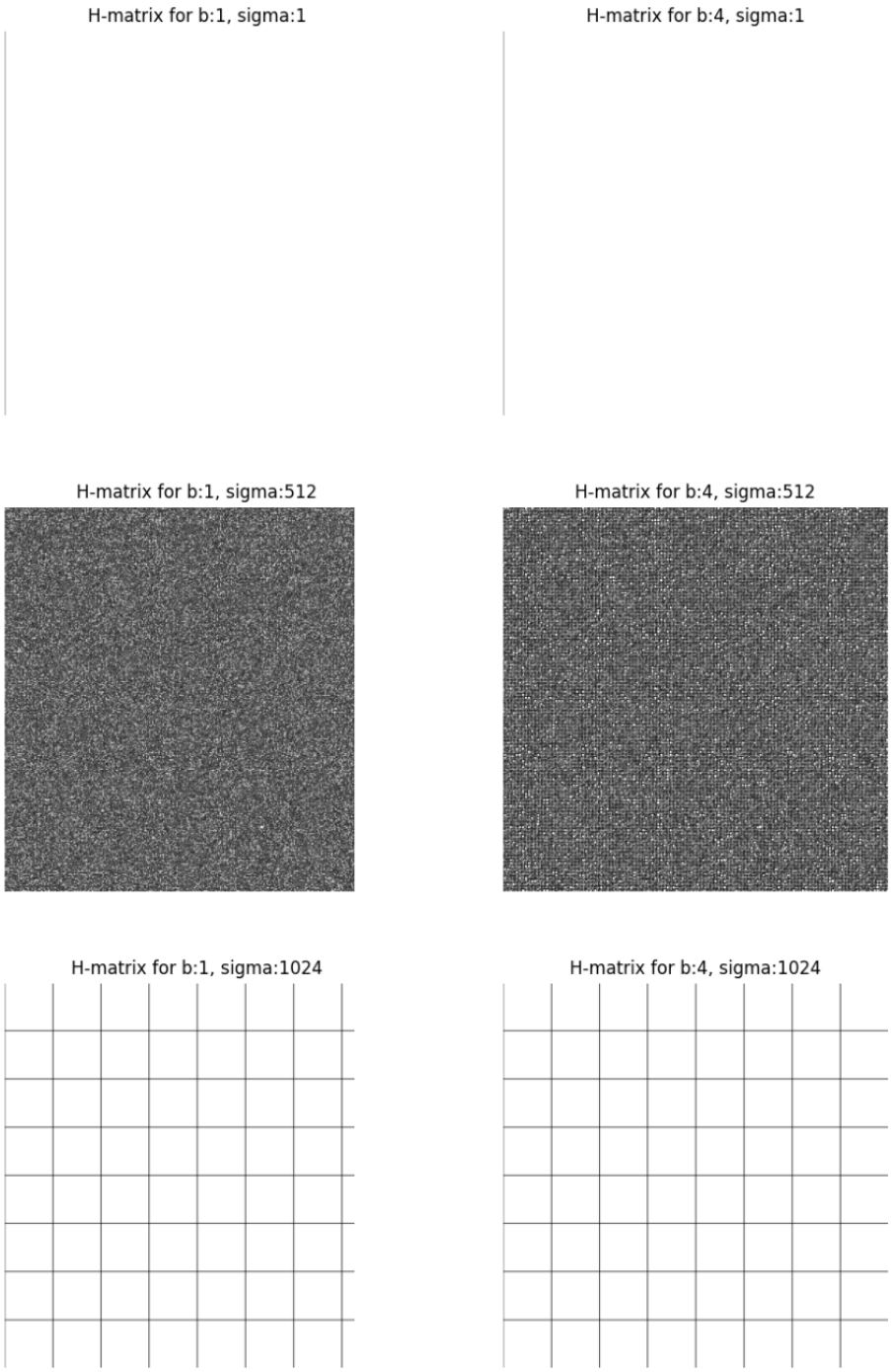
4.1 Macierz o 80 procent wartości zerowych



Rysunek 4: Wartości własne macierzy o 80% zer

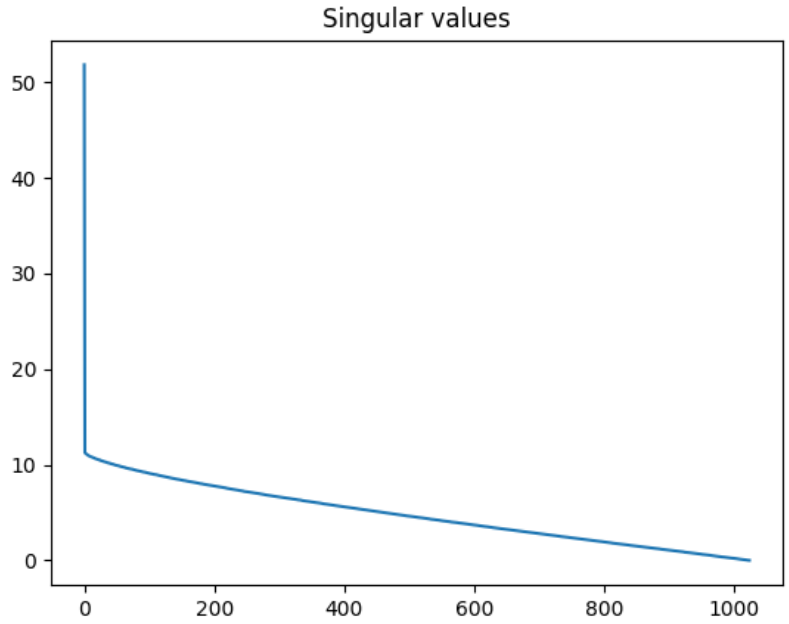
Maksymalny rank	Indeks σ	Wartoć σ	Czas kompresji[s]	Norma różnicy
1	2	15.239962	0.151924	59342.610841
1	1024	0.001667	49.869518	0.000283
1	512	6.153902	3.098952	58460.579358
4	2	15.239962	0.188971	59342.610841
4	1024	0.001667	18.080710	0.000297
4	512	6.153902	3.084782	58460.579358

Tabela 1: Wyniki pomiarów dla macierzy o 80% zer



Rysunek 5: Skompresowane macierze o 80% zer

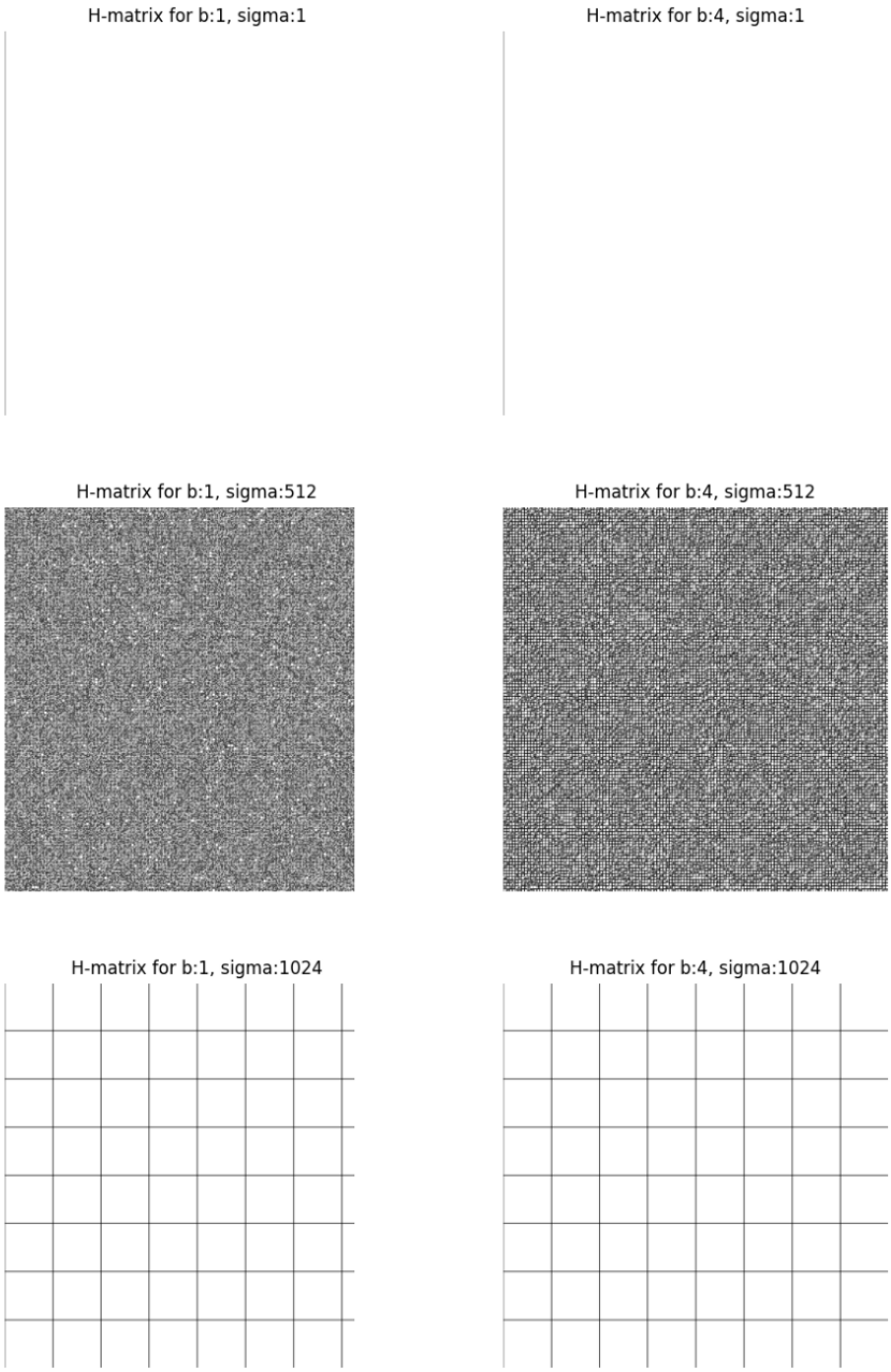
4.2 Macierz o 90 procent wartości zerowych



Rysunek 6: Wartości własne macierzy o 90% zer

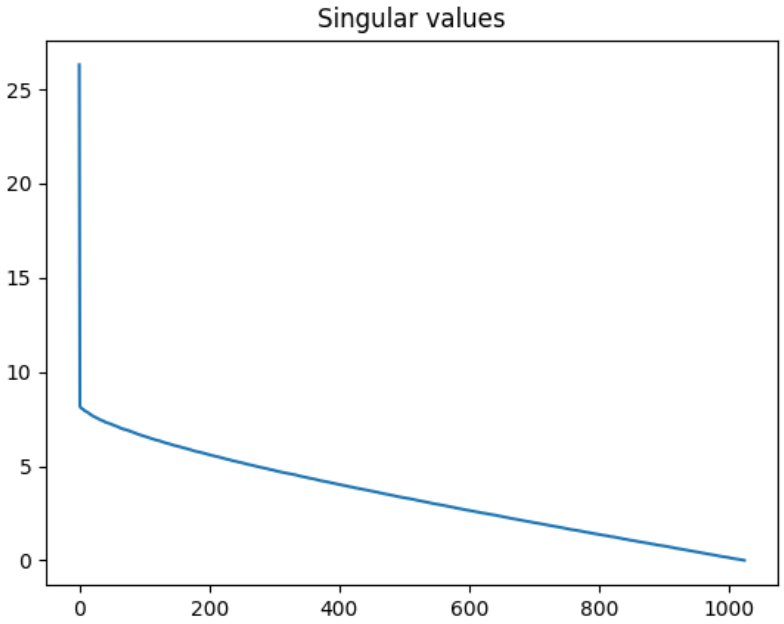
Maksymalny rank	Indeks σ	Wartoć σ	Czas kompresji[s]	Norma różnicy
1	2	11.255902	0.160033	32269.844675
1	1024	0.003064	36.824130	0.000881
1	512	4.518731	2.420678	31750.897428
4	2	11.255902	0.176159	32269.844675
4	1024	0.003064	15.255226	0.000839
4	512	4.518731	3.071007	31750.897428

Tabela 2: Wyniki pomiarów dla macierzy o 90% zer



Rysunek 7: Skompresowane macierze o 90% zer

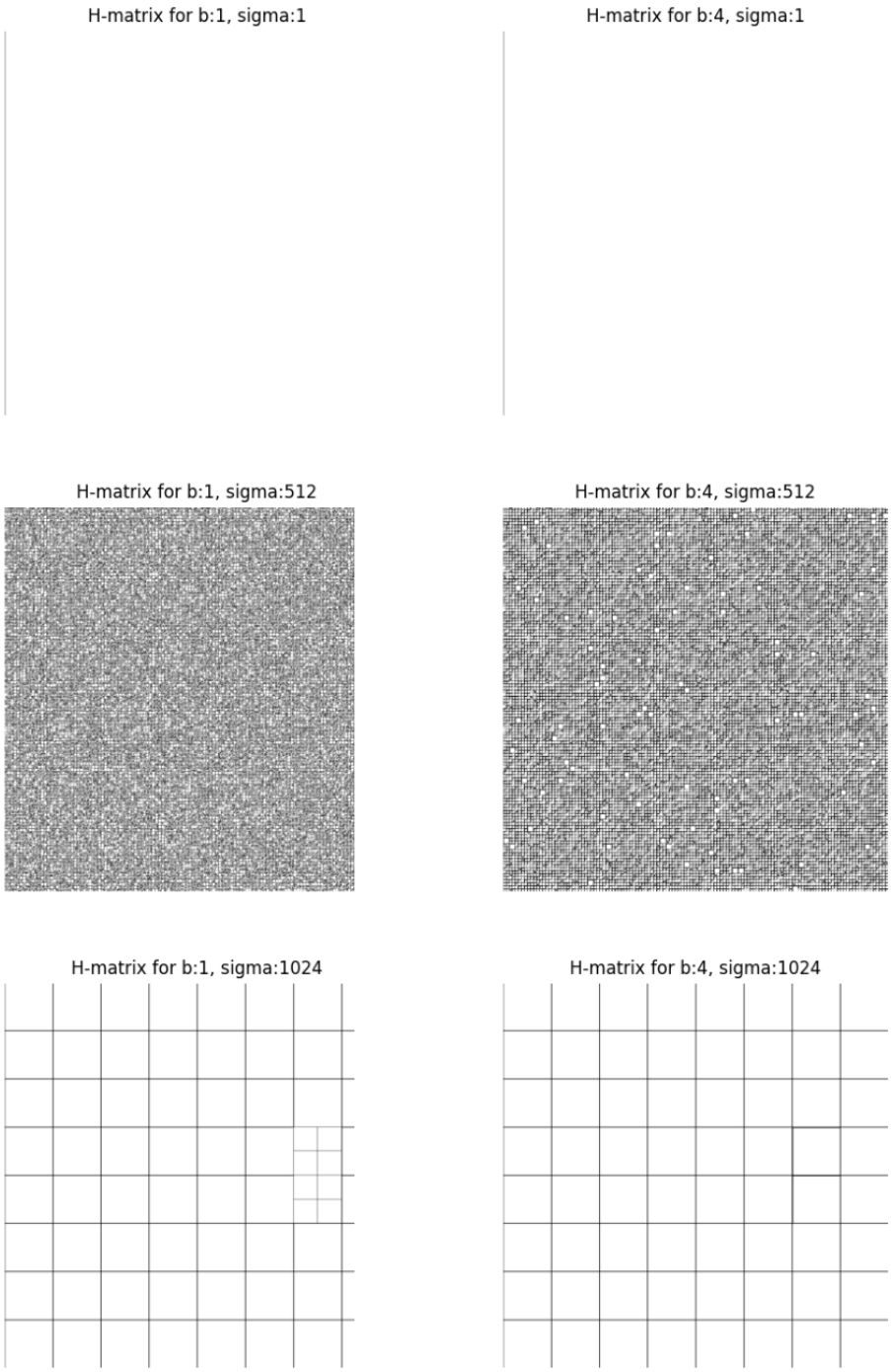
4.3 Macierz o 95 procent wartości zerowych



Rysunek 8: Wartości własne macierzy o 95% zer

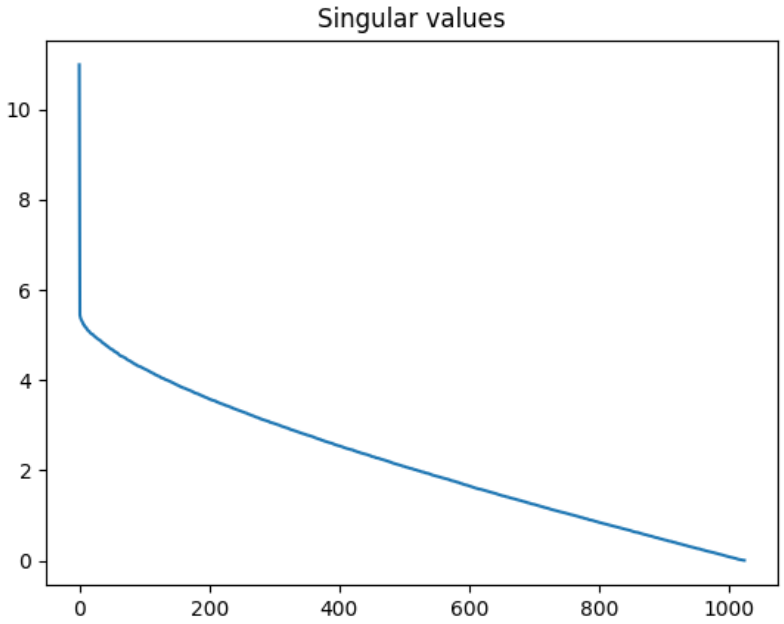
Maksymalny rank	Indeks σ	Wartoć σ	Czas kompresji[s]	Norma różnicy
1	2	8.187151	0.166085	16843.773072
1	1024	0.007090	34.293593	0.004943
1	512	3.256317	2.583114	16555.112550
4	2	8.187151	0.353532	16843.773072
4	1024	0.007090	15.760301	0.004486
4	512	3.256317	2.977887	16499.710490

Tabela 3: Wyniki pomiarów dla macierzy o 95% zer



Rysunek 9: Skompresowane macierze o 95% zer

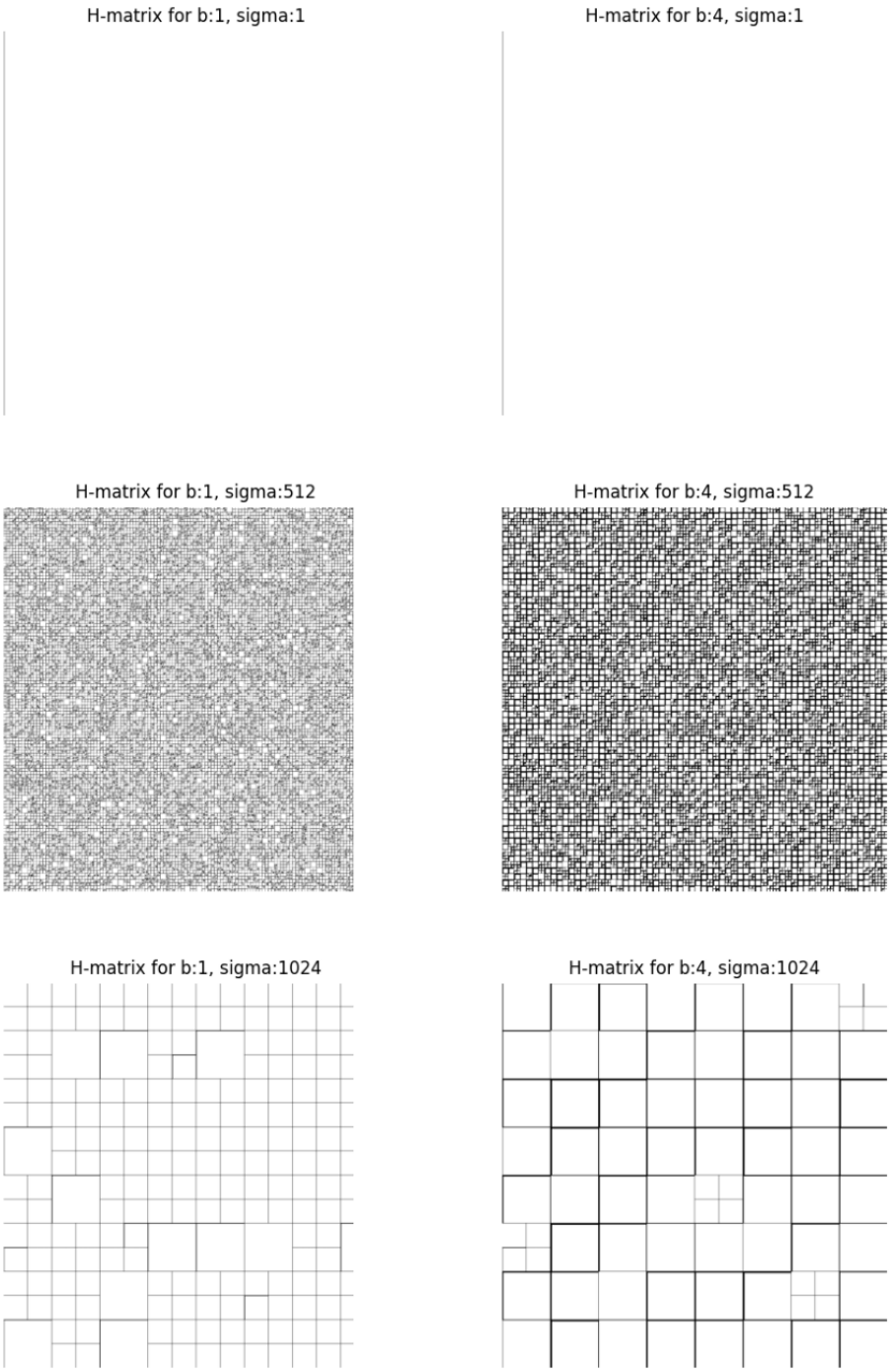
4.4 Macierz o 98 procent wartości zerowych



Rysunek 10: Wartości własne macierzy o 98% zer

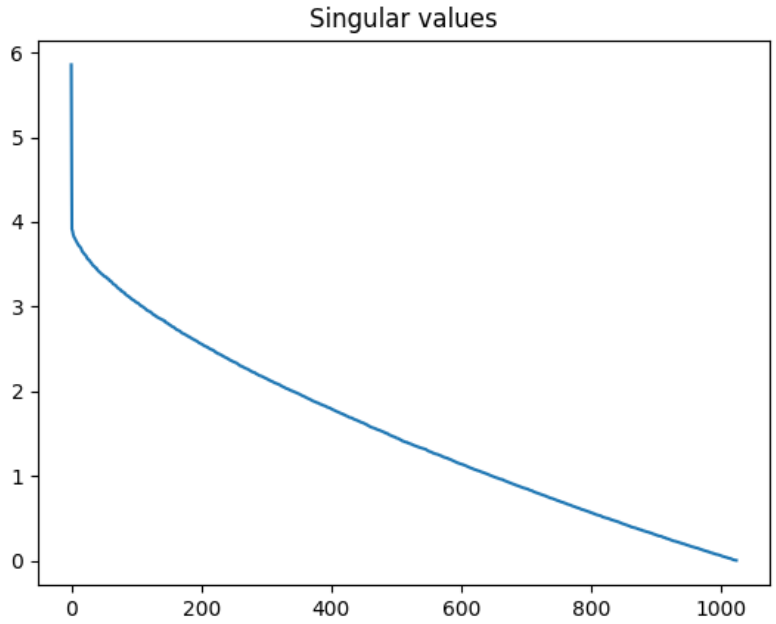
Maksymalny rank	Indeks σ	Wartoć σ	Czas kompresji[s]	Norma różnicy
1	2	5.443215	0.186192	6886.181377
1	1024	0.006911	22.125296	0.002079
1	512	2.037275	3.120790	6939.954720
4	2	5.443215	0.169572	6886.181377
4	1024	0.006911	9.719449	0.001712
4	512	2.037275	2.892564	6300.105302

Tabela 4: Wyniki pomiarów dla macierzy o 98% zer



Rysunek 11: Skompresowane macierze o 98% zer

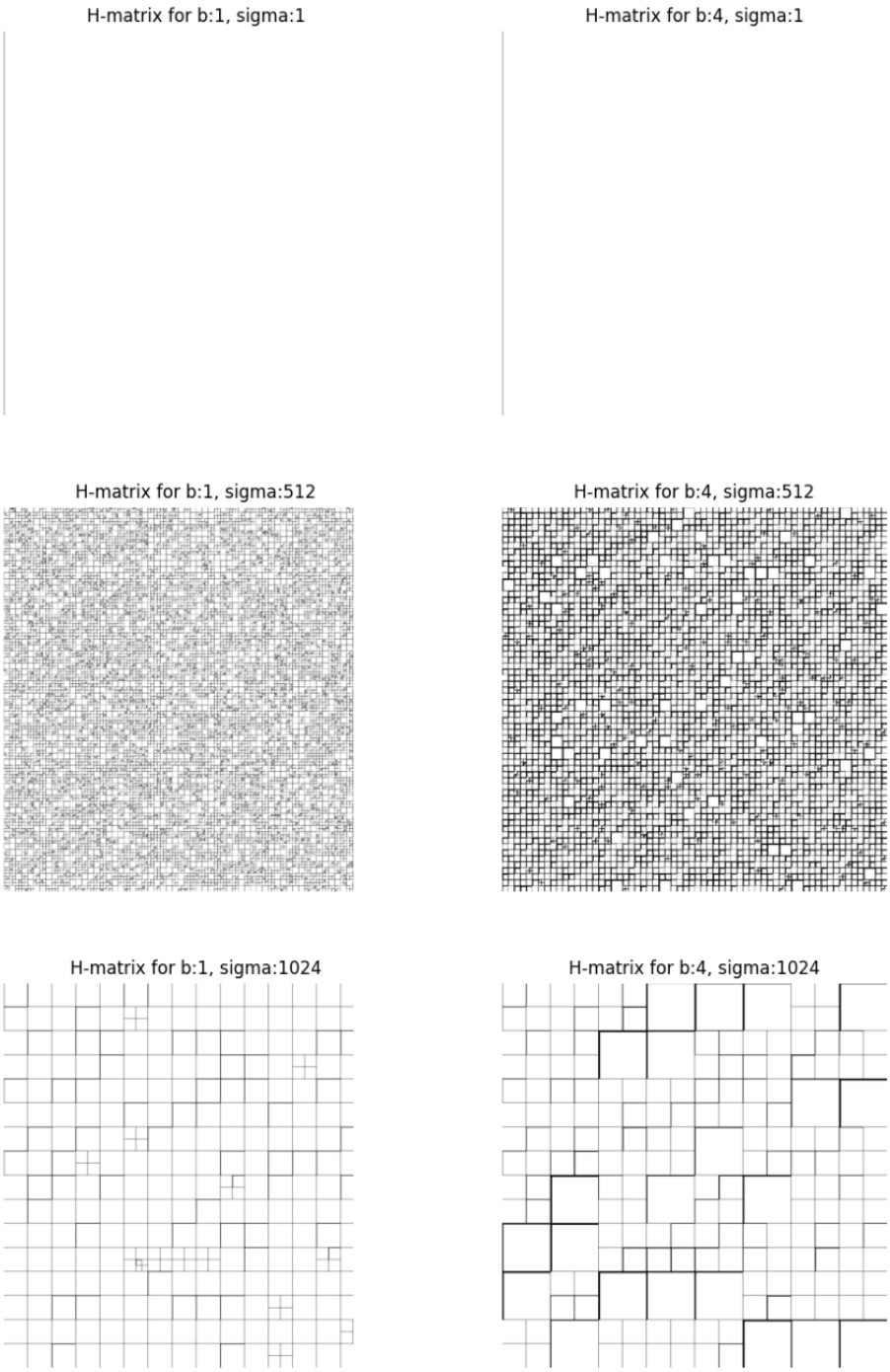
4.5 Macierz o 99 procent wartości zerowych



Rysunek 12: Wartości własne macierzy o 99% zer

Maksymalny rank	Indeks σ	Wartoć σ	Czas kompresji[s]	Norma różnicy
1	2	3.915814	0.150578	3479.470245
1	1024	0.002196	14.657351	0.000036
1	512	1.406898	3.109906	3360.245983
4	2	3.915814	0.228737	3479.470245
4	1024	0.002196	7.481430	0.000035
4	512	1.406898	3.482326	3175.621805

Tabela 5: Wyniki pomiarów dla macierzy o 99% zer



Rysunek 13: Skompresowane macierze o 99% zer

5 Wnioski

Jak można się było spodziewać błąd przybliżenia jest tym mniejszy im mniejsza będzie minimalna dopuszczalna wartość osobliwa. W przypadku, gdy najmniejsza wartość osobliwa była drugą wartością własną macierzy błąd był bardzo duży, a gdy była nią ostatnia wartość osobliwa błąd był znikomy. Błąd był też wysoki, gdy za minimalną wartość osobliwą braliśmy 512 wartość własną. Wynika z tego, że jeśli chcemy, aby nasze przybliżenie było dokładne można sobie pozwolić na zignorowanie najwyżej kilku ostatnich wartości osobliwych, w innych przypadkach błąd jest bardzo duży. W momencie gdy przybliżenie było coraz bardziej dokładne, rósł czas kompresji, a malał, gdy wzrastał maksymalny rank, gdyż wcześniej mogliśmy zakończyć rekurencyjne dzielenie macierzy. Widać, także, że czasami uzyskany błąd był nawet mniejszy, gdy maksymalny rank wynosił 4, co w połączeniu ze znacznie bo ponad dwukrotnie krótszym czasem kompresji sprawia, że taka wartość parametru rank jest lepsza.