Laboratorium
Metody Numeryczne

Instrukcja:
Na zajęciach należy wykonać poniższe zadania, uzupełnić plik main.py, wykonać test na platformie github, a następnie sporządzić sprawozdanie zawierające odpowiedzi z komentarzami.

Materiały przygotowujące:

In [1]:
```python
import numpy as np
import scipy
import matplotlib
import matplotlib.pyplot as plt
import scipy.linalg
from numpy.polynomial import polynomial as P
from numpy.core._multiarray_umath import ndarray
from numpy.polynomial import polynomial as P
import pickle

# zad1
def polly_A(x: np.ndarray):
    try:
        return P.polyfromroots(x)
    except Exception as e:
        print(f"Error in polly_A: {e}")
    return None

def roots_20(a: np.ndarray, num_iterations=20):
    try:
        roots_list = []

        for _ in range(num_iterations):
            perturbation = 1e-10 * np.random.normal(size=len(a))
            perturbed_a = a + perturbation
            roots = P.polyroots(perturbed_a)
            roots_list.append(roots)

        return a, np.array(roots_list)
    except Exception as e:
        print(f"Error in roots_20: {e}")
        return None


# zad 2


def frob_a(wsp: np.ndarray):
    try:
        # 1. Macierz Frobeniusa
        n = len(wsp) - 1
        frob_matrix = np.zeros((n, n))
        frob_matrix[0, 1:] = 1

        # 2. Wartości własne
        eigenvalues = np.linalg.eigvals(frob_matrix)

        # 3. Rozkład Schura
        schur_values, schur_vectors = scipy.linalg.schur(frob_matrix)

        # 4. Pierwiastki z polyroots
        roots = P.polyroots(wsp)

        return frob_matrix, eigenvalues, schur_values, roots
    except Exception as e:
        print(f"Error in frob_a: {e}")
        return None
```

Cel zajęć: Celem zajęć jest zapoznanie się z numerycznymi metodami rozwiązywania równań nieliniowych lub inaczej mówiąc metodami znajdowania miejsc zerowych funkcji. W związku z tym podczas zajęć będziemy rozważać następujący problem:

Dana jest funkcja $f(x)$, należy wyznaczyć argumenty funkcji x, dla których $f(x) = 0$ (funkcja jest równa zero).

Argumenty x^*, dla których $f(x^*)$=0$ nazywamy *pierwiastkami*.

Zadanie 1.
Dany jest wielomian postaci $W(x) = (x-1)(x-2)\cdot\ldots\cdot(x-20)$.

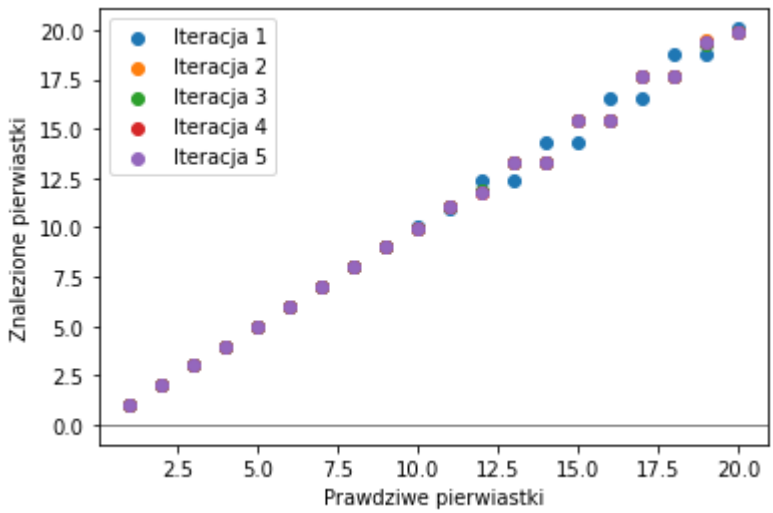
1. Zdefinuj funkcję *polly_A*, która obliczy współczynniki wielomianu a_i w postaci ogólnej wielomianu $w(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + 1$. Skonstruuj wektor tych współczynników. Użyj funkcji polyfromroots (https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.polynomial.polynomial.polyfromroots.html#numpy.polynomial.polyr oraz *linspace*.
2. Zdefinuj funkcję *roots_20*, która w pętli 20 iteracji będzie:

   i. konstruować wektor współczynników nowego wielomianu w następujący sposób: do każdego wygenerowanego wektora współczynników dodać losową wartość w postaci $(10^{-10})N(0,1)$. Użyj funkcji random_sample (https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.random_sample.html#numpy.random.random_sample).
   ii. wyliczyć pierwiaski tego wielomianu za pomocą metody polyroots (https://docs.scipy.org/doc/numpy/reference/generated/numpy.polynomial.polynomial.polyroots.html#n
3. Narysuj na wykresie te pierwiastki (w każdej iteracji dorysowywać pierwiastki na tym samym rysunku).
4. Określić, który pierwiastek jest najbardziej wrażliwy na zaburzenia.
5. Zaproponować sposób oszacowania uwarunkowania każdego z pierwiastków.

In [2]:
```
# Testowanie
true_roots = np.linspace(1, 20, 20)
coefficients = polly_A(true_roots)

for i in range(5):  # Testujemy na 5 iteracjach
    a, roots = roots_20(coefficients)
    plt.scatter(true_roots, np.real(roots[-1]), label=f'Iteracja {i+1}')

plt.axhline(0, color='black', linewidth=0.5)
plt.xlabel('Prawdziwe pierwiastki')
plt.ylabel('Znalezione pierwiastki')
plt.legend()
plt.show()
```



Zadanie 2.
Dany jest wielomian $w_1(x) = (x-1)^8$. Wyznacz numerycznie miejsca zerowego tego wielomianu poprzez wyznaczenie wartości własnych macierzy Frobeniusa. W związku z tym wykonaj następujące czynności:

1. Zaiplementuj funkcję tworzącą macierz Frobeniusa (https://github.com/KAIR-ISZ/public_lectures/blob/master/Metody%20Numeryczne%202019/Lecture%204%20(nonlinear%20ec%20R%C3%B3wnania%20nieliniowe.pdf) *frob_a*, dla zadanego wektora współczynników wielomianu $w(x)$.
2. Wyznacz wartości własne przekształconej macierzy za pomocą funkcji eigvals (https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eigvals.html#numpy.linalg.eigvals
3. Dokonaj rozkładu Schura macierzy zdefiniowanej w punkcie 1. użyj funkcji schure (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.schur.html#scipy.linalg.schur).
4. Porównaj wyniki z funkcją polyroots.

In [3]:
```
wsp_w1 = np.array([-1, 8, -28, 56, -70, 56, -28, 8, -1])  # Współczynniki wielomianu

frob_matrix, eigenvalues, schur_values, roots = frob_a(wsp_w1)

print("Macierz Frobeniusa:")
print(frob_matrix)
print("\nWartości własne:")
print(eigenvalues)
print("\nWartości z rozkładu Schura:")
print(schur_values)
print("\nPierwiastki z polyroots:")
print(roots)
```

```
Macierz Frobeniusa:
[[0. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]

Wartości własne:
[0. 0. 0. 0. 0. 0. 0. 0.]

Wartości z rozkładu Schura:
[[0. 1. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]

Pierwiastki z polyroots:
[0.98703727+0.j         0.99097217-0.00923541j 0.99097217+0.00923541j
 1.00023898-0.01276976j 1.00023898+0.01276976j 1.0090318 -0.00875353j
 1.0090318 +0.00875353j 1.01247683+0.j        ]
```

Zadanie 3.

Dla danego wielomianu $w_2(x) = 243x^7 - 486x^6 + 783x^5 - 990x^4 + 558x^3 - 28x^2 - 72x + 16$ wyznacz miejsca zerowe numerycznie, w taki sam sposób jak w zadaniu 2.

In [4]:
```
wsp_w2 = np.array([16, -72, -28, 558, -990, 783, -486, 243])  # Współczynniki wielomi

frob_matrix_w2, eigenvalues_w2, schur_values_w2, roots_w2 = frob_a(wsp_w2)

print("Macierz Frobeniusa:")
print(frob_matrix_w2)
print("\nWartości własne:")
print(eigenvalues_w2)
print("\nWartości z rozkładu Schura:")
print(schur_values_w2)
print("\nPierwiastki z polyroots:")
print(roots_w2)
```

```
Macierz Frobeniusa:
[[0. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]

Wartości własne:
[0. 0. 0. 0. 0. 0. 0.]

Wartości z rozkładu Schura:
[[0. 1. 1. 1. 1. 1. 1.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]]

Pierwiastki z polyroots:
[-3.33333333e-01+0.00000000e+00j  1.11022302e-15-1.41421356e+00j
  1.11022302e-15+1.41421356e+00j  3.33333333e-01+0.00000000e+00j
  6.66661120e-01+0.00000000e+00j  6.66669440e-01-4.80384808e-06j
  6.66669440e-01+4.80384808e-06j]
```

Zadanie 4.

Skonstruuj macierz diagonalną $\mathbf{A}_n$ (użyj do tego funkcji *diag* (https://numpy.org/doc/stable/reference/generated/numpy.diag.html)) której współczynniki $x_i = 2^i$ dla $i = \{1, 2, \ldots, n\}$ gdzie $n = \{10, 20, 30\}$.

1. Dla wszystkich macierzy $\mathbf{A}_n$ oblicz ich wartości własne przy użyciu eigvals (https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eigvals.html#numpy.linalg.eigvals i porównaj je z analitycznymi wartościami własnymi.
2. Zdefiniuj funcję *main.is_nonsingular*. Przy pomocy tej funkcji skonstruuj losowe macierze wektorów własnych $\mathbf{P}_n$ których wartości współczynników będą liczbami całkowitymi z zakresu (0,100). Na podstawie macierzy $\mathbf{A}_n$ i $\mathbf{P}_n$ oblicz macierze:
$$\mathbf{B}_n = \mathbf{P}_n\mathbf{A}_n\mathbf{P}_n^{-1}$$
Oblicz wartości własne dla uzyskanych macierzy i porównaj je z analitycznymi wartościami własnymi.
3. Bazując na macierzach $\mathbf{P}_n$ wygeneruj macierze ortonormalne $\mathbf{Q}_n$ (można do tego użyć rozkładu QR). Na podstawie macierzy $\mathbf{Q}_n$ oblicz macierze $\mathbf{C}_n$ w sposób analogiczny do macierze $\mathbf{B}_n$ używając macierzy $\mathbf{Q}_n$. Oblicz wartości własne dla uzyskanych macierzy i porównaj je z analitycznymi wartościami własnymi.
4. Bazując na analitycznych wartościach własnych dla wielomianów charakterystycznych macierzy $\mathbf{A}_n$ wygeneruj macierze Frobeniusa (https://github.com/KAIR-

ISZ/public_lectures/blob/master/Metody%20Numeryczne%202019/Lecture%204%20(nonlinear%20eq%20R%C3%B3wnania%20nieliniowe.pdf). Oblicz wartości własne dla uzyskanych macierzy i porównaj je z analitycznymi wartościami własnymi.

5. Porównaj otrzymane wyniki ze wszystkich punktów i wartości n

In [5]:

```
from scipy.linalg import eigvals, inv, qr

def generuj_analityczne_wartosci_wlasne(n):
    return np.array([2**i for i in range(1, n+1)])

def konstruuj_macierz_A(n):
    return np.diag([2**i for i in range(1, n+1)])

def porownaj_wartosci_wlasne_analityczne(n):
    macierz_A = konstruuj_macierz_A(n)
    analityczne_wartosci_wlasne = generuj_analityczne_wartosci_wlasne(n)
    numeryczne_wartosci_wlasne = eigvals(macierz_A)

    print(f"Porównanie dla A_{n}:")
    print("Analityczne Wartości Własne:", analityczne_wartosci_wlasne)
    print("Numeryczne Wartości Własne:", numeryczne_wartosci_wlasne)
    print("Zgodność:", np.allclose(analityczne_wartosci_wlasne, numeryczne_wartosci_w
    print()

def czy_singlowa(macierz):
    try:
        np.linalg.inv(macierz)
        return True
    except np.linalg.LinAlgError:
        return False

def konstruuj_macierz_B(P, A):
    P_odw = inv(P)
    return P @ A @ P_odw

def porownaj_wartosci_wlasne_B_analityczne(n):
    macierz_A = konstruuj_macierz_A(n)
    P = np.random.randint(0, 100, size=(n, n))
    macierz_B = konstruuj_macierz_B(P, macierz_A)

    analityczne_wartosci_wlasne = generuj_analityczne_wartosci_wlasne(n)
    numeryczne_wartosci_wlasne = eigvals(macierz_B)

    print(f"Porównanie dla B_{n}:")
    print("Analityczne Wartości Własne:", analityczne_wartosci_wlasne)
    print("Numeryczne Wartości Własne:", numeryczne_wartosci_wlasne)
    print("Zgodność:", np.allclose(analityczne_wartosci_wlasne, numeryczne_wartosci_w
    print()

def generuj_ortonormalna_macierz_Q(n):
    macierz_A = konstruuj_macierz_A(n)
    Q, _ = qr(macierz_A)
    return Q

def konstruuj_macierz_C(Q, A):
    Q_odw = inv(Q)
    return Q @ A @ Q_odw

def porownaj_wartosci_wlasne_C_analityczne(n):
    macierz_A = konstruuj_macierz_A(n)
    Q = generuj_ortonormalna_macierz_Q(n)
    macierz_C = konstruuj_macierz_C(Q, macierz_A)

    analityczne_wartosci_wlasne = generuj_analityczne_wartosci_wlasne(n)
    numeryczne_wartosci_wlasne = eigvals(macierz_C)

    print(f"Porównanie dla C_{n}:")
    print("Analityczne Wartości Własne:", analityczne_wartosci_wlasne)
    print("Numeryczne Wartości Własne:", numeryczne_wartosci_wlasne)
    print("Zgodność:", np.allclose(analityczne_wartosci_wlasne, numeryczne_wartosci_w
```

```
    print()

def konstruuj_macierz_Frobeniusa(n):
    analityczne_wartosci_wlasne = generuj_analityczne_wartosci_wlasne(n)
    macierz_Frobeniusa = np.diag(analityczne_wartosci_wlasne[:-1], 1)
    return macierz_Frobeniusa

def porownaj_wartosci_wlasne_Frobeniusa(n):
    macierz_Frobeniusa = konstruuj_macierz_Frobeniusa(n)
    analityczne_wartosci_wlasne = generuj_analityczne_wartosci_wlasne(n)
    numeryczne_wartosci_wlasne = eigvals(macierz_Frobeniusa)

    print(f"Porównanie dla Frobenius_{n}:")
    print("Analityczne Wartości Własne:", analityczne_wartosci_wlasne)
    print("Numeryczne Wartości Własne:", numeryczne_wartosci_wlasne)
    print("Zgodność:", np.allclose(analityczne_wartosci_wlasne, numeryczne_wartosci_w
```

In [6]:
```
# Wywołania funkcji
ns = [10, 20, 30]
```

In [7]:
```
for n in ns:
    porownaj_wartosci_wlasne_analityczne(n)
```

```
Porównanie dla A_10:
Analityczne Wartości Własne: [   2    4    8   16   32   64  128  256  512 1024]
Numeryczne Wartości Własne: [   2.+0.j    4.+0.j    8.+0.j   16.+0.j   32.+0.j   64.
+0.j  128.+0.j
  256.+0.j  512.+0.j 1024.+0.j]
Zgodność: True

Porównanie dla A_20:
Analityczne Wartości Własne: [      2       4       8      16      32      64     12
8     256     512
    1024    2048    4096    8192   16384   32768   65536  131072  262144
  524288 1048576]
Numeryczne Wartości Własne: [2.000000e+00+0.j 4.000000e+00+0.j 8.000000e+00+0.j 1.60
0000e+01+0.j
 3.200000e+01+0.j 6.400000e+01+0.j 1.280000e+02+0.j 2.560000e+02+0.j
 5.120000e+02+0.j 1.024000e+03+0.j 2.048000e+03+0.j 4.096000e+03+0.j
 8.192000e+03+0.j 1.638400e+04+0.j 3.276800e+04+0.j 6.553600e+04+0.j
 1.310720e+05+0.j 2.621440e+05+0.j 5.242880e+05+0.j 1.048576e+06+0.j]
Zgodność: True

Porównanie dla A_30:
Analityczne Wartości Własne: [         2          4          8         16         32
64
        128        256        512       1024       2048       4096
       8192      16384      32768      65536     131072     262144
     524288    1048576    2097152    4194304    8388608   16777216
   33554432   67108864  134217728  268435456  536870912 1073741824]
Numeryczne Wartości Własne: [2.00000000e+00+0.j 4.00000000e+00+0.j 8.00000000e+00+0.
j
 1.60000000e+01+0.j 3.20000000e+01+0.j 6.40000000e+01+0.j
 1.28000000e+02+0.j 2.56000000e+02+0.j 5.12000000e+02+0.j
 1.02400000e+03+0.j 2.04800000e+03+0.j 4.09600000e+03+0.j
 8.19200000e+03+0.j 1.63840000e+04+0.j 3.27680000e+04+0.j
 6.55360000e+04+0.j 1.31072000e+05+0.j 2.62144000e+05+0.j
 5.24288000e+05+0.j 1.04857600e+06+0.j 2.09715200e+06+0.j
 4.19430400e+06+0.j 8.38860800e+06+0.j 1.67772160e+07+0.j
 3.35544320e+07+0.j 6.71088640e+07+0.j 1.34217728e+08+0.j
 2.68435456e+08+0.j 5.36870912e+08+0.j 1.07374182e+09+0.j]
Zgodność: True
```

In [8]:
```
for n in ns:
    porownaj_wartosci_wlasne_B_analityczne(n)
```

```
Porównanie dla B_10:
Analityczne Wartości Własne: [   2    4    8   16   32   64  128  256  512 1024]
Numeryczne Wartości Własne: [1024.+0.j  512.+0.j  256.+0.j  128.+0.j   64.+0.j   32.
+0.j   16.+0.j
    8.+0.j    4.+0.j    2.+0.j]
Zgodność: False

Porównanie dla B_20:
Analityczne Wartości Własne: [      2       4       8      16      32      64     12
8     256     512
    1024    2048    4096    8192   16384   32768   65536  131072  262144
  524288 1048576]
Numeryczne Wartości Własne: [1.04857600e+06+0.j 5.24288000e+05+0.j 2.62144000e+05+0.
j
 1.31072000e+05+0.j 6.55360000e+04+0.j 3.27680000e+04+0.j
 1.63840000e+04+0.j 8.19200000e+03+0.j 4.09600000e+03+0.j
 2.04800000e+03+0.j 1.02400000e+03+0.j 5.12000000e+02+0.j
 2.56000000e+02+0.j 1.28000000e+02+0.j 6.39999999e+01+0.j
 3.20000000e+01+0.j 1.60000000e+01+0.j 8.00000002e+00+0.j
 4.00000004e+00+0.j 1.99999997e+00+0.j]
Zgodność: False

Porównanie dla B_30:
Analityczne Wartości Własne: [         2          4          8         16         32
64
        128        256        512       1024       2048       4096
       8192      16384      32768      65536     131072     262144
     524288    1048576    2097152    4194304    8388608   16777216
   33554432   67108864  134217728  268435456  536870912 1073741824]
Numeryczne Wartości Własne: [-1.06901485e+08       +0.j         -2.02759284e+07+2601
6156.88535729j
 -2.02759284e+07-26016156.88535729j -1.79844719e+06+23435666.42747455j
 -1.79844719e+06-23435666.42747455j  1.67772160e+07       +0.j
 -1.17247976e+07       +0.j          8.38860800e+06       +0.j
  4.19430400e+06       +0.j          2.09715200e+06       +0.j
  1.04857600e+06       +0.j          5.24288000e+05       +0.j
  2.62144000e+05       +0.j          1.31072000e+05       +0.j
  6.55360000e+04       +0.j          3.27680000e+04       +0.j
  1.63840000e+04       +0.j          8.19200000e+03       +0.j
  4.09600000e+03       +0.j          2.04800000e+03       +0.j
  1.02400000e+03       +0.j          5.12000000e+02       +0.j
  2.56000000e+02       +0.j          1.28000000e+02       +0.j
  6.40000000e+01       +0.j          3.20000001e+01       +0.j
  1.60000001e+01       +0.j          7.99999992e+00       +0.j
  1.99999997e+00       +0.j          4.00000004e+00       +0.j        ]
Zgodność: False
```

In [9]:
```
for n in ns:
    porownaj_wartosci_wlasne_C_analityczne(n)
```

```
Porównanie dla C_10:
Analityczne Wartości Własne: [   2    4    8   16   32   64  128  256  512 1024]
Numeryczne Wartości Własne: [   2.+0.j    4.+0.j    8.+0.j   16.+0.j   32.+0.j   64.
+0.j  128.+0.j
  256.+0.j  512.+0.j 1024.+0.j]
Zgodność: True

Porównanie dla C_20:
Analityczne Wartości Własne: [      2       4       8      16      32      64     12
8     256     512
    1024    2048    4096    8192   16384   32768   65536  131072  262144
  524288 1048576]
Numeryczne Wartości Własne: [2.000000e+00+0.j 4.000000e+00+0.j 8.000000e+00+0.j 1.60
0000e+01+0.j
 3.200000e+01+0.j 6.400000e+01+0.j 1.280000e+02+0.j 2.560000e+02+0.j
 5.120000e+02+0.j 1.024000e+03+0.j 2.048000e+03+0.j 4.096000e+03+0.j
 8.192000e+03+0.j 1.638400e+04+0.j 3.276800e+04+0.j 6.553600e+04+0.j
 1.310720e+05+0.j 2.621440e+05+0.j 5.242880e+05+0.j 1.048576e+06+0.j]
Zgodność: True

Porównanie dla C_30:
Analityczne Wartości Własne: [         2          4          8         16         32
64
        128        256        512       1024       2048       4096
       8192      16384      32768      65536     131072     262144
     524288    1048576    2097152    4194304    8388608   16777216
   33554432   67108864  134217728  268435456  536870912 1073741824]
Numeryczne Wartości Własne: [2.00000000e+00+0.j 4.00000000e+00+0.j 8.00000000e+00+0.
j
 1.60000000e+01+0.j 3.20000000e+01+0.j 6.40000000e+01+0.j
 1.28000000e+02+0.j 2.56000000e+02+0.j 5.12000000e+02+0.j
 1.02400000e+03+0.j 2.04800000e+03+0.j 4.09600000e+03+0.j
 8.19200000e+03+0.j 1.63840000e+04+0.j 3.27680000e+04+0.j
 6.55360000e+04+0.j 1.31072000e+05+0.j 2.62144000e+05+0.j
 5.24288000e+05+0.j 1.04857600e+06+0.j 2.09715200e+06+0.j
 4.19430400e+06+0.j 8.38860800e+06+0.j 1.67772160e+07+0.j
 3.35544320e+07+0.j 6.71088640e+07+0.j 1.34217728e+08+0.j
 2.68435456e+08+0.j 5.36870912e+08+0.j 1.07374182e+09+0.j]
Zgodność: True
```

In [10]:
```
for n in ns:
    porownaj_wartosci_wlasne_Frobeniusa(n)
```

```
Porównanie dla Frobenius_10:
Analityczne Wartości Własne: [   2    4    8   16   32   64  128  256  512 1024]
Numeryczne Wartości Własne: [0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j
0.+0.j 0.+0.j]
Zgodność: False
Porównanie dla Frobenius_20:
Analityczne Wartości Własne: [      2       4       8      16      32      64     12
8     256     512
    1024    2048    4096    8192   16384   32768   65536  131072  262144
  524288 1048576]
Numeryczne Wartości Własne: [0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j
0.+0.j 0.+0.j
 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j]
Zgodność: False
Porównanie dla Frobenius_30:
Analityczne Wartości Własne: [         2          4          8         16         32
64
        128        256        512       1024       2048       4096
       8192      16384      32768      65536     131072     262144
     524288    1048576    2097152    4194304    8388608   16777216
   33554432   67108864  134217728  268435456  536870912 1073741824]
Numeryczne Wartości Własne: [0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j
0.+0.j 0.+0.j
 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j
 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j]
Zgodność: False
```

Type *Markdown* and LaTeX: α^2