Zadanie numeryczne 6

Oleg Semenov

Spis treści

Wstęp

W rozwiązaniu danego zadania skorzystałem z Pythona 3.7 z dodatkowym użyciem bibliotek NumPy i Matplotlib. To rozwiązanie wykorzystuje interpolację Lagrange'a, żeby uzyskać funckję interpolacyjną z punktów wejściowych.

Interpolacja Lagrange'a

Interpolacja Lagrange'a zwaną także interpolacją wielomianową, to metoda numeryczna przybliżania funkcji wielomianem Lagrange'a stopnia n, przyjmującego w $n+1$ wartości takie same jak funkcja przybliżana.

Poszukiwany wzór interpolacyjny ma równanie:

$$f(x) = \sum_{j=1}^{n} l_j(x) f_j + E(x),$$

gdzie

$$l_j(x) = \frac{(x-x_1)\dots(x-x_{j-1})(x-x_{j+1})\dots(x-x_n)}{(x_j-x_1)\dots(x_j-x_{j-1})(x_j-x_{j+1})\dots(x_j-x_n)}$$

$E(x)$ to jest błąd interpolacji, który znika samoistnie z racji na to, że $f(x)$ jest co najwyżej stopnia $n-1$.

Wzór użyty w rozwiązaniu

W rozwiązaniu tego zadania użyłem wzoru, który znalazłem w **"Beginner's guide to mapping simplexes affinely"** w "Lagrange interpolation":

$$f(x) = (-1)\frac{\det\begin{pmatrix} 0 & f_0 & f_1 & \cdots & f_n \\ x^n & x_0^n & x_1^n & \cdots & x_n^n \\ x^{n-1} & x_0^{n-1} & x_1^{n-1} & \cdots & x_n^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}}{\det\begin{pmatrix} x_0^n & x_1^n & \cdots & x_n^n \\ x_0^{n-1} & x_1^{n-1} & \cdots & x_n^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}}$$

Tutaj można użyć rozszerzenia Laplace'a na pierwszej kolumnie, żeby uzyskać współczynniki wielomianu interpolacyjnego.

Kod Interpolacji Lagrange'a

Najpierw generuję macierz Vandermonde'a:

```
import numpy as np
mat = np.zeros((len(x), len(x)))
for i in range(len(x)):
    for j in range(len(x)):
        mat[i][j] = pow(x[i], j)
```

Później wykonuję pozostałe operacje potrzebne do wyliczenia współczynników używając interpolacji Lagrange'a:

```
import numpy as np
M = mat.T.tolist()

coeffs = [np.linalg.det((M+[y]+M)[d:d+len(x)]) for d in range(len(x)+1)]
coeffs = (coeffs / coeffs[0] * (-1)**(len(x)+1))[1:]
```

Rozwiązanie zadania numerycznego

Generowanie wyników

Współczynniki

Współczynniki są zapisywane do pliku `out.txt`

```
with open("out.txt", "w") as fd:
    for i in range(len(coeffs)):
        fd.write("{0} = {1:.4f}\n".format(chr(97+i), coeffs[i]))
```

Wykres

Przed generowaniem samego wykresu użyłem biblioteki NumPy, a dokładnie modułu Polynomial, żeby stworzyć z współczynników obiekt Polynomial. Zrobiłem to dla ułatwienia czytelności kodu.

```
from numpy.polynomial import Polynomial
poly = Polynomial(coeffs, [-1, 1])
```

Później generuję przedział gererowania punktów wyznaczanych przez wielomian interpolacyjny z próbkowaniem 100 punktów na przedziale $-1 \leqslant x \leqslant 1$.

```
import numpy as np
x1 = np.linspace(-1, 1, 100)
y1 = poly(x1)
```

Do wygenerowania wykresu użyłem biblioteki Matplotlib. Wykres zapisywany jest w formacie `.png`.

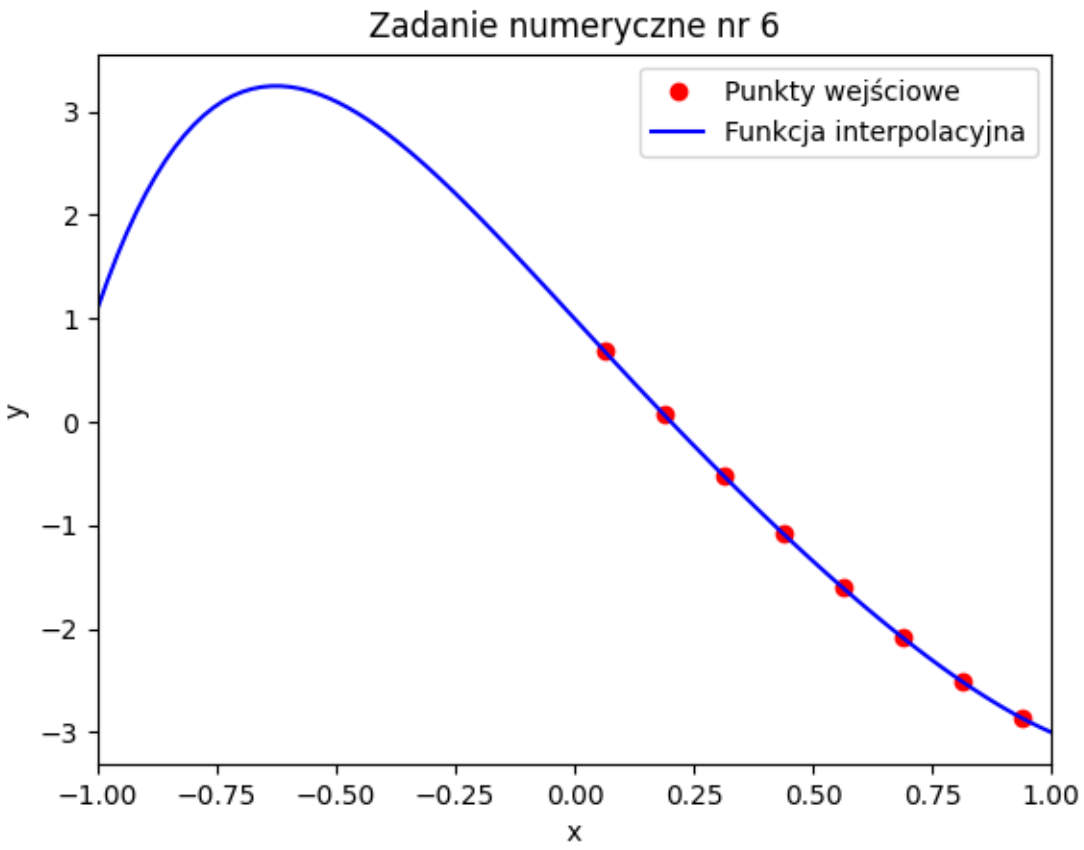
```
plt.plot(x, y, "or", label="Punkty wejściowe")
plt.plot(x1, y1, "b", label="Funkcja interpolacyjna")
plt.xlim(-1, 1)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Zadanie numeryczne nr 6")
plt.legend()
plt.savefig("plot.png")
```

Wyniki

Współczynniki

```
a = 1.0000
b = -5.0003
c = 0.0024
d = 1.9892
e = -1.9743
f = 0.9664
g = 0.0228
h = -0.0062
```

Wykres



Rysunek 1: Funkcja interpolacyjna a węzły interpolacji