PROJEKT - WARTOŚCI I WEKTORY WŁASNE MACIERZY

Łukasz Siemieniec, Konrad Skutnik, Natalia Wójcik

Politechnika Krakowska im.Tadeusza Kościuszki

11/05/2021

Opis projektu

Program został wykonany przy użyciu środowiska Python, oraz wykorzystuje algorytmy struktury danych.

Na program składa sie metoda potegowa

W wersji podstawowej służy do wyznaczenia maksymalnej wartości własnej i odpowiadajacego jej wektora własnego.

Polega na wykonaniu ciagu mnożeń przyjetego wektora startowego przez macierz, której dominujacej wartości własnej i odpowiadajacego wektora własnego poszukujemy.

Metoda wykorzystana do algorytmu pagerank – służy do klasyfikacji wezłów w sieci pod wzgledem ich ważności.

Opis projektu

Projekt jest podzielony na dwie cześci.

- W pierwszej cześci stworzyliśmy graf skierowany o kilkudziesieciu wezłach (N) i krawedziach (E), korzystajac z metody gnm-random-graph(N,E,directed=True) zawartej w bibliotece networkx.
- W drugiej cześci stworzyliśmy funkcje power2 o parametrach A-graf skierowany oraz alpha - prawdopodobieństwo poruszania sie po macierzy H i stworzyliśmy macierz Googla. Obliczyliśmy wartości własne macierzy G dla dwóch wartości parametru alpha i zobrazowaliśmy je na płaszczyźnie zespolonej. Funkcja zwracała wartości i wektory własne.

Biblioteki

```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
```

Rysunek 1: Fragment skryptu dotyczacy bibliotek

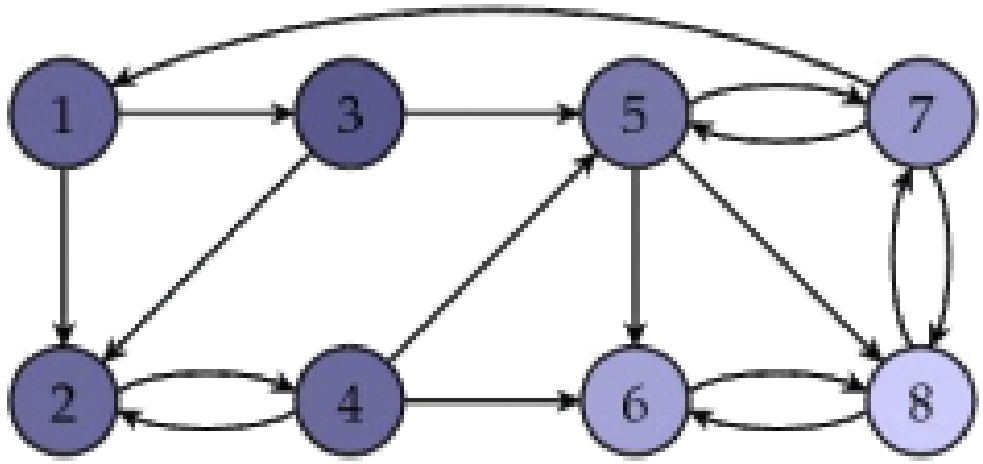
- Z całej biblioteki matplotlib używamy tylko pyplot.
- plt, nx i np to skróty bibliotek

Edge list

Nasze rozważania oparliśmy na strukturze danych używanych do reprezentowania wykresu jako listy jego krawedzi. Dane zostały obrane w sposób dowolny, na przykładzie sieci złożonej z 8 wezłów.

```
edgelist = [(1, 2), (1, 3), (2, 4), (3, 2), (3, 5), (4, 2),
            (4, 5), (4, 6), (5, 6), (5, 7), (6, 8), (7, 1),
            (7, 5), (7, 8), (8, 6), (8, 7)]
```

Rysunek 2: Edge list



Graf skierowany

- Zaczynamy od stworzenia macierzy sasiedztwa. Odwzorowuje ona połaczenia wierzchołków z krawedziami.
- Wiersze macierzy sasiedztwa odwzorowuja zawsze wierzchołki startowe krawedzi, a kolumny odwzorowuja wierzchołki końcowe krawedzi.

```
A = nx.DiGraph(edgelist)
```

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ m \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\
\begin{bmatrix}
a_{11} & a_{12} & \dots & a_{1n} \\
a_{21} & a_{22} & \dots & a_{2n} \\
a_{31} & a_{32} & \dots & a_{3n} \\
\vdots & \vdots & \vdots & \vdots \\
a_{m1} & a_{m2} & \dots & a_{mn}
\end{bmatrix}
\end{array}
$$

Graf skierowany

- W odniesieniu do otrzymanej macierzy sasiedztwa tworzymy graf stochastyczny B
- Przerabiamy graf stochastyczny na macierz H.
- Tworzymy wektor o długości n.

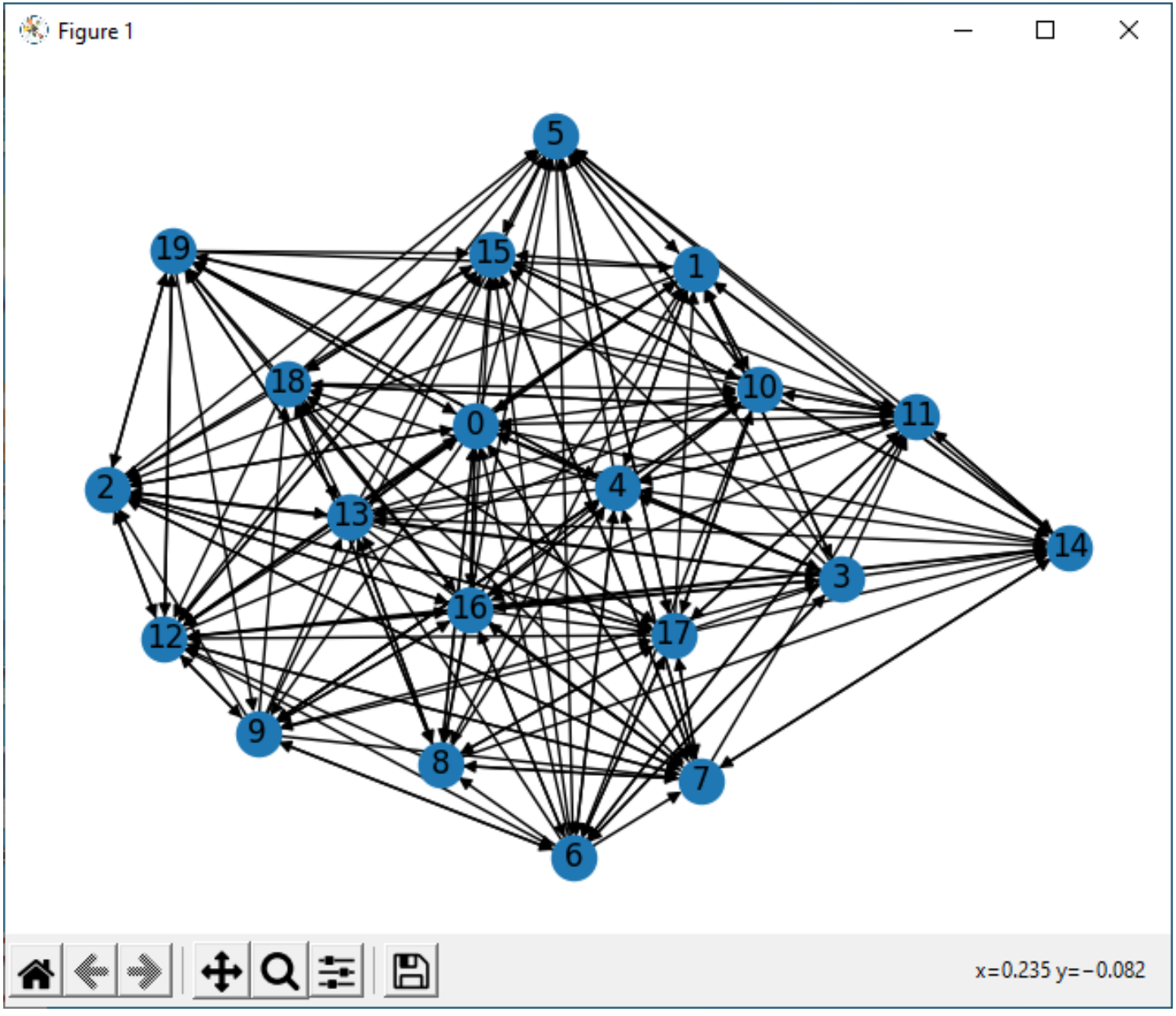
```
def power(A, B, debug=False):
    B = nx.stochastic_graph(A)
    H = nx.to_numpy_array(B).T
    n = H.shape[0]
    x = np.zeros(n)

    x[0] = 1
    for i in range(n):
        x = np.dot(H,x)
        if debug:
            print(x)
    return x
```

Rysunek 3: Funkcja power

Metoda nx.gnm-random-graph tworzy graf skierowany. W nawiasie zapisujemy ilość wezłów i krawedzi. Funkcja nx.draw rysuje graf A.

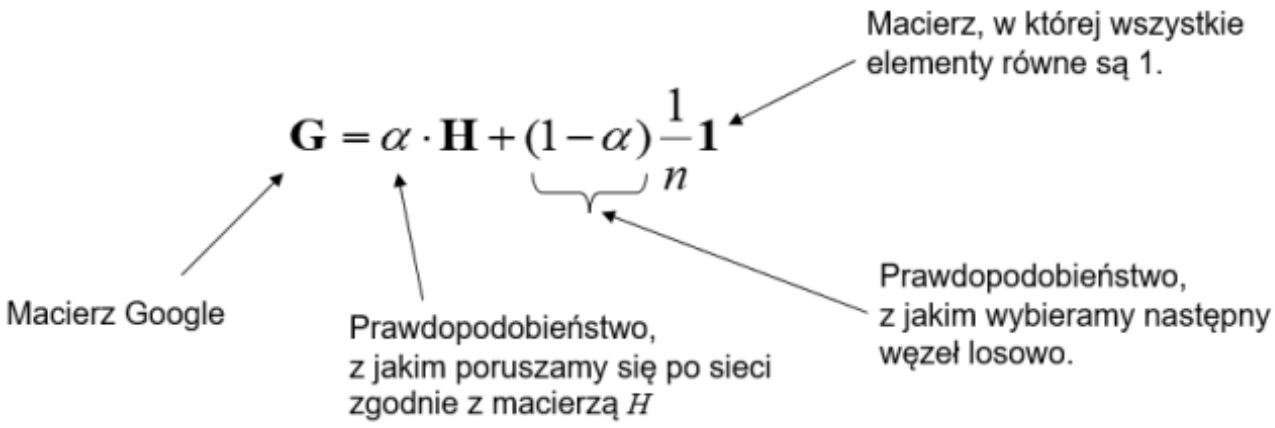
```
def cz1(N, E):
    A = nx.gnm_random_graph(N, E+10*N, directed=True)
    nx.draw(A, with_labels=True)
    plt.show()
```

Rysunek 4: Cześć kodu generujaca graf



Rysunek 5: Graf skierowany

Macierz Googla



Rysunek 6: Funkcja power2

Macierz Googla

```
def power2(A, alpha):
    B = nx.stochastic_graph(A)
    H = nx.to_numpy_matrix(B).T
    n = 8
    G = alpha*H+(1-alpha)*(1/n)*np.ones(n)
    wart, wekt = np.linalg.eig(G)
    print(H)
    return wart, wekt
```

Rysunek 7: Funkcja power2

Korzystamy z wartości A jako DiGraph'u, na podstawie naszej edge list.

```
def cz2():
    # dla alpha = 1
    wartosci, wektory = power2(A, 1)
    plt.scatter(wartosci.real, wartosci.imag)
    settings()
    plt.title("Wartosci własne dla alpha = 1")
    plt.xlabel("część rzeczywista")
    plt.ylabel("część urojona")
    plt.show()
    print("Jest spełniona dla alpha = 1")
```

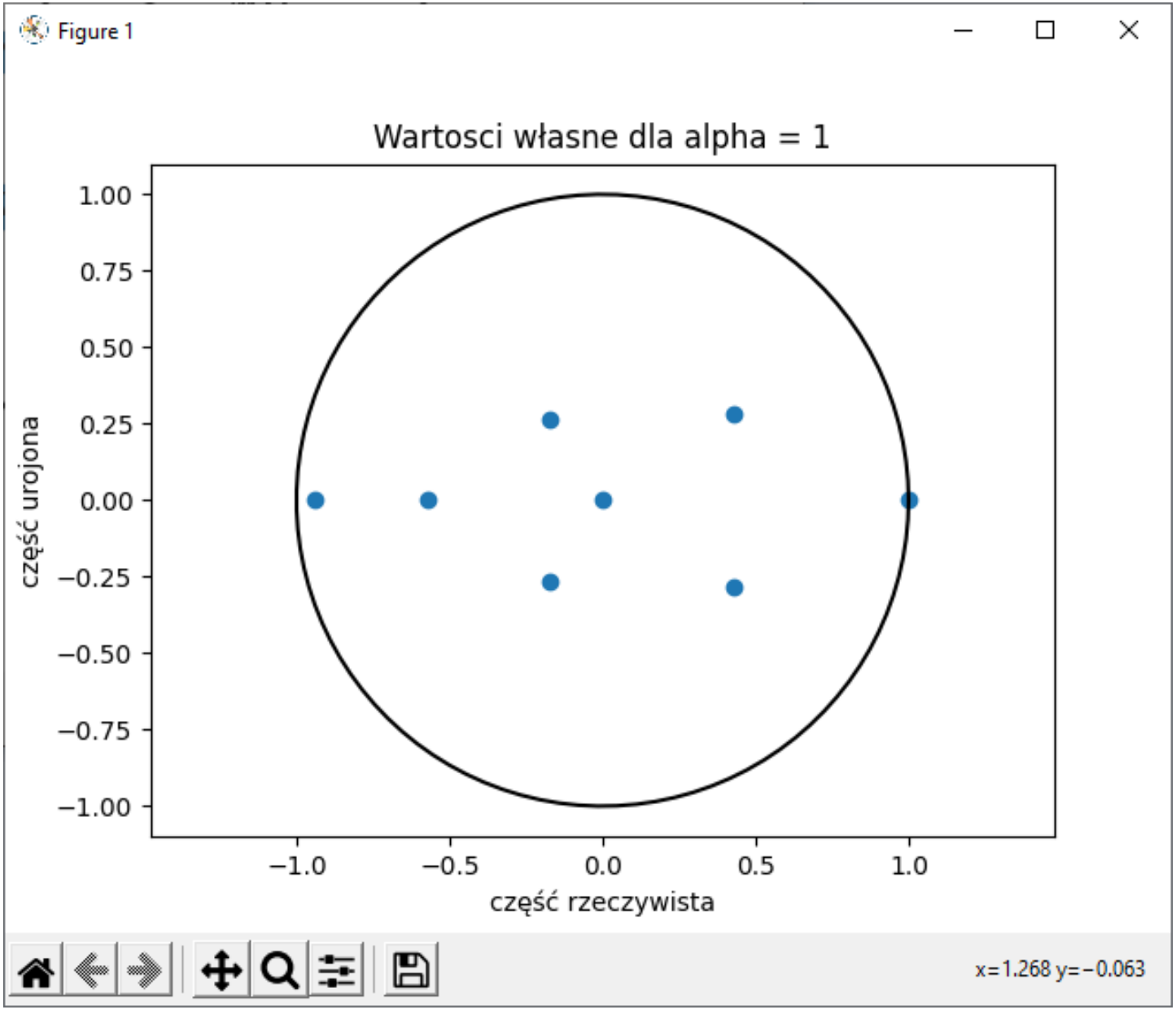


Figure 1
Wartosci własne dla alpha = 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
część urojona
−1.0
−0.5
0.0
0.5
1.0
część rzeczywista
x=1.268 y=−0.063

```
def cz2():
    # dla alpha = 0.6
    wartosci, wektory = power2(A, 0.6)
    plt.scatter(wartosci.real, wartosci.imag)
    settings()
    plt.title("Wartosci własne dla alpha = 0.6")
    plt.xlabel("część rzeczywista")
    plt.ylabel("część urojona")
    plt.show()
```

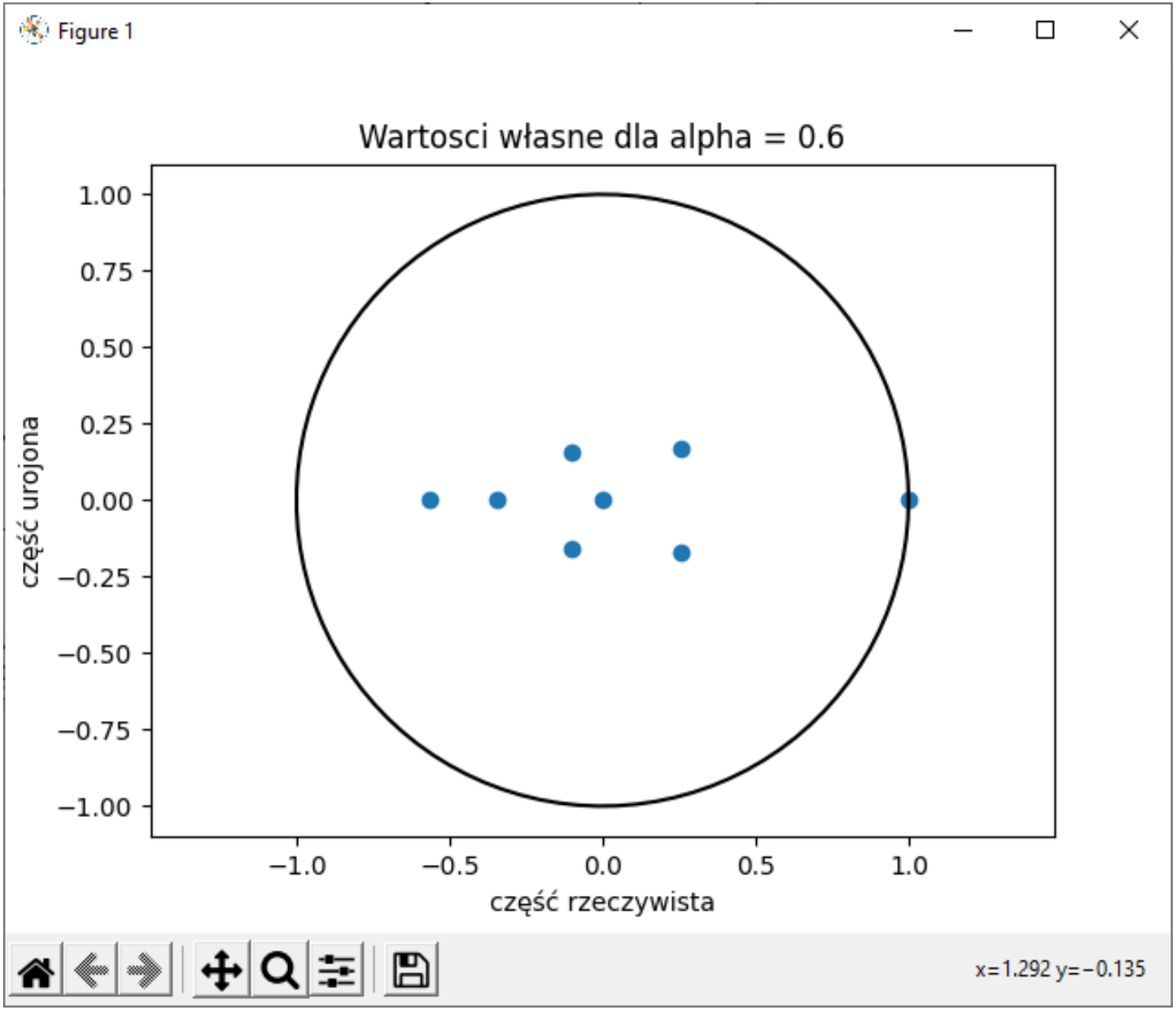


Figure 1
Wartosci własne dla alpha = 0.6
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
część urojona
−1.0
−0.5
0.0
0.5
1.0
część rzeczywista
x=1.292 y=−0.135

Bibliografia

https://pl.wikipedia.org/wiki/Macierz_stochastyczna
https://pl.xcv.wiki/wiki/Google_matrix
https://pl.wikipedia.org/wiki/Graf_(matematyka)
https://www.python.org/