MOwNiT – Podsumowanie interpolacji i aproksymacji

Przygotował:
Szymon Budziak

Metody interpolacji i aproksymacji, które były omawiane:

- interpolacja Lagrange'a
- interpolacja Newtona
- interpolacja Hermite'a
- funkcje sklejane
- aproksymacja średniokwadratowa wielomianami algebraicznymi
- aproksymacja średniokwadratowa trygonometryczna

Różnica między interpolacją, funkcjami sklejanymi a aproksymacją:

- **interpolacja** to metoda numeryczna polegająca na wyznaczaniu w danym przedziale tzw. funkcji interpolującej, która przyjmuje w nim z góry zadane wartości, w ustalonych punktach nazywanych węzłami.
- **funkcje sklejane** to metoda numeryczna polegająca na przybliżaniu nieznanej funkcji wielomianami niskiego stopnia. Funkcje sklejane bywają też nazywane splajnami.
- **aproksymacja** polega jest to przybliżanie funkcji zwanej funkcją aproksymowaną inną funkcją zwaną funkcją aproksymującą. Innymi słowy polega ona na dopasowaniu krzywej do punktów pomiarowych, przy czym krzywa nie przechodzi dokładnie przez te punkty, lecz odzwierciedla ogólny trend w danych.

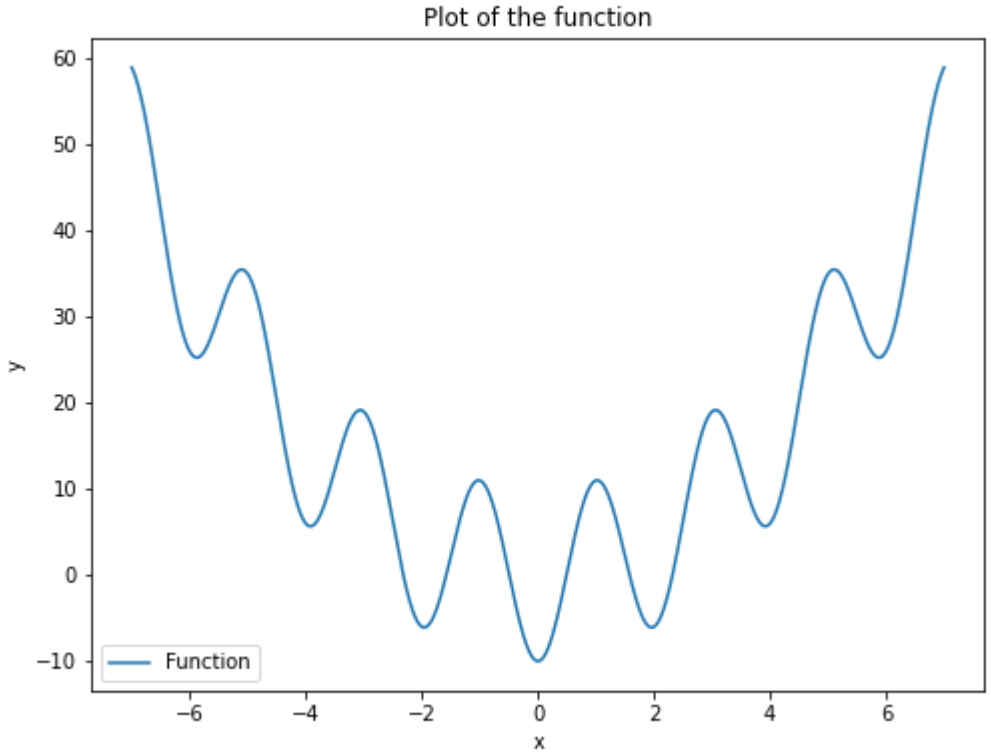
Dodatkowo dla wszystkich zagadnień związanych z interpolacją wyniki były przedstawiane dla różnego rozmieszczenia węzłów: równoodległe oraz Czebyszewa. Zostały również przedstawiony błędy obliczeniowy dla wszystkich metod interpolacji i aproksymacji. Liczby węzłów jakie zostały wzięte pod uwagę to: 3, 5, 8, 10, 15, 20, 30, 50, 80 dla wszystkich metod interpolacji. Dla funkcji sklejanych zostały użyte węzły: 4, 5, 7, 10, 15, 20, 50, 80, 150, 200, 500. Dla aproksymacji wielomianami algebraicznymi zostały użyte węzły: 4, 7, 10, 15, 20, 50 oraz stopnie wielomianu 2, 5, 8, 10, 12, 15 a dla aproksymacji trygonometrycznej użyte węzły to: 5, 7, 10,

20, 50, 100, natomiast stopnie: . Również tam gdzie zaobserwowano efekt Rungego został on przedstawiony i opisany.
Efekt Rungego jest to pogorszenie jakości interpolacji wielomianowej, mimo zwiększenia liczby jej węzłów. Początkowo ze wzrostem liczby węzłów *n* przybliżenie poprawia się, jednak po dalszym wzroście *n,* zaczyna się pogarszać, co jest szczególnie widoczne na końcach przedziałów.
Wszystkie powyższe zagadnienia zostały opracowane dla poniższej funkcji:

$$f(x) = x^2 - m \cdot \cos\left(\frac{\pi x}{k}\right)$$

k=1, m=10, [-7, 7]

Wykres funkcji



Wykres 1: Wykres funkcji omawianej we wszystkich tematach

1. Interpolacja Lagrange’a

Do interpolacji Lagrange’a został użyty wzór:

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

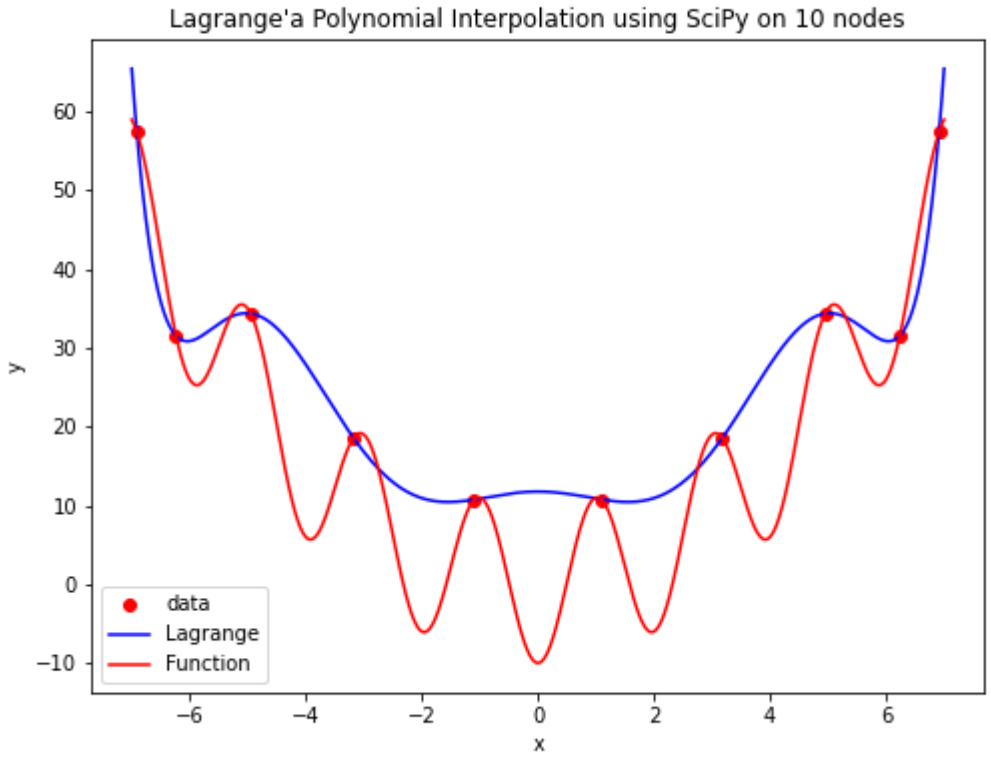
$$L_k(x) = \frac{d}{m} = \Pi_{i=0, i\neq k}^{n} \frac{x - x_i}{(x_k - x_i)}$$

$$d = (x - x_0)(x - x_1) \dots (x - x_{k-1}) \downarrow (x - x_{k+1}) \dots (x - x_n)$$

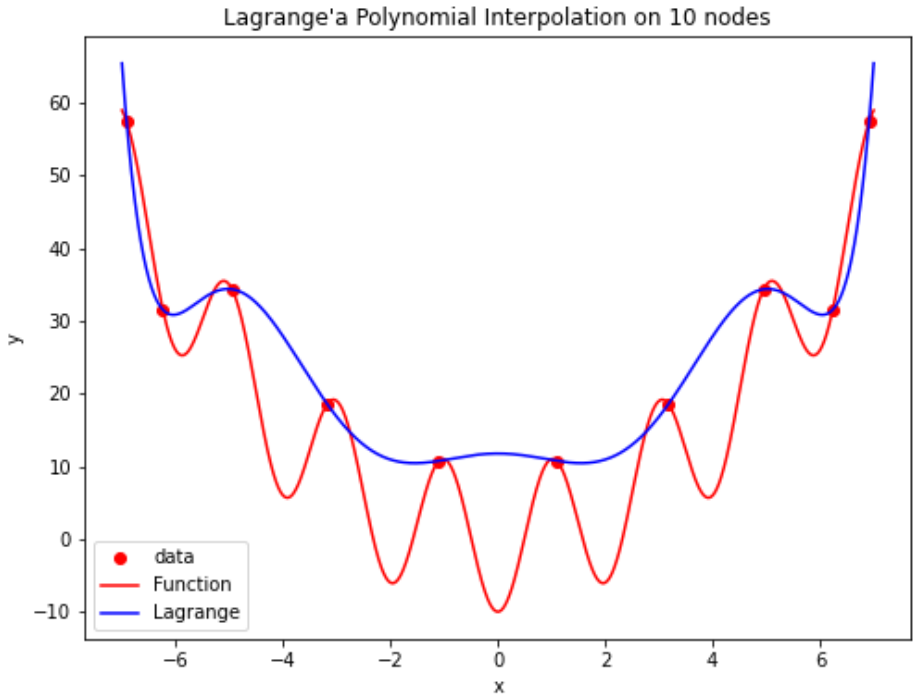
$$m = (x_k - x_0)(x_k - x_1) \dots (x_k - x_{k-1})(x_k - x_{k+1}) \dots (x_k - x_n)$$

Wykonanie interpolacji Lagrange’a zostało wykonane dla funkcji lagrange z pythonwej biblioteki SciPy oraz dla własnej implementacji. Dzięki możliwości skorzystania z biblioteki można było porównać działanie dla własnej implementacji interpolacji Lagrange'a. Wyniki obydwu funkcji niczym nie różnią się od siebie, dzięki czemu można wysunąć wniosek, że własna implementacja jest poprawna.

Przykładowe wykresy dla funkcji z biblioteki Scipy oraz własnej implementacji dla interpolacji Lagrange’a



Wykres 2: Wykres interpolacji Lagrange’a dla funkcji z biblioteki SciPy i 10 węzłów Czebyszewa



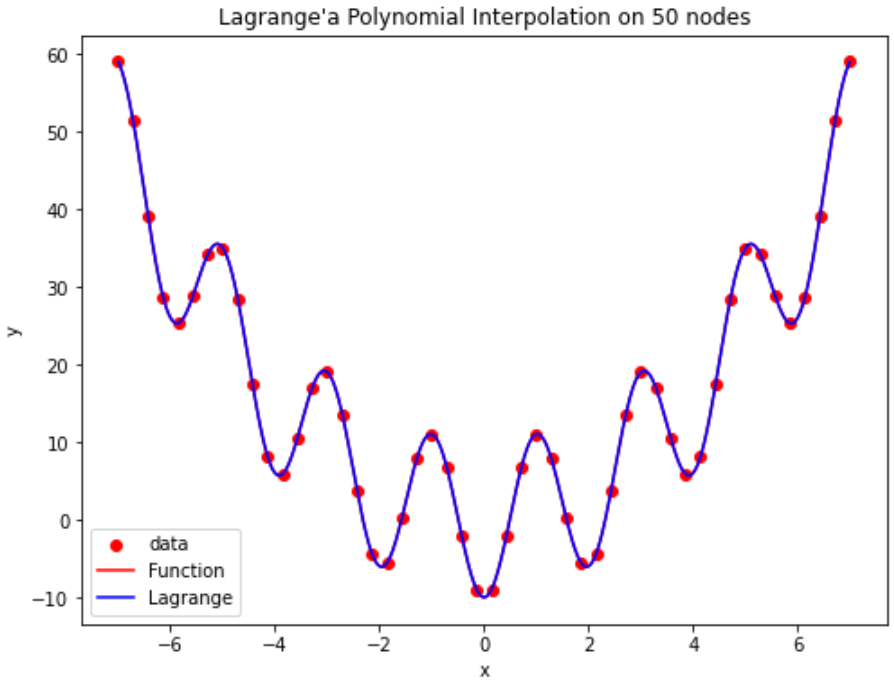
Wykres 3: Wykres interpolacji Lagrange’a dla własnej implementacji i 10 węzłów Czebyszewa

Błędy obliczeniowe dla własnej implementacji interpolacji Lagrange’a

Liczba węzłów	Błąd maksymalny (równoodlegle)	Błąd sumy kwadratów (równoodlegle)	Błąd maksymalny (Czebyszew)	Błąd sumy kwadratów (Czebyszew)
3	$1.959 * 10^{1}$	$1.999 * 10^{1}$	$9.491 * 10^{4}$	$1.485 * 10^{5}$
5	$1.990 * 10^{1}$	$2.042 * 10^{1}$	$1.049 * 10^{5}$	$1.220 * 10^{5}$
8	$1.999 * 10^{1}$	$2.211 * 10^{1}$	$1.498 * 10^{5}$	$9.713 * 10^{4}$
10	$1.996 * 10^{1}$	$2.194 * 10^{1}$	$9.640 * 10^{4}$	$1.120 * 10^{5}$
15	$2.830 * 10^{3}$	$1.907 * 10^{1}$	$4.910 * 10^{8}$	$6.619 * 104$
20	$4.460 * 10^{3}$	$1.498 * 10^{1}$	$8.529 * 10^{8}$	$3.593 * 10^{4}$
30	$3.272 * 10^{2}$	$2.720 * 10^{-2}$	$2.754 * 10^{6}$	$2.475 * 10^{-1}$
50	$2.653 * 10^{-3}$	$7.407 * 10^{-13}$	$4.215 * 10^{-5}$	$2.392 * 10^{-22}$
80	$2.054 * 10^{6}$	$1.278 * 10^{-13}$	$1.656 * 10^{13}$	$4.536 * 10^{-25}$

Tabela 1: Błąd obliczeniowy dla interpolacji Lagrange’a

Z tabeli można zauważyć, że najmniejszy błąd przy interpolacji Lagrange’a funkcja ma dla 50 węzłów.

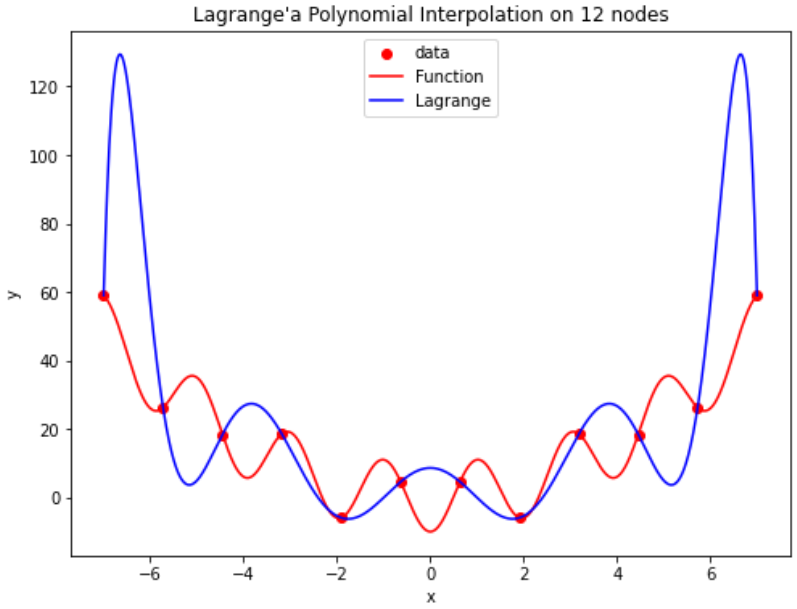


Wykres 4: Wykres interpolacji Lagrange’a dla własnej implementacji i 50 węzłów rozmieszczonych równoodlegle

Wykres dla interpolacji Lagrange’a jest identyczny oraz dla 50 węzłów Czebyszewa.

Efekt Rungego

W przypadku interpolacji Lagrange'a możemy zaobserwować, że już dla liczby węzłów 12 przy ich równoodległym rozmieszczeniu pojawia się efekt Rungego.



Wykres 5: Wykres interpolacji Lagrange’a dla własnej implementacji i 12 węzłów rozmieszczonych równoodlegle (przedstawienie efektu Rungego)

2. Interpolacja Newtona

Do interpolacji Newtona został użyty wzór:

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, ..., x_k] (x - x_0) ... (x - x_{k-1})$$

Budowanie tablicy ilorazów różnicowych:

$x_0 \quad f(x_0)$

$x_1 \quad f(x_1) \quad f[x_0, x_1]$

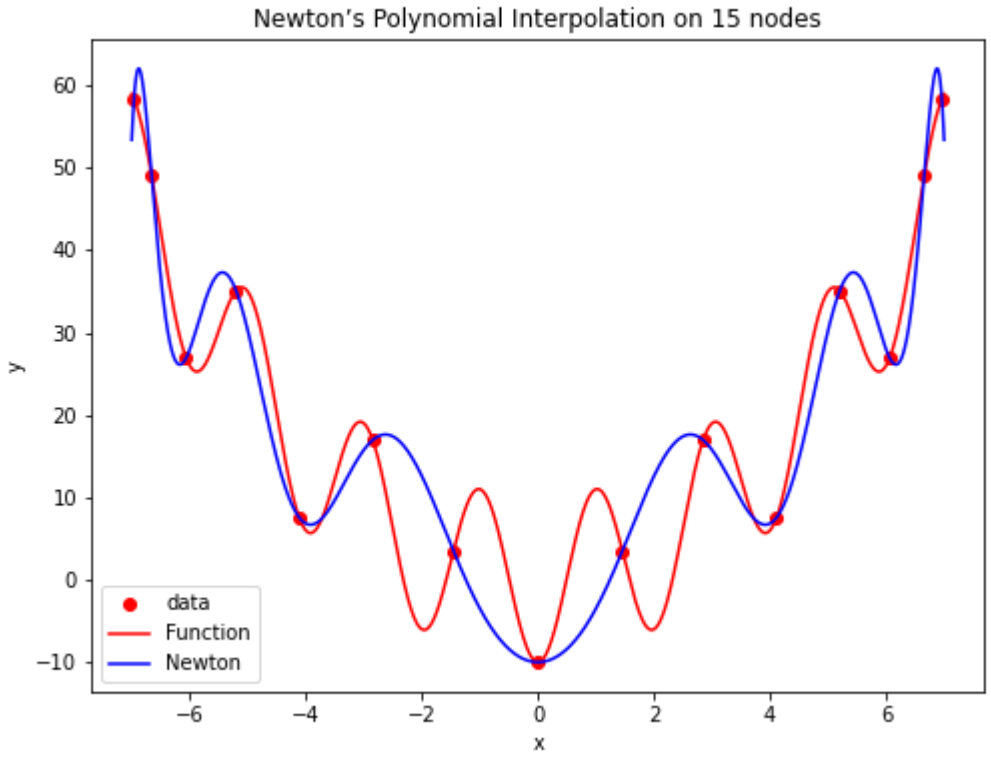
$x_2 \quad f(x_2) \quad f[x_0, x_1] \quad f[x_0, x_1, x_2]$

$\dots \quad \dots \quad \dots \quad \dots \quad \dots$

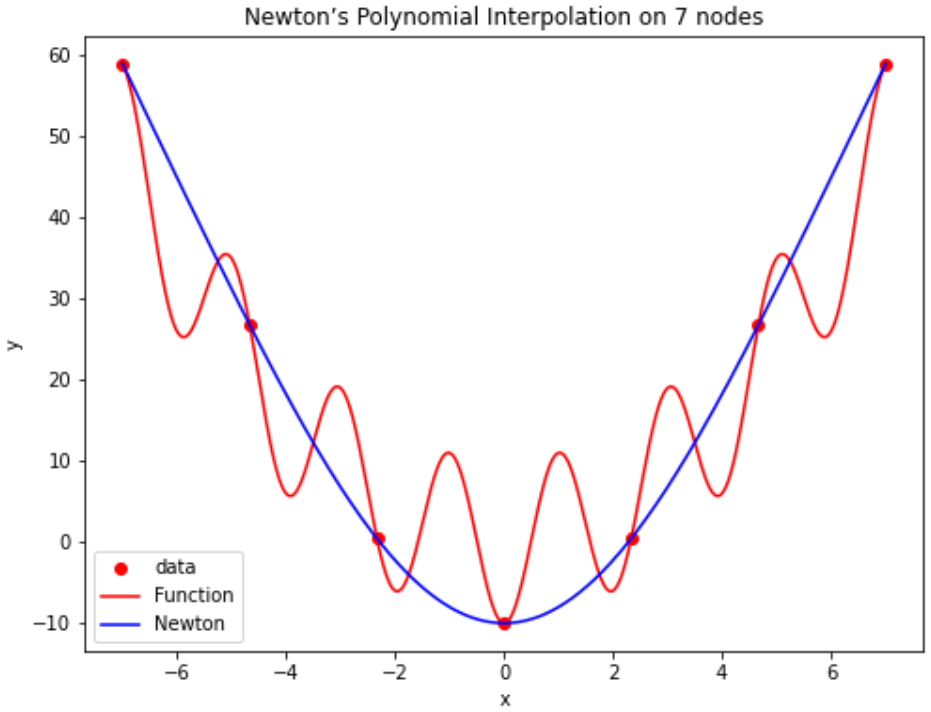
$x_n \quad f(x_n) \quad f[x_{n-1}, x_n] \quad .... \quad ... \quad f[x_0, ..., x_n]$

Wykonanie interpolacji Newtona zostało wykonane tylko dla własną implementacji.

Przykładowe wykresy dla interpolacji Newtona



Wykres 6: Wykres interpolacji Newtona dla własnej implementacji i 15 węzłów Czebyszewa

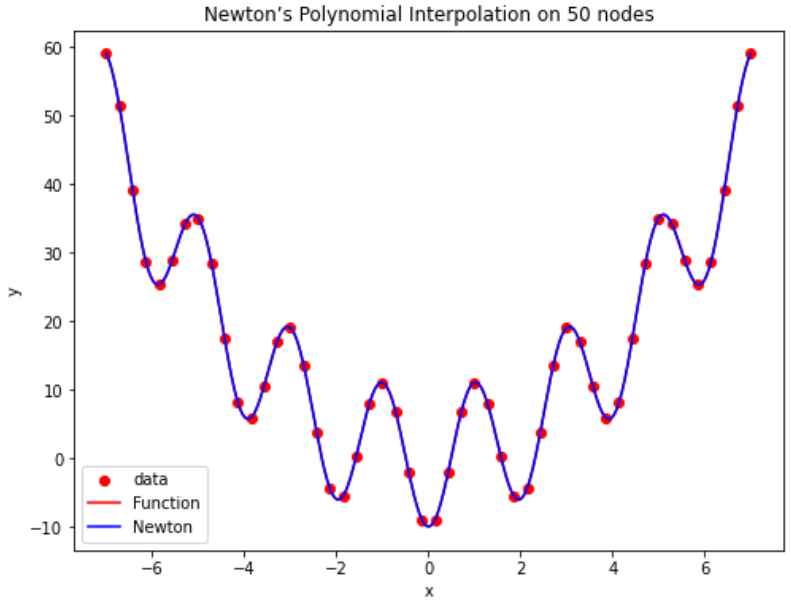


Wykres 7: Wykres interpolacji Newtona dla własnej implementacji i 7 węzłów rozmieszczonych równoodlegle

Błędy obliczeniowe dla własnej implementacji interpolacji Newtona

Liczba węzłów	Błąd maksymalny (równoodlegle)	Błąd sumy kwadratów (równoodlegle)	Błąd maksymalny (Czebyszew)	Błąd sumy kwadratów (Czebyszew)
3	19.594	$1.999 * 10^{1}$	$9.491 * 10^{4}$	$1.485 * 10^{5}$
5	19.908	$2.042 * 10^{1}$	$1.049 * 10^{5}$	$1.220 * 10^{5}$
8	19.999	$2.211 * 10^{1}$	$1.498 * 10^{5}$	$9.713 * 10^{4}$
10	19.960	$2.194 * 10^{1}$	$9.640 * 10^{4}$	$1.120 * 10^{5}$
15	2830.654	$1.907 * 10^{1}$	$4.910 * 10^{8}$	$6.619 * 10^{4}$
20	4460.178	$1.498 * 10^{1}$	$8.529 * 10^{8}$	$3.593 * 10^{4}$
30	327.215	$2.720 * 10^{-2}$	$2.754 * 10^{6}$	$2.475 * 10^{-1}$
50	0.005	$2.497 * 10^{-2}$	$1.832 * 10^{-4}$	$4.509 * 10^{-3}$
80	157778.405	$5.294 * 10^{7}$	$1.457 * 10^{11}$	$9.969 * 10^{15}$

Tabela 2: Błąd obliczeniowy dla interpolacji Newtona

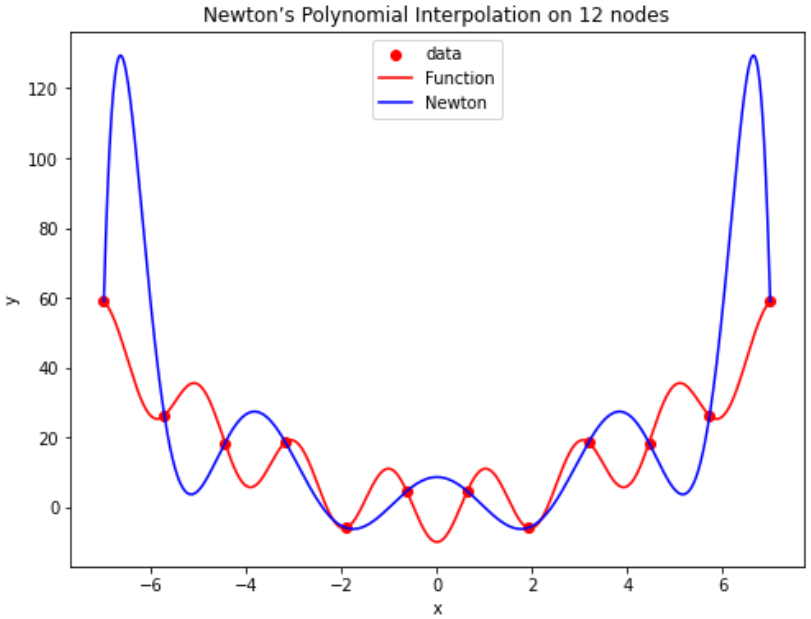


Wykres 8: Wykres interpolacji Newtona dla własnej implementacji i 50 węzłów rozmieszczonych równoodlegle

Z tabeli 2 możemy zauważyć, że minimalny błąd dla interpolacji Newtona jest dla liczby węzłów równej 50, tak jak w tabeli 1 (przypadek interpolacji Lagrange'a). Z tabeli możemy również wywnioskować, że interpolacja Newtona od 50 węzła zaczyna błędnie interpolować naszą funkcję.

Efekt Rungego

W przypadku interpolacji Newtona możemy zaobserwować, że dla liczby węzłów 12 przy ich równoodległym rozmieszczeniu pojawia się efekt Rungego.



Wykres 9: Wykres interpolacji Newtona dla własnej implementacji i 12 węzłów rozmieszczonych równoodlegle (przedstawienie efektu Rungego)

3. Interpolacja Hermite’a

Do interpolacji Hermite’a został użyty wzór:

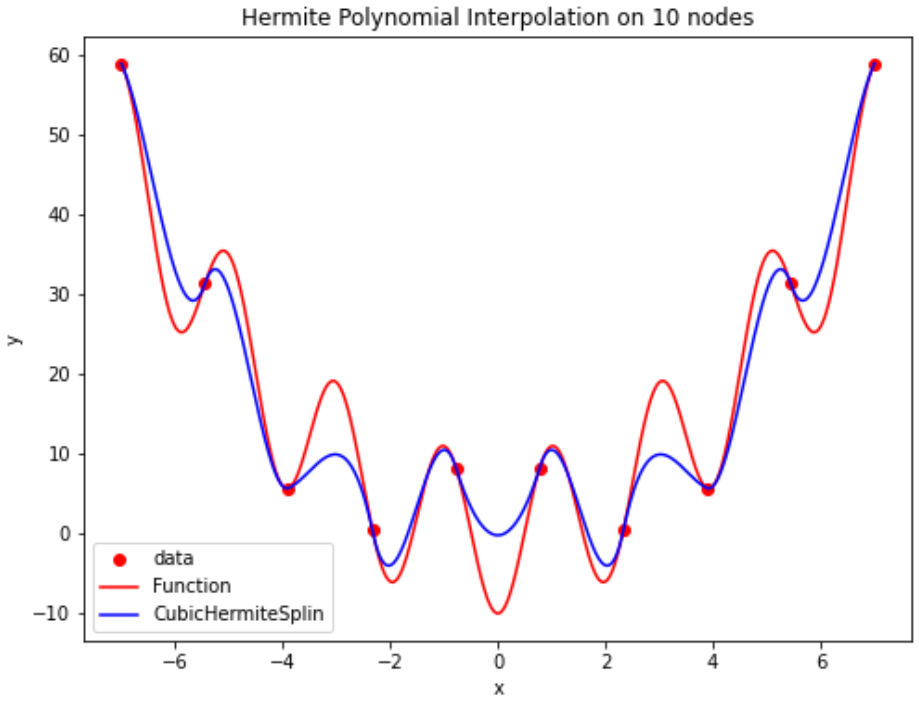
$$H_n(x) = \sum_{l=0}^{n} b_l \cdot p_l(x) = \sum_{i=0}^{k} \sum_{j=0}^{m_i - 1} b_{(s(i)+j)} \cdot p_{(s(i)+j)}(x)$$

Budowanie tablicy ilorazów różnicowych:

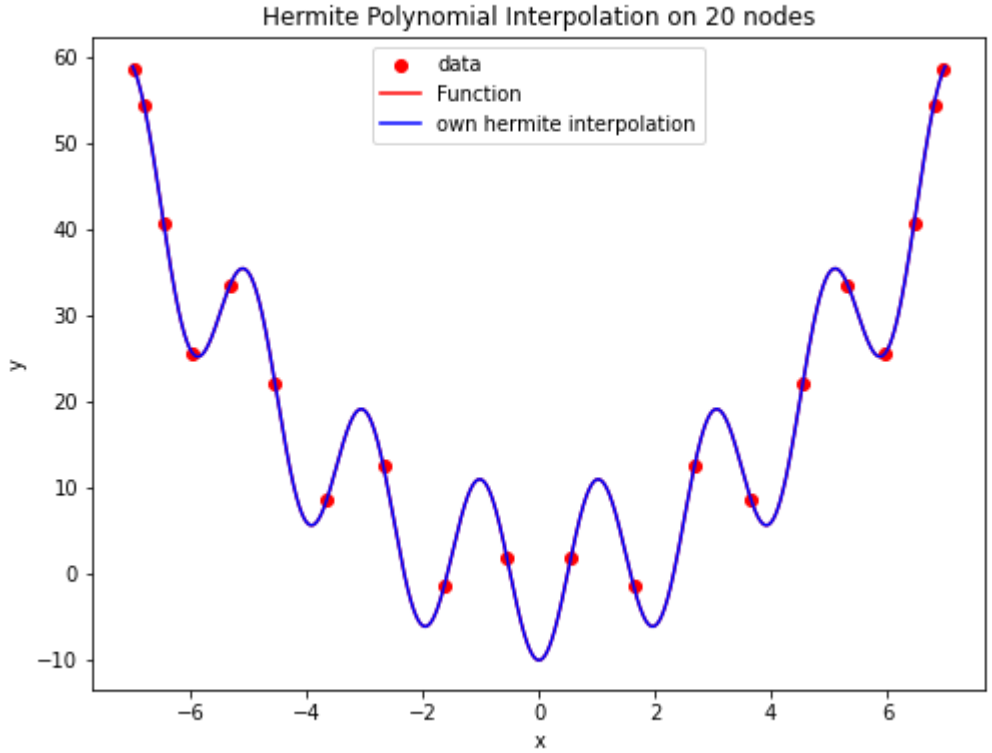
$$\begin{array}{llllll}
x_0 & f(x_0) & & & & \\
x_1 & f(x_1) & f[x_0,\ x_1] & & & \\
x_1 & f(x_1) & f'(x_1) & f[x_0,\ x_1,\ x_1] & & \\
x_1 & f(x_1) & f'(x_1) & \frac{f''(x_1)}{2!} & f[x_0,\ x_1,\ x_1,\ x_1] & \\
\ldots & \ldots & \ldots & \ldots & \ldots & \\
x_n & f(x_n) & f[x_{n-1},\ x_n] & .... & ... & f[x_0,\ ...,\ x_n]
\end{array}$$

Wykonanie interpolacji Hermite’a zostało wykonane dla funkcji CubicHermiteSpline z pythonwej biblioteki SciPy oraz dla własnej implementacji. Dzięki możliwości skorzystania z biblioteki można było porównać działanie dla własnej implementacji interpolacji Hermite’a. Wyniki obydwu funkcji niczym nie różnią się od siebie, dzięki czemu można wysunąć wniosek, że własna implementacja jest poprawna.

Przykładowe wykresy dla funkcji z biblioteki Scipy oraz własnej implementacji dla interpolacji Hermit’a



Wykres 10: Wykres interpolacji Hermite’a dla funkcji z biblioteki SciPy (interpolate.CubicHermiteSpline) i 10 węzłów rozmieszczonych równoodlegle



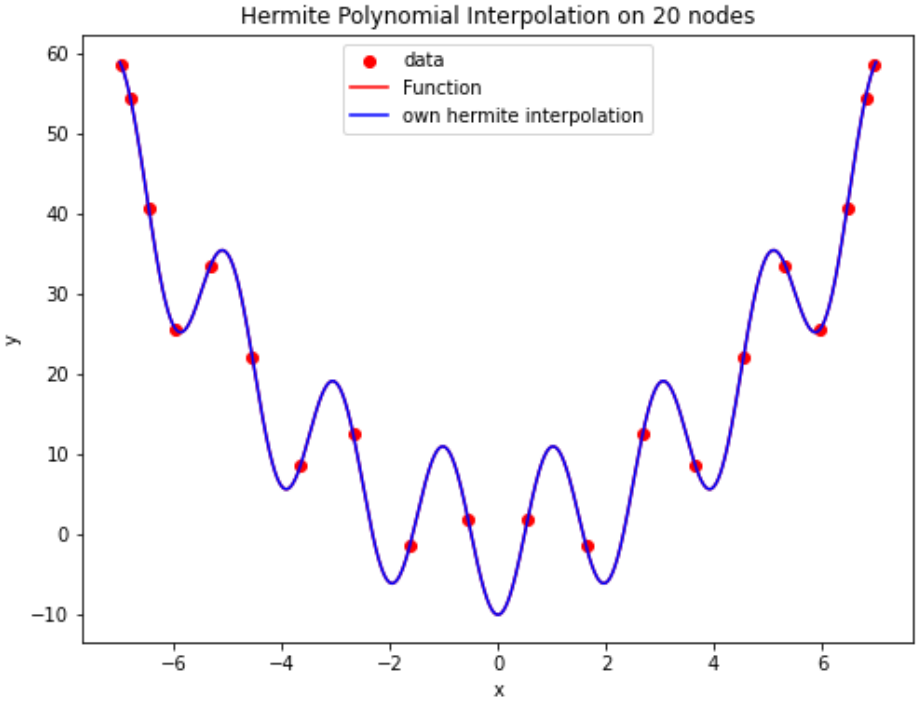
Wykres 11: Wykres interpolacji Hermite’a dla własnej implementacji i 20 węzłów Czebyszewa

Błędy obliczeniowe dla własnej implementacji interpolacji Hermite'a

Liczba węzłów	Błąd maksymalny (równoodległe)	Błąd sumy kwadratów (równoodległe)	Błąd maksymalny (Czebyszew)	Błąd sumy kwadratów (Czebyszew)
3	$1.920 * 10^{1}$	$2.386 * 10^{1}$	$9.906 * 10^{4}$	$1.872 * 10^{5}$
5	$6.158 * 10^{1}$	$2.915 * 10^{1}$	$6.877 * 10^{5}$	$1.959 * 10^{5}$
8	$1.999 * 10^{1}$	$4.741 * 10^{1}$	$1.498 * 10^{5}$	$2.158 * 10^{5}$
10	$1.307 * 10^{3}$	$2.997 * 10^{1}$	$1.260 * 10^{8}$	$1.078 * 10^{5}$
15	$9.049 * 10^{1}$	$5.393 * 10^{-2}$	$3.521 * 10^{5}$	$7.424 * 10^{-1}$
20	$6.102 * 10^{-2}$	$2.919 * 10^{-3}$	$1.107 * 10^{-1}$	$5.104 * 10^{-5}$
50	$8.324 * 10^{13}$	$1.026 * 10^{17}$	$2.058 * 10^{28}$	$4.664 * 10^{34}$
80	$4.615 * 10^{42}$	$1.504 * 10^{45}$	$4.964 * 10^{85}$	$7.548 * 10^{90}$

Tabela 3: Błąd obliczeniowy dla interpolacji Hermite'a

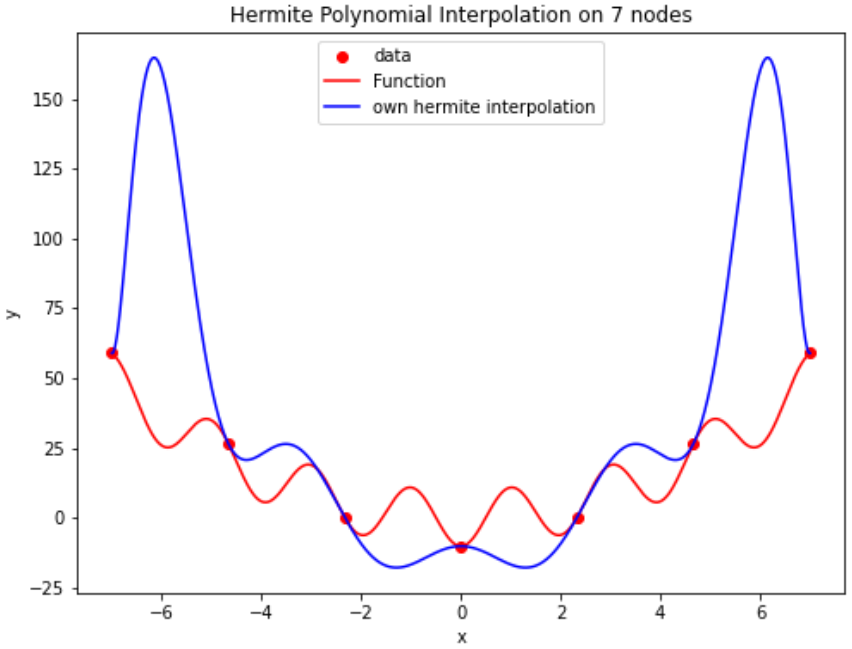
Z tabeli można zauważyć, że najmniejszy błąd przy interpolacji Hermite'a funkcja ma dla 20 węzłów.



Wykres 12: Wykres interpolacji Hermite'a dla własnej implementacji i 20 węzłów Czebyszewa

Efekt Rungego

W przypadku interpolacji Hermite'a możemy zaobserwować, że już dla liczby węzłów 7 przy ich równoodległym rozmieszczeniu pojawia się efekt Rungego.



Wykres 13: Wykres interpolacji Hermite’a dla własnej implementacji i 7 węzłów rozmieszczonych równoodlegle (przedstawienie efektu Rungego)

4. Funkcje sklejane

a) interpolacja funkcją sklejaną 3-go stopnia

Do interpolacji funkcją sklejaną 3-go stopnia został użyty wzór:

$$S_i(x) = a_i + b_i(x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3 \; dla \; i \in [1, \ldots, n-1]$$

gdzie każdy segment $S_i(x)$ jest interpolującym wielomianem drugiego rzędu w przedziale $[x_i, x_{i+1}]$.

Wypisując takie warunki dla każdego z ”wewnętrznych” punktów interpolacji i dodając warunki brzegowe można stworzyć układ równań i go rozwiązywać.

Aby była to funkcja sklejana 3-go stopnia, musi ona spełniać następujące warunki:

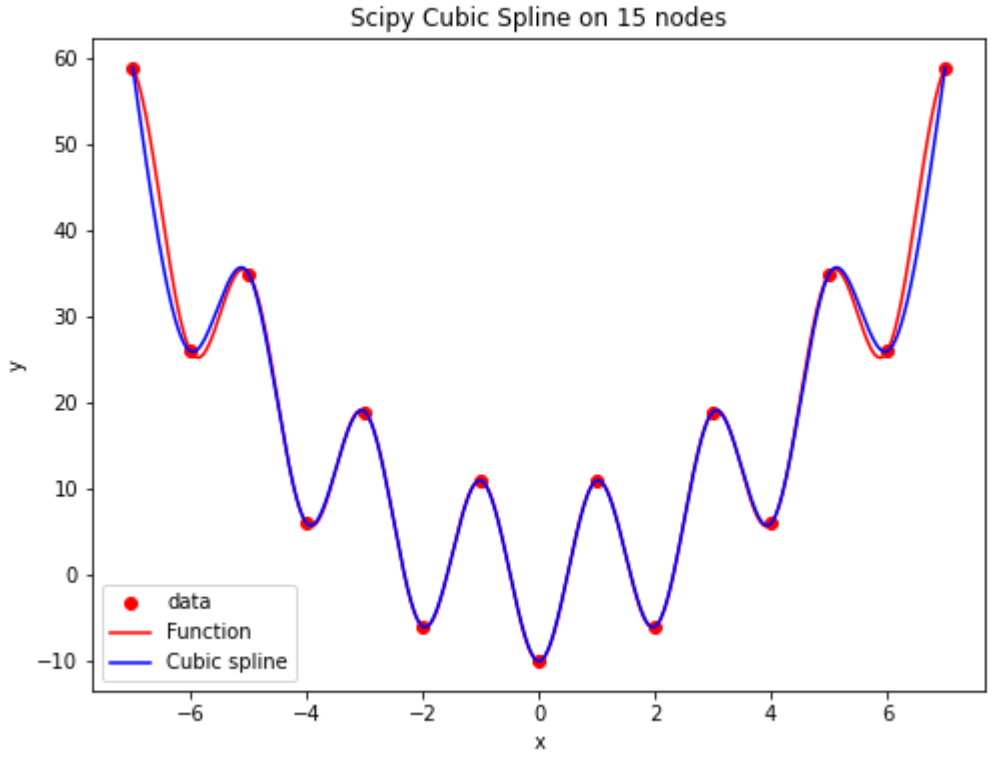
1) $S_i(x_{i+1}) = f(x_{i+1})$
2) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$
3) $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$
4) $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$

Pełne rozwiązanie oraz rozpisanie wzorów znajduje się w sprawozdaniu z funkcji sklejanych.

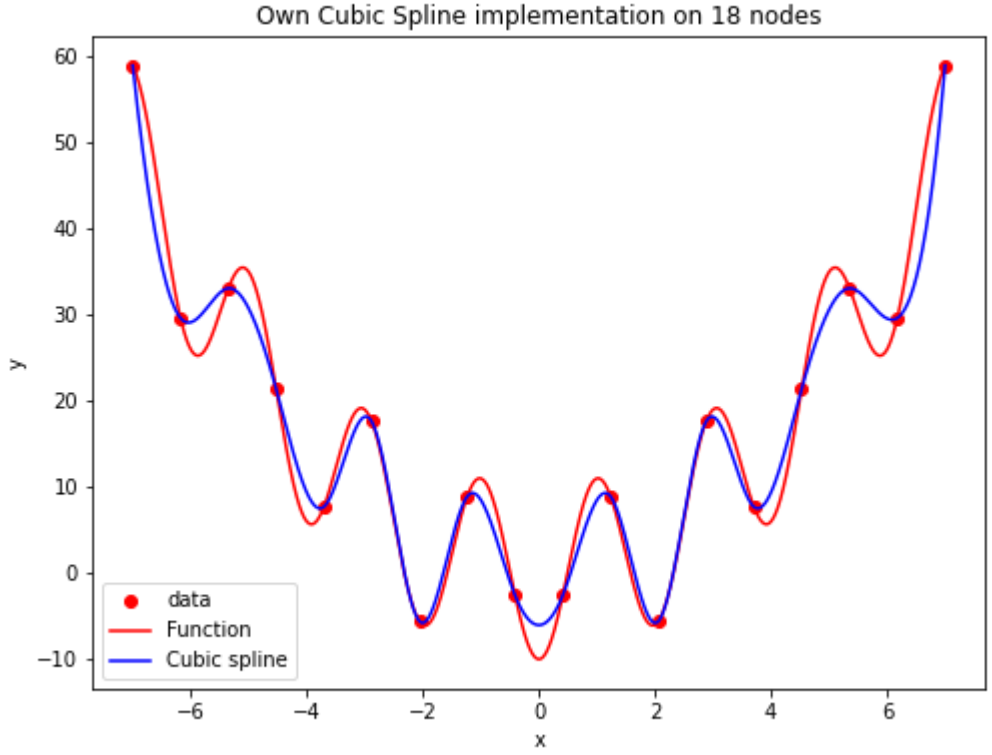
Warunki brzegowe jakie zostały użyte w przypadku funkcji sklejanej 3-go stopnia to cubic function oraz natural spline (free boundary). Również ich pełne rozpisanie znajduje się w sprawozdaniu z funkcji sklejanych.

Interpolacyjna funkcja sklejana 3-go stopnia została zaimplementowana i porównana dla dwóch funkcji. Pierwsza z biblioteki SciPy interpolate.CubicSpline w której jako argument możemy podać rodzaj warunków brzegowych które chcemy użyć. W tej funkcji zostały użyte warunki brzegowe “natural” oraz “clamped” (funkcja nie posiadała warunku “cubic”). Druga funkcja to własna implementacja interpolacyjnej funkcji sklejanej 3-go stopnia oraz użyte w niej warunki to “natural” oraz “cubic”.

Przykładowe wykresy dla funkcji z biblioteki SciPy interpolate.CubicSpline oraz własnej implementacji CubicSpline



Wykres 14: Wykres interpolacji 3-go stopnia z biblioteki SciPy funkcji sklejanej dla 15 węzłów interpolacji i warunku brzegowego “natural”



Wykres 15: Wykres interpolacji 3-go stopnia własnej implementacji funkcji sklejanej dla 18 węzłów interpolacji i warunku brzegowego “cubic”

b) interpolacja funkcją sklejaną 2-go stopnia

Do interpolacji funkcją sklejaną 2-go stopnia został użyty wzór:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \; dla \; i \in [0, \ldots, n - 1]$$

gdzie każdy segment $S_i(x)$ jest interpolującym wielomianem drugiego rzędu w przedziale $[x_i, x_{i+1}]$.

Aby była to funkcja sklejana 2-go stopnia, musi ona spełniać następujące warunki:

1) $S_i(x_i) = y_i \; dla \; i \in [0, 1, \ldots, n - 1]$

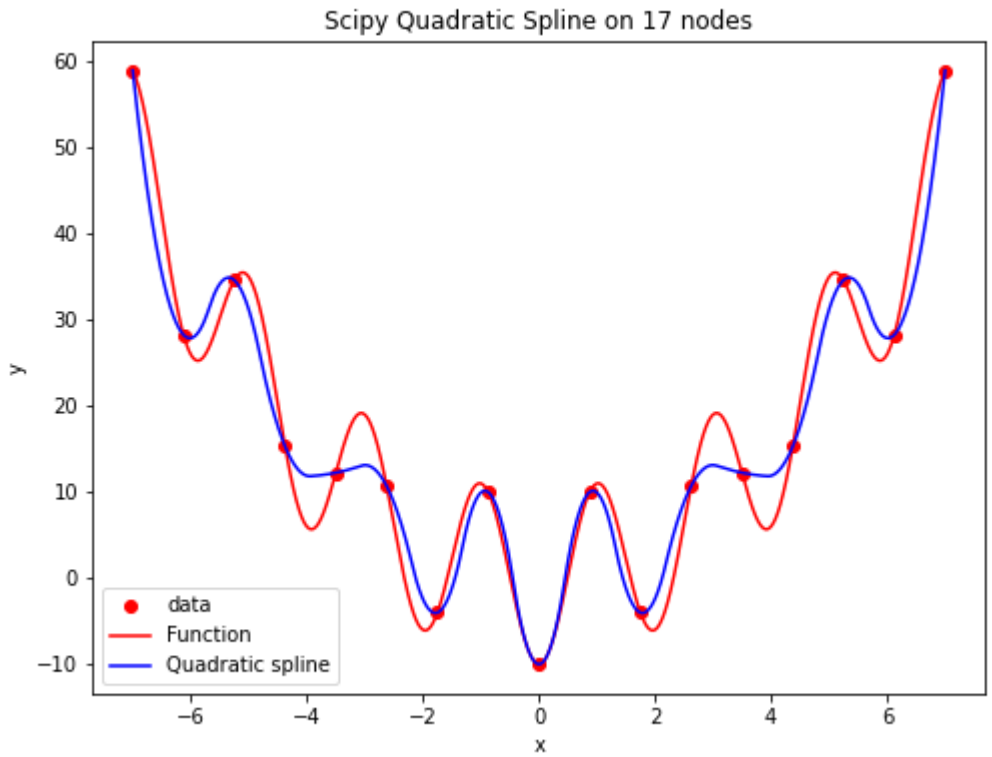
2) $S_{i+1}(x_{i+1}) = S_i(x_{i+1}) \; dla \; i \in [0, 1, \ldots, n - 2]$

3) $\frac{d}{dx}S_{i+1}(x_{i+1}) = \frac{d}{dx}S_i(x_{i+1}) \; dla \; i \in [0, 1, \ldots, n - 2]$

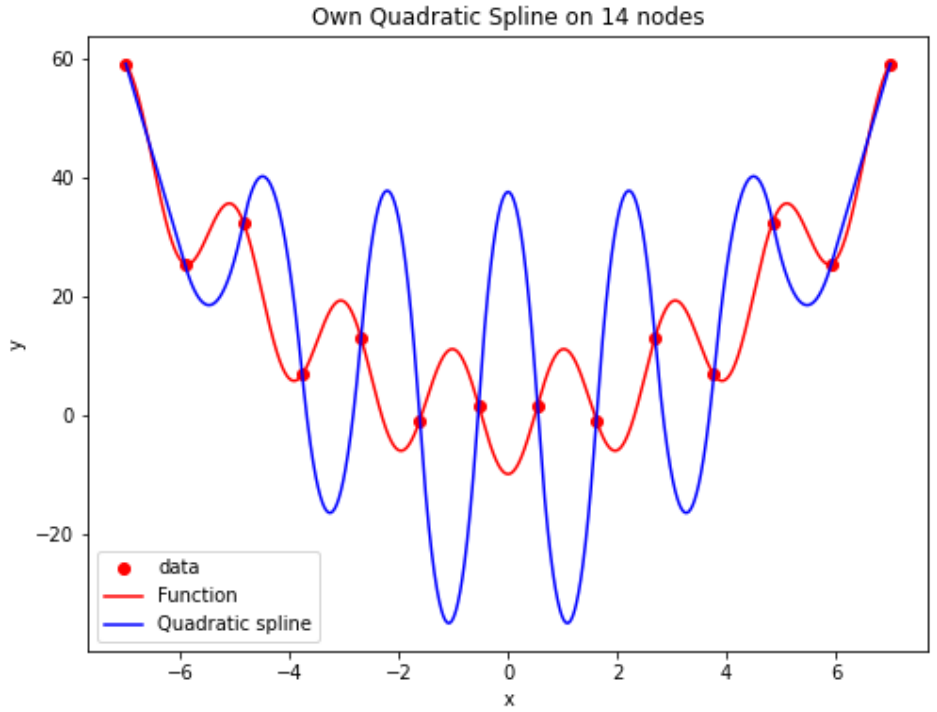
Pełne rozwiązanie oraz rozpisanie wzorów znajduje się w sprawozdaniu z funkcji sklejanych.

Warunki brzegowe jakie zostały użyte w przypadku funkcji sklejanej 2-go stopnia to natural spline (free boundary) oraz clamped boundary. Również ich pełne rozpisanie znajduje się w sprawozdaniu z funkcji sklejanych.

Przykładowe wykresy dla funkcji z biblioteki SciPy interpolate.interp1d(kind="quadratic") oraz z własnej implementacji QuadraticSpline



Wykres 16: Wykres interpolacji 2-go stopnia z biblioteki SciPy funkcji sklejanej dla 17 węzłów interpolacji



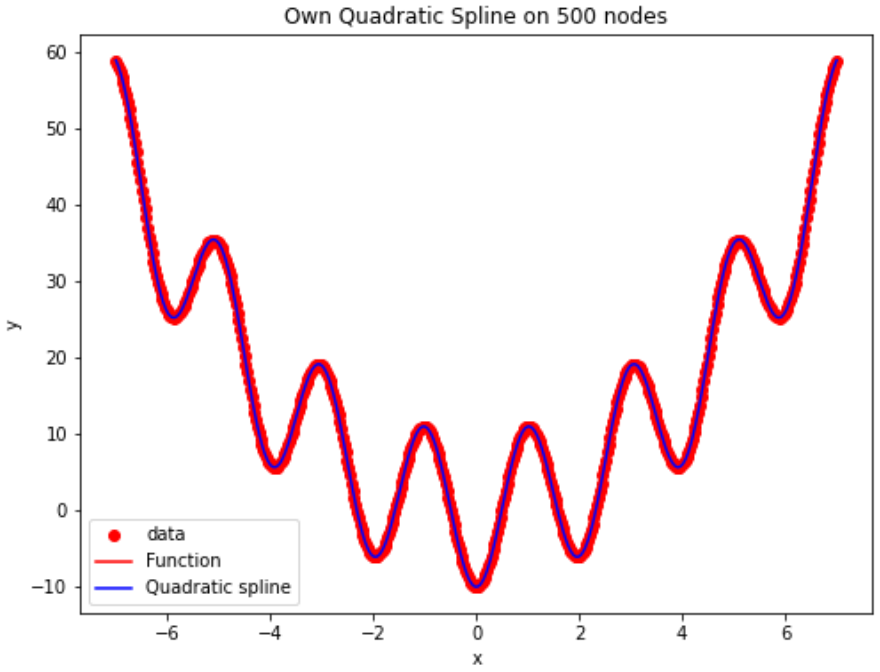
Wykres 17: Wykres interpolacji 2-go stopnia własnej implementacji funkcji sklejanej dla 14 węzłów interpolacji i warunku brzegowego “clamped”

Błędy obliczeniowe dla własnej implementacji interpolacji funkcją sklejaną 3-go i 2-go stopnia

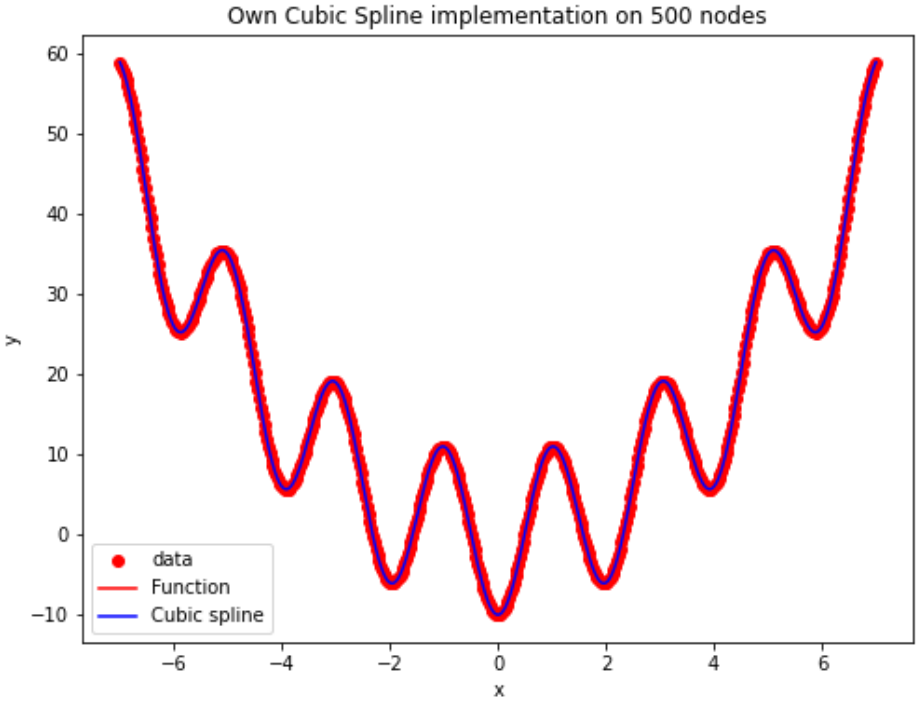
n	cubic natural max error	cubic natural sum square error	cubic cubic max error	cubic cubic sum square error	quadratic natural max error	quadratic natural sum square error	quadratic clamped max error	quadratic clamped sum square error
4	18,0931	88770,6254	25,7402	121278,6492	34,8143	419330,8664	22,5592	132957,3964
5	18,8908	96977,2575	19,1850	87627,9201	29,2712	183483,8881	19,6519	103699,1693
7	18,9961	99423,1191	19,1515	98463,9230	27,0817	135696,9435	19,2067	100168,7213
10	19,8336	95664,4153	19,8368	96728,4194	25,7986	135771,1214	20,8794	101118,0801
15	5,9490	2564,9495	12,5414	8837,6848	138,5584	3746392,9078	130,3114	3136503,4078
20	4,4214	2446,9062	6,4380	3533,1224	7,0324	10930,5846	8,4573	11936,6986
50	0,4308	4,0040	0,1459	0,4296	1,0540	535,8747	0,9752	452,2525
80	0,1552	0,3184	0,0267	0,0086	0,6259	205,1118	0,3757	72,8425
150	0,0415	0,0126	0,0025	0,0000	0,3294	57,6241	0,1065	5,9783
200	0,0236	0,0029	0,0008	0,0000	0,2464	32,3040	0,0597	1,8911
500	0,0035	0,0000	0,0000	0,0000	0,0982	5,1375	0,0095	0,0482

Tabela 4: Błąd obliczeniowy dla interpolacji 3-go i 2-go stopnia

Z tabeli 4 możemy zaobserwować, że dla naszej funkcji najlepsza interpolacja funkcjami sklejanymi zachodzi dla wielomianu stopnia n = 500. Możemy zauważyć, że błąd zmniejsza się wraz z coraz większą liczbą punktów interpolacji. Możemy wysunąć więc wniosek, że nie występuje efekt Rungego w przypadku interpolacji funkcją sklejaną 2-go oraz 3-go stopnia.



Wykres 18: Wykres interpolacji 2-go stopnia własnej implementacji funkcji sklejanej dla 500 węzłów interpolacji i warunku brzegowego “clamped”



Wykres 19: Wykres interpolacji 3-go stopnia własnej implementacji funkcji sklejanej dla 500 węzłów interpolacji i warunku brzegowego “cubic”

Możemy zauważyć, że wykresy są identyczne dla tak dużej liczby węzłów interpolacji (500) bez względu na to czy jest to interpolacja funkcją sklejaną 2-go czy 3-go stopnia oraz bez względu na użyte warunki brzegowe

5. Aproksymacja średniokwadratowa wielomianami algebraicznymi

Do aproksymacji średniokwadratowej wielomianami algebraicznymi został użyty wzór w postaci macierzowej:

$$\begin{pmatrix} \sum w_i & \sum w_i x_i & \sum w_i x_i^2 & \dots & \sum w_i x_i^m \\ \sum w_i x_i & \sum w_i x_i^2 & \sum w_i x_i^3 & \dots & \sum w_i x_i^{m+1} \\ . & . & . & . & . \\ \sum w_i x_i^m & \sum w_i x_i^{m+1} & \sum w_i x_i^{m+2} & \dots & \sum w_i x_i^{2m} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} =$$

$$= \begin{pmatrix} \sum w_i F_i \\ \sum w_i F_i x_i \\ \vdots \\ \sum w_i F_i x_i^m \end{pmatrix}$$

$$\underline{G \cdot A = B}$$

Jeżeli:

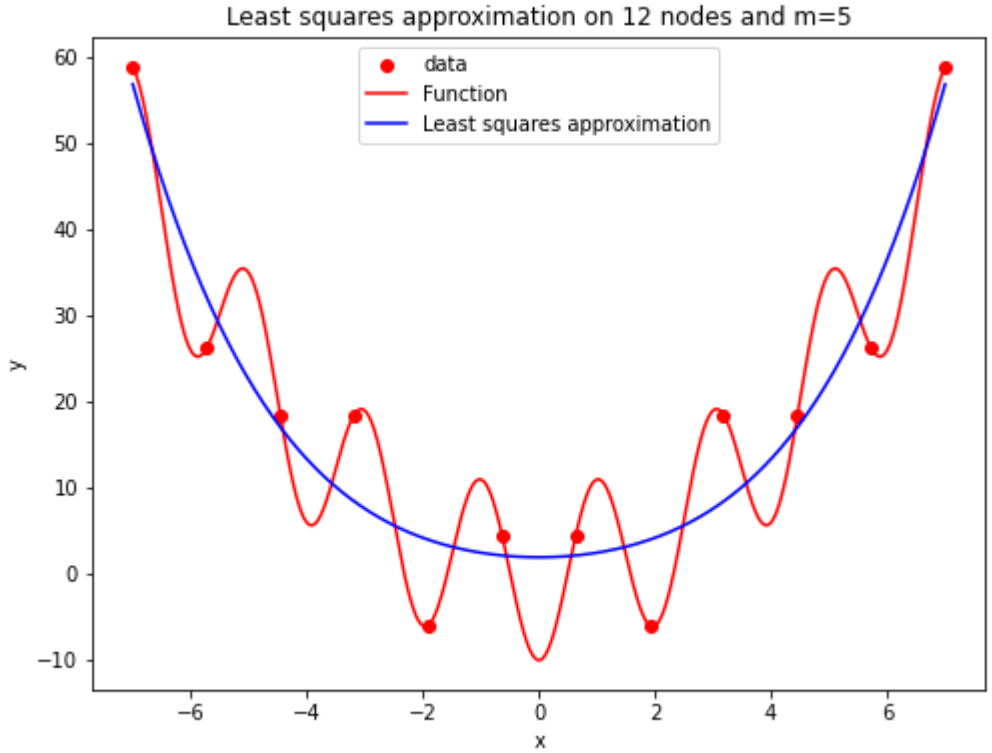
1) $x_0, x_1, \dots, x_n$ - są różne
2) m <= n

to det(G) ≠ 0 -> układ ma jedno rozwiązanie

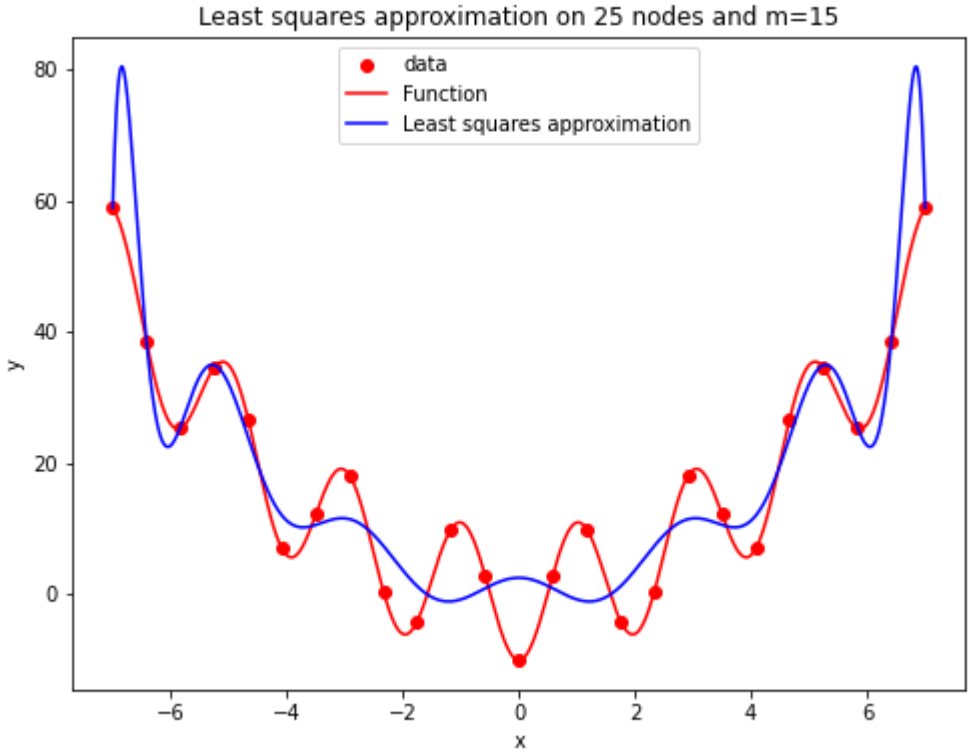
Jednak w praktyce:

1) m << n (korzystamy z dużej ilości informacji)
2) m - wysoki - by dobrze przybliżyć funkcję
3) m - niski - by wygładzić błędy
4) zwykle m <= 6

Przykładowe wykresy dla aproksymacji średniokwadratowej wielomianami algebraicznymi



Wykres 20: Wykres aproksymacji średniokwadratowej wielomianami algebraicznymi dla 12 węzłów i stopnia wielomianu 5



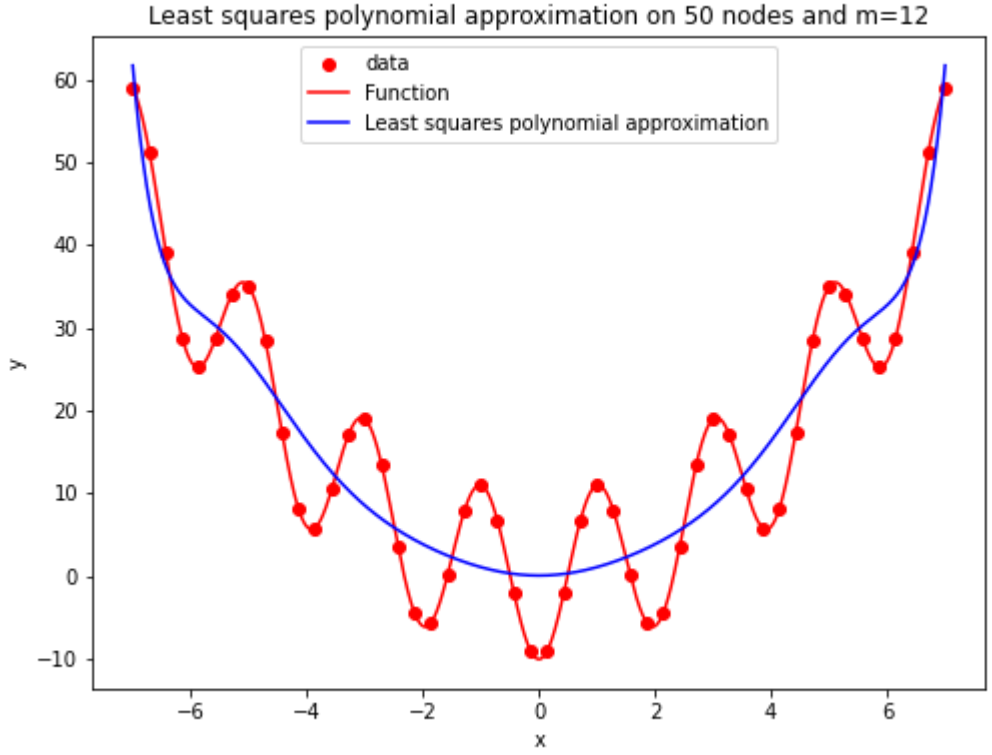
Wykres 21: Wykres aproksymacji średniokwadratowej wielomianami algebraicznymi dla 25 węzłów i stopnia wielomianu 15

Błędy obliczeniowe dla aproksymacji średniokwadratowej wielomianami algebraicznymi

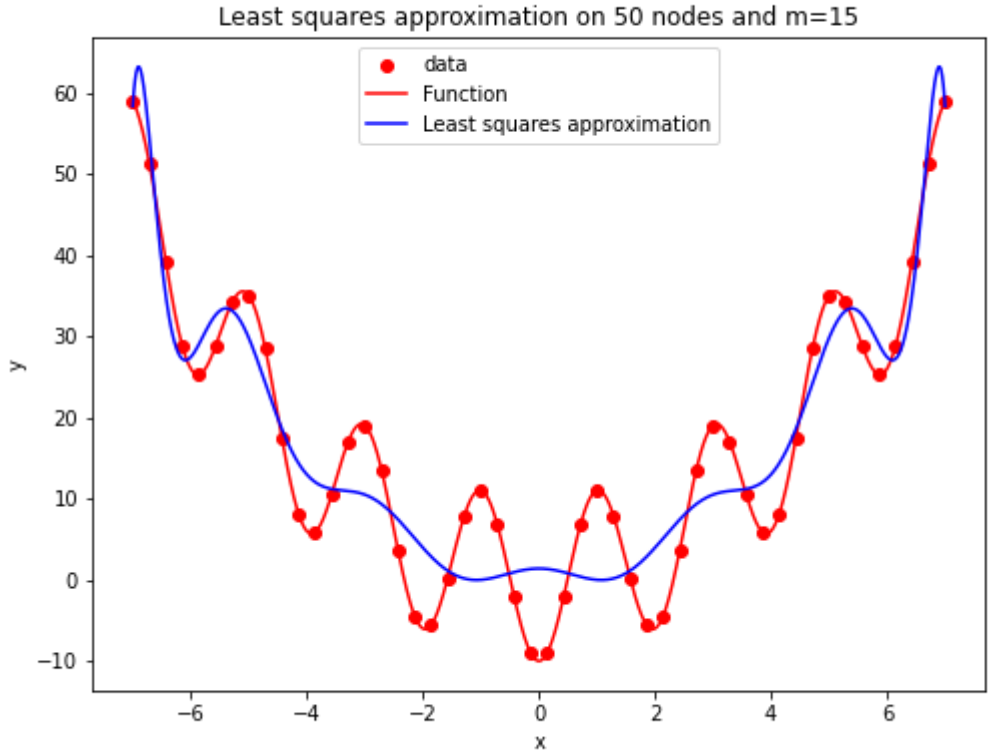
n	m	ls approximation max error	ls approximation sum square error
4	2	39,720	447149,781
4	5	74,296	1558274,095
4	8	42,603	243926,195
4	10	32,913	163247,870
4	12	24,697	103360,347
4	15	82,606	2021834,928
7	2	35,794	315682,230
7	5	19,548	101917,018
7	8	21,556	100850,157
7	10	19,404	79095,846
7	12	19,288	84321,192
7	15	26,093	116412,574
10	2	38,037	290068,935
10	5	15,138	60604,467
10	8	19,134	110209,233
10	10	19,961	96402,768
10	12	19,899	90047,612
10	15	19,962	96899,836
15	2	39,667	277774,633
15	5	11,474	50945,622
15	8	12,348	48299,409
15	10	12,615	49284,785
15	12	32,044	107582,077
15	15	2830,673	491020100,309
20	2	40,447	273766,589
20	5	11,116	50430,088
20	8	11,848	47777,983
20	10	11,623	45002,097
20	12	12,239	47608,040
20	15	50,619	147139,629
50	2	41,800	269707,928
50	5	10,825	49830,512
50	8	11,630	47703,101
50	10	11,334	44180,081
50	12	10,434	43353,587
50	15	11,386	34251,633

Tabela 5: Błąd obliczeniowy dla aproksymacji średniokwadratowej wielomianami algebraicznymi

Z tabeli 5 możemy zauważyć, że minimalny błąd jest dla liczby węzłów 50 oraz stopnia wielomianu równego 12 dla max error. Natomiast dla max square error minimalny błąd jest dla liczby węzłów 50 i stopnia wielomianu 15.



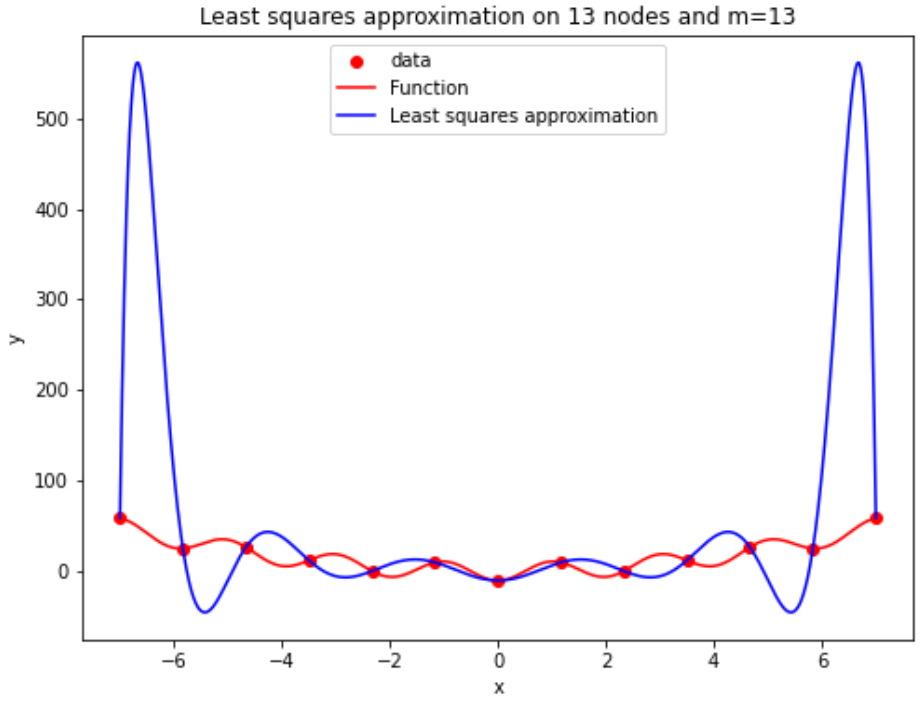
Wykres 22: Wykres aproksymacji średniokwadratowej wielomianami algebraicznymi dla 50 węzłów i stopnia wielomianu 12



Wykres 23: Wykres aproksymacji średniokwadratowej wielomianami algebraicznymi dla 50 węzłów i stopnia wielomianu 15

Efekt Rungego

Dla naszej funkcji efekt Rungego możemy już zaobserwować dla liczby węzłów 13 i stopnia wielomianu równego 13. Początkowo ze wzrostem liczby węzłów *n* przybliżenie poprawia się, jednak po dalszym wzroście *n,* zaczyna się pogarszać, co jest szczególnie widoczne na końcach przedziałów.



Wykres 24: Wykres aproksymacji średniokwadratowej wielomianami algebraicznymi dla 13 węzłów i stopnia wielomianu 13

6. Aproksymacja średniokwadratowa trygonometryczna

Do aproksymacji średniokwadratowej trygonometrycznej zostały użyte wzory:

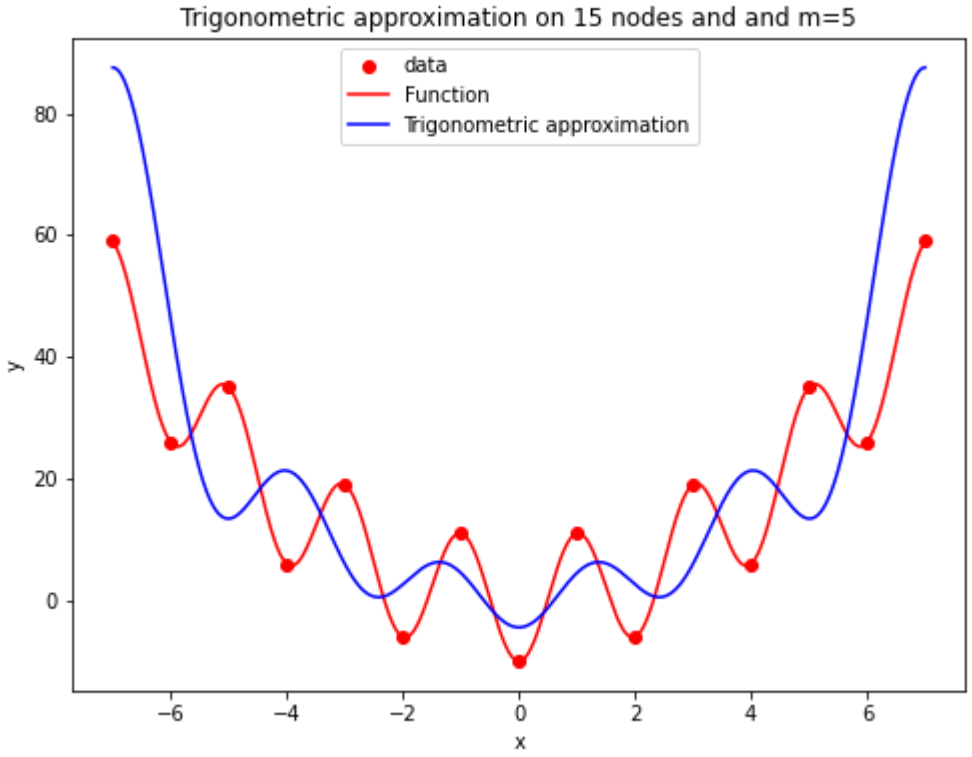
$$a_j = \frac{2}{n}\sum_{i=0}^{n-1} f(x_i) \cdot cos(j \cdot x_i)$$

$$b_j = \frac{2}{n}\sum_{i=0}^{n-1} f(x_i) \cdot sin(j \cdot x_i)$$

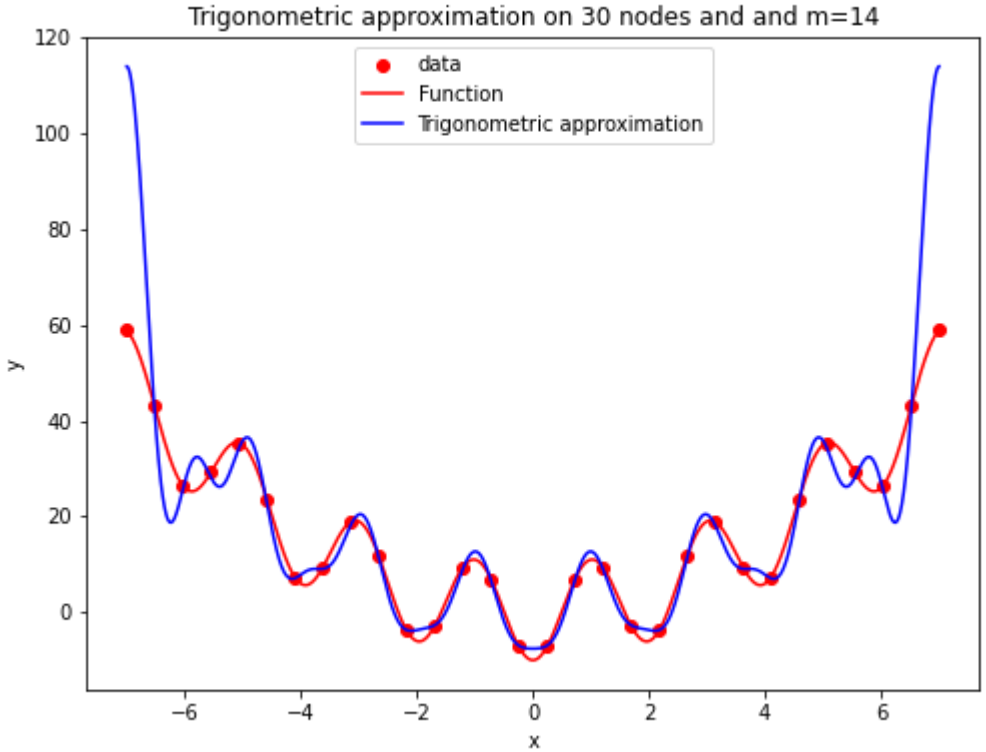
$$F_m(x) = \frac{1}{2} \cdot a_0 + \sum_{i=1}^{m} (a_j \cdot cos(j \cdot x) + b_j \cdot sin(j \cdot x))$$

Wielomianami trygonometrycznymi można aproksymować dowolną funkcję okresową, co wynika pośrednio z twierdzenia Weierstrassa dla funkcji okresowych. Zasady doboru stopnia wielomianu aproksymacyjnego różnią się od tych wykorzystanych w przypadku wielomianów algebraicznych. W przypadku wielomianów trygonometrycznych możemy od razu przyjąć najwyższy dopuszczalny stopień, równy $m = \lfloor \frac{n-1}{2} \rfloor$ (podłoga z $\frac{n-1}{2}$). Próba przyjęcia wyższego stopnia sprawia, że problem staje się źle uwarunkowany. Dodatkowo w przypadku aproksymacji średniokwadratowej wielomianami trygonometrycznymi musimy przeskalować każdy punkt aby był na przedziale -π do π. Następnie po przeskalowaniu wyliczamy nasze a_j oraz b_j. Kolejnym krokiem, jest ponowne przeskalowanie punktów jednak tym razem do wyjściowych wartości a następnie na tak przeskalowanych punktach wyliczamy funkcję F.

Przykładowe wykresy dla aproksymacji średniokwadratowej trygonometrycznej



Wykres 25: Wykres aproksymacji średniokwadratowej trygonometrycznej dla 15 węzłów i stopnia wielomianu 5



Wykres 26: Wykres aproksymacji średniokwadratowej trygonometrycznej dla 30 węzłów i stopnia wielomianu 14

Błędy obliczeniowe dla aproksymacji trygonometrycznej

n	m	trig approximation max error	trig approximation sum square error
5	2	69,280	1058389,857
5	3	-	-
5	5	-	-
5	10	-	-
5	20	-	-
5	24	-	-
7	2	52,005	563388,500
7	3	62,720	749160,920
7	5	-	-
7	10	-	-
7	20	-	-
7	24	-	-
10	2	39,250	357846,461
10	3	45,893	434183,625
10	5	-	-
10	10	-	-
10	20	-	-
10	24	-	-
20	2	21,751	92699,656
20	3	24,115	108061,905
20	5	24,607	143746,283
20	10	-	-
20	20	-	-
20	24	-	-
50	2	15,439	63406,988
50	3	16,468	63621,300
50	5	14,804	68067,661
50	10	22,470	27926,532
50	20	46,926	58154,935
50	24	56,640	70692,256
80	2	14,644	60185,225
100	2	15,288	59456,504
100	3	14,103	57609,981
100	5	12,683	57822,325
100	10	10,745	6917,472
100	20	22,901	14312,981
100	24	27,758	17371,916

Tabela 6: Błąd obliczeniowy dla aproksymacji średniokwadratowej trygonometrycznej

Przykład zmiany błędów obliczeniowych dla ustalonej liczby węzłów i zmiennego stopnia wielomianu

Przykład dla liczby węzłów 200:

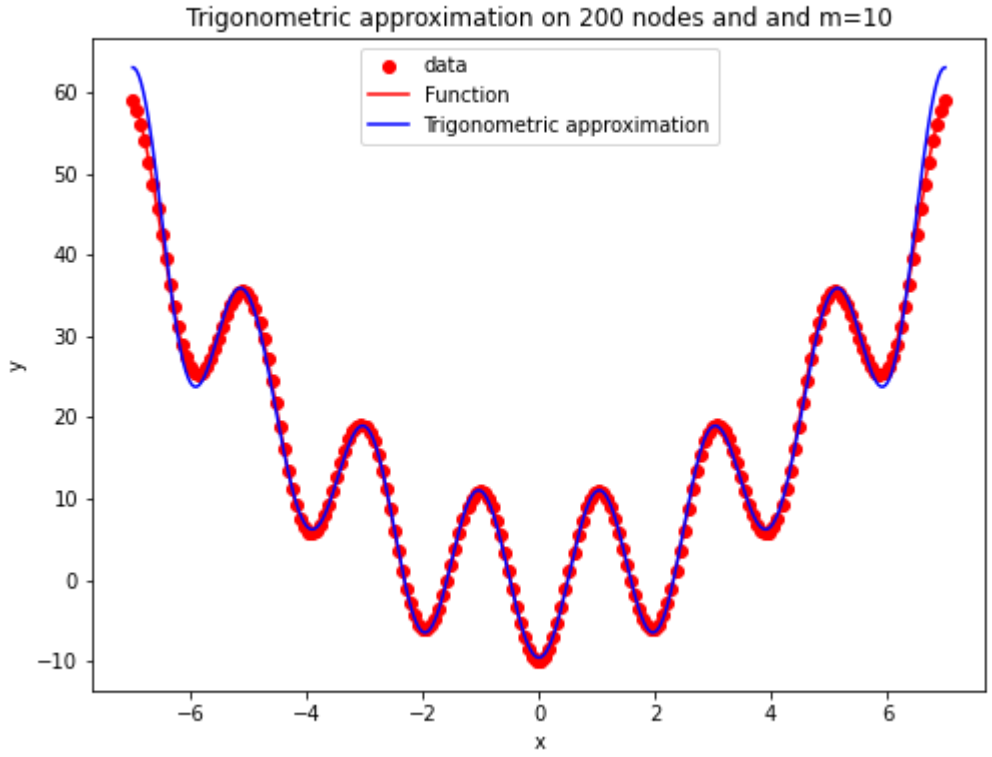
n\m	2	3	5	10	14	20	24	30	40	50	60	70	80
200	16,570	13,782	11,811	5,188	7,477	11,097	13,524	17,177	23,180	29,195	35,179	41,146	47,104

Tabela 7: Błąd obliczeniowy dla aproksymacji średniokwadratowej trygonometrycznej dla błędu maksymalnego punktów i liczby węzłów 200

n\m	2	3	5	10	14	20	24	30	40	50	60	70	80
200	58506,825	56158,434	55339,090	1752,956	2447,273	3551,367	4308,307	5467,850	7459,568	9522,986	11657,552	13863,292	16140,450

Tabela 8: Błąd obliczeniowy dla aproksymacji średniokwadratowej trygonometrycznej dla błędu sumy kwadratów punktów i liczby węzłów 200

Dla liczby węzłów 200 możemy zauważyć, że najmniejsze błędy są dla stopnia wielomianu 10 i wynosi on 5,188 dla błędu maksymalnego punktów oraz 1752,956 dla błędu sumy kwadratów punktów.



Wykres 27: Wykres aproksymacji średniokwadratowej trygonometrycznej dla 200 węzłów i stopnia wielomianu 10

Z powyższych tabel oraz wykresów możemy wywnioskować, że dla określonej liczby węzłów i dla zmiennego stopnia wielomianu zwiększanie stopnia wielomianu od pewnego przypadku nie zmniejsza błędu obliczeniowego.
Po przeanalizowaniu różnych przypadków dla aproksymacji trygonometrycznej możemy zauważyć, że wzrost liczby węzłów aproksymacji oraz stopnia wielomianu aproksymacyjnego nie powoduje zwiększenia się błędów obliczeniowych

Wnioski:

- w aproksymacji wielomianami trygonometrycznymi wzrost liczby węzłów aproksymacji oraz stopnia wielomianu aproksymacyjnego nie powoduje zwiększenia się błędów obliczeniowych.
- Aproksymacja wielomianami algebraicznymi może być źle uwarunkowana, co zawęża jej zastosowanie. Problem ten nie występuje w przypadku pozostałych omówionych metod.
- Ze względów wydajności i dokładności obliczeń należy starać się minimalizować stopień wielomianu aproksymacyjnego.
- Spośród wszystkich metod najlepszą pod względem przybliżania funkcji dla naszego przypadku wydaje się być metoda interpolacji funkcjami sklejanymi. Również dla dużej ilości liczby węzłów metoda aproksymacji trygonometrycznej daje dobre wyniki.

Literatura:

- Wykłady dr Rycerz z przedmiotu MOwNiT
- Oficjalna dokumentacja Numpy
- Wikipedia na temat interpolacji Lagrange'a, Newtona oraz efektu Rungego
- Wikipedia na temat interpolacji Hermite'a oraz krzywej sześciennej Hermita
- Wikipedia na temat interpolacji funkcjami sklejanymi 2-go oraz 3-go stopnia
- "Worksheet 5: Spline Interpolation Solutions" artykuł o interpolacji funkcjami sklejanymi z uniwersytetu w Stuttgart
- Wikipedia na temat Aproksymacji średniokwadratowej
- Jacek Złydach (JW2) – Metody Numeryczne – Sprawozdanie IV – Wstęp Teoretyczny