Zadanie 3:

3. Dana jest macierz

$$\mathbf{A} = \begin{bmatrix} \frac{19}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & -\frac{17}{12} \\ \frac{13}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & -\frac{11}{12} & \frac{13}{12} \\ \frac{5}{6} & \frac{5}{6} & \frac{5}{6} & -\frac{1}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{5}{6} & \frac{5}{6} & -\frac{1}{6} & \frac{5}{6} & \frac{5}{6} & \frac{5}{6} \\ \frac{13}{12} & -\frac{11}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & \frac{13}{12} \\ -\frac{17}{12} & \frac{13}{12} & \frac{5}{6} & \frac{5}{6} & \frac{13}{12} & \frac{19}{12} \end{bmatrix}. \tag{4}$$

Przy użyciu metody potęgowej znajdź jej dwie największe na moduł wartości własne i odpowiadające im wektory własne.

Kod w języku Python:

```
import numpy

def findFirst(arr):
    #pierwsze y1, potem y2,3, itd.
    prev_y=numpy.array([1,0,0,0,0,0])

    while True:
        zk=product(arr,prev_y)
        norm=calc_norm(zk)
        next_y=[zk[i]/norm for i in range(0,len(zk))]

        #warunek zakonczenia
        if abs(abs(prev_y[0])-abs(next_y[0])) < 1e-8:
            print("Znaleziony wektor własny: ", next_y)
            print("Znaleziona wartość własna:", norm)
            return next_y
        prev_y=next_y

def findSecond(arr,e1):
    #pierwsze y1, potem y2,3, itd.
    prev_y=numpy.array([1,0,0,0,0,0])

    while True:
        zk=product(arr,prev_y)
        #ortogonalizacja
        zk=ortho(zk,e1)
        norm=calc_norm(zk)
        next_y=[zk[i]/norm for i in range(0,len(zk))]

        #warunek zakonczenia
        if abs(abs(prev_y[0])-abs(next_y[0])) < 1e-13:
            print("Znaleziony wektor własny: ", next_y)
```

```
            print("Znaleziona wartość własna:", norm)
            return next_y
        prev_y=next_y


#mnozy macierz a razy wektor x i zwraca wektor wynikowy y
def product(a,x):
    y=[]
    for i in range(0,6):
        summ=0
        for j in range(0,6):
            summ=summ+a[i][j]*x[j]
        y.append(summ)
    return y


#liczy norme wektora
def calc_norm(y):
    norm=0
    for i in y:
        norm=norm+pow(i,2)
    return numpy.sqrt(norm)


#dokonuje ortogonalizacji wektora zk
def ortho(zk,e):
    prod=0
    for i in range(0,len(zk)):
        prod=prod+zk[i]*e[i]
    for i in range(0,len(zk)):
        zk[i]=zk[i]-e[i]*prod
    return zk

if __name__=="__main__":
    print("Szukanie dwoch najwiekszych na modul wartosci wlasnych macierzy")

    arr=numpy.array([
            [19/12,13/12,5/6,5/6,13/12,-17/12],
            [13/12,13/12,5/6,5/6,-11/12,13/12],
            [5/6,5/6,5/6,-1/6,5/6,5/6],
            [5/6,5/6,-1/6,5/6,5/6,5/6],
            [13/12,-11/12,5/6,5/6,13/12,13/12],
            [-17/12,13/12,5/6,5/6,13/12,19/12]
            ])

    #szuka pierwszej wartosci wlasnej
    e1=findFirst(arr)
    print()
    #szuka drugiej wartosci wlasnej
    e2=findSecond(arr,e1)
    print()

    print("Sprawdzenie wyników z wartosciami wlasnymi znalezionymi przez funkcję z
biblioteki numpy")
    eig=numpy.linalg.eig(arr)
    print("Znalezione wartosci wlasne z funkcji numpy.linalg.eig(): ", eig[0])
```

Wynik działania programu:

```
Szukanie dwoch najwiekszych na modul wartosci wlasnych macierzy
Znaleziony wektor własny:  [0.40824831975813375, 0.40824829046386274, 0.40824829046386274, 0.40824829046386274, 0.40824829046386263, 0.4082482611695916]
Znaleziona wartość własna: 3.9999999999999973

Znaleziony wektor własny:  [0.7071067642735245, -1.6913088194415014e-08, -1.691305529480605e-08, -1.691305529480605e-08, -1.6913088194415014e-08, -0.7071067980995694]
Znaleziona wartość własna: 3.0000000000000013

Sprawdzenie wyników z wartosciami wlasnymi znalezionymi przez funkcję z biblioteki numpy
Znalezione wartosci wlasne z funkcji numpy.linalg.eig():  [ 4.  3. -2. -1.  1.  2.]
```

e_1=[0.40824831975813375,
0.40824829046386274,
0.40824829046386274,
0.40824829046386274,
0.40824829046386263,
0.4082482611695916]
$\lambda_1 = 4$

e_2=[0.7071067642735245,
-1.6913088194415014e-08,
-1.691305529480605e-08,
-1.691305529480605e-08,
-1.6913088194415014e-08,
-0.7071067980995694]
$\lambda_2=3$

Omówienie wyników:

Metoda potęgowa obliczania wektorów i wartości własnych jest metodą iteracyjną. Aby obliczyć pierwszą wartość własną (o największym module) korzystamy z poniższego schematu:

Najpierw wybierz wektor y_1 taki, że jego norma wynosi 1. Następnie:

$$Ay_k = z_k$$

$$y_{k+1} = \frac{z_k}{||z_k||}$$

Kończymy iterować gdy kolejne y różnią się o mniej niż zadana tolerancja. Nasz wynikowy wektor własny to y, a wartość własna to $||z_k||$

Aby obliczyć drugą i kolejne wartości własne musimy dokonać ortogonalizacji z (przynajmniej co kilka kroków)

$$Ay_k = z_k$$

$$z_k = z_k - \sum_{i=1}^{n-1} e_i(e_i^T z_k)$$

Co dla drugiej największej co do modułu wartości własnej (n=2) będzie wyglądać jak poniżej:

$$z_k = z_k - e1(e_1^T z_k)$$

Następnie wyliczamy kolejne y jak poprzednio:

$$y_{k+1} = \frac{z_k}{||z_k||}$$

Metoda potęgowa daje szybkie i dokładne wyniki dla pierwszych z wartości własnych. Szybkość zbieżności metody potęgowej jest liniowa. W naszym konkretnym przypadku mamy do obliczenia tylko dwie wartości własne, ale dla kolejnych reortogonalizacja staje się kosztowna (k-1)*n operacji dla każdej iteracji (gdzie k-numer wartości własnej którą chcemy obliczyć, n-rozmiar wektora) co dla ostatniej wartości własnej dla każdej iteracji spowodowałoby 5*n dodatkowych operacji.
Metoda potęgowa działa tylko do macierzy symetrycznych.