Przemysław Rola, Juliusz Wasieleski
Informatyka, III rok, grupa 6
listopad 2023

Algorytmy macierzowe – kompresja Singular Value Decomposition – sprawozdanie

1. Opis ćwiczenia

Naszym zadaniem było , po wybraniu naszego ulubionego języka, wygenerowanie losowych macierzy.

2. Środowisko, biblioteki, założenia oraz użyte narzędzia

Ćwiczenie wykonaliśmy w języku Python przy użyciu Jupyer Notebooka. Do obliczeń, przechowywania danych użyliśmy bibliotek *numpy, pandas, scipy.*

Do rysowania wykresów użyliśmy biblioteki *matplotlib.*

Wszystkie obliczenia prowadziliśmy na komputerze Lenovo Y50-70 z systemem Windows 10 Pro w wersji 10.0.19045, procesor Intel Core i7-4720HQ 2.60GHz, 2601 MHz, rdzenie: 4, procesory logiczne: 8.

3. Implementacje

3.1 Minimal degree permutation

3.1.1 Pseudokod

Minimal degree permutation(matrix):

G(A) = {V, E}

***Dla** k=1 do n-1*

Adj(k) = {l :{k,l} ∈ E

V = V\{k}

E = E\{{k,l : l ∈ adj(k)

E = E ∪ {{x,y}: x ∈ adj(k), y ∈ adj(k)}

3.1.2 Istotne fragmenty implementacji

```
def minimum_degree_permutation(matrix):
    n,m = matrix.shape
    ADJ = {i:set() for i in range(n)}
    for i in range(n):
        for j in range(m):
            if matrix[i][j] != 0 and i != j:
                ADJ[i].add(j)
    permutation = []
    for i in range(n):
        deg_min = m+1
        for v, adj in ADJ.items():
            if len(adj) < deg_min:
                v_min = v
                deg_min = len(adj)
        for v in ADJ:
            ADJ[v] = ADJ[v].difference([v_min])
        for u in ADJ[v_min]:
            ADJ[u] =
(ADJ[u].union(ADJ[v_min].difference([u])))
        ADJ.pop(v_min)
        permutation.append(v_min)
    return permutation
```

3.2 Minimal degree permutation

3.2.1Pseudokod

Minimal degree permutation(matrix):

G(A) = {V, E}

***Dla** k=1 do n-1*

Succ(k) = {l: (k,l) ∈ E}

Pred(k) = {l: (l,k) ∈ E}

V = V \ {k}

E = E \ {k,l) :l ∈ succ(k)}

E = E \ {(l,k) :l ∈ pred(k)}

E = ∪ {(x,y) : x ∈ pred(k), y ∈ succ(l)}

3.2.2 Istotne fragmenty implementacji

```
def cuthill_mckee(matrix):
    def BFS():
        while Q:
            v = Q.popleft()
            if Visited[v]:
                continue
            permutation.append(v)
            Visited[v] = True
            for u in sorted(nx.neighbors(G, v), key = lambda x :
G.degree(x)):
                if not Visited[u]:
                    Q.append(u)
    n = len(matrix)

    G = nx.Graph()
    G.add_nodes_from(range(n))

    for i in range(n):
        for j in range(n):
            if matrix[i][j] != 0 and i != j:
                G.add_edge(i, j)

    permutation = []
    Visited = [False for i in range(n)]
    sorted_nodes = sorted([x for x in G.degree()], key = lambda x :
x[1])
    # print(sorted_nodes)
    sorted_nodes = list(map(lambda x : x[0], sorted_nodes))
    Q = deque()
    for s in sorted_nodes:
        if not Visited[s]:
            Q.append(s)
            BFS()

    return permutation
```

3.3 Minimal degree permutation

3.3.1 Pseudokod

Minimal degree permutation(matrix):

G(A) = {V, E}

***Dla** k=1 do n-1*

Succ(k) = {l: (k,l) ∈ E}

Pred(k) = {l: (l,k) ∈ E}

V = V \ {k}

E = E \ {k,l) :l ∈ succ(k)}

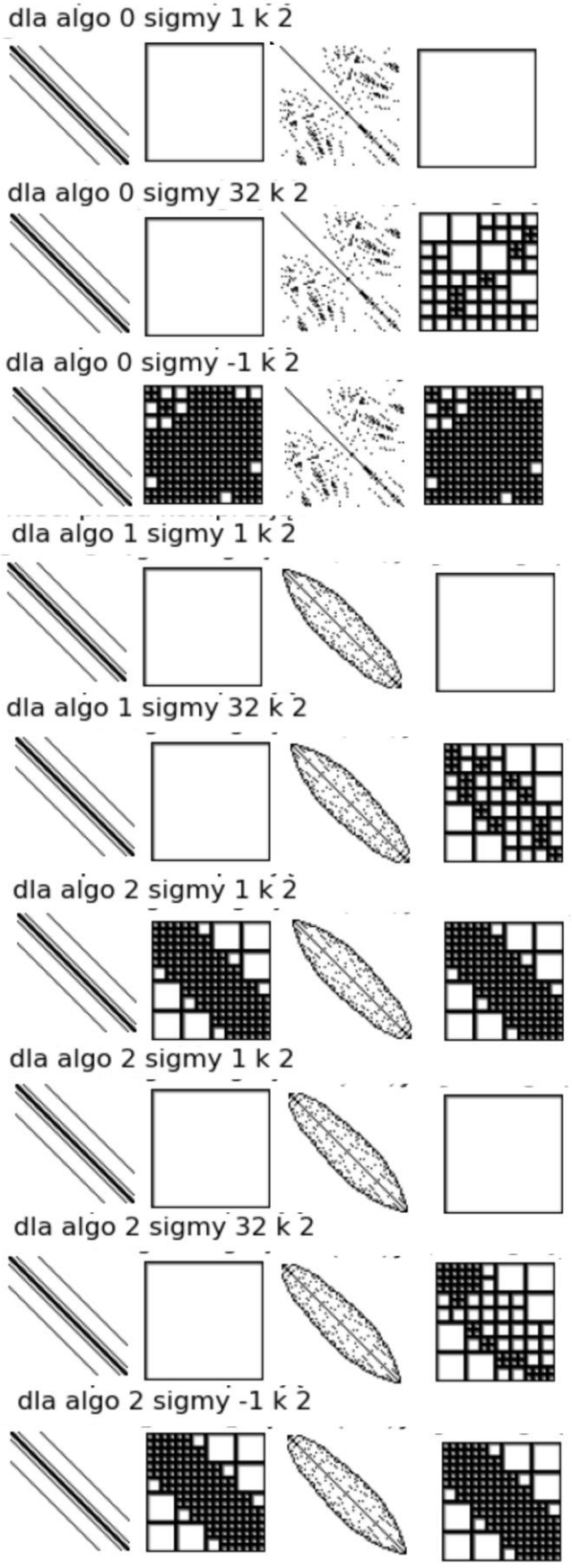
E = E \ {(l,k) :l ∈ pred(k)}

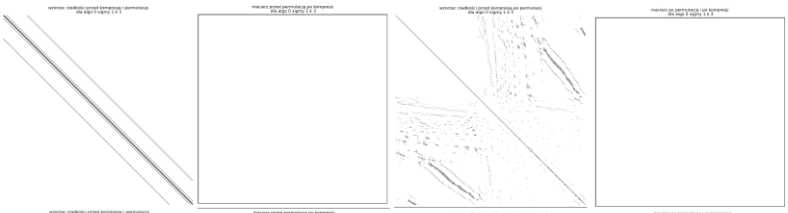
E = ∪ {(x,y) : x ∈ pred(k), y ∈ succ(l)}

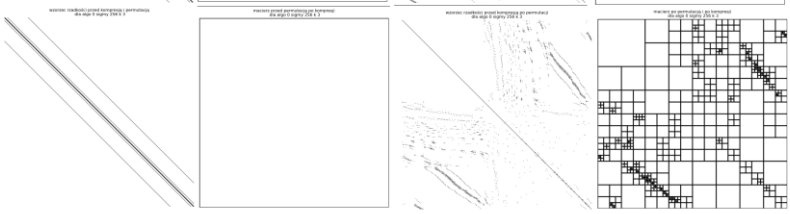
3.3.2 Istotne fragmenty implementacji

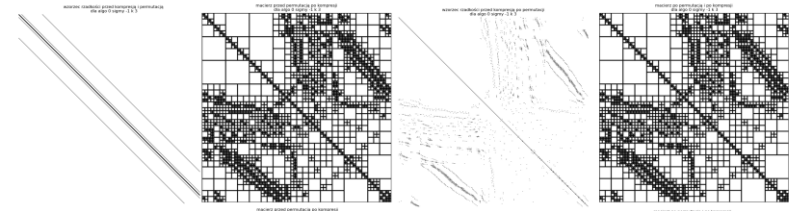
```
def reversed_cuthill_mckee(matrix):
    return cuthill_mckee(matrix)[::-1]
```

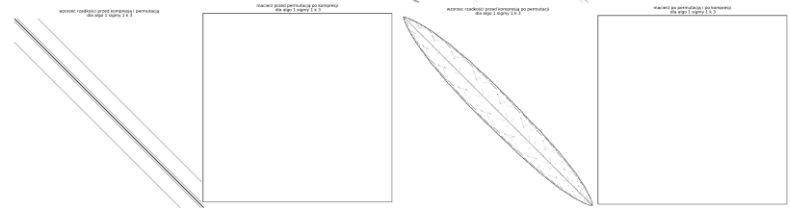
4 Analiza wykonanych pomiarów

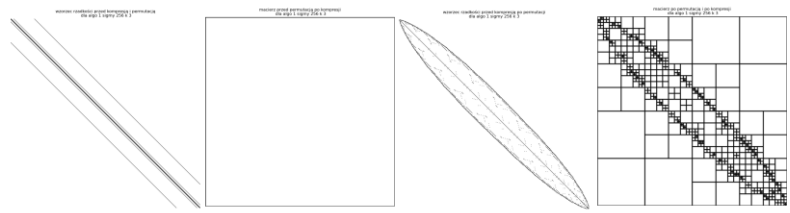


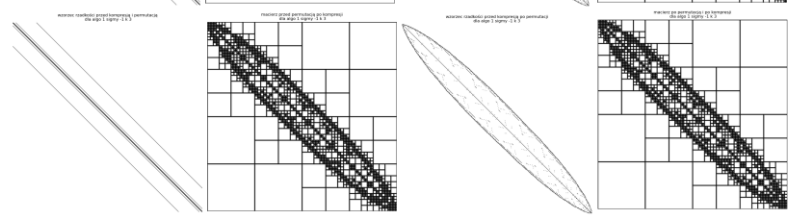

wzorzec rzadkości przed kompresją i permutacją
dla algo 0 sigmy -1 k 5
macierz przed permutacją po kompresji
dla algo 0 sigmy 1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 0 sigmy 1 k 5
macierz po permutacji i po kompresji
dla algo 0 sigmy 1 k 3

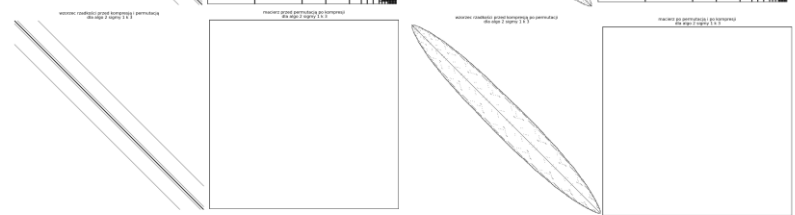

wzorzec rzadkości przed kompresją i permutacją
dla algo 0 sigmy 256 k 5
macierz przed permutacją po kompresji
dla algo 0 sigmy 256 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 0 sigmy 256 k 5
macierz po permutacji i po kompresji
dla algo 0 sigmy 256 k 3

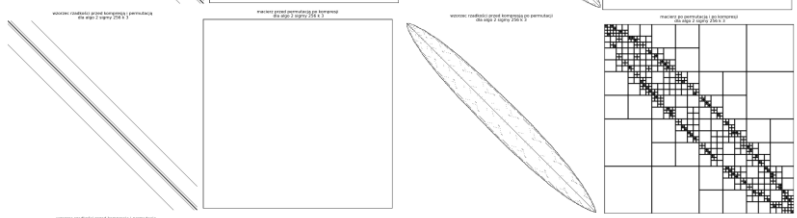

wzorzec rzadkości przed kompresją i permutacją
dla algo 0 sigmy -1 k 5
macierz przed permutacją po kompresji
dla algo 0 sigmy -1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 0 sigmy -1 k 5
macierz po permutacji i po kompresji
dla algo 0 sigmy -1 k 3

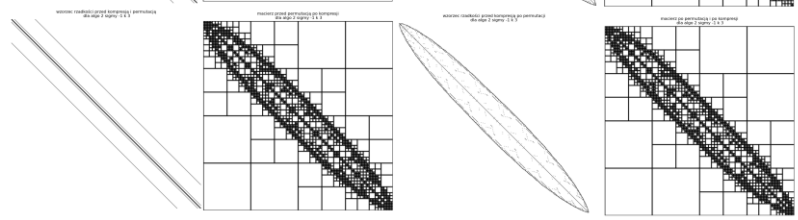

wzorzec rzadkości przed kompresją i permutacją
dla algo 1 sigmy 1 k 5
macierz przed permutacją po kompresji
dla algo 1 sigmy 1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 1 sigmy 1 k 5
macierz po permutacji i po kompresji
dla algo 1 sigmy 1 k 3

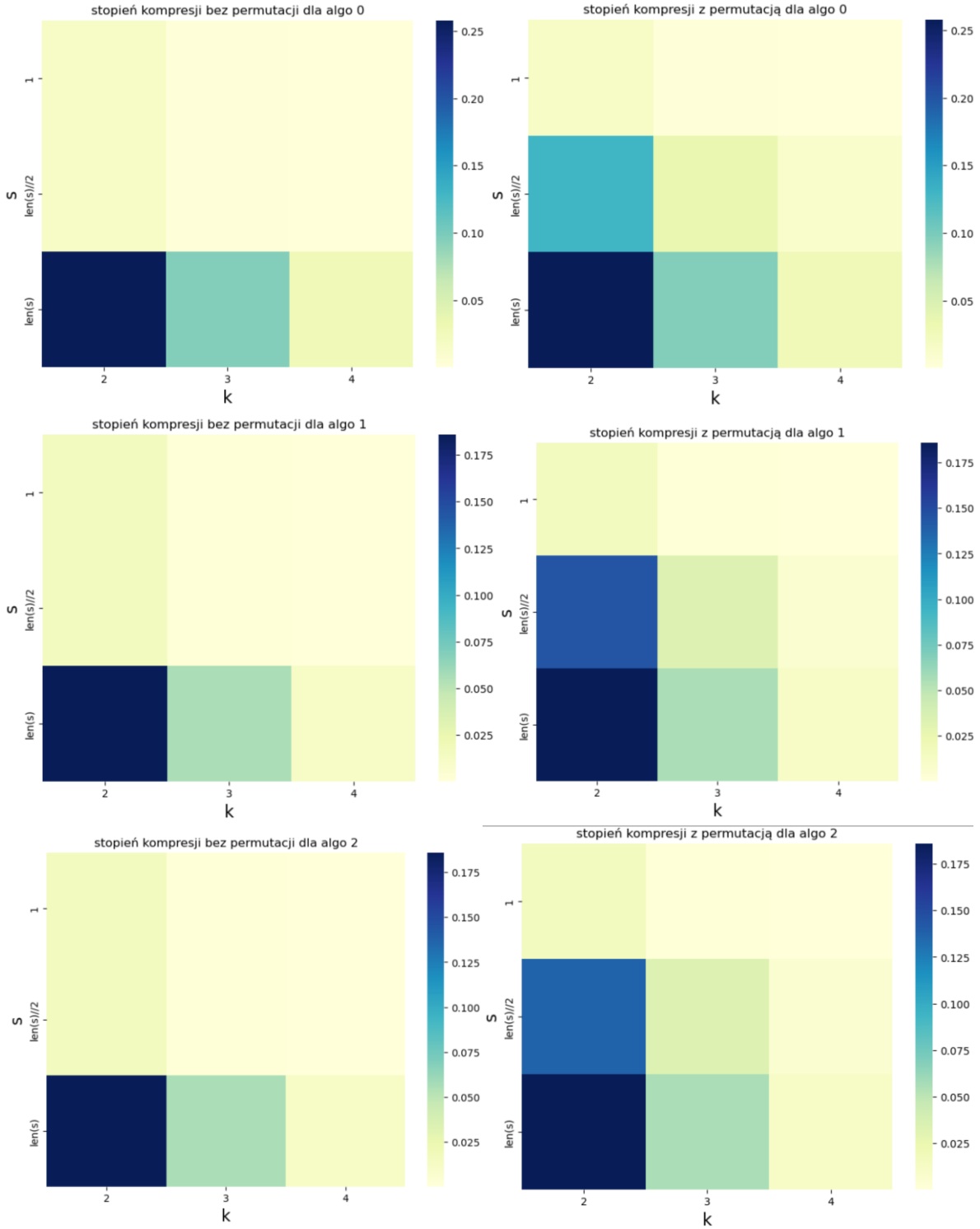

wzorzec rzadkości przed kompresją i permutacją
dla algo 1 sigmy 256 k 5
macierz przed permutacją po kompresji
dla algo 1 sigmy 256 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 1 sigmy 256 k 5
macierz po permutacji i po kompresji
dla algo 1 sigmy 256 k 3


wzorzec rzadkości przed kompresją i permutacją
dla algo 1 sigmy -1 k 5
macierz przed permutacją po kompresji
dla algo 1 sigmy -1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 1 sigmy -1 k 5
macierz po permutacji i po kompresji
dla algo 1 sigmy -1 k 3


wzorzec rzadkości przed kompresją i permutacją
dla algo 2 sigmy 1 k 5
macierz przed permutacją po kompresji
dla algo 2 sigmy 1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 2 sigmy 1 k 5
macierz po permutacji i po kompresji
dla algo 2 sigmy 1 k 3


wzorzec rzadkości przed kompresją i permutacją
dla algo 2 sigmy 256 k 5
macierz przed permutacją po kompresji
dla algo 2 sigmy 256 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 2 sigmy 256 k 5
macierz po permutacji i po kompresji
dla algo 2 sigmy 256 k 3


wzorzec rzadkości przed kompresją i permutacją
dla algo 2 sigmy -1 k 5
macierz przed permutacją po kompresji
dla algo 2 sigmy -1 k 3
wzorzec rzadkości przed kompresją po permutacji
dla algo 2 sigmy -1 k 5
macierz po permutacji i po kompresji
dla algo 2 sigmy -1 k 3


stopień kompresji bez permutacji dla algo 0
stopień kompresji z permutacją dla algo 0
stopień kompresji bez permutacji dla algo 1
stopień kompresji z permutacją dla algo 1
stopień kompresji bez permutacji dla algo 2
stopień kompresji z permutacją dla algo 2
s
k
1
len(s)//2
len(s)
2
3
4
0.25
0.20
0.15
0.10
0.05
0.175
0.150
0.125
0.100
0.075
0.050
0.025

5 Wnioski

-