Politechnika Wrocławska

Programowanie obliczeń komputerowych

W11IKW-SI0106W, W11IKW-SI0106L
rok akademicki 2024/25
semestr zimowy

Wykład 7

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 221

Politechnika Wrocławska

Plan

Biblioteka scipy

- szybka transformata Fouriera

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

https://docs.scipy.org/doc/scipy/tutorial/index.html

Politechnika Wrocławska

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)

https://docs.scipy.org/doc/scipy/tutorial/index.html

Politechnika Wrocławska

Szybka transformata Fouriera

Przykład pokazuje:

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

scipy_07.py

```
"""
Przykład pokazujący wykorzystanie funkcji fft
z modułu fft biblioteki scipy.
Wykorzystanie transformaty Fouriera
do analizy częstotliwościowej sygnału sinusoidalnego.

https://docs.scipy.org/doc/scipy/tutorial/fft.html
"""

from scipy.fft import fft, fftfreq
import numpy as np
import matplotlib.pyplot as plt

# Parametry sygnału
N = 600           # liczba próbek
T = 1.0 / 800.0   # okres próbkowania (odwrotność częstotliwości próbkowania)
                  # f_próbkowania = 1/T = 800 Hz

# Generowanie wektora czasu (od 0 do N*T z wykluczeniem końca)
x = np.linspace(0.0, N*T, N, endpoint=False)

# Definicja sygnału:
# Sygnał jest sumą dwóch sinusoid o częstotliwości 50 Hz i 80 Hz
# y = sin(2π * 50 * x) + 0.5 * sin(2π * 80 * x)
y = np.sin(50.0 * 2.0*np.pi*x) + 0.5*np.sin(80.0 * 2.0*np.pi*x)
```

Politechnika Wrocławska

Szybka transformata Fouriera

Przykład pokazuje:

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

scipy_07.py

```
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')
```

Politechnika Wrocławska

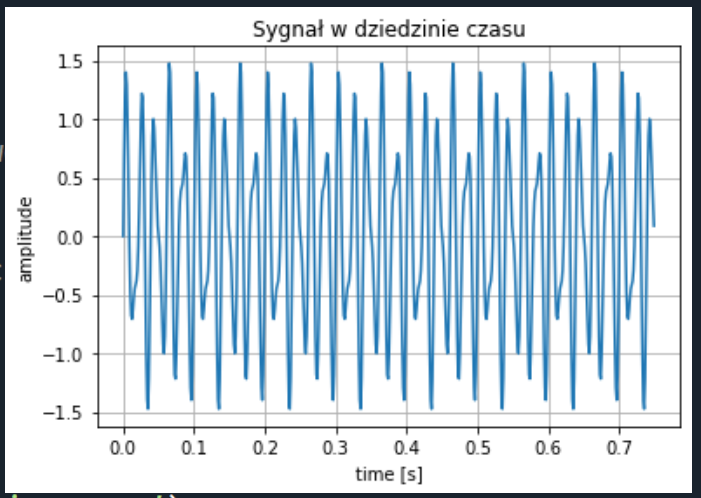
Szybka transformata Fouriera

Przykład pokazuje:

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

scipy_07.py

```
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')
```



Politechnika Wrocławska

Szybka transformata Fouriera

Przykład pokazuje:

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

scipy_07.py*

```
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()
```

Politechnika Wrocławska

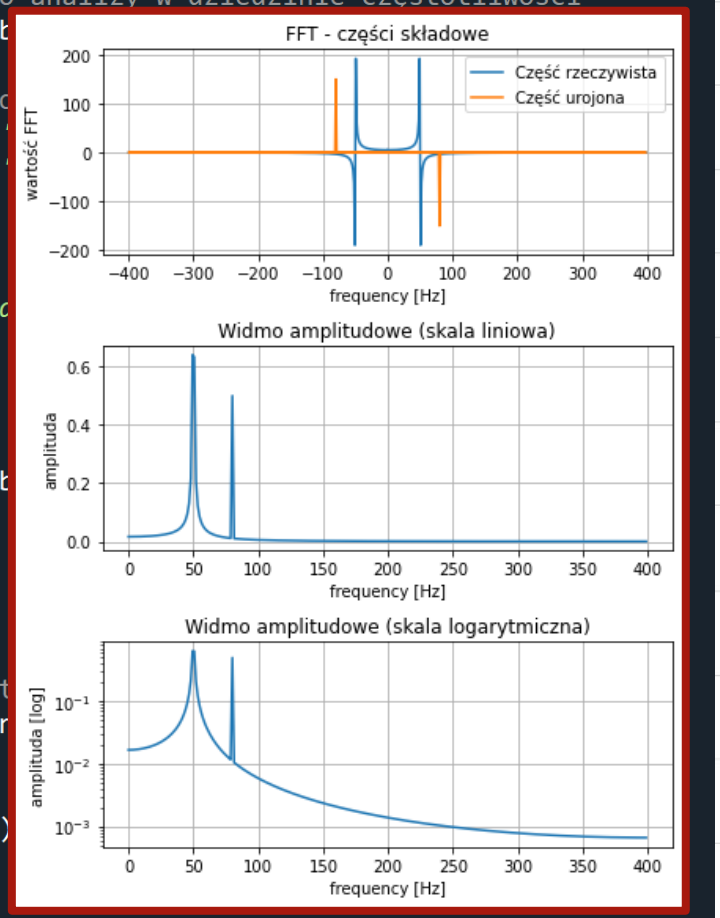
Szybka transformata Fouriera

Przykład pokazuje:

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

scipy_07.py*

```
41  # Przygotowanie trzech wykresów do analizy w dziedzinie częstotliwości
42  fig2, (ax21, ax22, ax23) = plt.sub
43
44  # Wykres części rzeczywistej i uro
45  ax21.plot(xf, np.real(yf), label='
46  ax21.plot(xf, np.imag(yf), label='
47  ax21.grid()
48  ax21.set_xlabel('frequency [Hz]')
49  ax21.set_ylabel('wartość FFT')
50  ax21.set_title('FFT - części skład
51  ax21.legend()
52
53  # Wykres modułu FFT (amplitudy) w
54  # Normalizacja amplitudy: 2.0/N *
55  ax22.plot(xf[:N//2], 2.0/N * np.ab
56  ax22.grid()
57  ax22.set_xlabel('frequency [Hz]')
58  ax22.set_ylabel('amplituda')
59  ax22.set_title('Widmo amplitudowe
60
61  # Wykres modułu FFT w paśmie dodat
62  ax23.semilogy(xf[:N//2], 2.0/N * r
63  ax23.grid()
64  ax23.set_xlabel('frequency [Hz]')
65  ax23.set_ylabel('amplituda [log]')
66  ax23.set_title('Widmo amplitudowe
67
68  plt.tight_layout()
69  plt.show()
```



Politechnika Wrocławska

Odwrotność macierzy

Przykład pokazuje:

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

scipy_08.py

```
"""
Przykład pokazujący wykorzystanie funkcji
do obliczania odwrotności macierzy
z modułu linalg biblioteki SciPy.

https://docs.scipy.org/doc/scipy/reference/tutorial/linalg.html
"""

import numpy as np
from scipy import linalg

# Definiujemy macierz A (3x3)
A = np.array([
    [1, 3, 5],
    [2, 5, 1],
    [2, 3, 8]
])
```

Politechnika Wrocławska

Odwrotność macierzy

Przykład pokazuje:

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

```
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))
```

Politechnika Wrocławska

Odwrotność macierzy

Przykład pokazuje:

- importow
  modułu li
  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]]
```

```
nalg.inv()

1.
kowej.
ostkową):")
```

Politechnika Wrocławska

Rozwiązywanie układów równań liniowych

Przykład pokazuje:

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

scipy_09.py

```
"""
Przykład pokazujący rozwiązywanie
układu równań liniowych A*x = b.

Pokazywane są podejścia:
1. wykorzystujące macierz odwrotną
   (wolniejsze i mniej stabilne numerycznie),
2. oparte o np.linalg.solve
   (szybsze i bardziej stabilne).

Sprawdzana jest także jakość uzyskanych rozwiązań.
Sprawdzane jest, czy uzyskane rozwiązanie (x)
spełnia równanie A*x = b.

https://docs.scipy.org/doc/scipy/tutorial/linalg.html
"""
```

Politechnika Wrocławska

Rozwiązywanie układów równań liniowych

Przykład pokazuje:

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

scipy_09.py*

```
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]])
```

Politechnika Wrocławska

Rozwiązywanie układów równań liniowych

Przykład pokazuje:

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

scipy_09.py

```
28  # Rozwiązanie układu równań liniowych A*x = b
29  # przy pomocy macierzy odwrotnej:
30  # Sposób: x = A^-1 * 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 * (A^-1 * b).
38  check_slow = A.dot(slow_solution) - b
39  print("\nSprawdzenie jakości (A*(A^-1*b) - b):")
40  print(check_slow)  # powinno być bliskie wektorowi zerowemu
```

Politechnika Wrocławska

Rozwiązywanie układów równań liniowych

Przykład pokazuje:

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

```
42 # Rozwiązanie układu równań liniowych A*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*x powinno być równe b.
53 check_fast = A.dot(fast_solution) - b
54 print("\nSprawdzenie jakości (A*np.linalg.solve(A,b) - b):")
55 print(check_fast)  # powinno być bliskie wektorowi zerowemu
```

Politechnika Wrocławska

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*(A^-1*b) - b):
[[0.00000000e+00]
 [1.77635684e-15]]

Rozwiązanie (poprzez np.linalg.solve):
[[-4. ]
 [ 4.5]]

Sprawdzenie jakości (A*np.linalg.solve(A,b) - b):
[[0.]
 [0.]]
```

Politechnika Wrocławska

Obliczanie wyznacznika macierzy

Przykład pokazuje:

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

scipy_10.py

```
"""
Przykład pokazujący obliczanie wyznacznika macierzy
za pomocą funkcji det z modułu linalg
biblioteki SciPy.

https://docs.scipy.org/doc/scipy/tutorial/linalg.html
"""

import numpy as np
from scipy import linalg

# Definiujemy przykładową macierz A
A = np.array([[1, 2],
              [3, 4]])

# Obliczamy wyznacznik macierzy A
det_A = linalg.det(A)

# Wyświetlamy otrzymany wyznacznik
print("Wyznacznik macierzy A to:", det_A)
```

Politechnika Wrocławska

Obliczanie normy macierzy

Przykład pokazuje:

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

scipy_11.py

```
"""
Przykład pokazujący obliczanie normy macierzy
przy użyciu funkcji linalg.norm z biblioteki SciPy.

Funkcja linalg.norm umożliwia obliczanie
różnych rodzajów norm macierzy,
m.in. normy Frobeniusa, normy l1, l2, czy normy maksimum (∞).

https://docs.scipy.org/doc/scipy/tutorial/linalg.html
"""

import numpy as np
from scipy import linalg

# Definiujemy przykładową macierz A
A = np.array([[1, 2],
              [3, 4]])
```

Politechnika Wrocławska

Obliczanie normy macierzy

Przykład pokazuje:

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

```
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

```
"""
Przykład pokazujący dopasowanie modelu
do danych metodą najmniejszych kwadratów
(least squares).

W poniższym przykładzie:
1. Generowane są dane pomiarowe (xi, zi),
   które są zaszumioną wersją funkcji
   y = c1*exp(-x) + c2*x.
2. Konstruowana są macierz A i wektor z.
   Rozwiązanie układu równań A*c = z
   pozwala na wyznaczenie parametrów c1 oraz c2.
3. Wykorzystywana jest funkcja linalg.lstsq
   do znalezienia parametrów c = [c1, c2]
   minimalizujących błąd.
4. Otrzymane parametry są wykorzystywane
   do narysowania dopasowanej krzywej
   i porównania jej z danymi pomiarowymi.

https://docs.scipy.org/doc/scipy/tutorial/linalg.html
"""
```

Politechnika Wrocławska

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*

```
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))
```

Politechnika Wrocławska

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

```
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()
```

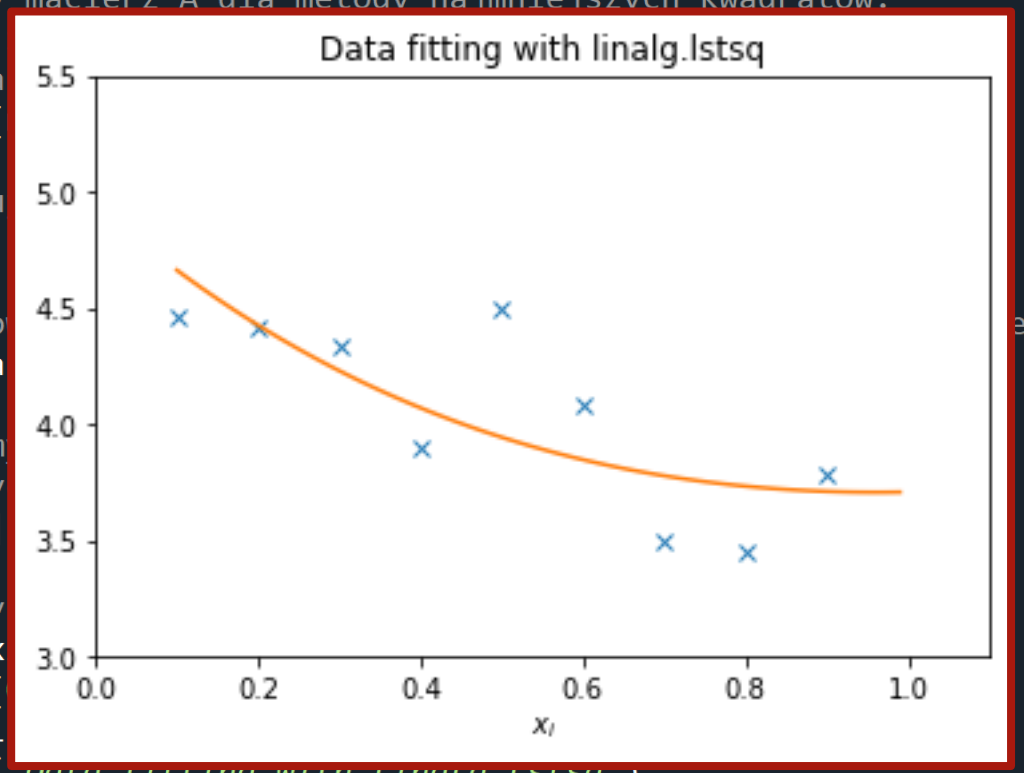
Dopasowanie - metoda najmniejszych kwadratów

Przykład pokazuje:

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

scipy_12.py

```
45  # Budujemy macierz A dla metody najmniejszych kwadratów:
46  # A * c =
47  # Pierwsza
48  A = np.c_[
49
50  # Rozwiązu
51  c, resid,
52
53  # Przygoto                                          ej krzywej
54  xi2 = np.a
55
56  # Wyliczam
57  # przy uży
58  yi2 = c[0]
59
60  # Rysujemy                                          (yi2)
61  plt.plot(x
62  plt.axis([
63  plt.xlabel
64  plt.title( Data fitting with linalg.lstsq )
65  plt.show()
```



Politechnika Wrocławska

Zagadnienie własne

Przykład pokazuje:

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

scipy_13.py

```
"""
Przykład pokazujący obliczanie wartości własnych
(eigenvalues) i wektorów własnych (eigenvectors)
macierzy przy użyciu funkcji linalg.eig z biblioteki SciPy.

W poniższym przykładzie:
1. Zdefiniowano macierz A.
2. Obliczono wartości własne (la)
   i wektory własne (v) macierzy A.
3. Wyświetlono obliczone wartości własne
   oraz odpowiadające im wektory własne.
4. Zweryfikowano, że wektory własne są
   znormalizowane (norma powinna być równa 1).
5. Sprawdzono poprawność, obliczając A*v - λ*v.
   Norma różnicy powinna być bliska zeru.

https://docs.scipy.org/doc/scipy/tutorial/linalg.html
"""
```

Politechnika Wrocławska

Zagadnienie własne

Przykład pokazuje:

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

```
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
```

Politechnika Wrocławska

Zagadnienie własne

Przykład pokazuje:

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

scipy_13.py

```
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))
```

Politechnika Wrocławska

Zagadnienie własne

Przykład pokazuje:

- importowanie modułu linalg biblioteki scipy
- wy
  eig
  wa
  we

scipy_13.py

```
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))
```

Politechnika Wrocławska

Podsumowanie

Wybrane funkcje biblioteki scipy

- szybka transformata Fouriera

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