Zagadnienie Hermite’a – Filip Żołnierczyk

Dane techniczne:

System operacyjny: windows 10

Procesor: AMD Ryzen 7 5800x

Ram: 32gb ddr4 3600mhz

Program napisany w języku python

Funkcja do interpolacji: f(x) = x*sin(kπ/x), k=2, [1/8, 0.4]

Opis zagadnienia:

Interpolacja Hermite’a to metoda numeryczna służąca do aproksymacji funkcji, która — w przeciwieństwie do klasycznej interpolacji wielomianowej (np. Lagrange’a) — bierze pod uwagę nie tylko wartości funkcji w zadanych punktach, ale również jej pochodne. Dzięki temu uzyskuje się dokładniejszą aproksymację, szczególnie wtedy, gdy dostępna jest dodatkowa informacja o zachowaniu funkcji (np. jej nachylenie) w punktach interpolacyjnych.

1. Postać Newtona dla interpolacji Hermite’a

$$H_{n(x)} = f[x_0] + (x - x_0)f[x_0,x_1] + (x - x_0)^2 f[x_0,x_0,x_1] + \dots \\ + (x - x_0)^k (x - x_1)^m \dots (x - x_n) f[x_0,x_0,\dots,x_n]$$

2. Dzielone różnice dla interpolacji Hermite’a

$$f[x_i,x_i] = f'(x_i)$$

$$f[x_i,x_i,x_j] = \backslash frac\{f[x_i,x_j] - f[x_i,x_i]\}\{x_j - x_i\}$$

$$f[x_i,x_j,x_k] = \backslash frac\{f[x_j,x_k] - f[x_i,x_j]\}\{x_k - x_i\}$$

$$f[x_0,x_1,\dots,x_n] = \backslash frac\{f[x_1,\dots,x_n] - f[x_0,\dots,x_{\{n-1\}}]\}\{x_n - x_0\}$$

3. Funkcje bazowe Hermite’a

$$H(x) = \backslash sum_{\{i=0\}_{i(x)}^{\{n\}f(x_i)h}} + \backslash sum_{\{i=0\}_{i(x)}^{\{n\}f'(x_i)k}}$$

Główne funkcje programu

- obliczanie współczynników różnic dzielonych uwzględniających zarówno wartości funkcji, jak i pochodne
- konstrukcja wielomianu interpolacyjnego w postaci Hermite’a, przy użyciu wielomianów bazowych
- analiza błędu aproksymacji poprzez obliczanie maksymalnej różnicy między wartościami funkcji a interpolacją oraz wykorzystanie estymatora wariancji

- generowanie wykresów przedstawiających porównanie funkcji oryginalnej i wielomianu interpolacyjnego

Opis funkcji :

def hermite_coefficients(x, f, df):

Funkcja oblicza współczynniki wielomianu Hermite’a na podstawie wartości funkcji i jej pochodnych w zadanych punktach. Tworzy rozszerzoną tablicę różnic dzielonych, która uwzględnia zarówno wartości funkcji, jak i pochodne.

def hermite_polynomial(x_values, coefficients, x):

Funkcja oblicza wartość wielomianu Hermite’a dla danego argumentu x. Wykorzystuje współczynniki wyznaczone wcześniej oraz wielomiany bazowe.

def max_error(f, hermite_f, x_range):

Funkcja oblicza maksymalny błąd interpolacji, czyli największą różnicę między wartością funkcji a jej interpolacją w zadanym zakresie punktów.

def variance_estimator(f, hermite_f, x_range):

Funkcja wyznacza średni błąd aproksymacji, stosując wzór podobny do estymatora wariancji. Sumuje kwadraty różnic między wartościami funkcji a interpolacją i normalizuje wynik.

def plot_results(x_values, f_values, hermite_values):

Funkcja generuje wykresy porównujące oryginalną funkcję z interpolacją Hermite’a, co pozwala wizualnie ocenić jakość aproksymacji.

Przebieg ćwiczenia:

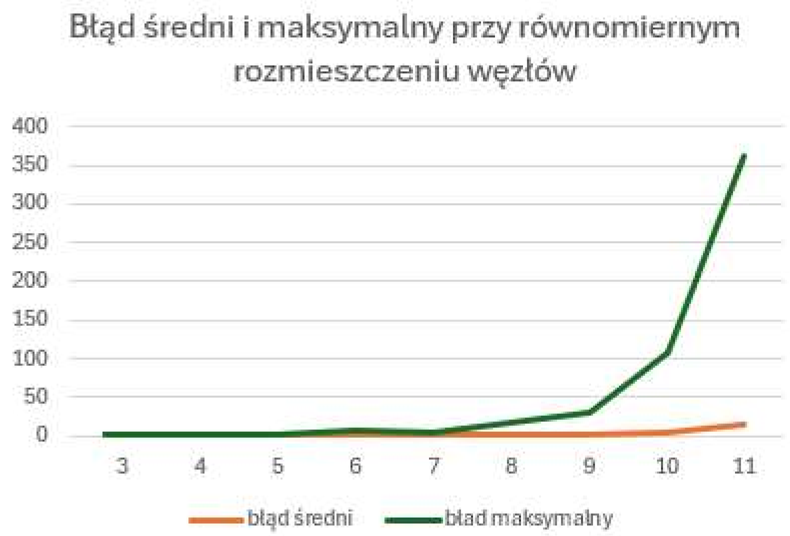
Użyłem programu do wykonywania obliczeń dla coraz kolejnych n zaczynając od 2 do momentu wystąpienia efektu Rungego. Poniżej znajdują się wartości n dla których błędy średnie i maksymalne były najmniejsze oraz wykresy ukazujące zmiany tych błędów wraz z zwiększającym się n na których możemy zaobserwować efekt Rungego.

Analiza najmniejszych błędów w interpolacji

1. Interpolacja z węzłami Czebyszewa
   Najmniejszy błąd średni występuje dla n = 17 i wynosi 0.104855.
   Najmniejszy błąd maksymalny przypada na n = 18 i wynosi 0.416979.
   Wartości błędów pozostają relatywnie niskie i stabilne dla n od około 7 do 18, co pokazuje, że metoda Czebyszewa skutecznie ogranicza wzrost błędu.

2. Interpolacja z równomiernym rozmieszczeniem węzłów

- Najmniejszy błąd średni występuje dla n = 7 i wynosi 0.206143076.
- Najmniejszy błąd maksymalny również przypada na n = 7 i wynosi 0.899538562.
- Dla większej liczby węzłów (np. n = 12, 13) obserwujemy gwałtowny wzrost błędów, co jest efektem Rungego.



Podsumowanie wykresów:

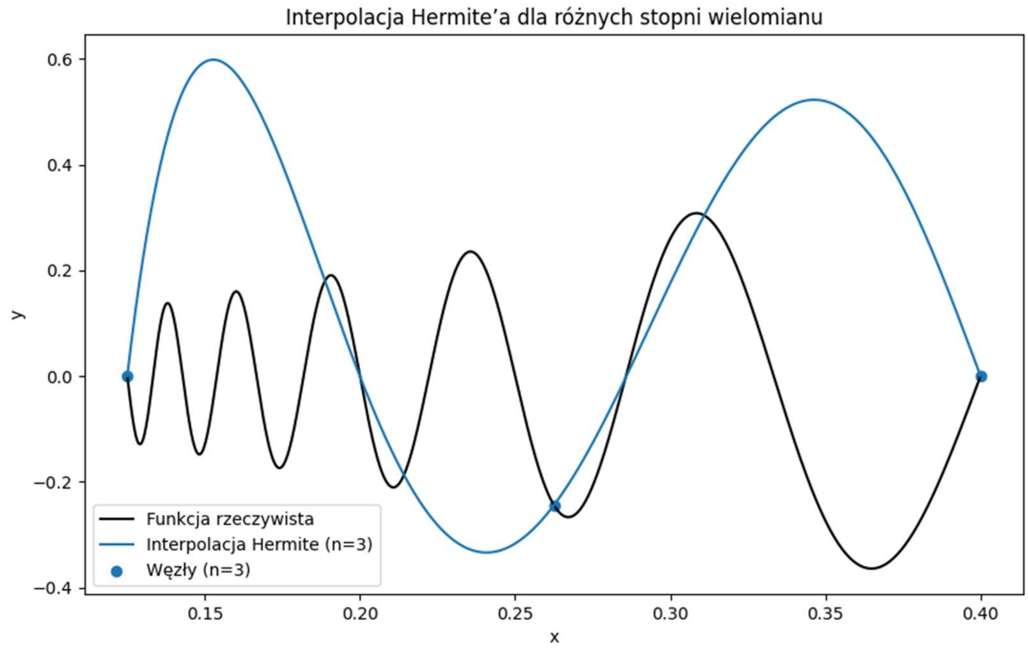
1. Wykres górny (Równomierne rozmieszczenie węzłów):
   - Przy równomiernym rozmieszczeniu węzłów błąd maksymalny gwałtownie rośnie dla większej liczby węzłów, szczególnie po 10 węzłach.
   - Wskazuje to na efekt Rungego – zwiększanie liczby węzłów w interpolacji wielomianowej przy równomiernym rozmieszczeniu prowadzi do dużych błędów na krańcach przedziału.
   - Błąd średni (pomarańczowa linia) rośnie znacznie wolniej niż błąd maksymalny, ale również wykazuje tendencję wzrostową.
2. Wykres dolny (Węzły Czebyszewa):
   - Przy zastosowaniu węzłów Czebyszewa błąd maksymalny rośnie znacznie wolniej niż w przypadku równomiernego rozmieszczenia.
   - Chociaż dla większej liczby węzłów nadal obserwuje się pewien wzrost błędu maksymalnego, jest on znacznie mniej dramatyczny niż na pierwszym wykresie.
   - Błąd średni również rośnie, ale w dużo bardziej kontrolowany sposób.

Wnioski:

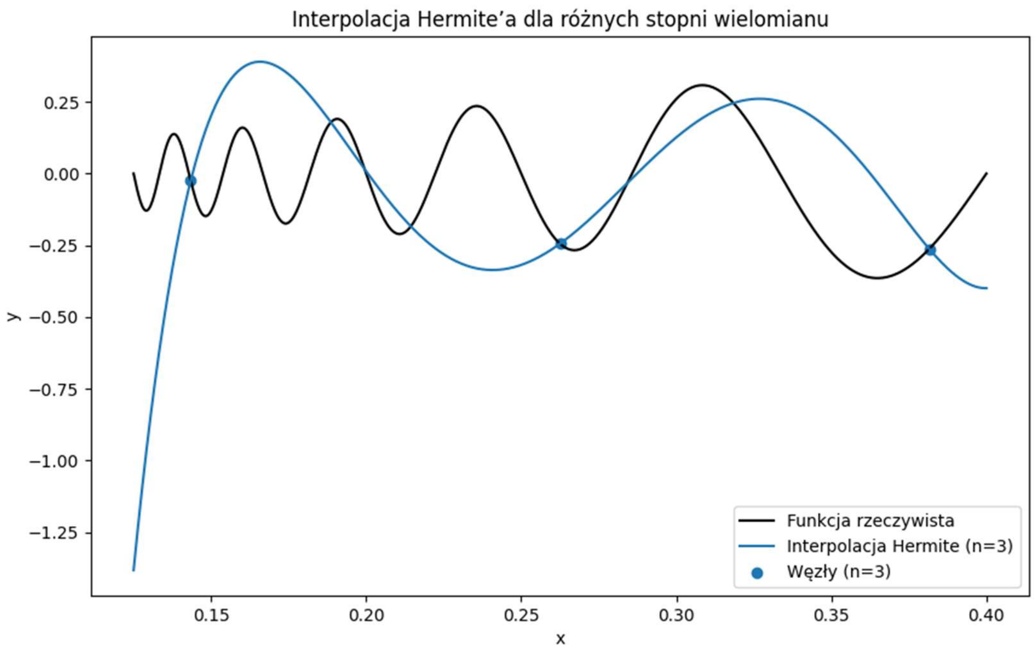
- Równomierne rozmieszczenie węzłów prowadzi do znacznych błędów w interpolacji wielomianowej, zwłaszcza dla dużej liczby węzłów (efekt Rungego).
- Węzły Czebyszewa znacząco redukują błąd maksymalny, co sprawia, że są lepszym wyborem do interpolacji wielomianowej.

Poniżej przedstawiono wykresy wygenerowane przez program dla wybranych wartości parametru n. Celem jest zilustrowanie, w jaki sposób zwiększa się dokładność przybliżenia wraz ze wzrostem stopnia kolejnych wielomianów interpolacyjnych. Są to wykresy dla interpolacji z równomiernym rozmieszczeniem węzłów oraz z użyciem węzłów Czybyszewa.

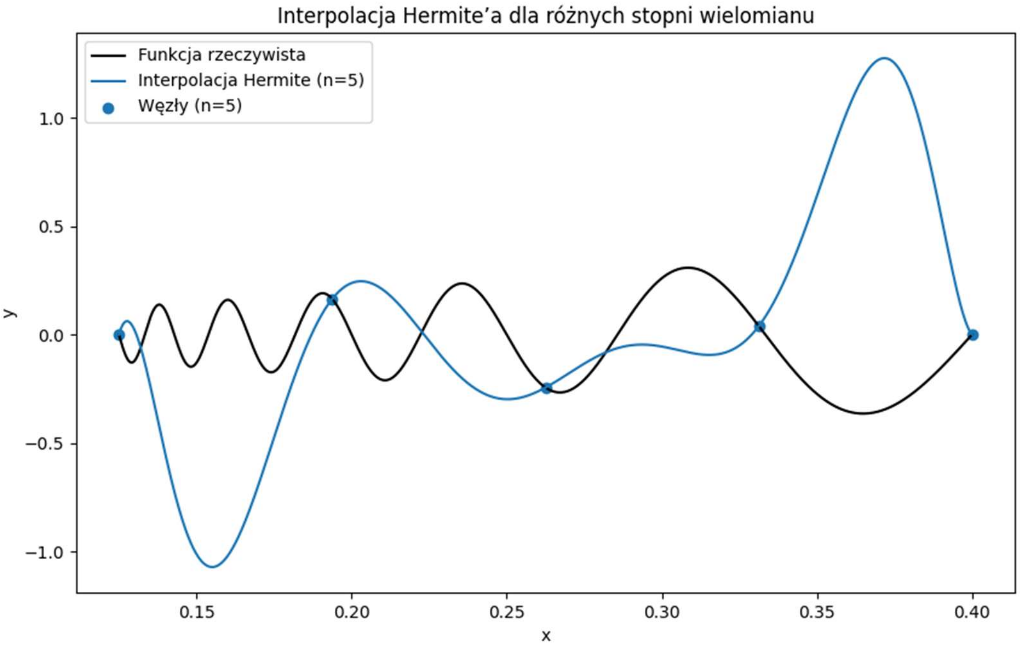
Wykres dla węzłów rozmieszczonych równomiernie i n = 3.



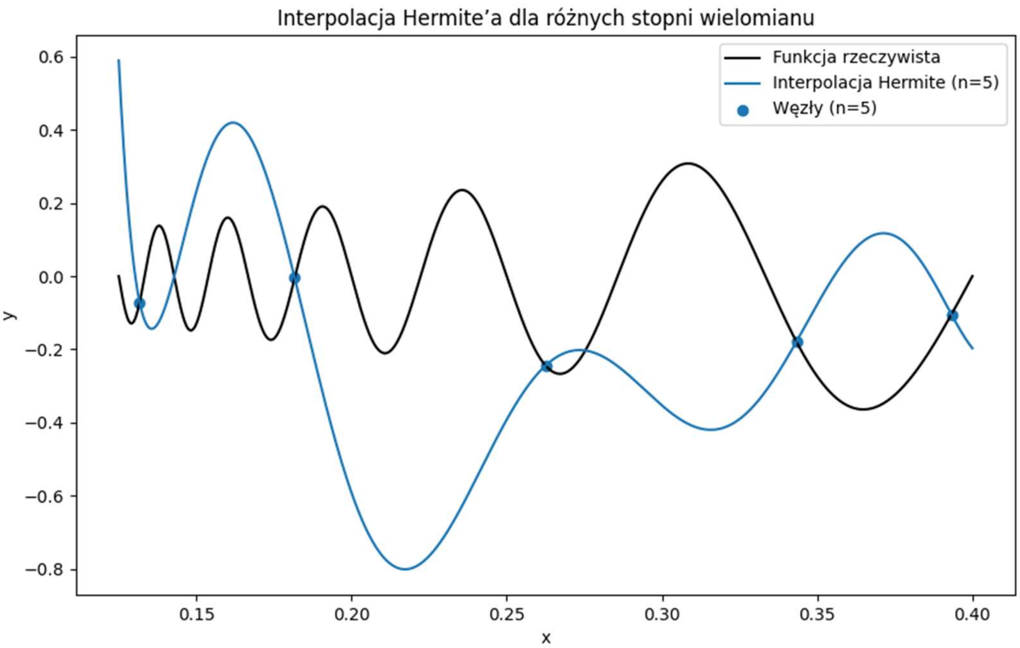
Wykres dla węzłów Czybyszewa i n = 3.



Wykres dla węzłów rozmieszczonych równomiernie i n = 5.

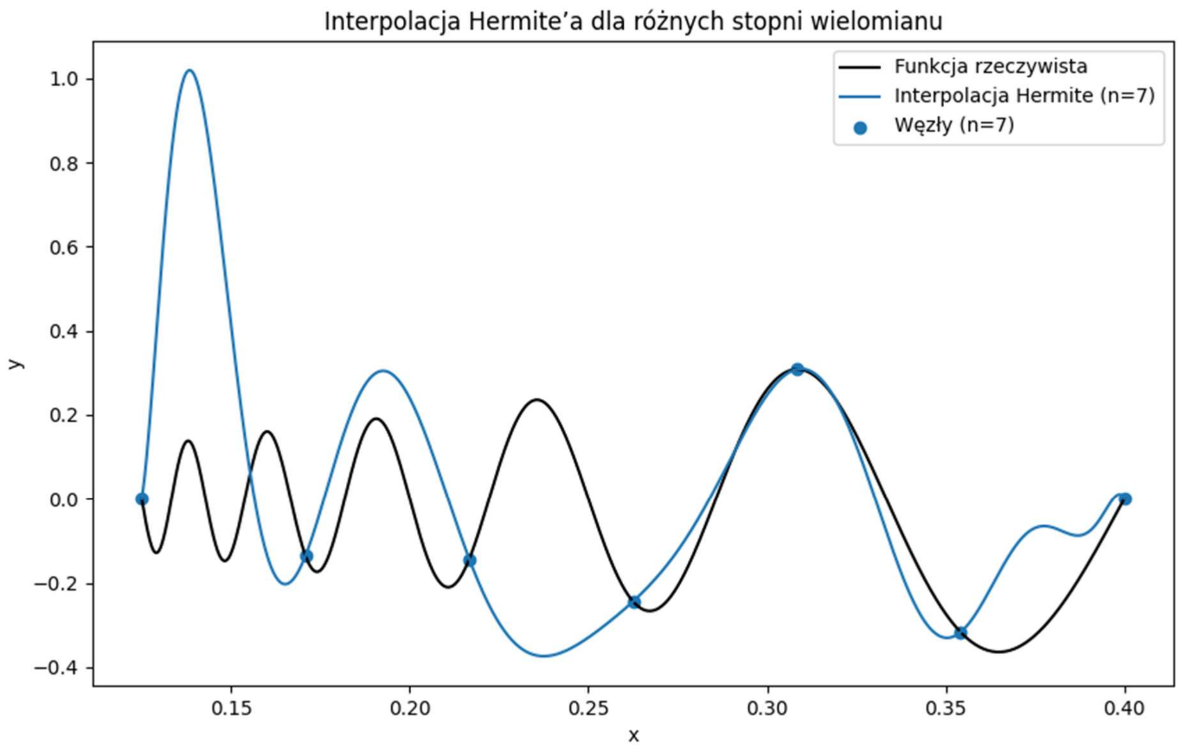


Wykres dla węzłów Czybyszewa i n = 5

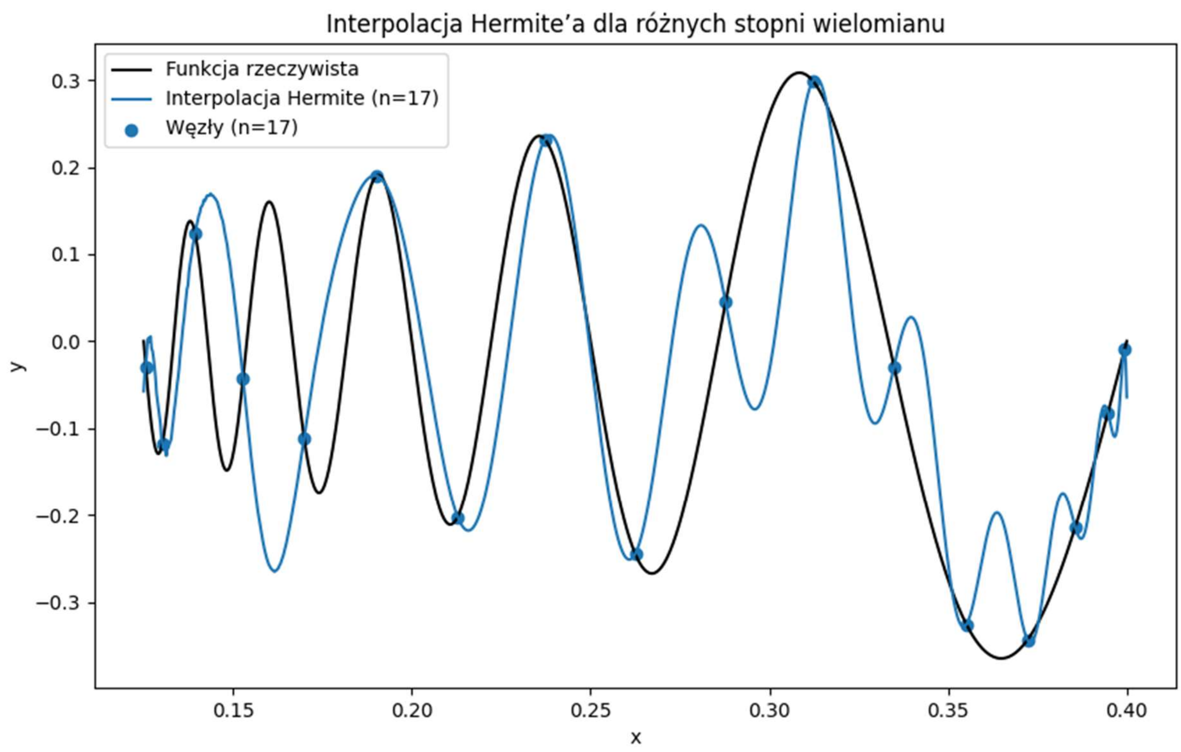


Wykresy najdokładniejszych aproksymacji:

Wykres dla węzłów rozmieszczonych równomiernie i n = 7

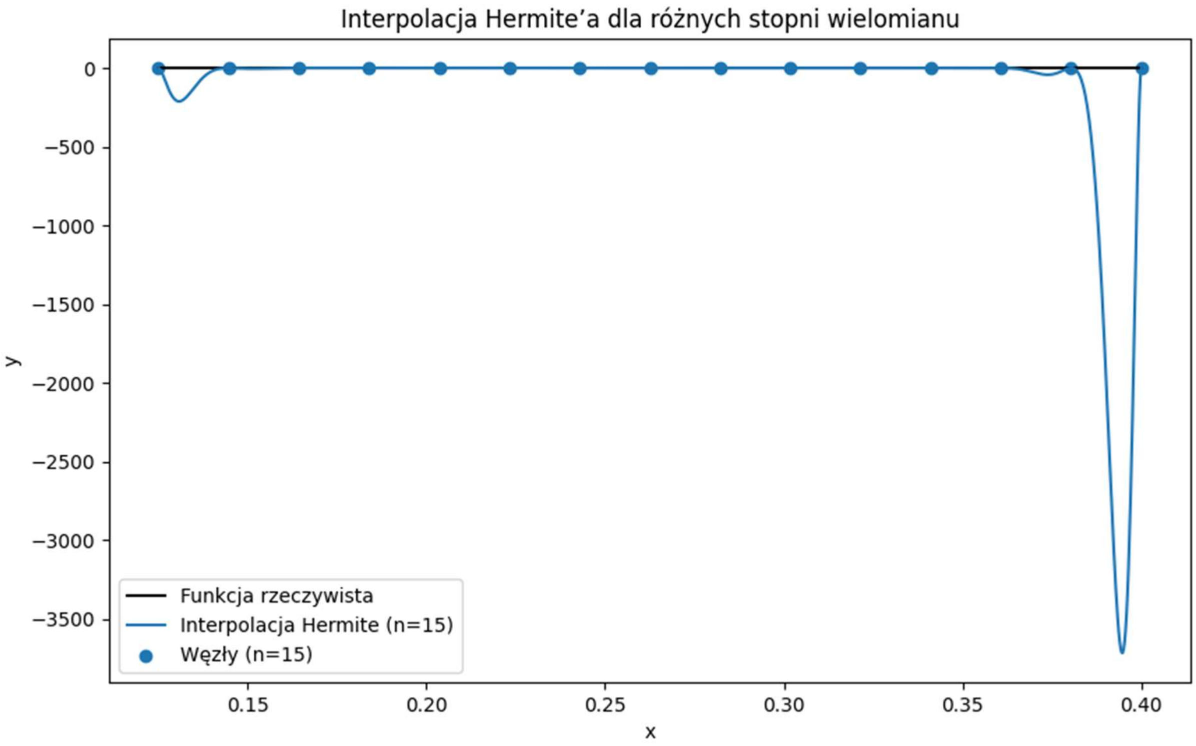


Wykres dla węzłów Czybyszewa i n = 18



Zilustrowanie efektu Rungego:

Wykres dla węzłów rozmieszczonych równomiernie i n = 15



Wykres dla węzłów Czybyszewa i n = 25