Interpolacja za pomocą węzłów Czebyszewa

Michael Tryfanau

3 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:

$$\begin{split} W_n(x) = c_{0,0} + c_{0,1}(x - x_0) + c_{0,2}(x - x_0)(x - x_1) + \dots + \\ + c_{0,n}(x - x_0)(x - x_1)\dots(x - x_{n-1}) \end{split} \tag{3}$$

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

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

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

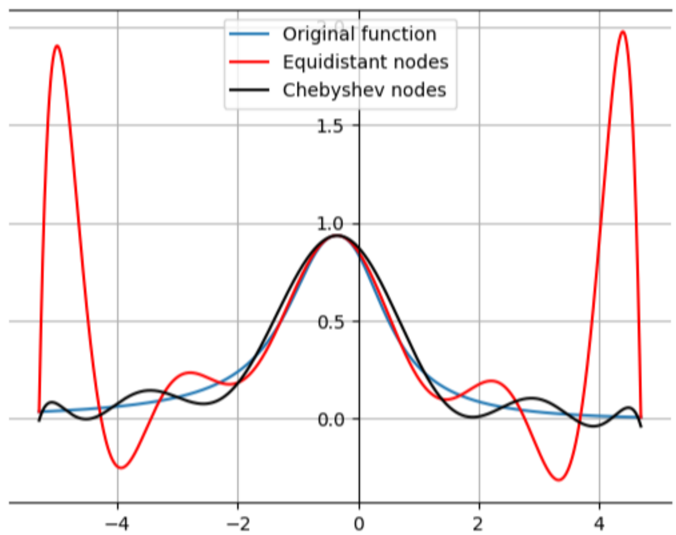
Za pomocą wzorów 3 oraz 4 możemy uzyskać wielomian interpolacyjny, używając tych współczynników:

$c_{0,i}$	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 równoodległych 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(x_test, f_true, f_int):
    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 węzłach interpolacji:

i	Błędy na węzłach równooddalonych	Błędy na węzłach Czebyszewa
0	0,00000000	0,00000000
1	0,00000000	0,00000000
2	0,00000000	0,00000000
3	$2{,}77555756 \times 10^{-17}$	0,00000000
4	$5{,}55111512 \times 10^{-17}$	$2{,}77555756 \times 10^{-17}$
5	$2{,}22044605 \times 10^{-16}$	$1{,}11022302 \times 10^{-16}$
6	$3{,}88578059 \times 10^{-16}$	$2{,}77555756 \times 10^{-16}$
7	$4{,}16333634 \times 10^{-16}$	$2{,}41473508 \times 10^{-15}$
8	$1{,}57512892 \times 10^{-15}$	$1{,}91513472 \times 10^{-15}$
9	$1{,}55084279 \times 10^{-14}$	$3{,}94823063 \times 10^{-14}$
10	$4{,}69589645 \times 10^{-15}$	$1{,}13103971 \times 10^{-15}$

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

Sumując, dostajemy następujące wartości błędów bezwzględnych interpolacji:

$$\text{Dla węzłów równooddalonych: } R_{int} = 2{,}288967626551397 \times 10^{-14}$$
$$\text{Dla węzłów Czebyszewa: } R_{cheb} = 4{,}535954944984155 \times 10^{-14}$$

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

3.3.6 Interpretacja otrzymanych wyników

Hipoteza się sprawdziła częściowo. Dostaliśmy coś nieprzewidzianego: gdy wykresy, które dostaliśmy, jest oczekiwany dla takiej procedury, otrzymany błąd ogólny dla węzłów Czebyszewa okazał się wyższy od tego dla węzłów równooddalonych.

4 Dodatkowe badania

4.1 Przygotowanie

Sprawdzimy działalność programu z innymi danymi wejściowymi. Nowa funckja $g(x)$:

$$g(x) = \frac{25}{1 + 3x^2} \tag{5}$$

Parametry przedziału:

$$a = -5,\ b = 5,\ n = 10$$

4.2 Przeprowadzenie badań

Badanie zostało przeprowadzone za użyciem tego samego programy Pythonowego, też za użyciem funkcji wskazanych w listingach w tym dokumencie.

4.3 Węzły interpolacji

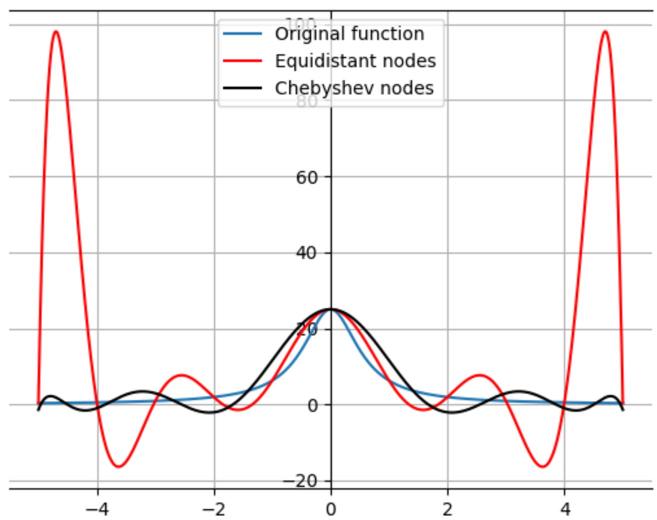
Nowe węzły interpolacji:

i	Węzeł równooddalony	Węzeł Czebyszewa
0	-5	4,949107209404663
1	-4	4,548159976772592
2	-3	3,7787478717712912
3	-2	2,7032040872779883
4	-1	1,4086627842071489
5	0	$1{,}4163847244119948 \times 10^{-15}$
6	1	$-1{,}4086627842071484$
7	2	$-2{,}703204087277986$
8	3	$-3{,}778747871771291$
9	4	$-4{,}548159976772591$
10	5	$-4{,}949107209404663$

Tabela 4: Tabela nowych węzłów interpolacji

4.4 Wyniki

Nowe wykresy:



Rysunek 2: Wykresy funkcji $g(x)$ oraz wielomianów interpolacyjnych

Nowe błędy na węzłach interpolacji:

i	Błędy na węzłach równooddalonych	Błędy na węzłach Czebyszewa
0	0,00000000	0,00000000
1	0,00000000	0,00000000
2	0,00000000	0,00000000
3	$2{,}22044605 \times 10^{-16}$	0,00000000
4	0,00000000	$4{,}44089210 \times 10^{-16}$
5	$3{,}55271368 \times 10^{-15}$	$3{,}55271368 \times 10^{-15}$
6	$1{,}24344979 \times 10^{-14}$	$7{,}99360578 \times 10^{-15}$
7	$1{,}64979141 \times 10^{-13}$	$1{,}99840144 \times 10^{-14}$
8	$4{,}34208225 \times 10^{-13}$	$3{,}03201908 \times 10^{-13}$
9	$8{,}05466804 \times 10^{-13}$	$3{,}01370040 \times 10^{-13}$
10	$2{,}72848411 \times 10^{-12}$	$1{,}25055521 \times 10^{-12}$

Tabela 5: Tabela błędów interpolacji dla funkcji $g(x) = \frac{25}{1+3x^2}$

Sumując, dostajemy następujące nowe wartości błędów bezwzględnych interpolacji:

$$\text{Dla węzłów równooddalonych: } R_{int} = 4{,}14934753223406 \times 10^{-12}$$
$$\text{Dla węzłów Czebyszewa: } R_{cheb} = 1{,}88710158610661 \times 10^{-12}$$

Koniecznie należy odznaczyć, że tym razem $R_{cheb} < R_{int}$.

4.5 Interpretacja otrzymanych wyników

Tym razem hipoteza się potwierdziła. Nie tylko wykres wielomianu interpolacyjnego, dostanego za pomocą węzłów Czebyszewa, jest bliższy do wykresu funkcji $g(x)$, ale błąd bezwzględny interpolacji też jest mniejszy. Tym samym, można stwierdzić, że poziom mocy efektu Rungego oraz efektywności węzłów Czebyszewa zależy od badanej funkcji.

5 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 tylko w części. Stwierdzono, że różnica w błędach bezwzględnych jest spowodowana właściwościami funkcji $f(x)$, bo uruchomienie programu z funkcją $g(x)$ prowadzi do potwierdzenia oryginalnej hipotezy.