Laboratorium 3 - Interpolacja

Piotr Karamon

25.03.2024r.

Treści zadań

Zadania

1. Dane są trzy węzły interpolacji $(-1, 2.4)$, $(1, 1.8)$, $(2, 4.5)$, proszę obliczyć wielomian interpolacyjny 2-go stopnia, wykorzystując:

   (a) jednomiany

   (b) wielomiany Lagrange'a

   (c) wielomiany wg wzoru Newton'a

   Pokazać że trzy reprezentacje dają ten sam wielomian

2. Wyrazić następujący wielomian metodą Hornera $p(t) = 3t^3 - 7t^2 + 5t - 4$

3. Ile mnożeń trzeba wykonać do ewaluacji wielomianu $p(t)$ stopnia $n - 1$ w danym punkcie t jeżeli wybieramy jako reprezentacje:

   (a) jednomiany

   (b) wielomiany Lagrange'a

   (c) wielomiany Newtona

Zadania domowe

1. Znaleźć kompromis między granicą błędu a zachowaniem wielomianu interpolacyjnego dla funkcji Rungego $f(t) = 1/(1 + 25t^2)$, dla równoodległych węzłów na przedziale $[-1, 1]$

2. (a) sprawdzić czy pierwsze siedem wielomianów Legendre'a są wzajemnie ortogonalne

   (b) sprawdzić czy one spełniają wzór na rekurencję

   (c) wyrazić każdy z siedmiu pierwszych jednomianów 1, t, $\ldots$, t^6 jako liniową kombinację pierwszych siedmiu wielomianów Legendre'a, p_0, $\ldots$, p_6

3. Dana jest funkcja określona w trzech punktach x_0, x_1, x_2, rozmieszczonych w jednakowych odstępach $(x_1 = x_0 + h, x_2 = x_1 + h)$:

   $f(x0) = y0, \quad f(x1) = y1, \quad f(x2) = y2$

   Proszę wykonać interpolację danej funkcji sklejanymi funkcjami sześciennymi.

Zadanie 1.

Jednomiany(metoda algebraiczna)

$$f(x) = ax^2 + bx + c$$

Tworzymy układ równań:

$$\begin{aligned} ax_0^2 + bx_0 + c &= y_0 \\ ax_1^2 + bx_1 + c &= y_1 \\ ax_2^2 + bx_2 + c &= y_2 \end{aligned}$$

Następnie podstawiamy dane z węzłów interpolacji:

$$\begin{aligned} a(-1)^2 + b(-1) + c &= 2.4 \\ a(1)^2 + b(1) + c &= 1.8 \\ 4a + 2b + c &= 4.5 \end{aligned}$$

Układ równań rozwiążemy przy użyciu języka Python i biblioteki numpy.

```python
import numpy as np
A = np.array([[ 1, -1, 1],
              [ 1,  1, 1],
              [ 4,  2, 1]])
b = np.array([2.4, 1.8, 4.5])

solution = np.linalg.solve(A, b)
print(solution)
```

```
[ 1.  -0.3  1.1]
```

Czyli: $P(x) = x^2 - 0.3x + 1.1$

Wielomiany Lagrange'a

$$L_k(x) = \prod_{i=0, i\neq k}^{n} \left(\frac{x - x_i}{x_k - xi} \right)$$

$$P_n(x) = \sum_{k=0}^{n} f(x_k) L_k(x)$$

$$L_0 = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$L_1 = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$L_2 = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

$$P_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2$$
$$P_2(x) = \frac{(x-1)(x-2)}{(-1-1)(-1-2)}2.4 + \frac{(x+1)(x-2)}{(1+1)(1-2)}1.8 + \frac{(x+1)(x-1)}{(2+1)(2-1)}4.5$$
$$P_2(x) = \frac{(x-1)(x-2)}{6}2.4 + \frac{(x+1)(x-2)}{-2}1.8 + \frac{(x+1)(x-1)}{3}4.5$$
$$P_2(x) = \left(\frac{1}{6}\cdot 2.4\cdot(x^2-3x+2)\right) - \left(\frac{1}{2}\cdot 1.8\cdot(x^2-x-2)\right) + \left(\frac{1}{3}\cdot 4.5\cdot(x^2-1)\right)$$
$$P_2(x) = (0.4x^2 - 1.2x + 0.8) - (0.9x^2 - 0.9x - 1.8) + (1.5x^2 - 1.5)$$
$$P_2(x) = 1.0x^2 - 0.3x + 1.1$$

Tak samo jak w ostatniej metodzie otrzymujemy $P(x) = x^2 - 0.3x + 1.1$.

Wielomiany wg wzoru Newton'a

Wielomian interpolacyjny będzie miał wzór:

$$P(x) = f[x_0] + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x1,x2]$$

1. Wypisujemy wzory na ilorazy różnicowe

$$f[x_0] = f(x_0)$$
$$f[x_1] = f(x_1)$$
$$f[x_2] = f(x_2)$$
$$f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0}$$
$$f[x_0,x_1,x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$$

2. Podstawiamy węzły interpolacji:

$$f[x_0] = f(-1) = 2.4$$
$$f[x_1] = f(1) = 1.8$$
$$f[x_2] = f(2) = 4.5$$
$$f[x_0,x_1] = \frac{f[x_1]-f[x_0]}{x_1-x_0} = \frac{1.8-2.4}{1-(-1)} = \frac{-0.6}{2} = -0.3$$
$$f[x_1,x_2] = \frac{f[x_2]-f[x_1]}{x_2-x_1} = \frac{4.5-1.8}{2-1} = \frac{2.7}{1} = 2.7$$
$$f[x_0,x_1,x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} = \frac{2.7-(-0.3)}{2-(-1)} = \frac{3.0}{3} = 1.0$$

3. Ilorazy różnicowe podstawiamy do wzoru na $P(x)$

$$\begin{aligned} P(x) &= f[x_0] + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] \\ &= 2.4 + (x - (-1))(-0.3) + (x - (-1))(x - 1)(1.0) \\ &= 2.4 - 0.3(x + 1) + 1.0(x + 1)(x - 1) \\ &= 2.4 - 0.3x - 0.3 + 1.0(x^2 - 1) \\ &= 2.1 - 0.3x + x^2 - 1 \\ &= x^2 - 0.3x + 1.1 \end{aligned}$$

Ponownie otrzymaliśmy ten sam wielomian $P(x) = x^2 - 0.3x + 1.1$

Wnioski

Każda z tych metod, pomimo różnych podejść i technik obliczeniowych, doprowadziła do identycznego wyniku, co potwierdza empirycznie równoważność i skuteczność tych technik interpolacji.

Zadanie 2

$$\begin{aligned} p(t) &= 3t^3 - 7t^2 + 5t - 4 \\ &= t(3t^2 - 7t + 5) - 4 \\ &= t(t(3t - 7) + 5) - 4 \end{aligned}$$

Zadanie 3

Wielomian $p(t)$ stopnia $n-1$ interesuje nas liczba mnożeń potrzebna by obliczyć wartość tego wielomianu w zależności od reprezentacji:

1. **Jednomiany** - można użyć algorytmu Hornera, który wymaga $n - 1$ mnożeń.
2. **Wielomiany Lagrange'a** - Aby obliczyć L_k potrzebujemy $n-1$ mnożeń. Wyrazów jest n. Każde L_k również mnożymy przez odpowiadający y_k. Sumarycznie dostajemy: $(n-1) \cdot n + n = n^2 - n + n = n^2$ mnożeń. Trzeba zaznaczyć, że w obliczenia nie są wliczone mnożenia potrzebne do obliczenia mianowników L_k ponieważ są one stałe, zatem wystarczy jest obliczyć raz, a nie dla każdej innej wartości t.
3. **Wielomiany wg wzoru Newton'a** - Możemy zastosować algorytm Hornera dla postaci Newtona. Dzięki temu potrzebne będzie tylko $n-1$ mnożeń. Podobnie jak przy wielomianach Lagrange'a trzeba zaznaczyć, że w ilość mnożeń nie są wliczone mnożenia potrzebne by obliczyć ilorazy różnicowe. Z racji tego że jest to wykonywane tylko raz a nie dla każdej innej wartości t.

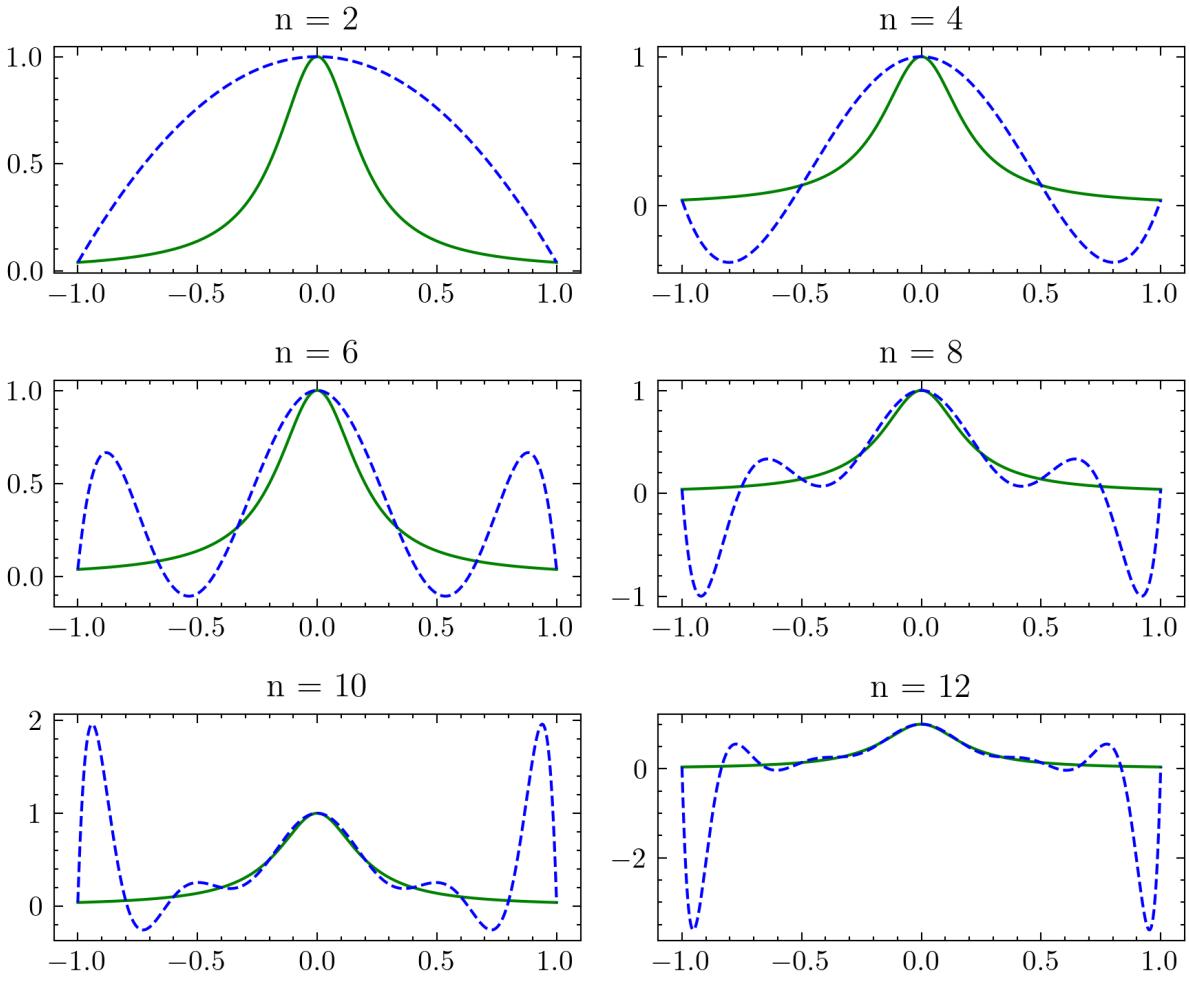
Wnioski: Metoda Hornera, stosowana dla jednomianów i wielomianów Newtona, okazuje się najbardziej efektywna obliczeniowo, wymagając tylko $n - 1$ mnożeń. W porównaniu, wielomiany Lagrange'a wymagają znacznie więcej operacji mnożenia, co czyni je mniej efektywnymi dla wielomianów wysokiego stopnia.

Zadanie domowe 1

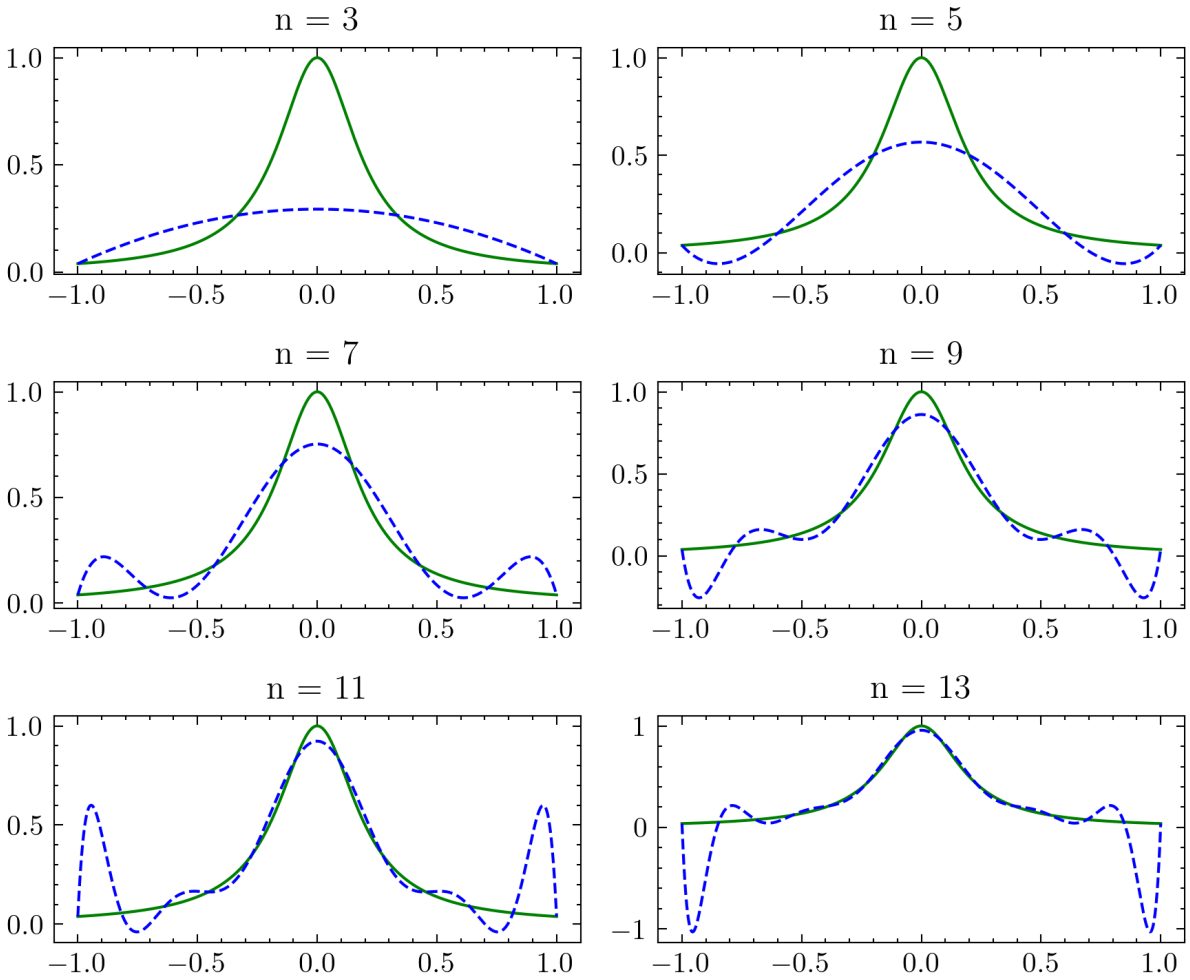
Funkcja Rungego:

$$f(t) = \frac{1}{1 + 25t^2},\ t \in [-1, 1]$$

Zielonym kolorem zaznaczona jest funkcja Rungego, przerywaną linią natomiast zaznaczony jest wielomian interpolacyjny o stopniu n.



Rysunek 1: Wielomiany interpolacyjne stopnia parzystego dla funkcji Rungego.



Rysunek 2: Wielomiany interpolacyjne stopnia nieparzystego dla funkcji Rungego.

Wnioski: Widzimy, że dla wielomianów interpolacyjnych o dużych stopniach różnica między wielomianem a funkcją Rungego jest szczególnie widoczna w okolicy -1 i 1. Dobrym kompromisem jest wielomian stopnia 9. W tej sytuacji jednakże powinniśmy rozważyć specjalny dobór węzłów lub inne typy interpolacji(np. interpolacja funkcjami sześciennymi).

Zadanie domowe 2

a) Ortogonalność

Wielomiany Legendre'a to wielomiany określone wzorem:

$$P_n = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n \quad (n = 0, 1, \ldots).$$

Pierwsze siedem wielomianów Legendre'a:

$$
\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \frac{1}{2}(3x^2 - 1) \\
P_3(x) &= \frac{1}{2}(5x^3 - 3x) \\
P_4(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3) \\
P_5(x) &= \frac{1}{8}(63x^5 - 70x^3 + 15x) \\
P_6(x) &= \frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5)
\end{aligned}
$$

Wielomiany są wzajemnie ortogonalne gdy:

$$\int_{-1}^{1} P_m(x)P_n(x)dx = 0, \quad \text{dla} \quad m \neq n$$

Całki zostaną policzone symbolicznie używając biblioteki SymPy. Najpierw definiujemy potrzebne wielomiany:

```
import sympy as sp
x = sp.symbols('x')

# pobrane z tablic
P_0 = 1
P_1 = x
P_2 = (1/2)*(3*x**2 - 1)
P_3 = (1/2)*(5*x**3 - 3*x)
P_4 = (1/8)*(35*x**4 - 30*x**2 + 3)
P_5 = (1/8)*(63*x**5 - 70*x**3 + 15*x)
P_6 = (1/16)*(231*x**6 - 315*x**4 + 105*x**2 - 5)
```

Potem liczymy całkę dla każdej pary wielomianów:

```
legendre_polynomials = [P_0, P_1, P_2, P_3, P_4, P_5, P_6]
for i, p1 in enumerate(legendre_polynomials):
    for j, p2 in enumerate(legendre_polynomials[:i]):
        integral = sp.integrate(p1*p2, (x, -1, 1))
        print(f'integral for p_{i} and p_{j} = {integral}')
```

```
integral for p_1 and p_0 = 0
integral for p_2 and p_0 = 0
integral for p_2 and p_1 = 0
integral for p_3 and p_0 = 0
integral for p_3 and p_1 = 0
integral for p_3 and p_2 = 0
integral for p_4 and p_0 = 0
integral for p_4 and p_1 = 0
integral for p_4 and p_2 = 0
integral for p_4 and p_3 = 0
integral for p_5 and p_0 = 0
integral for p_5 and p_1 = 0
integral for p_5 and p_2 = 0
integral for p_5 and p_3 = 0
integral for p_5 and p_4 = 0
integral for p_6 and p_0 = 0
integral for p_6 and p_1 = 0
integral for p_6 and p_2 = 0
integral for p_6 and p_3 = 0
integral for p_6 and p_4 = 0
integral for p_6 and p_5 = 0
```

Jak widać wszystkie całki postaci $\int_{-1}^{1} P_m(x)P_n(x)dx$ wynoszą zero dla pierwszych siedmiu wielomianów Legendre'a.

Wzór rekurencyjny

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ (n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x) \end{aligned}$$

W celu weryfikacji tej rekurencyjnej zależności skorzystamy ponownie z biblioteki SymPy.

```python
from sympy import Rational

def legendre(n):
    match n:
        case 0: return 1
        case 1: return x
        case _:
            return sp.expand(Rational(1, n)*((2*n-1)*x*legendre(n-1) - (n-1)*legendre(n-2)))

for i in range(0, 7):
    print(f"legendre({i}) = {legendre(i)}, diff = {sp.simplify(legendre(i) -legendre_polynomials[i])}")
```

```
legendre(0) = 1, diff = 0
legendre(1) = x, diff = 0
legendre(2) = 3*x**2/2 - 1/2, diff = 0
legendre(3) = 5*x**3/2 - 3*x/2, diff = 0
legendre(4) = 35*x**4/8 - 15*x**2/4 + 3/8, diff = 0
legendre(5) = 63*x**5/8 - 35*x**3/4 + 15*x/8, diff = 0
legendre(6) = 231*x**6/16 - 315*x**4/16 + 105*x**2/16 - 5/16, diff = 0
```

Jak widać wielomiany Legendre'a wyliczone ze wzoru rekurencyjnego zgadzają się z wielomianami pobranymi z tablic, o czym świadczy `diff=0` w każdym wierszu wyniku.

Liniowe kombinacje

Chcą wyrazić każdy z siedmiu pierwszych jednomianów jako liniową kombinację pierwszych siedmiu wielomianów Legendre'a musimy stworzyć układy równań. Układy te pozyskamy z równań postaci

$$x^i = p_0 \cdot P_0(x) + p_1 \cdot P_1(x) + p_2 \cdot P_2(x) + p_3 \cdot P_3(x) + p_4 \cdot P_4(x) + p_5 \cdot P_5(x) + p_6 \cdot P_6(x)$$

. Musimy podstawić wzory na wielomiany Legendre'a a następnie wyznaczyć współczynniki $p_0, ..., p_6$.

Do tego celu wykorzystamy bibliotekę SymPy.

```python
p_0, p_1, p_2, p_3, p_4, p_5, p_6 = sp.symbols('p_0, p_1, p_2, p_3, p_4, p_5, p_6')

for i in range(0, 7):
    equation = sp.Eq(x**i, sp.expand(p_0 *P_0 + p_1 *P_1 + p_2 *P_2 + p_3 *P_3 + p_4 *P_4 + p_5 *P_5
    ↪  +p_6*P_6))
    coeffs_eq = sp.collect(equation.lhs - equation.rhs, x, evaluate=False)
    eqs = [sp.Eq(coeffs_eq.get(x**i, 0), 0) for i in range(7)]
    solutions = sp.solve(eqs, (p_0, p_1, p_2, p_3, p_4, p_5, p_6), rational=True)
    print(f'x**{i}', solutions)
```

```
x**0 p_0: 1, p_1: 0, p_2: 0, p_3: 0, p_4: 0, p_5: 0, p_6: 0
x**1 p_0: 0, p_1: 1, p_2: 0, p_3: 0, p_4: 0, p_5: 0, p_6: 0
x**2 p_0: 1/3, p_1: 0, p_2: 2/3, p_3: 0, p_4: 0, p_5: 0, p_6: 0
x**3 p_0: 0, p_1: 3/5, p_2: 0, p_3: 2/5, p_4: 0, p_5: 0, p_6: 0
x**4 p_0: 1/5, p_1: 0, p_2: 4/7, p_3: 0, p_4: 8/35, p_5: 0, p_6: 0
x**5 p_0: 0, p_1: 3/7, p_2: 0, p_3: 4/9, p_4: 0, p_5: 8/63, p_6: 0
x**6 p_0: 1/7, p_1: 0, p_2: 10/21, p_3: 0, p_4: 24/77, p_5: 0, p_6: 16/231
```

Podsumowując:

$$
\begin{aligned}
1 &= 1 \cdot P_0 \\
x &= 1 \cdot P_1 \\
x^2 &= \frac{1}{3} \cdot P_0 + \frac{2}{3} \cdot P_2 \\
x^3 &= \frac{3}{5} \cdot P_1 + \frac{2}{5} \cdot P_3 \\
x^4 &= \frac{1}{5} \cdot P_0 + \frac{4}{7} \cdot P_2 + \frac{8}{35} \cdot P_4 \\
x^5 &= \frac{3}{7} \cdot P_1 + \frac{4}{9} \cdot P_3 + \frac{8}{63} \cdot P_5 \\
x^6 &= \frac{1}{7} \cdot P_0 + \frac{10}{21} \cdot P_2 + \frac{24}{77} \cdot P_4 + \frac{16}{231} \cdot P_6
\end{aligned}
$$

Zadanie domowe 3

Mamy funkcję określoną w trzech punktach x_0, x_1, x_2 takich, że $x_1 = x_0 + h$ i $x_2 = x_1 + h$: $f(x_0) = y_0$, $f(x_1) = y_1$, $f(x_2) = y_2$

Jako że mamy 3 punkty to musimy znaleźć dwie funkcje sześcienne:

- $S_0(x)$ na odcinku $[x_0, x_1]$
- $S_1(x)$ na odcinku $[x_1, x_2]$.

By wyznaczyć S_0 i S_1 potrzebujemy warunków. Wiemy, że funkcje S_0 powinny przechodzić przez węzły interpolacji, oraz chcemy by nasza funkcja była ciągła, zatem:

$$
\begin{aligned}
S_0(x_0) &= y_0 \\
S_0(x_1) &= y_1 \\
S_1(x_1) &= y_1 \\
S_2(x_2) &= y_2
\end{aligned}
$$

Poza tym chcemy aby nasza funkcja była "gładka", a zatem żądamy by pierwsza i druga pochodna S_0 i S_1 były sobie równe w węzłach "wewnętrznych"w naszym przypadku jest to tylko węzeł x_1. Zatem:

$$
\begin{aligned}
S_0'(x_1) &= S_1'(x_1) \\
S_0''(x_1) &= S_1''(x_1)
\end{aligned}
$$

To daje nam łącznie 6 równań, natomiast mamy 8 niewiadomych. Wybór ostatnich dwóch równań jest zależny od tego co chcemy osiągnąć i co opisuje funkcja która interpolujemy. Z racji na abstrakcyjny charakter tego zadania wybiorę opcję najprostszą czyli, zażądam by druga pochodna była równa zero w punktach brzegowych, czyli:

$$
\begin{aligned}
S_0''(x_0) &= 0 \\
S_1''(x_2) &= 0
\end{aligned}
$$

Pierwsze cztery równania mają postać:

$$
\begin{aligned}
a_0(x_0)^3 + b_0(x_0)^2 + c_0(x_0) + d_0 &= y_0 \\
a_0(x_1)^3 + b_0(x_1)^2 + c_0(x_1) + d_0 &= y_1 \\
a_1(x_1)^3 + b_1(x_1)^2 + c_1(x_1) + d_1 &= y_1 \\
a_1(x_2)^3 + b_1(x_2)^2 + c_1(x_2) + d_1 &= y_2
\end{aligned}
$$

Kolejne związane z pochodnymi:

$$
\begin{aligned}
3a_0(x_1)^2 + 2b_0(x_1) + c_0 &= 3a_1(x_1)^2 + 2b_1(x_1) + c_1 \\
6a_0(x_1) + 2b_0 &= 6a_1(x_1) + 2b_1
\end{aligned}
$$

I warunki brzegowe:

$$
\begin{aligned}
6a_0(x - x_0) + 2b_0 &= 0 \\
6a_1(x - x_1) + 2b_1 &= 0
\end{aligned}
$$

Współczynniki zostaną policzone przy użyciu biblioteki SymPy.

```
import sympy as sp

a0, b0, c0, d0, a1, b1, c1, d1, x0, h, x = sp.symbols('a0 b0 c0 d0 a1 b1 c1 d1 x0 h x')
y0, y1, y2 = sp.symbols('y0 y1 y2')

S_0 = a0*x**3 + b0*x**2 +c0*x + d0
S_1 = a1*x**3 + b1*x**2 +c1*x + d1

solution = sp.solve((
    sp.Eq(S_0.subs(x,x0), y0),
    sp.Eq(S_0.subs(x,x0+h), y1),
    sp.Eq(S_1.subs(x,x0+h), y1),
    sp.Eq(S_1.subs(x,x0+2*h), y2),
    sp.Eq(sp.diff(S_0, x).subs(x,x0+h), sp.diff(S_1, x).subs(x, x0+h)),
    sp.Eq(sp.diff(S_0, x, x).subs(x,x0+h), sp.diff(S_1, x, x).subs(x, x0+h)),
    sp.Eq(sp.diff(S_0, x, x).subs(x,x0), 0),
    sp.Eq(sp.diff(S_1, x, x).subs(x,x0+2*h), 0),
), (a0, b0, c0, d0, a1, b1, c1, d1))


latex_str = "\\begin{align*}\n"
for key, value in solution.items():
    latex_str += f"  {sp.latex(key)} &= {sp.latex(value)} \\\\ \n"
latex_str += "\\end{align*}"

print(latex_str)
```

$$a_0 = \frac{y_0 - 2y_1 + y_2}{4h^3}$$
$$a_1 = \frac{-y_0 + 2y_1 - y_2}{4h^3}$$
$$b_0 = \frac{-3x_0y_0 + 6x_0y_1 - 3x_0y_2}{4h^3}$$
$$b_1 = \frac{6hy_0 - 12hy_1 + 6hy_2 + 3x_0y_0 - 6x_0y_1 + 3x_0y_2}{4h^3}$$
$$c_0 = \frac{-5h^2y_0 + 6h^2y_1 - h^2y_2 + 3x_0^2y_0 - 6x_0^2y_1 + 3x_0^2y_2}{4h^3}$$
$$c_1 = \frac{-11h^2y_0 + 18h^2y_1 - 7h^2y_2 - 12hx_0y_0 + 24hx_0y_1 - 12hx_0y_2 - 3x_0^2y_0 + 6x_0^2y_1 - 3x_0^2y_2}{4h^3}$$
$$d_0 = \frac{4h^3y_0 + 5h^2x_0y_0 - 6h^2x_0y_1 + h^2x_0y_2 - x_0^3y_0 + 2x_0^3y_1 - x_0^3y_2}{4h^3}$$
$$d_1 = \frac{6h^3y_0 - 4h^3y_1 + 2h^3y_2 + 11h^2x_0y_0 - 18h^2x_0y_1 + 7h^2x_0y_2 + 6hx_0^2y_0 - 12hx_0^2y_1 + 6hx_0^2y_2 + x_0^3y_0 - 2x_0^3y_1 + x_0^3y_2}{4h^3}$$

Obliczenia zostały wykonane symbolicznie z racji na treść zadania.

Wnioski: Wzory opisujące interpolację funkcjami sześciennymi potrafią być bardzo skomplikowane. Nie należy je w taki sposób wyprowadzać, ponieważ jest to niepraktyczne. Należy zebrać węzły interpolacji i stworzyć układ równań który można obliczyć już nie symbolicznie a numerycznie.

Bibliografia

- Marian Bubak, Katarzyna Rycerz: *Metody Obliczeniowe w Nauce i Technice. Interpolacja*
- Marian Bubak, Katarzyna Rycerz: *Metody Obliczeniowe w Nauce i Technice. Funkcja sklejane - spline functions*