Interpolacja za pomocą węzłów Czebyszewa

Michael Tryfanau

12 lutego 2025

Spis treści

1 Cel projektu

Celem projektu było badanie interpolacji z używaniem różnie wybranych węzłów:

- Węzłów równie odległych od siebie na wybranym przedziale
- Węzłów Czebyszewa

i porównanie tych dwóch algorytmów.

2 Podstawowe dane, metody i zasoby

2.1 Środki techniczne

Obliczenia wykonywano na własnym laptopie, gdzie w IDE **Visual Studio Code** napisano program w języku `Python`. Użyto bibliotek: `math` oraz `matplotlib`, `numpy`. Dokument ze sprawozdaniem o projekcie sporządzono za pomocą LaTeX.

2.2 Dane o wybranej funkcji i przedziale

Interpolowana funkcja:

$$f(x) = sin(\frac{1}{x^2 + e^x}) \tag{1}$$

Dla projektu użyto interpolacji **Newtona**.
Parametry przedziału:

$$a = -5{,}3,\ b = 4{,}7,\ n = 10$$

czyli 11 węzłów.

3 Przebieg eksperymentu

3.1 Hipoteza

Autor przypuszcza, że za pomocą węzłów Czebyszewa uda się otrzymać wykres, będący znacznie bliższym do tego funkcji 1 w porównaniu do wykresu wielomianu interpolacyjnego otrzymanego za użyciem węzłów równooddalonych. W dodatku błąd interpolacji będzie widocznie mniejszy.

3.2 Węzły interpolacji

3.2.1 Teoria

Według wykładu generujemy węzły Czebyszewa w następujący sposób:

$$x_i^* = \frac{b-a}{2} \cdot cos(\frac{2i+1}{2n+1}\pi) + \frac{a+b}{2}, \ 0 \leq i \leq n \tag{2}$$

3.2.2 Zastosowanie teorii

Z tą wiedzą generujemy węzły interpolacji za pomocą następnej funkcji:

Listing 1: Funkcja generacji węzłów

```
def generate_xes(a, b, n):
    #vars for the function graph
    x=np.linspace(a,b,1000)
    #classic interpolation nodes
    x_int=(np.linspace(a,b,n+1))
    #Chebyshev nodes
    x_ch=[((b-a)/2)*cos(pi*(2*i+1)/(2*n+2))+((a+b)/2) for i in
        range(n+1)]
    res = { "function": x, "interpolation": x_int, "chebyshev":
        x_ch }
    return res
```

3.2.3 Tabela

Poprzez wywoływanie tej funkcji otrzymano następujące węzły interpolacji:

i	Węzeł równooddalony	Węzeł Czebyszewa
0	$-5{,}3$	4,649107209404663
1	$-4{,}3$	4,248159976772592
2	$-3{,}3$	3,4787478717712914
3	$-2{,}3$	2,4032040872779885
4	$-1{,}3$	1,108662784207149
5	$-0{,}3$	$-0{,}2999999999999984$
6	0,7	$-1{,}7086627842071482$
7	1,7	$-3{,}003204087277986$
8	2,7	$-4{,}078747871771291$
9	3,7	$-4{,}848159976772591$
10	4,7	$-5{,}249107209404663$

Tabela 1: Tabela węzłów interpolacji

3.3 Przebieg interpolacji

3.3.1 Teoria

Wielomian interpolacyjny Newtona wygląda następująco:

$$W_n(x) = c_{0,0} + c_{0,1}(x - x_0) + c_{0,2}(x - x_0)(x - x_1) + \cdots + \\ + c_{0,n}(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \quad (3)$$

Gdzie współczynniki $c_{0,0}, c_{0,1}, \ldots, c_{0,n}$ są wyznaczane jako:

$$c_{i,j} = \begin{cases} j = 0 & \Rightarrow y_i \\ 1 \leq j \leq n & \Rightarrow \frac{c_{i+1,j-1} - c_{i,j-1}}{x_{i+j} - x_i} \end{cases} \quad (4)$$

Użyjemy równań 3 i 4 w następującej części.

3.3.2 Zastosowanie teorii

A interpolujemy my za pomocą funkcji:

Listing 2: Funkcja interpolacji Newtona

```
def interpolate(x, y):
    newton_table = [[y[len(y)-i]]+[0]*(i-1) for i in
        range(len(y), 0, -1)]
    for i in range(len(y)-2, -1, -1):
        for j in range(1, len(newton_table[i])):
            newton_table[i][j]=(newton_table[i+1][j-1]-
                newton_table[i][j-1])/(x[i+j]-x[i])
    return [lambda z:
        newton_table[0][0]+sum([newton_table[0][i]*prod([(z-x[j])
        for j in range(i)]) for i in range(1,
        len(newton_table[0]))]), newton_table[0]]
```

Ta funkcja zwraca gotowy wielomian interpolacyjny (poprzez lambdę), za pomocą którego można wstawić węzły do funkcji pomocniczej, która stwarza dla dowolnej funkcji F i zbioru X jego obraz, ale też i współczynniki $c_{0,0}, c_{0,1}, \ldots, c_{0,n}$, których wartości są przesyłane do pliku.

3.3.3 Wielomiany interpolacyjne

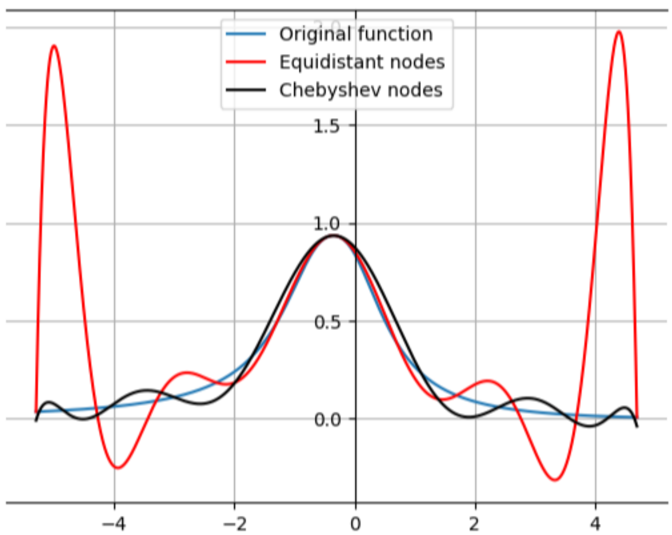
Za pomocą wzorów 3 oraz 4 możemy uzyskać wielomian interpolacyjny, używając tych współczynników:

$c_{0,i}$	Współczynniki na węzłach r.o.	Współczynniki na węzłach Czebyszewa
0	0,035586017532897744	0,007929762965125289
1	0,018431307949521192	−0,00855625888676866
2	0,009470538039314861	0,005014907212819489
3	0,006125719371200611	−0,002567607805399241
4	0,004908579157430042	0,0020823868829243477
5	−0,002836472496128449	−0,00022539177128791
6	−0,0006963035621500522	−0,0008218729724849621
7	0,0009532899678599745	−0,00038410634637789934
8	−0,0004045393494760657	−0,00010989126129072859
9	0,00011033517688382007	$-2{,}459359655731805 \times 10^{-5}$
10	$-2{,}2623531398049946 \times 10^{-5}$	$-4{,}867662770044368 \times 10^{-6}$

Tabela 2: Tabela współczynników wielomianów interpolacji Newtona dla funkcji $f(x) = sin(\frac{1}{x^2+e^x})$

3.3.4 Wykres

Wykres funkcji $f(x) = sin(\frac{1}{x^2+e^x})$ oraz wielomianów interpolacji Newtona za pomocą węzłów równooddalonych i Czebyszewa na przedziale $[-5{,}3, 4{,}7]$:



Rysunek 1: Wykresy funkcji $f(x)$ oraz wielomianów interpolacyjnych

3.3.5 Błąd interpolacji

Obliczenie Obliczamy błąd bezwzględny interpolacji za pomocą następującej funkcji:

Listing 3: Funkcja obliczenia błędów interpolacji

```
def compute_error(f_true, f_int):
    x_test = np.linspace(-5.3,4.7,15)
    f_true = test_function(x_test, f_true)
    f_int = test_function(x_test, f_int)
    return (np.abs(np.array(f_true) - np.array(f_int)))
```

Wartości błędów Na początku obliczmy błędy na 15 równooddalonych punktach:

x	Błędy interpolacji z węzłami r. o.	Błędy interpolacji z węzłami Czebyszewa
$-5{,}3$	$0{,}00000000$	$4{,}38220520 \times 10^{-2}$
$-4{,}58571429$	$7{,}63106706 \times 10^{-1}$	$4{,}75465723 \times 10^{-2}$
$-3{,}87142857$	$3{,}09823489 \times 10^{-1}$	$3{,}66211457 \times 10^{-2}$
$-3{,}15714286$	$6{,}30026280 \times 10^{-2}$	$2{,}77583611 \times 10^{-2}$
$-2{,}44285714$	$3{,}74659227 \times 10^{-2}$	$8{,}37210537 \times 10^{-2}$
$-1{,}72857143$	$5{,}90027507 \times 10^{-2}$	$4{,}52634569 \times 10^{-3}$
$-1{,}01428571$	$2{,}35557369 \times 10^{-2}$	$7{,}54338907 \times 10^{-2}$
$-0{,}3$	$2{,}22044605 \times 10^{-16} \approx 0$	$2{,}22044605 \times 10^{-16} \approx 0$
$0{,}41428571$	$3{,}90820161 \times 10^{-2}$	$1{,}15443679 \times 10^{-1}$
$1{,}12857143$	$7{,}26110104 \times 10^{-2}$	$5{,}39733204 \times 10^{-3}$
$1{,}84285714$	$4{,}25305435 \times 10^{-2}$	$9{,}31967385 \times 10^{-2}$
$2{,}55714286$	$6{,}90339701 \times 10^{-2}$	$2{,}99847274 \times 10^{-2}$
$3{,}27142857$	$3{,}32844140 \times 10^{-1}$	$3{,}89014916 \times 10^{-2}$
$3{,}98571429$	$8{,}09514610 \times 10^{-1}$	$4{,}99702517 \times 10^{-2}$
$4{,}7$	$4{,}69589645 \times 10^{-15} \approx 0$	$4{,}57148579 \times 10^{-2}$

Tabela 3: Tabela błędów interpolacji dla funkcji $f(x) = sin(\frac{1}{x^2+e^x})$

Sumując, dostajemy następujące wartości błędów bezwzględnych interpolacji:

$$\text{Dla węzłów równooddalonych: } R_{int} = 2{,}6215735246718834$$
$$\text{Dla węzłów Czebyszewa: } R_{cheb} = 0{,}6980384997037218$$

Tym samym $R_{cheb} < R_{int}$.

3.3.6 Interpretacja otrzymanych wyników

Hipoteza się sprawdziła. Wykresy, które dostaliśmy, są oczekiwane dla takiej procedury. Otrzymany błąd ogólny dla węzłów Czebyszewa okazał się znacznie mniejszy od tego dla węzłów równooddalonych.

4 Wnioski

Tym eksperymentem zbudowano wielomiany interpolacyjne Newtona dla węzłów równooddalonych i węzłów Czebyszewa. Zbadano działanie wybrania węzłów Czebyszewa na efektywność interpolacji wielomianowej Newtona.

Postawiona hipoteza się sprawdziła. Stwierdzono istnienie różnicy w błędach bezwzględnych oraz wykresach wielomianów interpolacyjnych, co potwierdza efektywność zastosowania węzłów Czebyszewa przy interpolacji wielomianowej Newtona.