1 Metoda potęgowa

Napisz funkcję obliczającą metodą potęgową dominującą wartość własną (największą co do modułu) i odpowiadający jej wektor własny dla danej macierzy rzeczywistej symetrycznej. Sprawdź poprawność działania programu porównując własną implementację z wynikami funkcji bibliotecznej. Przedstaw na wykresie zależność czasu obliczeń od rozmiaru macierzy (rozmiary macierzy 100x100, 500x500, ...).

Metoda potęgowa

```
ef eigenvalue(A, v):
    Av = A.dot(v)
    return v.dot(Av)

def power_iteration(A, max_iteration, eps):
    n, d = A.shape

    v = np.ones(d) / np.sqrt(d)
    ev = eigenvalue(A, v)

    for i in range(max_iteration):
        Av = A.dot(v)
        v_new = Av / np.linalg.norm(Av)

        ev_new = eigenvalue(A, v_new)
        if np.abs(ev - ev_new) < eps:
            break

        v = v_new
        ev = ev_new
```

Test poprawności metody potęgowej

```
def check_power_iteration():
    np.set_printoptions(suppress=True)
    j = 0
    for i in range(4, 10):
        j += 1
        print("TEST NR {0}".format(j))
        arr = generate_matrix(i)
        copy_arr = np.copy(arr)
        e_val, e_vec = power_iteration(arr, 10000, 0.01)
        print("my power iteration")
        print(e_val)
        print(e_vec)
        e_val, e_vec = np.linalg.eig(copy_arr)
        print("library power iteration")
```

```
        print(e_val)
        print(e_vec)
```

Wyniki testu

```
/home/pawel/Desktop/Git/Mownit/venv5/bin/python /home/pawel/Desktop/Git/Mownit/lab7/lab7.py
TEST NR 1
my power iteration
2.3555950355507784
[0.46492757 0.58433482 0.29709365 0.59508868]
library power iteration
[ 2.35617638  0.44327354 -0.02520221  0.01104449]
[[-0.45789106 -0.81786286 -0.59685589  0.5791018 ]
 [-0.58535903  0.09619395  0.40882811 -0.23818015]
 [-0.30173573  0.51989411  0.60519248 -0.72694788]
 [-0.59719857  0.22706206 -0.33221182  0.28188313]]
TEST NR 2
my power iteration
2.4764547651856015
[0.37899588 0.44500583 0.45055301 0.39318594 0.54839651]
library power iteration
[ 2.47699283+0.j         -0.43712863+0.j          0.33322165+0.j
 -0.09293182+0.12283141j -0.09293182-0.12283141j]
[[ 0.3791919 +0.j          0.16606097+0.j          0.42120657+0.j
  -0.06323675-0.02518301j -0.06323675+0.02518301j]
 [ 0.44433522+0.j         -0.46573167+0.j          0.45090886+0.j
   0.3245928 -0.03743279j  0.3245928 +0.03743279j]
 [ 0.45166587+0.j          0.65083769+0.j         -0.50790692+0.j
  -0.60878633+0.14008626j -0.60878633-0.14008626j]
 [ 0.39238182+0.j         -0.56410563+0.j          0.31779441+0.j
   0.69364972+0.j          0.69364972-0.j        ]
 [ 0.54846529+0.j         -0.11710214+0.j         -0.51019947+0.j
  -0.04042139-0.12480569j -0.04042139+0.12480569j]]
```

```
TEST NR 3
my power iteration
3.1610114289494073
[0.46415555 0.41056299 0.32920423 0.4685199  0.30535586 0.44143985]
library power iteration
[ 3.16651633+0.j          0.30681025+0.77356812j  0.30681025-0.77356812j
 -0.53151604+0.j          0.3186342 +0.j         -0.11161213+0.j        ]
[[ 0.48319571+0.j         -0.06501184+0.4054583j  -0.06501184-0.4054583j
   0.1475632 +0.j         -0.11184696+0.j         -0.32734606+0.j        ]
 [ 0.41158203+0.j          0.01971352-0.13972194j  0.01971352+0.13972194j
  -0.14698129+0.j         -0.6436335 +0.j         -0.62448147+0.j        ]
 [ 0.35159517+0.j          0.58094204+0.j          0.58094204-0.j
   0.34800612+0.j         -0.07714919+0.j          0.29750844+0.j        ]
 [ 0.45139965+0.j         -0.42658768-0.09114496j -0.42658768+0.09114496j
   0.48081964+0.j         -0.00666049+0.j          0.64223435+0.j        ]
 [ 0.27177826+0.j         -0.18249594-0.47942597j -0.18249594+0.47942597j
   0.03693789+0.j          0.74308587+0.j         -0.0431116 +0.j        ]
 [ 0.44258096+0.j         -0.05906439+0.13055824j -0.05906439-0.13055824j
  -0.7765059 +0.j          0.12269149+0.j          0.00570933+0.j        ]]
```

```
TEST NR 4
my power iteration
3.876538613881307
[0.17634243 0.30287284 0.45610419 0.50153448 0.29464997 0.30025087
 0.49054494]
library power iteration
[ 3.87407109+0.j         -0.12068354+0.47170485j -0.12068354-0.47170485j
 -0.36819329+0.06869145j -0.36819329-0.06869145j  0.75767196+0.j
  0.46771772+0.j        ]
[[-0.170693  +0.j          -0.05121927-0.21919491j -0.05121927+0.21919491j
   0.1525842 -0.12504562j  0.1525842 +0.12504562j -0.33327782+0.j
   0.15698995+0.j        ]
 [-0.3022533 +0.j          0.09267689+0.33718465j  0.09267689-0.33718465j
  -0.42503887+0.01602538j -0.42503887-0.01602538j -0.22726798+0.j
  -0.32000195+0.j        ]
 [-0.46111521+0.j         -0.63924449+0.j         -0.63924449-0.j
  -0.06148994+0.02272342j -0.06148994-0.02272342j  0.43417323+0.j
  -0.01479038+0.j        ]
 [-0.5013719 +0.j          0.52362313-0.07000303j  0.52362313+0.07000303j
   0.37689751-0.14077484j  0.37689751+0.14077484j  0.03890462+0.j
  -0.44240743+0.j        ]
 [-0.29659051+0.j          0.19638152+0.04864883j  0.19638152-0.04864883j
  -0.00991884-0.03321684j -0.00991884+0.03321684j  0.50053899+0.j
   0.74968654+0.j        ]
 [-0.29293335+0.j         -0.08153649+0.22861148j -0.08153649-0.22861148j
   0.57066365+0.j          0.57066365-0.j         -0.60591045+0.j
   0.10588499+0.j        ]
 [-0.4916604 +0.j         -0.02308096-0.19737279j -0.02308096+0.19737279j
  -0.48884492+0.21948132j -0.48884492-0.21948132j -0.17201291+0.j
  -0.32213084+0.j        ]]
```

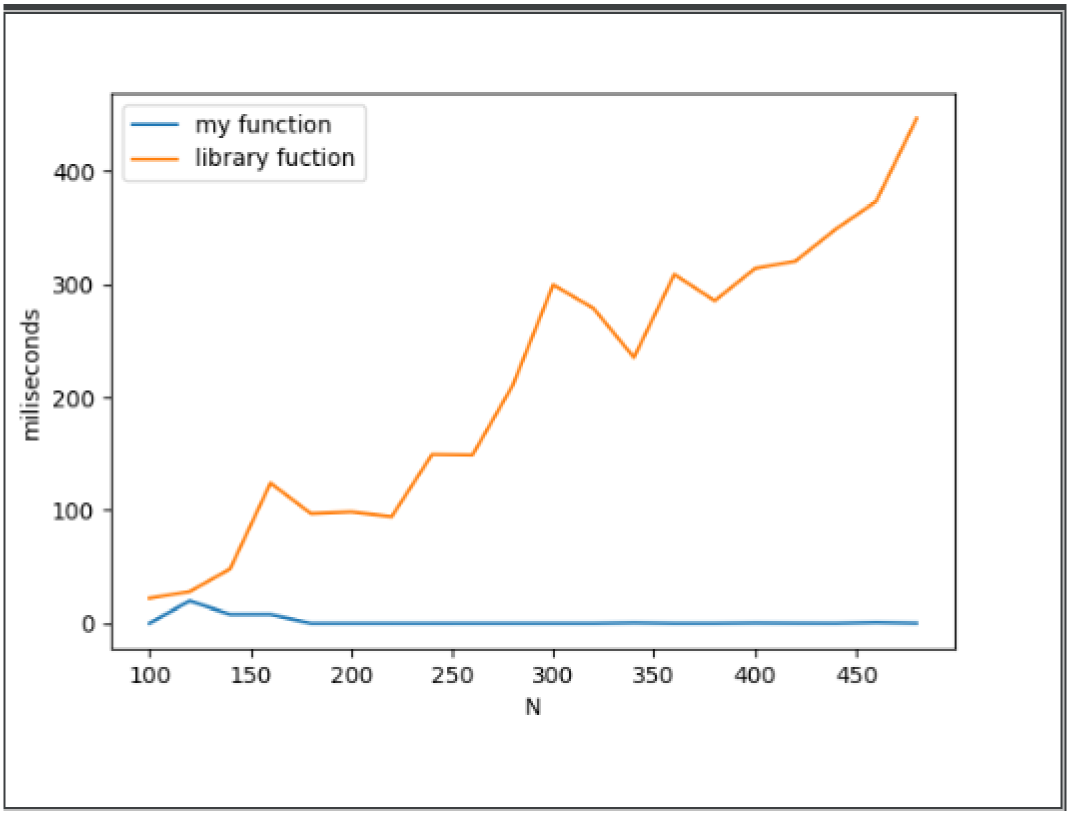
```
TEST NR 5
my power iteration
3.5140567624506014
[0.15545049 0.25756025 0.26309358 0.28111055 0.50330033 0.43382848
 0.42065125 0.37787649]
library power iteration
[ 3.51411409+0.j          -0.8101885 +0.j           0.24997824+0.35045082j
  0.24997824-0.35045082j -0.297245  +0.48230665j -0.297245  -0.48230665j
 -0.41998753+0.j           0.03989756+0.j        ]
[[-0.15674224+0.j          0.41632366+0.j          0.62656371+0.j
   0.62656371-0.j         -0.18324108-0.0451794j  -0.18324108+0.0451794j
  -0.38415822+0.j         -0.62097717+0.j        ]
 [-0.25645034+0.j         -0.28431106+0.j          0.06207453-0.13244923j
   0.06207453+0.13244923j  0.54026844+0.j          0.54026844-0.j
   0.01517325+0.j         -0.39930396+0.j        ]
 [-0.26240006+0.j         -0.25690799+0.j         -0.43967144+0.17870113j
  -0.43967144-0.17870113j -0.14313092+0.38426911j -0.14313092-0.38426911j
   0.51597996+0.j          0.2952666 +0.j        ]
 [-0.27987526+0.j         -0.47609551+0.j          0.13782796+0.22828916j
   0.13782796-0.22828916j  0.20678194-0.14434752j  0.20678194+0.14434752j
   0.35862923+0.j         -0.0442263 +0.j        ]
 [-0.50264767+0.j         -0.21205345+0.j         -0.01353381+0.19427772j
  -0.01353381-0.19427772j -0.12222839-0.04173932j -0.12222839+0.04173932j
  -0.06424642+0.j          0.24056505+0.j        ]
 [-0.43560243+0.j          0.56942169+0.j          0.13401166-0.17265016j
   0.13401166+0.17265016j -0.18157362-0.49275405j -0.18157362+0.49275405j
  -0.64643635+0.j         -0.36674615+0.j        ]
 [-0.42135762+0.j          0.27271416+0.j         -0.31098085+0.00397063j
  -0.31098085-0.00397063j  0.31338211-0.00312109j  0.31338211+0.00312109j
  -0.09223792+0.j          0.33541485+0.j        ]
 [-0.37753774+0.j         -0.09794104+0.j         -0.12259315-0.30367118j
  -0.12259315+0.30367118j -0.11607913+0.19130452j -0.11607913-0.19130452j
   0.16378827+0.j          0.24683593+0.j        ]]
```

```
TEST NR 6
my power iteration
4.328805970952725
[0.35744436 0.2637974  0.35182123 0.32204973 0.27992185 0.42004786
 0.25489183 0.39395963 0.31651187]
library power iteration
[ 4.32836216+0.j         -0.82149155+0.20908341j -0.82149155-0.20908341j
  0.86498953+0.j          0.51239008+0.20516377j  0.51239008-0.20516377j
 -0.11483576+0.j          0.01678536+0.4746772j   0.01678536-0.4746772j ]
[[-0.35809808+0.j         -0.03746048-0.40447854j -0.03746048+0.40447854j
  -0.13398046+0.j         -0.36335631+0.15985709j -0.36335631-0.15985709j
  -0.28707469+0.j         -0.10160543-0.12785927j -0.10160543+0.12785927j]
 [-0.26469859+0.j          0.20739617+0.1256972j   0.20739617-0.1256972j
   0.04504005+0.j          0.0474096 -0.08054468j  0.0474096 +0.08054468j
   0.31315389+0.j          0.56829985+0.j          0.56829985-0.j        ]
 [-0.353986  +0.j          0.38504423-0.21366114j  0.38504423+0.21366114j
   0.18457914+0.j          0.04136109-0.01293903j  0.04136109+0.01293903j
  -0.18665302+0.j         -0.15954162+0.3054172j  -0.15954162-0.3054172j ]
 [-0.32155238+0.j          0.1924917 +0.05030218j  0.1924917 -0.05030218j
  -0.04730426+0.j         -0.10163304+0.22843683j -0.10163304-0.22843683j
  -0.47394535+0.j         -0.15550247-0.05843763j -0.15550247+0.05843763j]
 [-0.27959625+0.j         -0.07891799+0.23067786j -0.07891799-0.23067786j
  -0.75346094+0.j          0.06583902-0.41847891j  0.06583902+0.41847891j
   0.43730808+0.j          0.32969755-0.0091588j   0.32969755+0.0091588j ]
 [-0.41959954+0.j         -0.50353317+0.j         -0.50353317-0.j
  -0.14135939+0.j          0.62657064+0.j          0.62657064-0.j
   0.15483819+0.j         -0.095514  +0.05860164j -0.095514  -0.05860164j]
 [-0.25300371+0.j         -0.15201961+0.19168602j -0.15201961-0.19168602j
   0.39066615+0.j         -0.34223113+0.13470874j -0.34223113-0.13470874j
  -0.26805938+0.j         -0.12783564+0.22698095j -0.12783564-0.22698095j]
 [-0.39323366+0.j          0.33597128+0.19685512j  0.33597128-0.19685512j
  -0.08989326+0.j          0.04969847-0.0362465j   0.04969847+0.0362465j
   0.45412508+0.j         -0.2611537 -0.36544184j -0.2611537 +0.36544184j]
 [-0.31641269+0.j         -0.13482046-0.0065342j  -0.13482046+0.0065342j
   0.44195788+0.j         -0.10763002-0.21588254j -0.10763002+0.21588254j
   0.25838068+0.j          0.25834924-0.21526122j  0.25834924+0.21526122j]]
```

Wioski:
Zaimplementowana funkcja daje wyniki zgodne z funkcją biblioteczną

Funkcja mieżąca czas

```python
def test_power_iteration():
    my_func = []
    lib_func = []
    sizes = []
    for size in range (100,500,20):
        sizes.append(size)
        arr = generate_matrix(size)
        copy_arr = np.copy(arr)

        start_time = time.time()
        np.linalg.eig(copy_arr)
        diff_time = time.time() - start_time
        lib_func.append(diff_time * 1000)

        start_time = time.time()
        power_iteration(arr,1000000,0.1)
        diff_time = time.time() - start_time
        my_func.append(diff_time*1000)




    fig = plt.figure()
    ax1 = fig.add_axes((0.1, 0.2, 0.8, 0.7))
    ax1.set_xlabel('N')
    ax1.set_ylabel('miliseconds')
    plt.plot(sizes, my_func, label="my function")
    plt.plot(sizes, lib_func, label="library fuction")
    plt.legend(loc="upper left")

    plt.show()
```



Wnioski:

-funkcja oparta na metodzie potęgowej jest szybsza od biliotecznej, zapewne to wynika z tego, że wyliczamy tylko jedną wartość własną i tylko dla niej wektor własny

- na wykresie można zauważyć wzrost czasu potrzebnego na wykonanie funkcji opartej ma teodzi potęgowej przy wielkości macierzy około 120x120.

2.Odwrotna metoda potęgowa

```python
def inverse_power_method(A, sgn, eps, max_iterations):
    size, d = A.shape
    x0 = np.array([1.5 for i in range(size)])
    x0 = np.array(x0/np.linalg.norm(x0, ord=np.inf))

    for i in range(size):
        A[i][i] = A[i][i] - sgn

    flag = False
    LU = scipy.linalg.lu_factor(A)

    for i in range(max_iterations):
        x1 = scipy.linalg.lu_solve(LU, x0)
        x2 = x1/np.linalg.norm(x1, ord=np.inf)

        if np.linalg.norm(x2 - x0) < eps:
            flag = True

        x0 = x2
        if flag:
            break

    return x1/np.linalg.norm(x1)
```

Test metody:

```python
def check_inverse_power_method():
    A = generate_matrix(3)

    e_val, e_vec = scipy.linalg.eig(A)
    print("Library fuction:")
    print(e_val)
    print(e_vec)
    print("My function:")
    print("Sigma equal approximately ", e_val[0])
    print(inverse_power_method(A, e_val[0] + 0.1, 0.01, 10000))
    print("Sigma equal approximately  ", e_val[1])
    print(inverse_power_method(A, e_val[1] + 0.2, 0.01, 10000))
    print("Sigma equal approximately  ", e_val[2])
    print(inverse_power_method(A, e_val[2] + 0.5, 0.01, 10000))
```

Wyniki testu

```
Library fuction:
[-0.01099261+0.j  1.15303112+0.j  0.54376929+0.j]
[[-0.99918178  0.32738701 -0.44765381]
 [ 0.04023321  0.9128558  -0.78456132]
 [-0.00412967  0.2439509   0.42903333]]
My function:
Sigma equal approximately  (-0.010992607466682l8+0j)
[ 0.99918178 -0.04023321  0.00412967]
Sigma equal approximately   (1.1530311178552062+0j)
[0.32738701 0.9128558  0.2439509 ]
Sigma equal approximately   (0.5437692898339123+0j)
[0.32738701 0.9128558  0.2439509 ]
```

Wnioski

-funkcja działa poprawnie