

Algorytmy macierzowe - laboratorium 4

Marta Bukowińska

Jakub Karczewski

15.01.2024

1 Treść zadania

1. Wybrać ulubiony język programowania.
2. Wygenerować macierz o rozmiarze $2^{3k} = 2^k \cdot 2^k \cdot 2^k$ dla $k = 2, 3, 4$, która opisuje strukturę topologii trójwymiarowej siatki zbudowanej z elementów sześciennych.
 - Wiersz w macierzy odpowiada wierzchołkowi siatki.
 - Niezerowe losowe wartości w kolumnach oznaczają sąsiadujące wierzchołki siatki.
3. Zaimplementować rekurencyjną procedurę kompresji macierzy z Zadania 3.
4. Narysować skompresowaną macierz, wykorzystując rysowacz z Zadania 3.
5. Wykonać mnożenie skompresowanej macierzy przez wektor (20 punktów).
6. Wykonać mnożenie skompresowanej macierzy przez samą siebie (20 punktów).

2 Fragmenty kodu

2.1 Mnożenie macierzy przez wektor

```
1 def matrix_vector_mult(v, X):
2     if not v.sons:
3         if v.rank == 0:
4             return np.zeros(len(X))
5         return np.matmul(v.U, np.matmul(v.V, X))
6     rows = len(X)
7     X1 = X[: rows // 2]
8     X2 = X[rows // 2 :]
9     Y11 = matrix_vector_mult(v.sons[0], X1)
10    Y12 = matrix_vector_mult(v.sons[1], X2)
```

```

11 Y21 = matrix_vector_mult(v.sons[2], X1)
12 Y22 = matrix_vector_mult(v.sons[3], X2)
13 return np.hstack((np.add(Y11, Y12), np.add(Y21, Y22)))

```

2.2 Mnożenie macierzy przez macierz

```

1 def rSVDofCompressed(v, w):
2     U = np.hstack((v.U, w.U))
3     V = np.vstack((v.V, w.V))
4     M = np.matmul(U, V)
5     return CreateTree(M, 0, M.shape[0]-1, 0, M.shape[1]-1, r, eps)
6
7 def create_subnode(v, U_slice, V_slice):
8     u = Node()
9     u.rank = v.rank
10    u.U = v.U[U_slice, :]
11    u.V = v.V[:, V_slice]
12    u.sons = []
13    return u
14
15 def matrix_matrix_add(v, w):
16     if not v.sons and not w.sons:
17         if v.rank == 0 and w.rank == 0:
18             res = Node(0)
19             return res
20         if v.rank != 0 and w.rank != 0:
21             return rSVDofCompressed(v, w)
22     if v.rank != 0 and w.rank == 0:
23         return v
24     if v.rank == 0 and w.rank != 0:
25         return w
26     if v.sons and w.sons:
27         res = Node()
28         res.sons.append(matrix_matrix_add(v.sons[0], w.sons[0]))
29         res.sons.append(matrix_matrix_add(v.sons[1], w.sons[1]))
30         res.sons.append(matrix_matrix_add(v.sons[2], w.sons[2]))
31         res.sons.append(matrix_matrix_add(v.sons[3], w.sons[3]))
32         return res
33     if not v.sons and w.sons:
34         half = len(v.U) // 2
35         u11 = create_subnode(v, slice(0, half), slice(0, half))
36         u12 = create_subnode(v, slice(0, half), slice(half, None))
37         u21 = create_subnode(v, slice(half, None), slice(0, half))
38         u22 = create_subnode(v, slice(half, None), slice(half, None))
39         res = Node()
40         res.sons.append(matrix_matrix_add(u11, w.sons[0]))
41         res.sons.append(matrix_matrix_add(u12, w.sons[1]))
42         res.sons.append(matrix_matrix_add(u21, w.sons[2]))
43         res.sons.append(matrix_matrix_add(u22, w.sons[3]))
44         return res
45     if v.sons and not w.sons:
46         half = len(w.U) // 2
47         u11 = create_subnode(w, slice(0, half), slice(0, half))
48         u12 = create_subnode(w, slice(0, half), slice(half, None))

```

```

49     u21 = create_subnode(w, slice(half, None), slice(0, half))
50     u22 = create_subnode(w, slice(half, None), slice(half, None))
51     res = Node()
52     res.sons.append(matrix_matrix_add(v.sons[0], u11))
53     res.sons.append(matrix_matrix_add(v.sons[1], u12))
54     res.sons.append(matrix_matrix_add(v.sons[2], u21))
55     res.sons.append(matrix_matrix_add(v.sons[3], u22))
56     return res

```

```

1 def matrix_matrix_mult(v, w):
2     if not v.sons and not w.sons:
3         if v.rank == 0 and w.rank == 0:
4             res = Node(0)
5             return res
6         if v.rank != 0 and w.rank != 0:
7             res = Node(v.rank)
8             res.U = v.U * (v.V * w.U)
9             res.V = w.V
10            return res
11
12    if v.rank != 0 and w.rank == 0:
13        return w
14
15    if v.rank == 0 and w.rank != 0:
16        return v
17
18    if v.sons and w.sons:
19        res = Node()
20        res.sons.append(
21            matrix_matrix_add(
22                matrix_matrix_mult(v.sons[0], w.sons[0]),
23                matrix_matrix_mult(v.sons[1], w.sons[2])
24            )
25        )
26        res.sons.append(
27            matrix_matrix_add(
28                matrix_matrix_mult(v.sons[0], w.sons[1]),
29                matrix_matrix_mult(v.sons[1], w.sons[3])
30            )
31        )
32        res.sons.append(
33            matrix_matrix_add(
34                matrix_matrix_mult(v.sons[2], w.sons[0]),
35                matrix_matrix_mult(v.sons[3], w.sons[2])
36            )
37        )
38        res.sons.append(
39            matrix_matrix_add(
40                matrix_matrix_mult(v.sons[2], w.sons[1]),
41                matrix_matrix_mult(v.sons[3], w.sons[3])
42            )

```

```

43     )
44     return res
45
46 if not v.sons and w.sons:
47     half = len(v.U) // 2
48     u11 = create_subnode(v, slice(0, half), slice(0, half))
49     u12 = create_subnode(v, slice(0, half), slice(half, None))
50     u21 = create_subnode(v, slice(half, None), slice(0, half))
51     u22 = create_subnode(v, slice(half, None), slice(half, None))
52     res = Node()
53     res.sons.append(
54         matrix_matrix_add(
55             matrix_matrix_mult(u11, w.sons[0]),
56             matrix_matrix_mult(u12, w.sons[2])
57         )
58     )
59     res.sons.append(
60         matrix_matrix_add(
61             matrix_matrix_mult(u11, w.sons[1]),
62             matrix_matrix_mult(u12, w.sons[3])
63         )
64     )
65     res.sons.append(
66         matrix_matrix_add(
67             matrix_matrix_mult(u21, w.sons[0]),
68             matrix_matrix_mult(u22, w.sons[2])
69         )
70     )
71     res.sons.append(
72         matrix_matrix_add(
73             matrix_matrix_mult(u21, w.sons[1]),
74             matrix_matrix_mult(u22, w.sons[3])
75         )
76     )
77     return res
78
79 if v.sons and not w.sons:
80     half = len(w.U) // 2
81     u11 = create_subnode(w, slice(0, half), slice(0, half))
82     u12 = create_subnode(w, slice(0, half), slice(half, None))
83     u21 = create_subnode(w, slice(half, None), slice(0, half))
84     u22 = create_subnode(w, slice(half, None), slice(half, None))
85     res = Node()
86     res.sons.append(
87         matrix_matrix_add(
88             matrix_matrix_mult(v.sons[0], u11),
89             matrix_matrix_mult(v.sons[1], u21)
90         )
91     )
92     res.sons.append(
93         matrix_matrix_add(
94             matrix_matrix_mult(v.sons[0], u12),
95             matrix_matrix_mult(v.sons[1], u22)
96         )
97     )

```

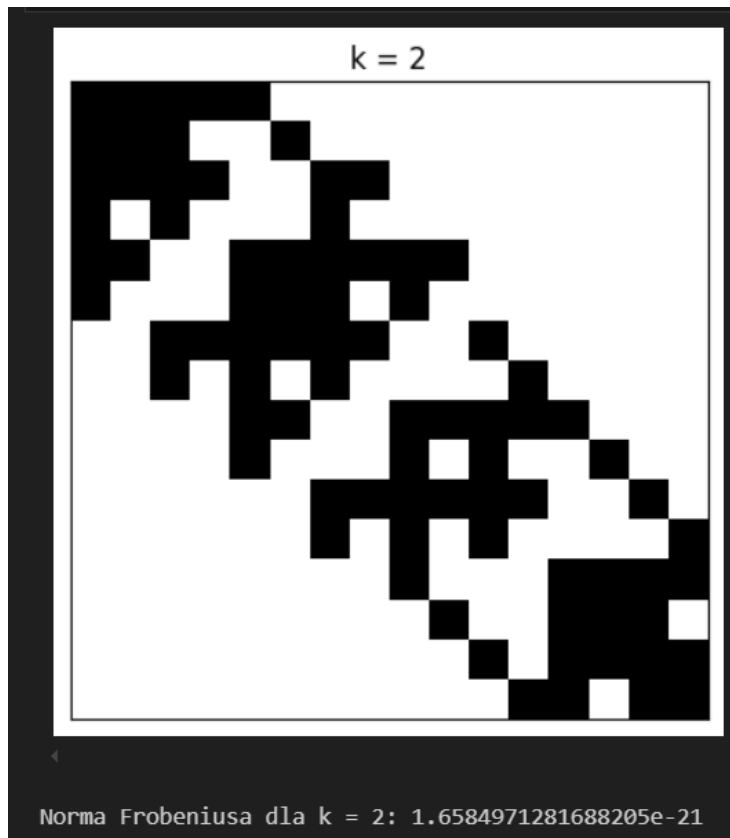
```

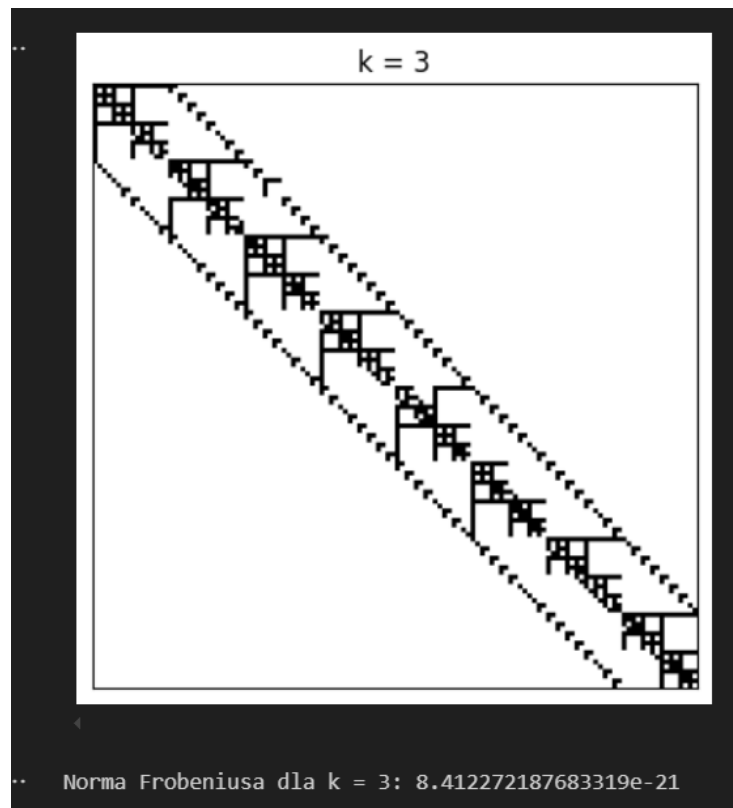
98     res.sons.append(
99         matrix_matrix_add(
100             matrix_matrix_mult(v.sons[2], u11),
101             matrix_matrix_mult(v.sons[3], u21)
102         )
103     )
104     res.sons.append(
105         matrix_matrix_add(
106             matrix_matrix_mult(v.sons[2], u12),
107             matrix_matrix_mult(v.sons[3], u22)
108         )
109     )
110     return res

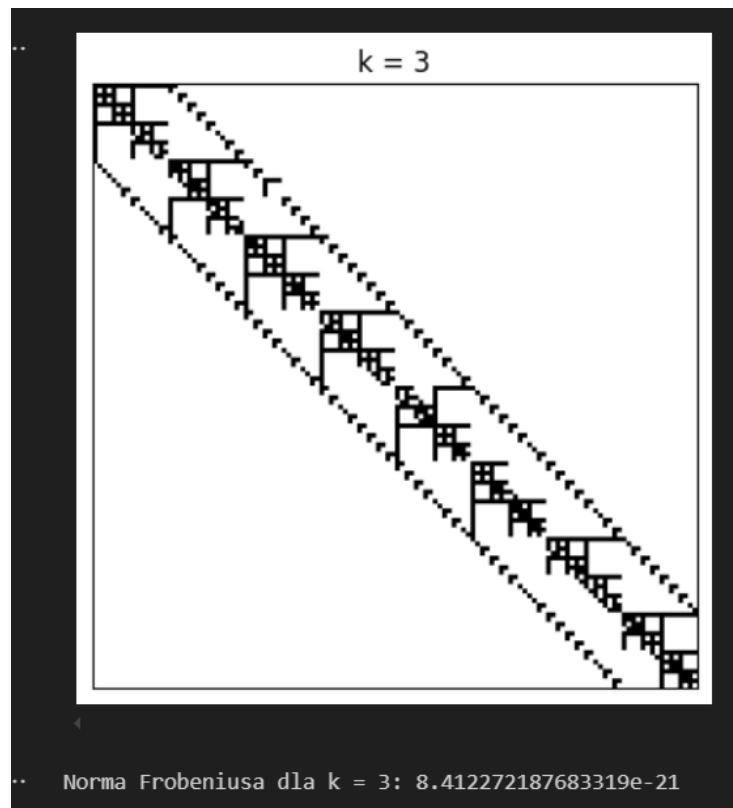
```

3 Wykresy

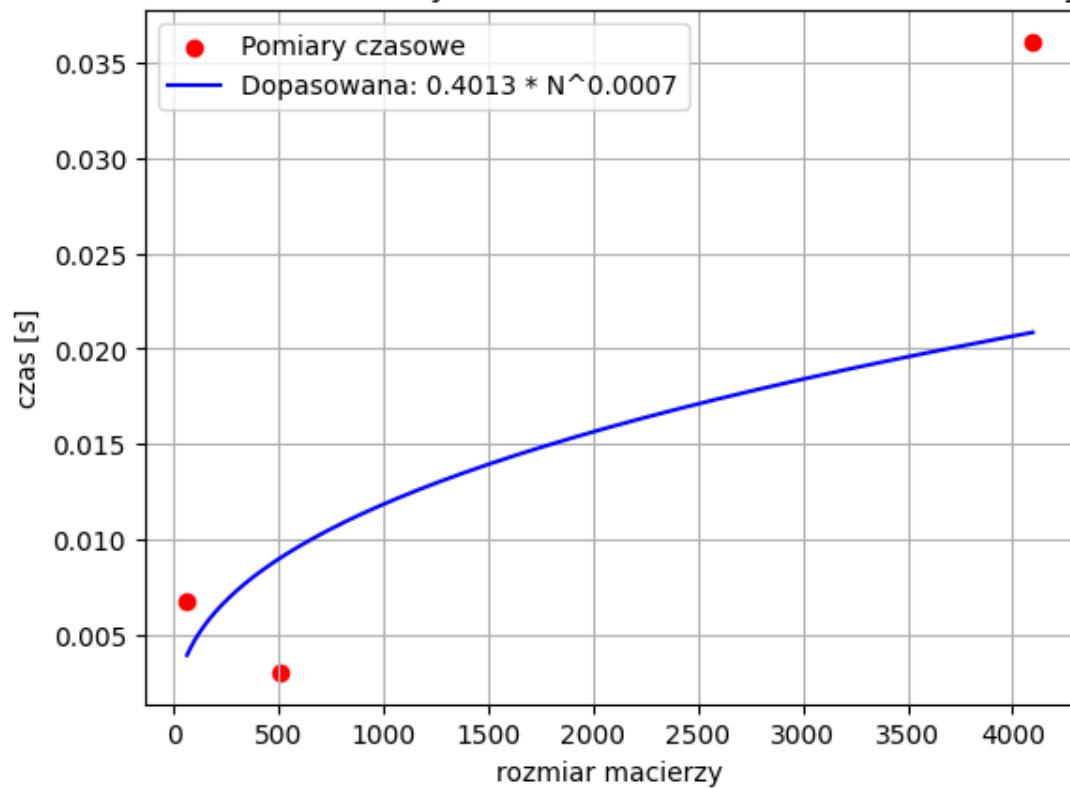
3.1 Mnożenie macierzy przez wektor



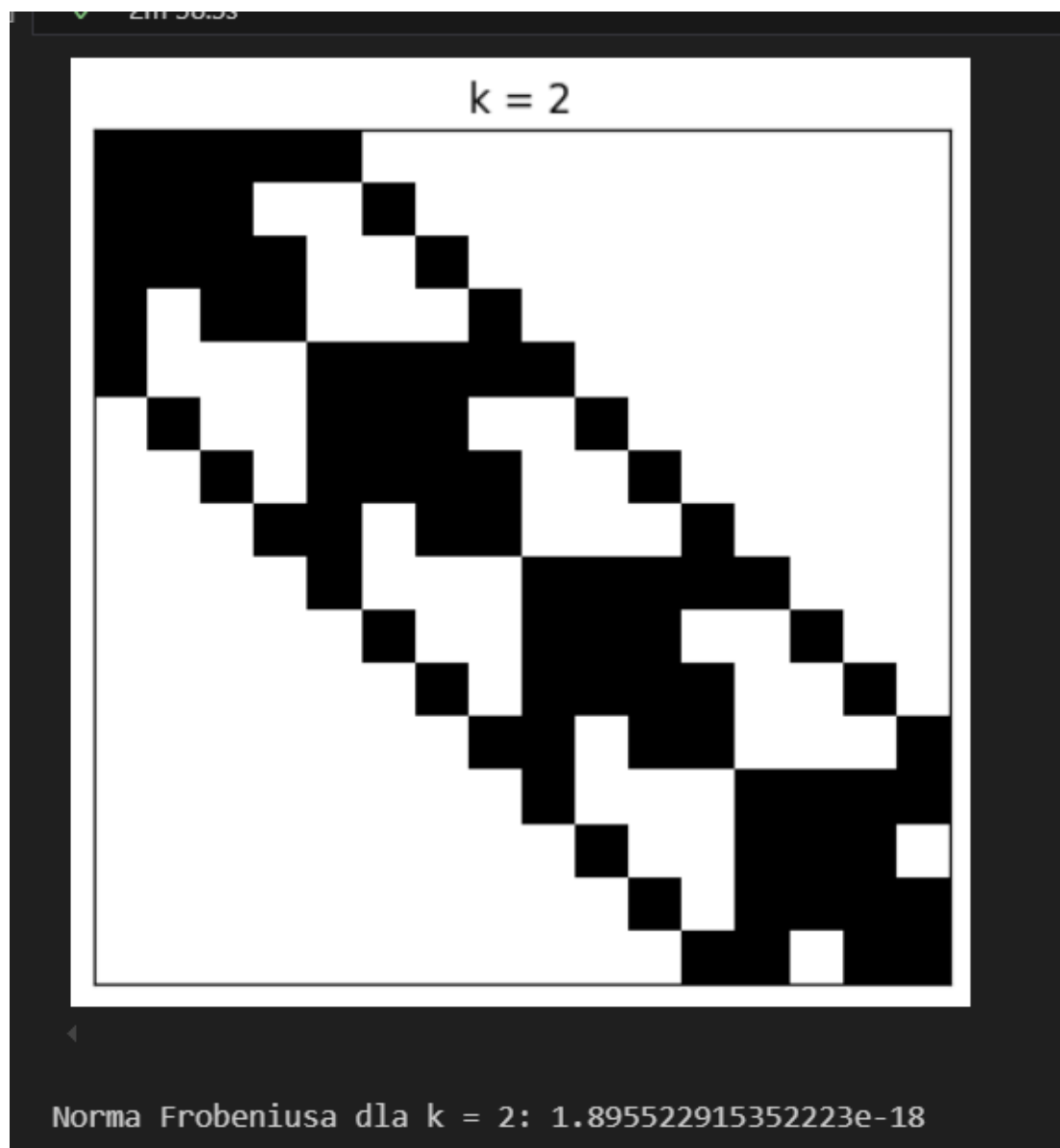


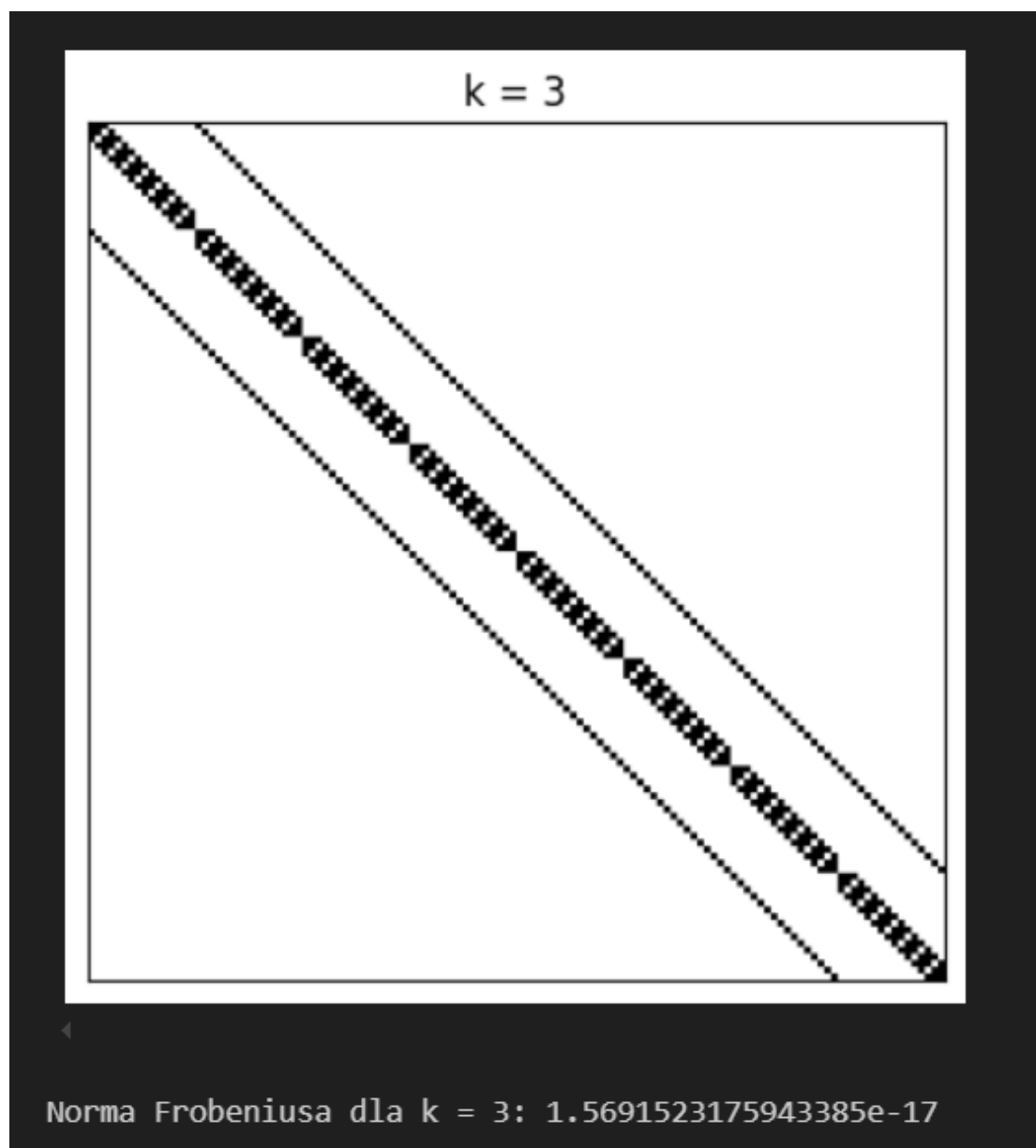


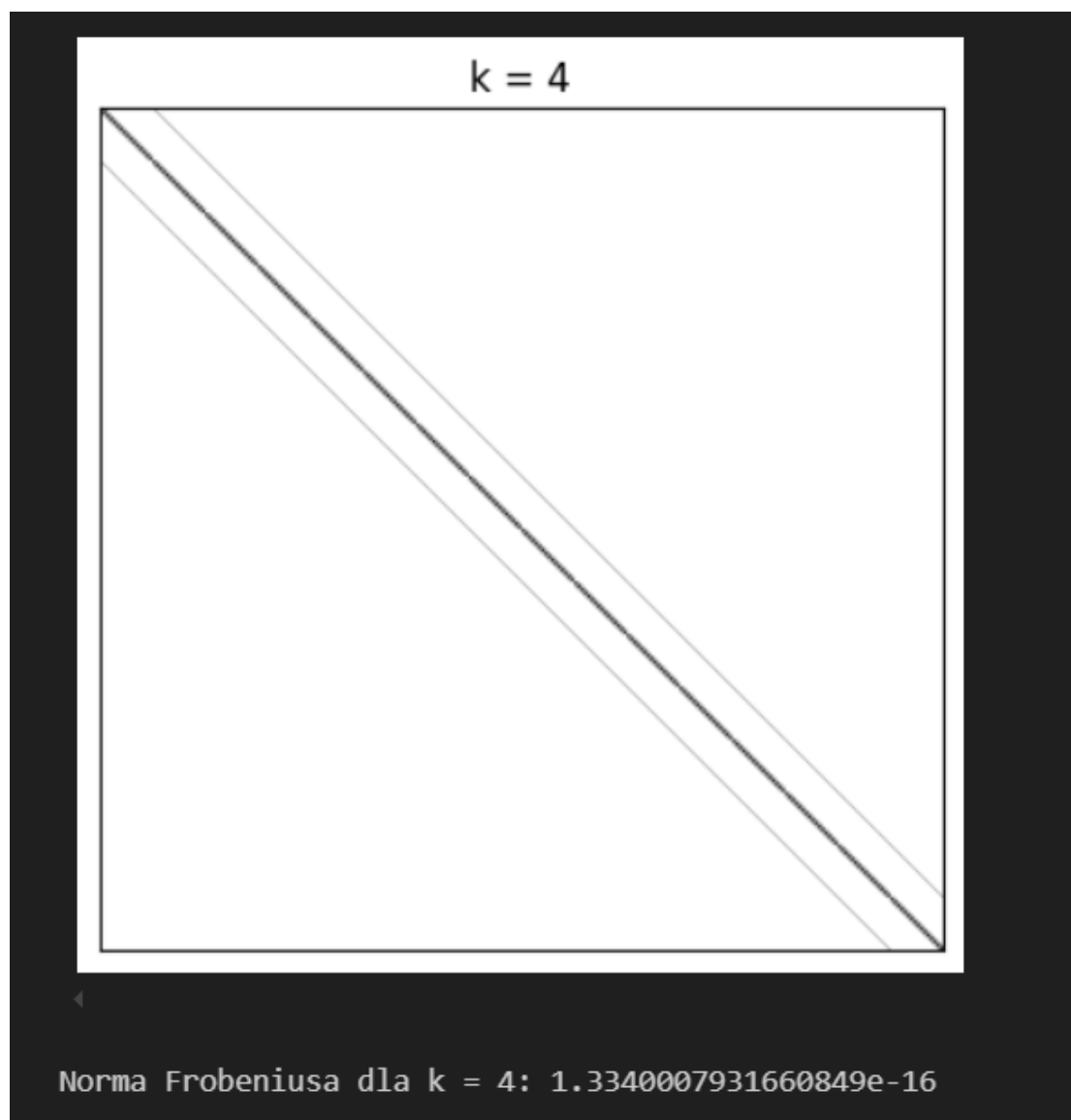
Zależność czasu wykonania mnożenia od rozmiaru macierzy



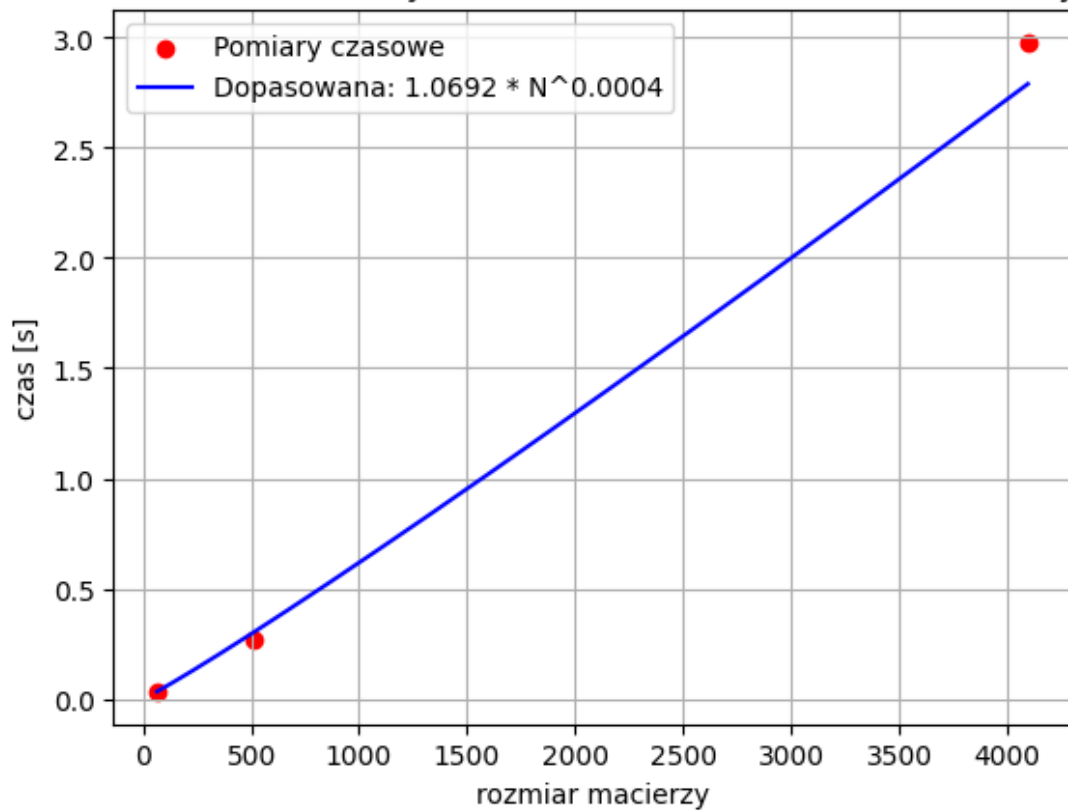
3.2 Mnożenie macierzy przez macierz







Zależność czasu wykonania mnożenia od rozmiaru macierzy



4 Wnioski