

Programowanie obliczeń komputerowych

W110PA-SI0075G

rok akademicki 2024/25

semestr zimowy

Wykład 7

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 221

Plan

Biblioteka scipy

- szybka transformata Fouriera
- odwrotność macierzy
- układ równań liniowych
- wyznacznik macierzy
- norma macierzy
- metoda najmniejszych kwadratów
- algebraiczne zagadnienie własne

Biblioteka scipy

Narzędzia numeryczne dotyczące m. in.:

- funkcji specjalnych (scipy.special)
- całkowania numerycznego (scipy.integrate)
- problemów optymalizacyjnych (scipy.optimize)
- interpolacji (scipy.interpolate)
- transformaty Fouriera (scipy.fft)
- zagadnień algebry liniowej (scipy.linalg)

Szybka transformata Fouriera

Przykład
pokazuje:

- importowanie funkcji z modułu fft biblioteki scipy
- wykorzystanie funkcji fft do analizy sygnału sinusoidalnego

```
scipy_07.py X
1  """
2  Przykład pokazujący wykorzystanie funkcji fft
3  z modułu fft biblioteki scipy.
4  Wykorzystanie transformaty Fouriera
5  do analizy częstotliwościowej sygnału sinusoidalnego.
6
7  https://docs.scipy.org/doc/scipy/tutorial/fft.html
8  """
9
10 from scipy.fft import fft, fftfreq
11 import numpy as np
12 import matplotlib.pyplot as plt
13
14 # Parametry sygnału
15 N = 600          # liczba próbek
16 T = 1.0 / 800.0 # okres próbkowania (odwrotność częstotliwości próbkowania)
17                 # f_próbkowania = 1/T = 800 Hz
18
19 # Generowanie wektora czasu (od 0 do N*T z wykluczeniem końca)
20 x = np.linspace(0.0, N*T, N, endpoint=False)
21
22 # Definicja sygnału:
23 # Sygnał jest sumą dwóch sinusoid o częstotliwości 50 Hz i 80 Hz
24 # y = sin(2π * 50 * x) + 0.5 * sin(2π * 80 * x)
25 y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)
26
```

Szybka transformata Fouriera

Przykład
pokazuje:

- importowanie funkcji z modułu fft biblioteki scipy
- wykorzystanie funkcji fft do analizy sygnału sinusoidalnego

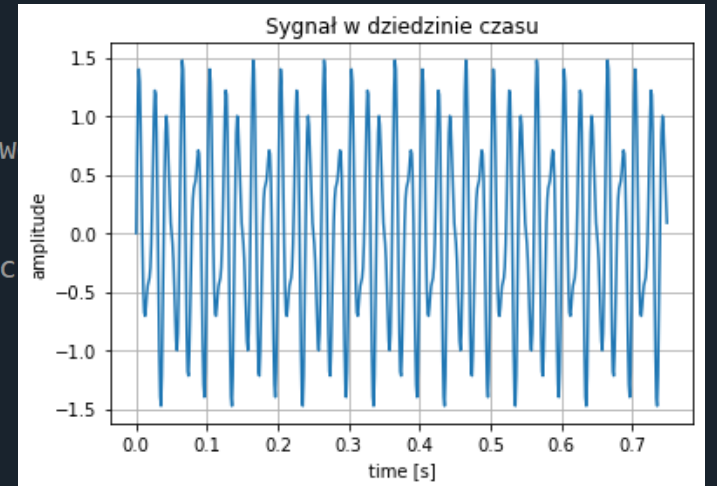
```
scipy_07.py X
18
19 # Generowanie wektora czasu (od 0 do N*T z wykluczeniem końca)
20 x = np.linspace(0.0, N*T, N, endpoint=False)
21
22 # Definicja sygnału:
23 # Sygnał jest sumą dwóch sinusoid o częstotliwości 50 Hz i 80 Hz
24 # y = sin(2π * 50 * x) + 0.5 * sin(2π * 80 * x)
25 y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)
26
27 # Obliczenie FFT sygnału
28 yf = fft(y)
29
30 # Obliczenie wektora częstotliwości
31 xf = fftfreq(N, T)
32
33 # Wykres sygnału w dziedzinie czasu
34 fig, ax = plt.subplots()
35 ax.plot(x, y)
36 ax.grid()
37 ax.set_xlabel('time [s]')
38 ax.set_ylabel('amplitude')
39 ax.set_title('Sygnał w dziedzinie czasu')
```

Szybka transformata Fouriera

Przykład
pokazuje:

- importowanie funkcji z modułu fft biblioteki scipy
- wykorzystanie funkcji fft do analizy sygnału sinusoidalnego

```
scipy_07.py X
18
19 # Generowanie wektora czasu (od 0 do N*T z wykluczeniem końca)
20 x = np.linspace(0.0, N*T, N, endpoint=False)
21
22 # Definicja sygnału:
23 # Sygnał jest sumą dwóch sinusoid o częstotliwości 50 Hz i 80 Hz
24 # y = sin(2π * 50 * x) + 0.5 * sin(2π * 80 * x)
25 y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)
26
27 # Obliczenie FFT sygnału
28 yf = fft(y)
29
30 # Obliczenie wektora częstotliw
31 xf = fftfreq(N, T)
32
33 # Wykres sygnału w dziedzinie c
34 fig, ax = plt.subplots()
35 ax.plot(x, y)
36 ax.grid()
37 ax.set_xlabel('time [s]')
38 ax.set_ylabel('amplitude')
39 ax.set_title('Sygnał w dziedzinie czasu')
```



Szybka transformata Fouriera

Przykład
pokazuje:

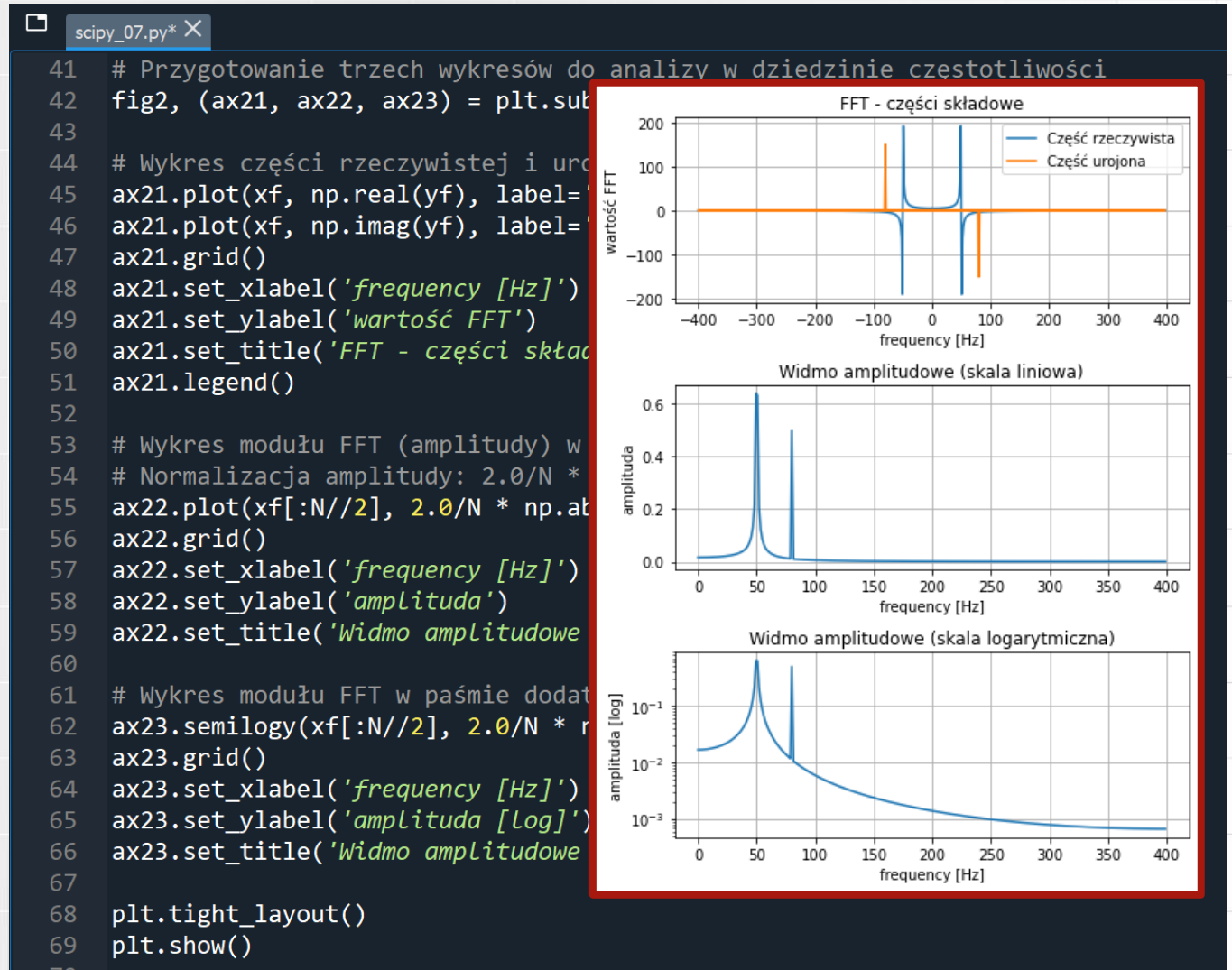
- importowanie funkcji z modułu fft biblioteki scipy
- wykorzystanie funkcji fft do analizy sygnału sinusoidalnego

```
scipy_07.py* X
41 # Przygotowanie trzech wykresów do analizy w dziedzinie częstotliwości
42 fig2, (ax21, ax22, ax23) = plt.subplots(3,1, figsize=(6,8))
43
44 # Wykres części rzeczywistej i urojonej transformaty
45 ax21.plot(xf, np.real(yf), label='Część rzeczywista')
46 ax21.plot(xf, np.imag(yf), label='Część urojona')
47 ax21.grid()
48 ax21.set_xlabel('frequency [Hz]')
49 ax21.set_ylabel('wartość FFT')
50 ax21.set_title('FFT - części składowe')
51 ax21.legend()
52
53 # Wykres modułu FFT (amplitudy) w paśmie dodatnich częstotliwości
54 # Normalizacja amplitudy: 2.0/N * |yf|
55 ax22.plot(xf[:N//2], 2.0/N * np.abs(yf[0:N//2]))
56 ax22.grid()
57 ax22.set_xlabel('frequency [Hz]')
58 ax22.set_ylabel('amplituda')
59 ax22.set_title('Widmo amplitudowe (skala liniowa)')
60
61 # Wykres modułu FFT w paśmie dodatnich częstotliwości w skali logarytmicznej
62 ax23.semilogy(xf[:N//2], 2.0/N * np.abs(yf[0:N//2]))
63 ax23.grid()
64 ax23.set_xlabel('frequency [Hz]')
65 ax23.set_ylabel('amplituda [Log]')
66 ax23.set_title('Widmo amplitudowe (skala logarytmiczna)')
67
68 plt.tight_layout()
69 plt.show()
70
```

Szybka transformata Fouriera

Przykład
pokazuje:

- importowanie funkcji z modułu fft biblioteki scipy
- wykorzystanie funkcji fft do analizy sygnału sinusoidalnego



Odwrotność macierzy

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji inv do obliczenia macierzy odwrotnej

```
scipy_08.py X
1  """
2  Przykład pokazujący wykorzystanie funkcji
3  do obliczania odwrotności macierzy
4  z modułu linalg biblioteki SciPy.
5
6  https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html
7  """
8
9  import numpy as np
10 from scipy import linalg
11
12 # Definiujemy macierz A (3x3)
13 A = np.array([
14     [1, 3, 5],
15     [2, 5, 1],
16     [2, 3, 8]
17 ])
18
```

Odwrotność macierzy

Przykład
pokazuje:

- importowanie modułu `linalg` biblioteki `scipy`
- wykorzystanie funkcji `inv` do obliczenia macierzy odwrotnej

```
17
18
19 # Wyświetlenie oryginalnej macierzy
20 print("Macierz A:")
21 print(A)
22
23 # Obliczenie macierzy odwrotnej A^-1 przy użyciu funkcji linalg.inv()
24 invA = linalg.inv(A)
25
26 # Wyświetlenie macierzy odwrotnej
27 print("\nMacierz odwrotna A^-1:")
28 print(invA)
29
30 # Sprawdzenie poprawności wyniku poprzez pomnożenie A * A^-1.
31 # Wynikiem powinna być macierz zbliżona do macierzy jednostkowej.
32 print("\nWynik mnożenia A * A^-1 (powinien dać macierz jednostkową):")
33 print(A.dot(invA))
34
```

Odwrotność macierzy

Przykład
pokazuje:

- importowa
modułu lin
biblioteki
- wykorzyst
inv do obl
macierzy

Macierz A:

```
[[1 3 5]  
 [2 5 1]  
 [2 3 8]]
```

Macierz odwrotna A^{-1} :

```
[[ -1.48  0.36  0.88]  
 [ 0.56  0.08 -0.36]  
 [ 0.16 -0.12  0.04]]
```

Wynik mnożenia $A * A^{-1}$ (powinien dać macierz jednostkową):

```
[[ 1.00000000e+00 -1.11022302e-16 -5.55111512e-17]  
 [ 3.05311332e-16  1.00000000e+00  1.87350135e-16]  
 [ 2.22044605e-16 -1.11022302e-16  1.00000000e+00]]
```

```
alg.inv()
```

```
1.  
kowej.  
ostkową):")
```

Rozwiązywanie układów równań liniowych

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji solve do rozwiązania układu równań

```
scipy_09.py X
1  """
2  Przykład pokazujący rozwiązywanie
3  układu równań liniowych  $A*x = b$ .
4
5  Pokazywane są podejścia:
6  1. wykorzystujące macierz odwrotną
7     (wolniejsze i mniej stabilne numerycznie),
8  2. oparte o np.linalg.solve
9     (szybsze i bardziej stabilne).
10
11 Sprawdzana jest także jakość uzyskanych rozwiązań.
12 Sprawdzane jest, czy uzyskane rozwiązanie ( $x$ )
13 spełnia równanie  $A*x = b$ .
14
15 https://docs.scipy.org/doc/scipy/tutorial/linalg.html
16 """
17
```

Rozwiązywanie układów równań liniowych

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji solve do rozwiązania układu równań

```
scipy_09.py* X
19 import numpy as np
20 from scipy import linalg
21
22 # Definiujemy macierz A i wektor b
23 A = np.array([[1, 2],
24              [3, 4]])
25 b = np.array([[5],
26              [6]])
27
```

Rozwiązywanie układów równań liniowych

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji solve do rozwiązania układu równań

```
scipy_09.py X
28 # Rozwiązanie układu równań liniowych  $A \cdot x = b$ 
29 # przy pomocy macierzy odwrotnej:
30 # Sposób:  $x = A^{-1} \cdot b$ 
31 # Jest to podejście wolniejsze i mniej stabilne numerycznie.
32 slow_solution = linalg.inv(A).dot(b)
33 print("Rozwiązanie (poprzez mnożenie przez  $A^{-1}$ ):")
34 print(slow_solution)
35
36 # Sprawdzenie jakości rozwiązania:
37 # Powinniśmy otrzymać b, gdy pomnożymy  $A \cdot (A^{-1} \cdot b)$ .
38 check_slow = A.dot(slow_solution) - b
39 print("\nSprawdzenie jakości ( $A \cdot (A^{-1} \cdot b) - b$ ):")
40 print(check_slow) # powinno być bliskie wektorowi zerowemu
```

Rozwiązywanie układów równań liniowych

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji solve do rozwiązania układu równań

```
42 # Rozwiązanie układu równań liniowych  $A \cdot x = b$ 
43 # przy pomocy np.linalg.solve:
44 # Jest to metoda szybsza i bardziej stabilna
45 # numerycznie niż używanie macierzy odwrotnej.
46 fast_solution = np.linalg.solve(A, b)
47 print("\nRozwiązanie (poprzez np.linalg.solve):")
48 print(fast_solution)
49
50 # Sprawdzenie jakości rozwiązania otrzymanego
51 # przez np.linalg.solve:
52 # Podobnie jak wcześniej,  $A \cdot x$  powinno być równe  $b$ .
53 check_fast = A.dot(fast_solution) - b
54 print("\nSprawdzenie jakości ( $A \cdot np.linalg.solve(A,b) - b$ ):")
55 print(check_fast) # powinno być bliskie wektorowi zerowemu
```

Rozwiązywanie układów równań liniowych

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji solve do rozwiązania układu równań

```
Rozwiązanie (poprzez mnożenie przez  $A^{-1}$ ):
```

```
[[ -4. ]  
 [ 4.5]]
```

```
Sprawdzenie jakości ( $A \cdot (A^{-1} \cdot b) - b$ ):
```

```
[[ 0.00000000e+00]  
 [ 1.77635684e-15]]
```

```
Rozwiązanie (poprzez np.linalg.solve):
```

```
[[ -4. ]  
 [ 4.5]]
```

```
Sprawdzenie jakości ( $A \cdot \text{np.linalg.solve}(A,b) - b$ ):
```

```
[[ 0.]  
 [ 0.]]
```


Obliczanie wyznacznika macierzy

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji det do obliczania wyznacznika macierzy

```
scipy_10.py X
1  """
2  Przykład pokazujący obliczanie wyznacznika macierzy
3  za pomocą funkcji det z modułu linalg
4  biblioteki SciPy.
5
6  https://docs.scipy.org/doc/scipy/tutorial/linalg.html
7  """
8
9  import numpy as np
10 from scipy import linalg
11
12 # Definiujemy przykładową macierz A
13 A = np.array([[1, 2],
14              [3, 4]])
15
16 # Obliczamy wyznacznik macierzy A
17 det_A = linalg.det(A)
18
19 # Wyświetlamy otrzymany wyznacznik
20 print("Wyznacznik macierzy A to:", det_A)
21
```

Obliczanie normy macierzy

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji norm do obliczania normy macierzy

```
scipy_11.py X
1  """
2  Przykład pokazujący obliczanie normy macierzy
3  przy użyciu funkcji linalg.norm z biblioteki SciPy.
4
5  Funkcja linalg.norm umożliwia obliczanie
6  różnych rodzajów norm macierzy,
7  m.in. normy Frobeniusa, normy l1, l2, czy normy maksimum (∞).
8
9  https://docs.scipy.org/doc/scipy/tutorial/linalg.html
10 """
11
12 import numpy as np
13 from scipy import linalg
14
15 # Definiujemy przykładową macierz A
16 A = np.array([[1, 2],
17              [3, 4]])
```

Obliczanie normy macierzy

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji norm do obliczania normy macierzy

```
18
19 # Wyświetlenie różnych rodzajów norm macierzy A
20 # Domyślnie linalg.norm(A) oblicza normę Frobeniusa.
21 print("Norma Frobeniusa (domyślna):", linalg.norm(A))
22
23 # Norma Frobeniusa (jawnie podana)
24 print("Norma Frobeniusa (fro):", linalg.norm(A, 'fro'))
25
26 # Norma l1: maksymalna suma bezwzględnych wartości w kolumnie
27 print("Norma l1:", linalg.norm(A, 1))
28
```

```
Norma Frobeniusa (domyślna): 5.477225575051661
Norma Frobeniusa (fro): 5.477225575051661
Norma l1: 6.0
Norma -1: 4.0
Norma ∞: 7.0
```

Dopasowanie - metoda najmniejszych kwadratów

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji lstsq do rozwiązywania układu równań (w sensie najmniejszych kwadratów)

```
scipy_12.py X
1  """
2  Przykład pokazujący dopasowanie modelu
3  do danych metodą najmniejszych kwadratów
4  (least squares).
5
6  W poniższym przykładzie:
7  1. Generowane są dane pomiarowe (xi, zi),
8     które są zaszumioną wersją funkcji
9      $y = c1*exp(-x) + c2*x$ .
10 2. Konstruowana są macierz A i wektor z.
11   Rozwiązanie układu równań  $A*c = z$ 
12   pozwala na wyznaczenie parametrów c1 oraz c2.
13 3. Wykorzystywana jest funkcja linalg.lstsq
14   do znalezienia parametrów  $c = [c1, c2]$ 
15   minimalizujących błąd.
16 4. Otrzymane parametry są wykorzystywane
17   do narysowania dopasowanej krzywej
18   i porównania jej z danymi pomiarowymi.
19
20 https://docs.scipy.org/doc/scipy/tutorial/linalg.html
21 """
22
```

Dopasowanie - metoda najmniejszych kwadratów

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji lstsq do rozwiązywania układu równań (w sensie najmniejszych kwadratów)

```
scipy_12.py* X
23 import numpy as np
24 from scipy import linalg
25 import matplotlib.pyplot as plt
26
27 # Inicjalizacja generatora liczb losowych
28 rng = np.random.default_rng()
29
30 # Definiujemy prawdziwe wartości parametrów modelu
31 c1, c2 = 5.0, 2.0
32
33 # Tworzymy wektor x (xi) składający się z 10 punktów
34 # (od 0.1 do 1.0 z krokiem 0.1)
35 xi = np.arange(0.1, 1, 0.1)
36
37 # Wyliczamy wartości teoretyczne (bez szumu)
38 yi = c1 * np.exp(-xi) + c2 * xi
39
40 # Dodajemy szum do danych, aby zasymulować pomiary
41 # Szum to 5% maksymalnej wartości yi,
42 # pomnożone przez losową liczbę z rozkładu normalnego
43 zi = yi + 0.05 * np.max(yi) * rng.standard_normal(len(yi))
44
```

Dopasowanie - metoda najmniejszych kwadratów

Przykład
pokazuje:

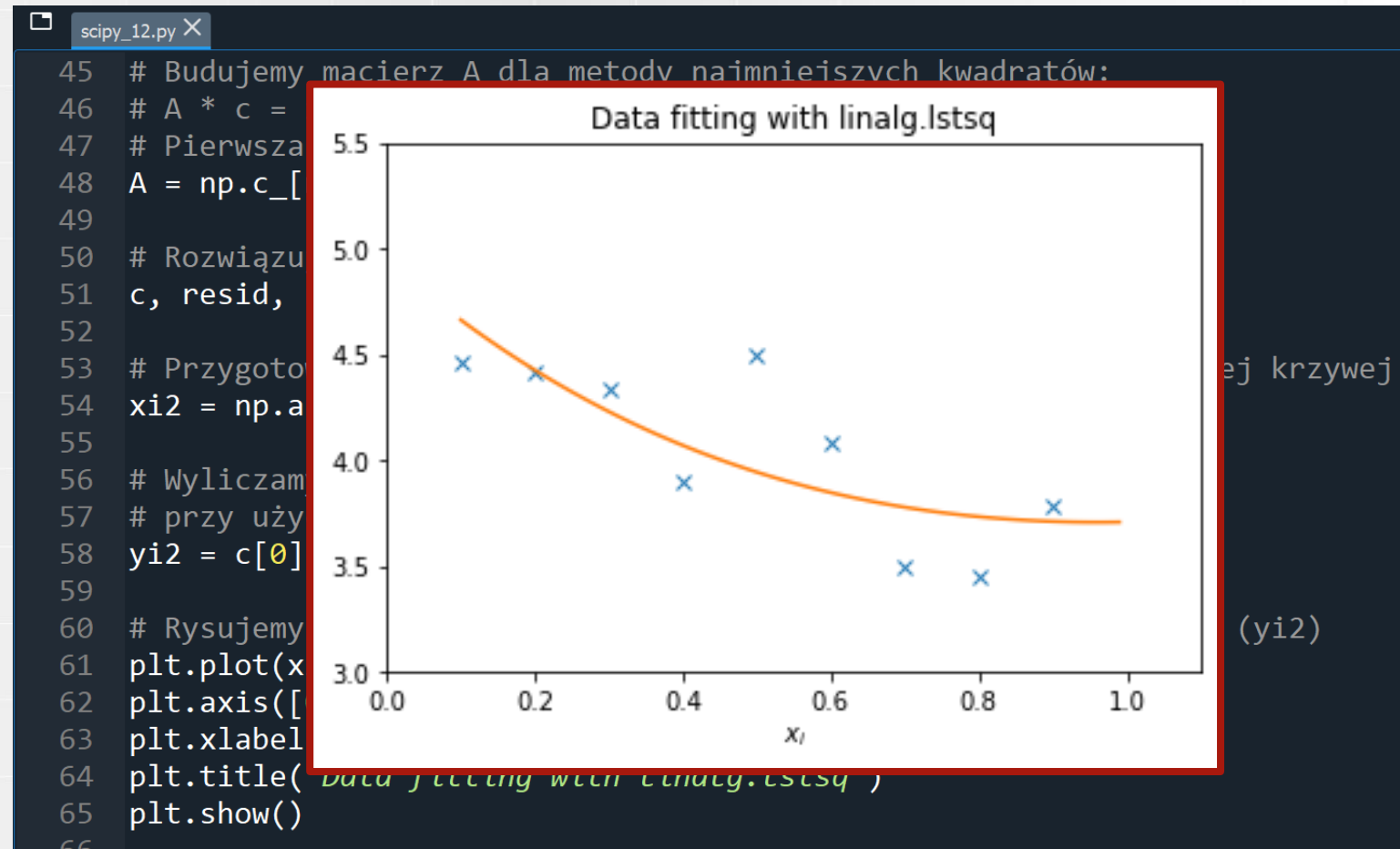
- importowanie modułu `linalg` biblioteki `scipy`
- wykorzystanie funkcji `lstsq` do rozwiązywania układu równań (w sensie najmniejszych kwadratów)

```
scipy_12.py X
45 # Budujemy macierz A dla metody najmniejszych kwadratów:
46 # A * c = z, gdzie c = [c1, c2]
47 # Pierwsza kolumna: exp(-xi), druga kolumna: xi
48 A = np.c_[np.exp(-xi)[:], np.newaxis], xi[:], np.newaxis]]
49
50 # Rozwiązujemy układ A * c ≈ z metodą najmniejszych kwadratów
51 c, resid, rank, sigma = linalg.lstsq(A, zi)
52
53 # Przygotowujemy gęstszy wektor x (xi2) do rysowania dopasowanej krzywej
54 xi2 = np.arange(0.1, 1, 0.01)
55
56 # Wyliczamy wartości dopasowanej funkcji
57 # przy użyciu znalezionych parametrów c
58 yi2 = c[0] * np.exp(-xi2) + c[1] * xi2
59
60 # Rysujemy dane pomiarowe (zi) jako 'x' oraz dopasowaną krzywą (yi2)
61 plt.plot(xi, zi, 'x', xi2, yi2)
62 plt.axis([0, 1.1, 3.0, 5.5])
63 plt.xlabel('$x_i$')
64 plt.title('Data fitting with linalg.lstsq')
65 plt.show()
66
```

Dopasowanie - metoda najmniejszych kwadratów

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji lstsq do rozwiązywania układu równań (w sensie najmniejszych kwadratów)



Zagadnienie własne

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji eig do wyznaczania wartości własnych i wektorów własnych

```
scipy_13.py X
1  """
2  Przykład pokazujący obliczanie wartości własnych
3  (eigenvalues) i wektorów własnych (eigenvectors)
4  macierzy przy użyciu funkcji linalg.eig z biblioteki SciPy.
5
6  W poniższym przykładzie:
7  1. Zdefiniowano macierz A.
8  2. Obliczono wartości własne ( $\lambda$ )
9     i wektory własne ( $v$ ) macierzy A.
10 3. Wyświetlono obliczone wartości własne
11    oraz odpowiadające im wektory własne.
12 4. Zweryfikowano, że wektory własne są
13    znormalizowane (norma powinna być równa 1).
14 5. Sprawdzone poprawność, obliczając  $A*v - \lambda*v$ .
15    Norma różnicy powinna być bliska zeru.
16
17 https://docs.scipy.org/doc/scipy/tutorial/linalg.html
18 """
```


Zagadnienie własne

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji eig do wyznaczania wartości własnych i wektorów własnych

```
19
20 import numpy as np
21 from scipy import linalg
22
23 # Zdefiniowano macierz A
24 A = np.array([[1, 2],
25              [3, 4]])
26
27 # Obliczono wartości własne (la) i wektory własne (v)
28 la, v = linalg.eig(A)
29
30 # Rozpakowano dwie wartości własne
31 l1, l2 = la
```

Zagadnienie własne

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy
- wykorzystanie funkcji eig do wyznaczania wartości własnych i wektorów własnych

```
scipy_13.py X
32
33 # Wyświetlono wartości własne
34 print("Wartości własne (l1, l2):", l1, l2)
35
36 # Wyświetlono pierwszy wektor własny v[:,0]
37 print("Pierwszy wektor własny:", v[:, 0])
38
39 # Wyświetlono drugi wektor własny v[:,1]
40 print("Drugi wektor własny:", v[:, 1])
41
42 # Sprawdzono normy wektorów własnych
43 # (powinny być znormalizowane)
44 # Suma kwadratów modułów elementów powinna być bliska 1
45 print("Normy wektorów własnych:", np.sum(abs(v**2), axis=0))
46
47 # Wybrano pierwszy wektor własny do weryfikacji
48 v1 = np.array(v[:, 0]).T
49
50 # Sprawdzono A*v1 - l1*v1. Jeśli v1 jest wektorem własnym,
51 # norma tego wyrażenia powinna być mała (bliska zeru).
52 print("Sprawdzenie A*v1 - l1*v1:", linalg.norm(A.dot(v1) - l1*v1))
53
```

Zagadnienie własne

Przykład
pokazuje:

- importowanie modułu linalg biblioteki scipy

• wyliczenie wartości własnych i wektorów własnych

```
scipy_13.py X
32
33 # Wyświetlono wartości własne
34 print("Wartości własne (l1, l2):", l1, l2)
35
36 # Wyświetlono pierwszy wektor własny v[:,0]
37 print("Pierwszy wektor własny:", v[:, 0])
38

Wartości własne (l1, l2): (-0.3722813232690143+0j)
(5.372281323269014+0j)
Pierwszy wektor własny: [-0.82456484  0.56576746]
Drugi wektor własny: [-0.41597356 -0.90937671]
Normy wektorów własnych: [1.  1.]
Sprawdzenie A*v1 - l1*v1: 5.551115123125783e-17

52 print("Sprawdzenie A*v1 - l1*v1:", linalg.norm(A.dot(v1) - l1*v1))
53
```

Podsumowanie

Wybrane funkcje biblioteki scipy

- szybka transformata Fouriera
- odwrotność macierzy
- układ równań liniowych
- wyznacznik macierzy
- norma macierzy
- metoda najmniejszych kwadratów
- algebraiczne zagadnienie własne