# Aproksymacja

Joanna Klimek

Informatyka, WIET

14.04.2021r.

Niestety nie udało mi się wykonać pierwszego zadania.

Materiały (kod programów) dołączony do zadania.

**Zadanie 2.**

Wykonaj aproksymację średniokwadratową ciągłą funkcji $f(x) = \frac{1}{1+25x^2}$ w przedziale $[-1,1]$ wielomianem dziesiątego stopnia używając wielomianów w bazie naturalnej i macierzy Hilberta.

Postać naszych wielomianów:

$$W_n(x) = x^n$$

Wyznaczam kolejne wielomiany:

| | |
|---|---|
| $W_0(x)$ | $1$ |
| $W_1(x)$ | $x$ |
| $W(x)$ | $x^2$ |
| $W_3(x)$ | $x^3$ |
| $W_4(x)$ | $x^4$ |
| $\dots$ | $\dots$ |

Wielomiany wyznaczone przez program:

```
W( 0 ) = 1.0

W( 1 ) = 0.0 + 1.0 x**1

W( 2 ) = 0.0 + 0.0 x**1 + 1.0 x**2

W( 3 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 1.0 x**3

W( 4 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 1.0 x**4

W( 5 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 1.0 x**5

W( 6 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 0.0 x**5 + 1.0 x**6

W( 7 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 0.0 x**5 + 0.0 x**6 +
1.0 x**7

W( 8 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 0.0 x**5 + 0.0 x**6 +
0.0 x**7 + 1.0 x**8

W( 9 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 0.0 x**5 + 0.0 x**6 +
0.0 x**7 + 0.0 x**8 + 1.0 x**9

W( 10 ) = 0.0 + 0.0 x**1 + 0.0 x**2 + 0.0 x**3 + 0.0 x**4 + 0.0 x**5 + 0.0 x**6 +
0.0 x**7 + 0.0 x**8 + 0.0 x**9 + 1.0 x**10
```

Układ równań ma postać:

$$\sum_{i=0}^{n} c_i \cdot \int_a^b w_i(x) \cdot w_j(x) dx = \int_a^b f(x) \cdot w_j(x)\, dx, \quad j = 0,1, \dots 10$$

Z wykorzystaniem macierzy Hilberta (oznaczenie H), układ w postaci macierzowej:

$$H \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{10} \end{bmatrix} = \begin{bmatrix} \int_{-1}^{1} f(x) \cdot w_0(x)\, dx \\ \vdots \\ \int_{-1}^{1} f(x) \cdot w_{10}(x)\, dx \end{bmatrix}$$

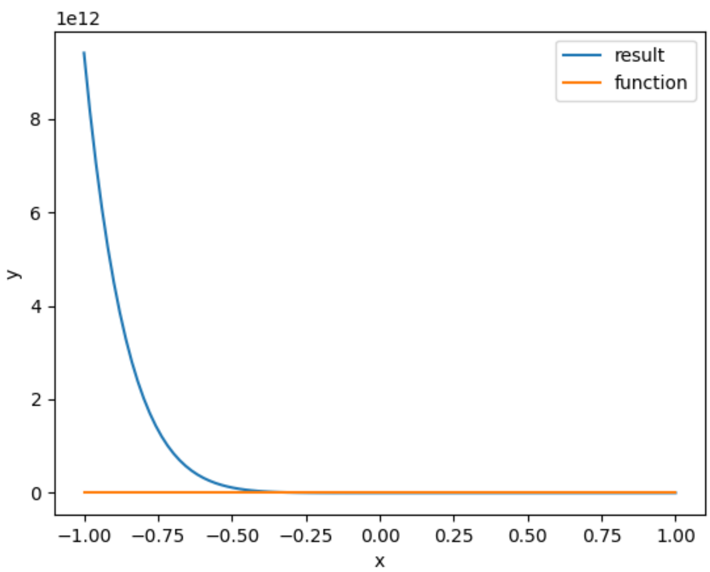
Wartości $c_n$ wyznaczone przez program:

```
c0 = 1381146.2691274732

c1 = -145697587.32102653

c2 = 3805679814.350263

c3 = -42806622111.349594

c4 = 256373714494.8848

c5 = -905589975651.4287

c6 = 1979912618983.387

c7 = -2709069150432.543

c8 = 2257632623566.453

c9 = -1047600010848.4458

c10 = 207486385658.61096
```

Wielomian aproksymujący ma postać:

$$w(x) = c_0 * W_0(x) + c_1 * W_1(x) + c_2 * W_2(x) + \dots + c_{10} * W_{10}(x)$$

Otrzymany wykres wielomianu aproksymującego:



Kod w języku Python wykorzystany w zadaniu (dołączony również w materiałach):

```
from numpy.polynomial.polynomial import Polynomial
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt


nmax = 10

fx_odwr = Polynomial((1, 0, 25), domain=[-1.,  1.]) # odwrotność funkcji aproksymowanej

Wx = Polynomial((0, 1), domain=[-1.,  1.]) # wielomian R = x potrzebny do mnożenia z wielomianami

W0 = Polynomial((1), domain=[-1.,  1.])
W1 = Polynomial((0, 1), domain=[-1.,  1.])

print("\n\n ======= WYZNACZONE WIELOMIANY =======\n")
print("\nW( 0 ) =", W0)
print("\nW( 1 ) =", W1)

polynomials = []
polynomials.append(W0)
polynomials.append(W1)

# wyznaczenie wielomianow
for n in range(2, nmax+1):
    Wn = polynomials[n-1] * Wx
    print("\nW(", n, ") =", Wn)
    polynomials.append(Wn)

"""
calki_lewastrona = [None] * (nmax+1)
for i in range(len(calki_lewastrona)):
    calki_lewastrona[i] = [None] * (nmax+1)

for i in range(0, nmax+1):
    for j in range(0, nmax + 1):
        if i <= j:
            P = polynomials[i]
            R = polynomials[j]
            P_func = lambda x: P(x) * R(x)
            calka = integrate.quad(P_func, -1, 1)
            # print(calka)
            calki_lewastrona[i][j] = calka[0]
"""

# macierz hilberta
calki_lewastrona = [None] * (nmax+1)
for i in range(len(calki_lewastrona)):
    calki_lewastrona[i] = [None] * (nmax+1)

for i in range(0, nmax+1):
    for j in range(0, nmax + 1):
        calki_lewastrona[i][j] = 1 / (i+1 + j+1 - 1)
#print(calki_lewastrona)

calki_prawastrona = [None] * (nmax+1)
for i in range(len(calki_prawastrona)):
```

```python
    P = polynomials[i]
    P_func = lambda x: P(x) / (1 + (x**2 * 25))
    calki_prawastrona[i] = integrate.quad(P_func, -1, 1)[0]

a = np.array(calki_lewastrona)
b = np.array(calki_prawastrona)
c_list = np.linalg.solve(a, b)

print("\n\n ======= WYZNACZONE Cn =======\n")
for i in range(len(c_list)):
    print("\nc" + str(i) + " =", c_list[i])

result = Polynomial(0)

for n in range(0, nmax+1):
    result = result + (polynomials[n] * c_list[n])

#print(result)

x = np.linspace(-1, 1, 100)

fig, ax = plt.subplots()
ax.plot(x, result(x), label="result")
ax.plot(x, 1/fx_odwr(x), label="function")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend()
plt.show()
```

**Zadanie 3.**

Wykonaj aproksymację średniokwadratową ciągłą funkcji $f(x) = \frac{1}{1+25x^2}$ w przedziale $[-1,1]$ wielomianem dziesiątego stopnia używając wielomianów Legendre'a.

Postać wielomianów Legendre'a:

$$P_n(x) = \frac{1}{n! \cdot 2^n} \frac{d^n}{dx^n} [(x^2-1)^n], x \in [-1,1], n = 0,1 \ldots$$

Spełniają one wzór rekurencyjny (dla $n \geq 1$):

$$(2n+1) \cdot x \cdot P_n(x) = (n+1)P_{n+1}(x) + n \cdot P_{n-1}(x)$$

Po przekształceniach wzór dla $n \geq 2$:

$$P_n(x) = \frac{(2n-1) \cdot x \cdot P_{n-1}(x) - (n-1) \cdot P_{n-2}(x)}{n}$$

Na podstawie znanych $P_0(x) = 1$ i $P_1(x) = x$ wyznaczam kolejne wielomiany:

| | |
|---|---|
| $P_0(x)$ | $1$ |
| $P_1(x)$ | $x$ |
| $P_2(x)$ | $\frac{3x^2-1}{2}$ |
| $P_3(x)$ | $\frac{5x^3-3x}{2}$ |
| $P_4(x)$ | $\frac{35x^4-30x^2+3}{8}$ |
| … | … |

Wielomiany wyznaczone przez program:

```
P( 0 ) = 1.0

P( 1 ) = 0.0 + 1.0 x**1

P( 2 ) = -0.5 + 0.0 x**1 + 1.5 x**2

P( 3 ) = 0.0 - 1.5 x**1 + 0.0 x**2 + 2.5 x**3

P( 4 ) = 0.375 + 0.0 x**1 - 3.75 x**2 + 0.0 x**3 + 4.375 x**4

P( 5 ) = 0.0 + 1.875 x**1 + 0.0 x**2 - 8.75 x**3 + 0.0 x**4 + 7.875 x**5

P( 6 ) = -0.3125 + 0.0 x**1 + 6.5625 x**2 + 0.0 x**3 - 19.6875 x**4 + 0.0 x**5 +
14.4375 x**6

P( 7 ) = 0.0 - 2.1875 x**1 + 0.0 x**2 + 19.6875 x**3 + 0.0 x**4 - 43.3125 x**5 +
0.0 x**6 + 26.8125 x**7

P( 8 ) = 0.2734375 + 0.0 x**1 - 9.84375 x**2 + 0.0 x**3 + 54.140625 x**4 +
0.0 x**5 - 93.84375 x**6 + 0.0 x**7 + 50.2734375 x**8

P( 9 ) = 0.0 + 2.4609375 x**1 + 0.0 x**2 - 36.09375 x**3 + 0.0 x**4 +
140.765625 x**5 + 0.0 x**6 - 201.09375 x**7 + 0.0 x**8 + 94.9609375 x**9

P( 10 ) = -0.24609375 + 0.0 x**1 + 13.53515625 x**2 + 0.0 x**3 - 117.3046875 x**4 +
0.0 x**5 + 351.9140625 x**6 + 0.0 x**7 - 427.32421875 x**8 + 0.0 x**9 +
180.42578125 x**10
```

Wielomiany te są ortogonalne na zadanym przedziale stąd nasz układ równań upraszcza się do postaci:

$$c_n \int_{-1}^{1} P_n(x)P_n(x)dx = \int_{-1}^{1} P_n(x)f(x)dx \quad dla\ n = 0,1,2\ldots10$$

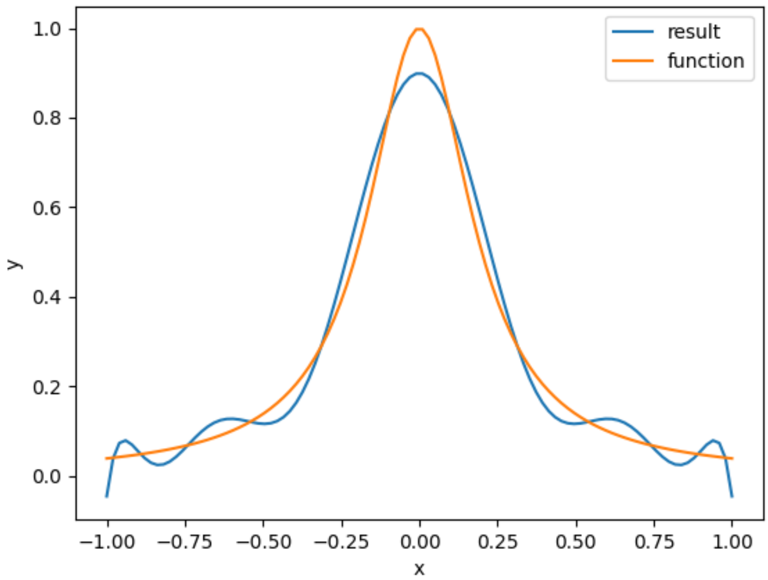
Wartości $c_n$ wyznaczone przez program:

```
c0 = 0.2746801533890033

c1 = 0.0

c2 = -0.4691044294892092

c3 = 0.0

c4 = 0.4271685744265475

c5 = 0.0

c6 = -0.3461031273980904

c7 = 0.0

c8 = 0.26638148602333966

c9 = 0.0

c10 = -0.19914364758543476
```

Wielomian aproksymujący ma postać:

$$w(x) = c_0 * P_0(x) + c_1 * P_1(x) + c_2 * P_2(x) + \ldots + c_{10} * P_{10}(x)$$

Otrzymany wykres:



Kod w języku Python wykorzystany w zadaniu (dołączony również w materiałach):

```
from numpy.polynomial.polynomial import Polynomial
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt


nmax = 10

fx_odwr = Polynomial((1, 0, 25), domain=[-1.,  1.]) # odwrotność funkcji
aproksymowanej

Px = Polynomial((0, 1), domain=[-1.,  1.]) # wielomian R = x potrzebny do
mnożenia z wielomianami Legendre’a

P0 = Polynomial((1), domain=[-1.,  1.])
P1 = Polynomial((0, 1), domain=[-1.,  1.])

print("\n\n ======= WYZNACZONE WIELOMIANY =======\n")
print("\nP( 0 ) =", P0)
print("\nP( 1 ) =", P1)

polynomials = []
polynomials.append(P0)
polynomials.append(P1)

# wyznaczenie wielomianow Legendre’a
for n in range(2, nmax+1):
    Pn = (((2 * n - 1) * Px * polynomials[n-1]) - ((n - 1) * polynomials[n-
2])) / n
    print("\nP(", n, ") =", Pn)
    polynomials.append(Pn)


c_list = []

for n in range(0, nmax+1):

    # obliczenie calki lewej strony rownania
    L = polynomials[n] * polynomials[n]
    L = L.integ()
    calkaL = L(1) - L(-1)
    #print(calkaL)

    # obliczenie calki prawej strony rownania
    P = polynomials[n]
    P_func = lambda x: P(x) / (1 + (x**2 * 25))
    calkaR = integrate.quad(P_func, -1, 1)
    #print(calkaR)

    c_list.append(calkaR[0] / calkaL)

print("\n\n ======= WYZNACZONE Cn =======\n")
for i in range(len(c_list)):
    print("\nc" + str(i) + " =", c_list[i])

result = Polynomial(0)

for n in range(0, nmax+1):
    result = result + (polynomials[n] * c_list[n])
```

```
#print(result)

x = np.linspace(-1, 1, 100)

fig, ax = plt.subplots()
ax.plot(x, result(x), label="result")
ax.plot(x, 1/fx_odwr(x), label="function")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend()
plt.show()
```

**Zadanie 4.**

Wykonaj aproksymację średniokwadratową ciągłą funkcji $f(x) = \frac{1}{1+25x^2}$ w przedziale $[-1,1]$ wielomianem dziesiątego stopnia używając wielomianów Czebyszewa. Aproksymacja ta jest tańszym obliczeniowo zamiennikiem aproksymacji jednostajnej.

Postać wielomianów Czebyszewa:

$$T_n(x) = \cos\,[n \cdot arccosx], x \in [-1,1], n = 0,1 \dots$$

Spełniają one wzór rekurencyjny (dla $n \geq 2$):

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

Na podstawie znanych $T_0(x) = 1$ i $T_1(x) = x$ wyznaczam kolejne wielomiany:

| | |
|---|---|
| $T_0(x)$ | $1$ |
| $T_1(x)$ | $x$ |
| $T_2(x)$ | $2x^2 - 1$ |
| $T_3(x)$ | $4x^3 - 3x$ |
| $T_4(x)$ | $8x^4 - 8x^2 + 1$ |
| … | … |

Wielomiany wyznaczone przez program:

```
T( 0 ) = 1.0

T( 1 ) = 0.0 + 1.0 x**1

T( 2 ) = -1.0 + 0.0 x**1 + 2.0 x**2

T( 3 ) = 0.0 - 3.0 x**1 + 0.0 x**2 + 4.0 x**3

T( 4 ) = 1.0 + 0.0 x**1 - 8.0 x**2 + 0.0 x**3 + 8.0 x**4

T( 5 ) = 0.0 + 5.0 x**1 + 0.0 x**2 - 20.0 x**3 + 0.0 x**4 + 16.0 x**5

T( 6 ) = -1.0 + 0.0 x**1 + 18.0 x**2 + 0.0 x**3 - 48.0 x**4 + 0.0 x**5 + 32.0 x**6

T( 7 ) = 0.0 - 7.0 x**1 + 0.0 x**2 + 56.0 x**3 + 0.0 x**4 - 112.0 x**5 + 0.0 x**6 +
64.0 x**7

T( 8 ) = 1.0 + 0.0 x**1 - 32.0 x**2 + 0.0 x**3 + 160.0 x**4 + 0.0 x**5 -
256.0 x**6 + 0.0 x**7 + 128.0 x**8

T( 9 ) = 0.0 + 9.0 x**1 + 0.0 x**2 - 120.0 x**3 + 0.0 x**4 + 432.0 x**5 +
0.0 x**6 - 576.0 x**7 + 0.0 x**8 + 256.0 x**9

T( 10 ) = -1.0 + 0.0 x**1 + 50.0 x**2 + 0.0 x**3 - 400.0 x**4 + 0.0 x**5 +
1120.0 x**6 + 0.0 x**7 - 1280.0 x**8 + 0.0 x**9 + 512.0 x**10
```

Wielomiany te są ortogonalne na zadanym przedziale z wagą $p(x) = \frac{1}{\sqrt{1-x^2}}$ stąd nasz układ równań upraszcza się do postaci:

$$c_n \int_{-1}^{1} p(x)T_n(x)T_n(x)dx = \int_{-1}^{1} p(x)T_n(x)f(x)dx \quad dla\ n = 0,1,2\ldots10$$

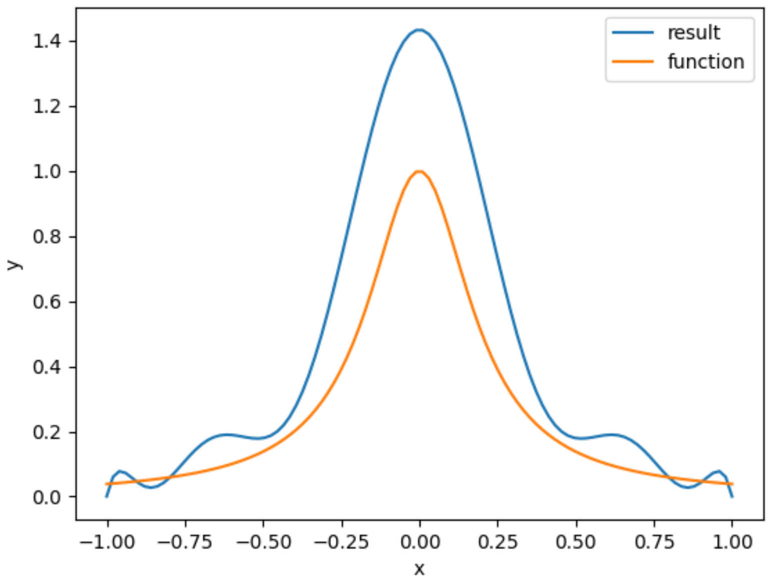
Wartości $c_n$ wyznaczone przez program:

```
c0 = 0.30805850470026985

c1 = 0.0

c2 = -0.4436560262843584

c3 = 0.0

c4 = 0.2827821450659605

c5 = 0.0

c6 = -0.18835223615073873

c7 = 0.0

c8 = 0.12619714047407346

c9 = 0.0

c10 = -0.08469403752362883
```

Wielomian aproksymujący ma postać:

$$w(x) = c_0 * T_0(x) + c_1 * T_1(x) + c_2 * T_2(x) + \ldots + c_{10} * T_{10}(x)$$

Otrzymany wykres:



Kod w języku Python wykorzystany w zadaniu (dołączony również w materiałach):

```
from numpy.polynomial.polynomial import Polynomial
import numpy as np
from scipy import integrate
import matplotlib.pyplot as plt


nmax = 10

fx_odwr = Polynomial((1, 0, 25), domain=[-1.,  1.]) # odwrotność funkcji
aproksymowanej

Tx = Polynomial((0, 1), domain=[-1.,  1.]) # wielomian R = x potrzebny do
mnożenia z wielomianami Czebyszewa

T0 = Polynomial((1), domain=[-1.,  1.])
T1 = Polynomial((0, 1), domain=[-1.,  1.])

print("\n\n ======= WYZNACZONE WIELOMIANY =======\n")
print("\nT( 0 ) =", T0)
print("\nT( 1 ) =", T1)

polynomials = []
polynomials.append(T0)
polynomials.append(T1)

# wyznaczenie wielomianow Czebyszewa
for n in range(2, nmax+1):
    Tn = ((2 * Tx * polynomials[n-1]) - (polynomials[n-2]))
    print("\nT(", n, ") =", Tn)
    polynomials.append(Tn)


c_list = []

for n in range(0, nmax+1):

    # obliczenie calki lewej strony rownania
    L = polynomials[n] * polynomials[n]
    L = L.integ()
    calkaL = L(1) - L(-1)
    #print(calkaL)

    # obliczenie calki prawej strony rownania
    T = polynomials[n]

    T_func = lambda x: T(x) / ((1 + (x**2 * 25)) * (np.sqrt(1 - x**2)))
    calkaR = integrate.quad(T_func, -1, 1)
    #print(calkaR)

    c_list.append(calkaR[0] / calkaL)

print("\n\n ======= WYZNACZONE Cn =======\n")
for i in range(len(c_list)):
    print("\nc" + str(i) + " =", c_list[i])

result = Polynomial(0)

for n in range(0, nmax+1):
    result = result + (polynomials[n] * c_list[n])
```

```
#print(result)

x = np.linspace(-1, 1, 100)

fig, ax = plt.subplots()
ax.plot(x, result(x), label="result")
ax.plot(x, 1/fx_odwr(x), label="function")
ax.set_xlabel("x")
ax.set_ylabel("y")
ax.legend()
plt.show()
```