Metody numeryczne

Instrukcja 4

Aproksymacja i interpolacja

Robert Kazała

1 Aproksymacja i interpolacja

1.1 Aproksymacja

Aproksymacja polega na opisaniu:
- zbyt złożonej funkcji, funkcją uproszczoną lub
- funkcji w postaci dyskretnej (np. zbioru punktów pomiarowych).

W każdym z tych przypadków poszukuje się funkcji, która dobrze przybliża funkcję pierwotną. Zadanie aproksymacji optymalnej w bazie jednomianów polega na dobraniu wielomianu aproksymującego P(x) w taki sposób, aby najlepiej przybliżał on funkcję aproksymowaną f(x).

$$P(x)=a_m x^m+a_{m-1} x^{m-1}+...+a_1 x+a_0=\sum_{k=1}^{m} a_k x^k=\boldsymbol{p}(x)\boldsymbol{a}$$

p = [1 x . . . x_m] - wektor wierszowy jednomianów
a = [a_0 a_1 . . . a_m]^T - wektor kolumnowy nieznanych parametrów aproksymacji.

Kryteriów oceny jakości aproksymacji jest wiele. W praktyce najczęściej wykorzystywanym kryterium jest kryterium najmniejszych kwadratów.
Błąd ε dla metody najmniejszych kwadratów określa zależność:

$$\varepsilon=\sum_{i=0}^{n}[f(x_i)-P(x_i)]=\sum_{i=0}^{n}[f(x_i)-\boldsymbol{p}(x_i)\boldsymbol{a}]^2$$

Warunkiem koniecznym istnienia minimum funkcji błędu

$$\varepsilon=\sum_{i=0}^{n}[f(x_i)-P(x_i)]=\sum_{i=0}^{n}[f(x_i)-\boldsymbol{p}(x_i)\boldsymbol{a}]^2$$

Przyjmując oznaczenie macierzy

$$\boldsymbol{A}=\sum_{i=0}^{n}\boldsymbol{p}^T(x_i)\boldsymbol{p}(x_i) \qquad \boldsymbol{B}=[\boldsymbol{p}^T(x_0) \quad \boldsymbol{p}^T(x_1) \quad ... \quad \boldsymbol{p}^T(x_n)]$$

$$\boldsymbol{A}\boldsymbol{a}=\boldsymbol{B}\boldsymbol{F} \qquad \boldsymbol{a}=\boldsymbol{A}^{-1}\boldsymbol{B}\boldsymbol{F}$$

$$\boldsymbol{A}=\boldsymbol{B}\boldsymbol{B}^T$$

$$\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{a}=\boldsymbol{B}\boldsymbol{F} \qquad \boldsymbol{a}=(\boldsymbol{B}\boldsymbol{B}^{-1})\boldsymbol{B}\boldsymbol{F}$$

Przykład 1

```
import matplotlib.pyplot as plt
import numpy as np

# funkcja aproksymowana
x = np.array( [ 2.0, 4.0, 6.0, 8.0 ] )[None, :] # wektor wierszowy
F = np.array( [ 2.0, 11.0, 28.0, 40.0 ] )[:, None]   # wektor kolumnowy

# metoda najmniejszych kwadratow
# funkcja aproksymujaca P = a0 + a1*x
# a = [a0 a1]^T
# p = [1 x]
B = np.zeros((2, x.shape[1]))
B[0, :] = x**0
B[1, :] = x**1
# a = ((B*B')^(-1)) * B*F
a = np.dot( np.linalg.inv( np.dot( B, np.transpose(B) ) ), np.dot(B, F) )

P = a[0] + a[1]*x   # funkcja aproksymujaca
E = np.sum( (F - np.transpose(P))**2 )  # blad aproksymacji
```

1.2 Interpolacja

Interpolacja funkcji f(x) jest szczególnym przypadkiem aproksymacji gdy m = n.
W węzłach interpolacji x = [x_0, x_1, . . . , x_n] wartość funkcji interpolacyjnej P(x) jest równa wartościom funkcji interpolowanej f(x).

Wyrażenia dotyczące aproksymacji punktowej są również adekwatne dla przypadku interpolacji.

a = [a_0 a_1 . . . a_n]^T - wektor kolumnowy nieznanych parametrów interpolacji.
F = [f_0 f_1 . . . f_n] - wektor wartości wielomianu w punktach interpolacji

p(x) = [1 x, x^2 . . . x^n]

Równanie opisujące zadanie interpolacji ma postać

$$\boldsymbol{B}^T \boldsymbol{a} = \boldsymbol{F}$$

Macierz *B* możemy utworzyć za pomocą zależności

$$\boldsymbol{B} = [\boldsymbol{p}^T(x_0) \quad \boldsymbol{p}^T(x_1) \quad \ldots \quad \boldsymbol{p}^T(x_n)]$$

Po przekształceniu możemy uzyskać wektor nieznanych parametrów

$$\boldsymbol{a} = (\boldsymbol{B}^T)^{-1} \boldsymbol{F}$$

Przykład 2

```
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return 1 / x

# funkcja interpolowana
x = np.array( [ 2.0, 2.5, 4.0 ] )[None, :] # wektor wierszowy
F = np.transpose( f(x) )

# funkcja interpolujaca (n = m = 3) P = a0 + a1*x + a2*x^2
# a = [a0 a1 a2]^T
# p = [1 x x^2]
B = np.zeros((x.shape[1], x.shape[1]))
B[0, :] = x**0
B[1, :] = x**1
B[2, :] = x**2

# a = ((B*B')^(-1)) * B*F
# a = np.dot( np.linalg.inv( np.dot( B, np.transpose(B) ) ), np.dot(B, F)
)
# a = ((B')^(-1)) * F   # zapis uproszczony (mozliwy bo B jest macierza
kwadratowa)
a = np.dot( np.linalg.inv( np.transpose(B) ), F )

# funkcja interpolujaca w postaci wielomianu
p = np.poly1d( np.squeeze( np.fliplr(np.transpose(a)) ) )

# wspolczynniki wielomianu interpolacyjnego
for idx, val in enumerate(a):
    print('a[%d] = %g' % (idx, val))
print(p)
vx = np.linspace(x[:, 0], x[:, -1], 100)
plt.plot(vx, f(vx))
plt.plot(vx, p(vx))
```

1.3 Zadania do wykonania

Zadanie 1

Przeanalizować i uruchomić przykład 1. Zaprezentować działanie programu dla różnych danych wejściowych oraz różnych stopni wielomianu aproksymujacego.

Zadanie 2

Przeanalizować i uruchomić przykład 2. Zaprezentować działanie programu dla różnych danych wejściowych oraz różnych stopni wielomianu interpolacyjnego.

Zadanie 3

Napisać własną wersję programu do aproksymacji z przykładu 1, wykorzystującą pętle zamiast obliczeń tablicowych i macierzowych (odpowiednik języka C).

Zadanie 4

Napisać własną wersję programu do interpolacji z przykładu 2, wykorzystującą pętle zamiast obliczeń tablicowych i macierzowych (odpowiednik języka C).

Zadanie 5

Zapoznać się z działaniem funkcji *interp1d* z pakietu *scipy.interpolate*. Zaprezentować działanie funkcji na własnych przykładach.