MOwNiT

Aproksymacja Wielomianowa

Opracował:
Kacper Kłusek

Dane sprzętowe:

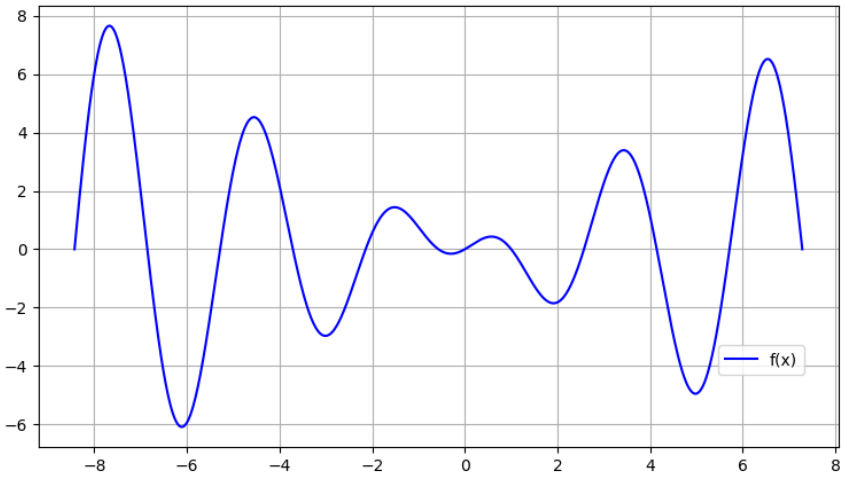
Do obliczeń użyłem języka Python 3.10.4 , system operacyjny wykorzystywany przeze mnie to Ubuntu 20.04.
Procesor to 16 wątkowy AMD® Ryzen 7 4800h 8 x 2.9Ghz,
Pamięć RAM 16GB DDR4

Obliczenia są wykonywane przez załączony do opracowania program wykorzystujący bibliotekę numpy do operacji na wektorach i macierzach. Wykresy rysowane były z pomocą biblioteki matplotlib.

Zadana funkcja:

$$f(x) = -k \cdot x \cdot \sin(m(x-1))$$

k=1, m=2, [-3pi+1, 2pi+1]



Doświadczenie:

Doświadczenie polegało na uruchomieniu programu obliczającego wielomian aproksymacyjny dla liczby węzłów $\in \{5, 7, 10, 12, 14, 30, 50, 70, 100\}$.
Doświadczenie wykonywano dla m stałych oraz zależnych od n.
Do sprawozdania załączono najciekawsze przypadki.
Doświadczenie było wykonywane jedynie dla węzłów równoodległych.
Błędy były liczone zgodnie z wzorem na SSE (Sum Squared Error).
Wykresy były rysowane przez obliczenie wartości interpolowanej funkcji dla 1000 równoodległych punktów na podanym w punkcie wyżej przedziale.
Na każdym z wykresów widnieją:

- **f** - funkcja bazowa (zielona)
- **F** - funkcja aproksymacyjna (niebieska)

Użyte wzory:

$F(x)$ - zadana na zbiorze dyskretnym $\{x_i\}, i = 0, 1, \ldots, n$

Szukamy takich współczynników a_j, że :

$$min \sum_{i=0}^{n} w(x_i)[F(x_i) - f(x_i)]^2$$

$$\sum_{i=0}^{n} w(x_i) \left[F(x_i) - \sum_{j=0}^{m} a_j x_i^j \right] x_i^{k \leftarrow \frac{\partial f}{\partial a_k}} = 0, k = 0, 1, \ldots, m$$

$$\sum_{i=0}^{n} w(x_i) x_i^k \sum_{j=0}^{m} a_j x_i^j = \sum_{i=0}^{n} w(x_i) F(x_i) x_i^k, k = 0, 1, \ldots, m$$

$$\sum_{j=0}^{m} \underbrace{\left(\sum_{i=0}^{n} w(x_i) x_i^{j+k} \right)}_{g_{k,j}} a_j = \underbrace{\sum_{i=0}^{n} w(x_i) F(x_i) x_i^k}_{b_k}$$

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

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

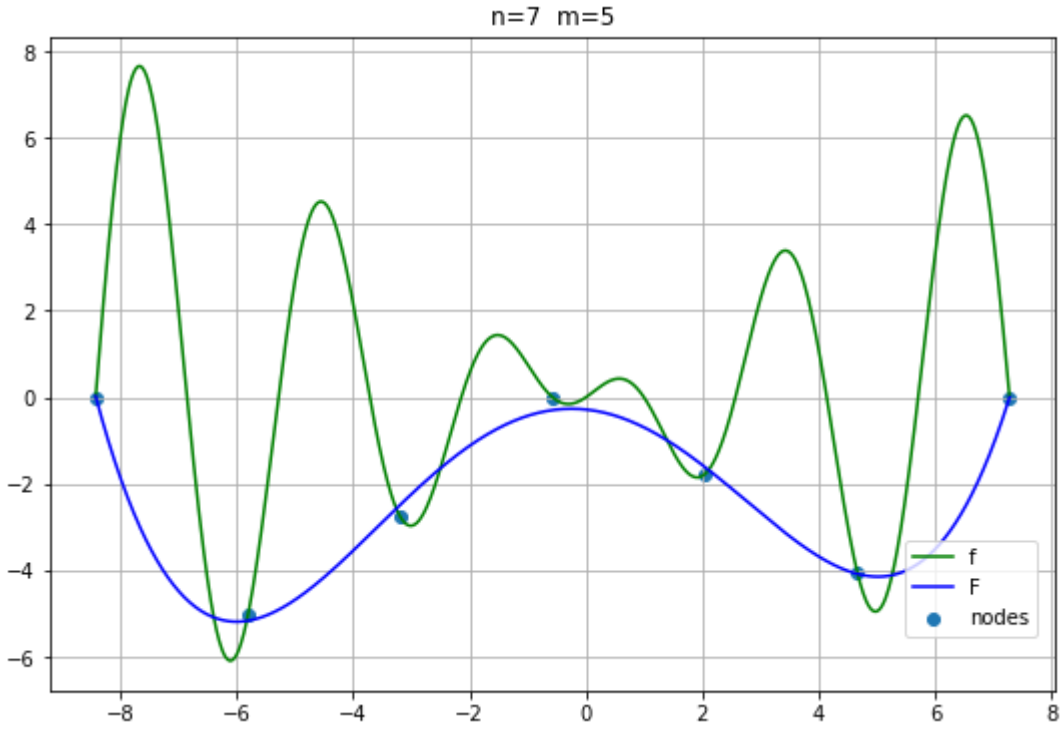
$$\underline{G \cdot A = B}$$

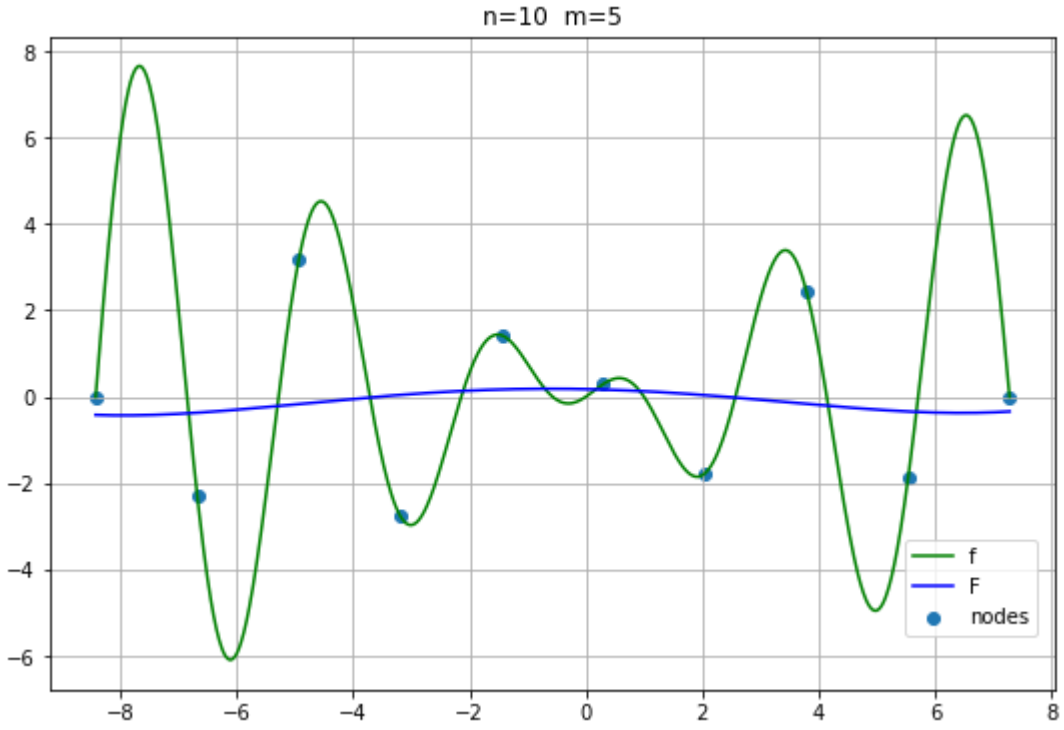
Wektor **A** jest wektorem współczynników wielomianu stopnia **m**

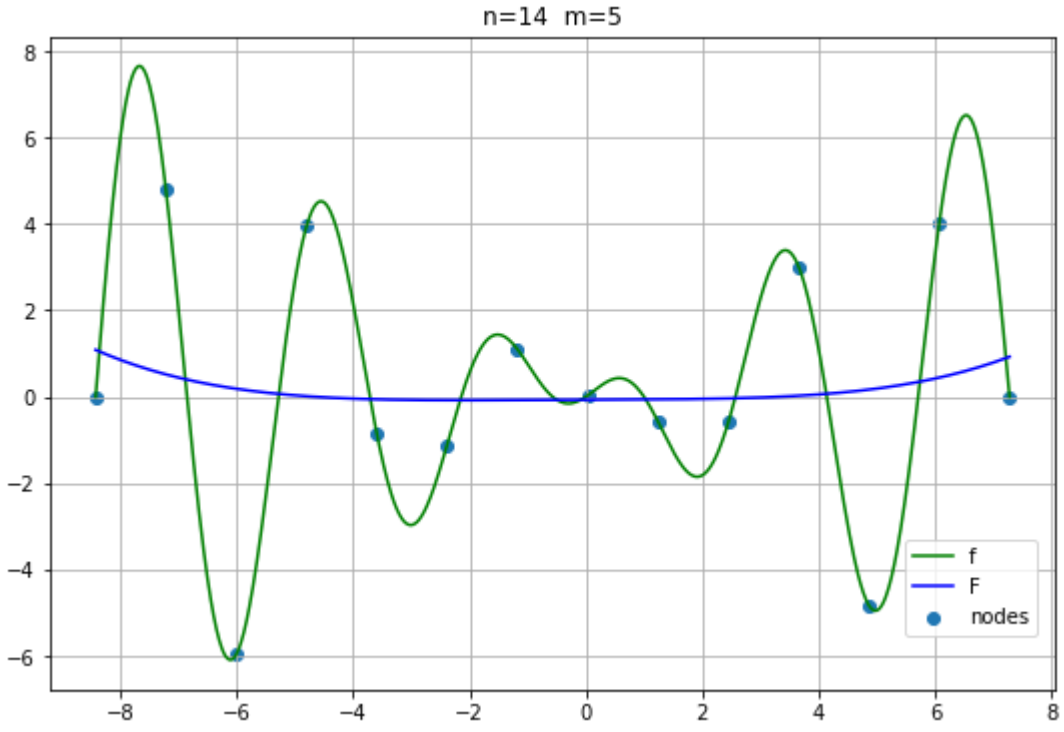
Wyniki doświadczenia:

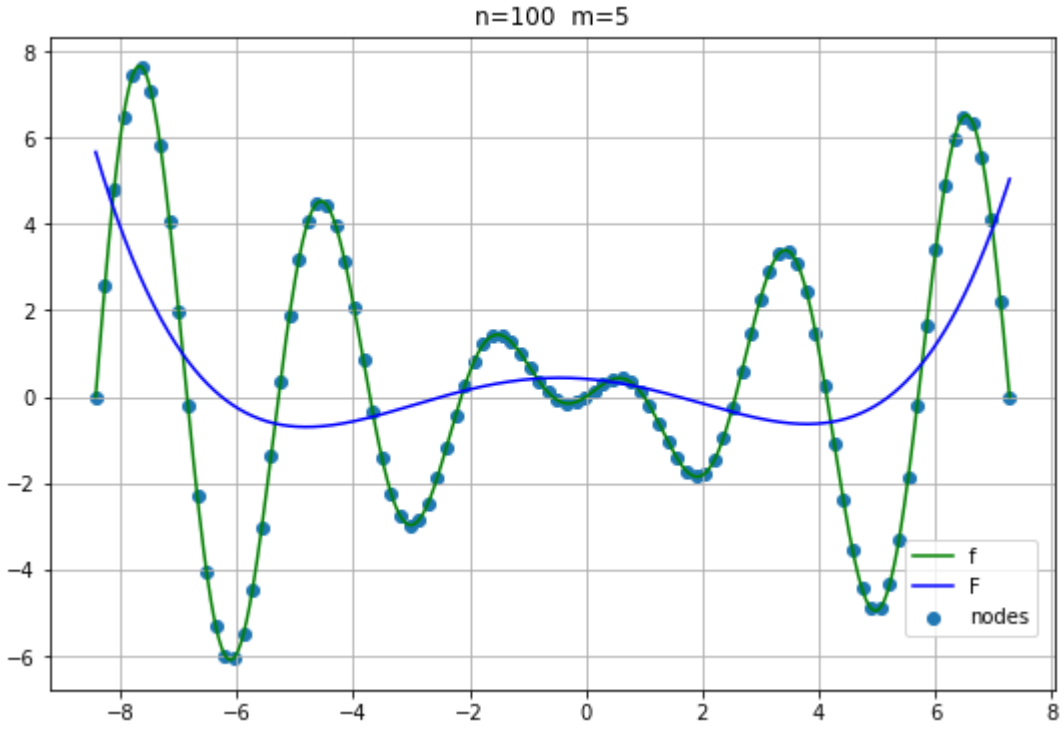
Dla stałego **m** równego 5.

Liczba węzłów	stopień wielomianu	sse	max
7	5	20642.685	10.765
10	5	10843.750	8.083
12	5	10037.369	7.362
14	5	9550.142	6.959
20	5	8753.656	6.297
25	5	8429.513	6.191
30	5	8237.270	6.143
50	5	7929.253	6.038
70	5	7836.692	5.991




n=10 m=5
f
F
nodes

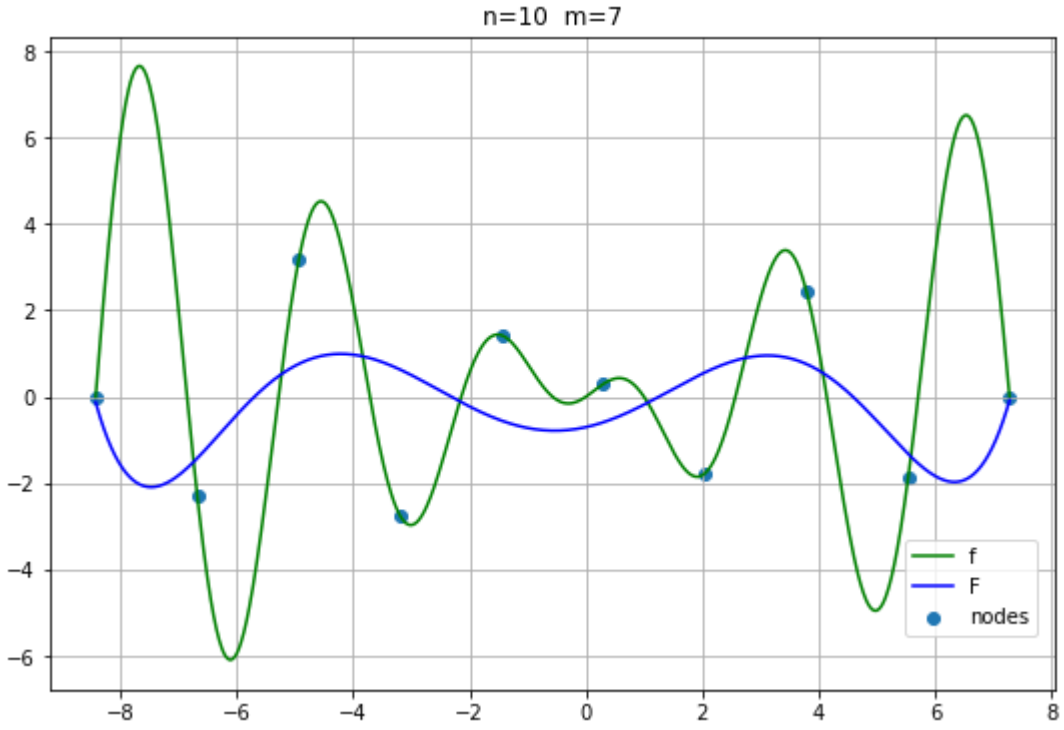

n=14 m=5
f
F
nodes

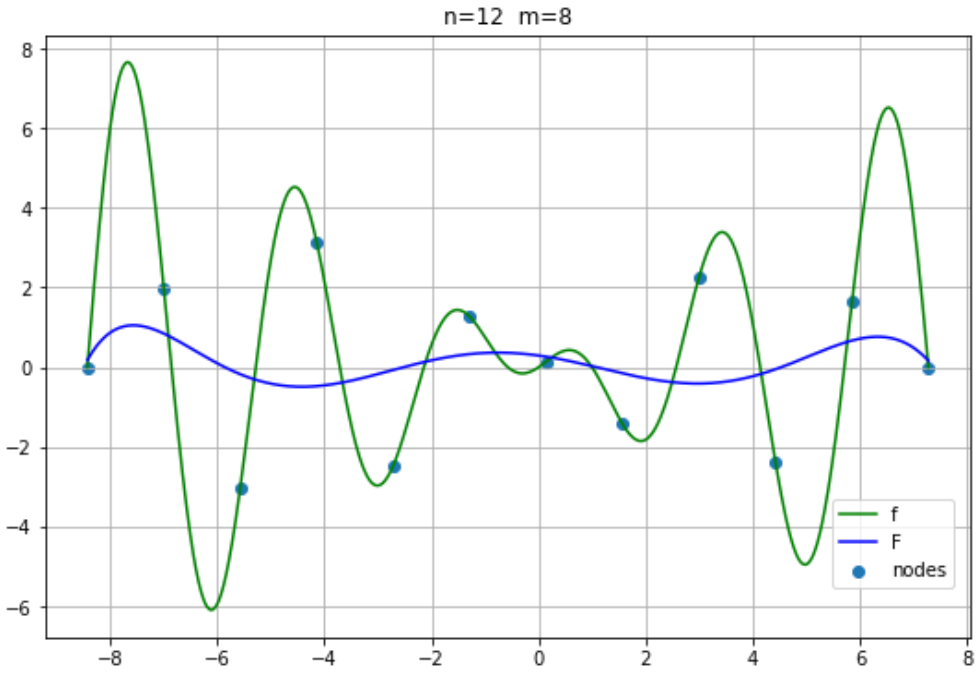


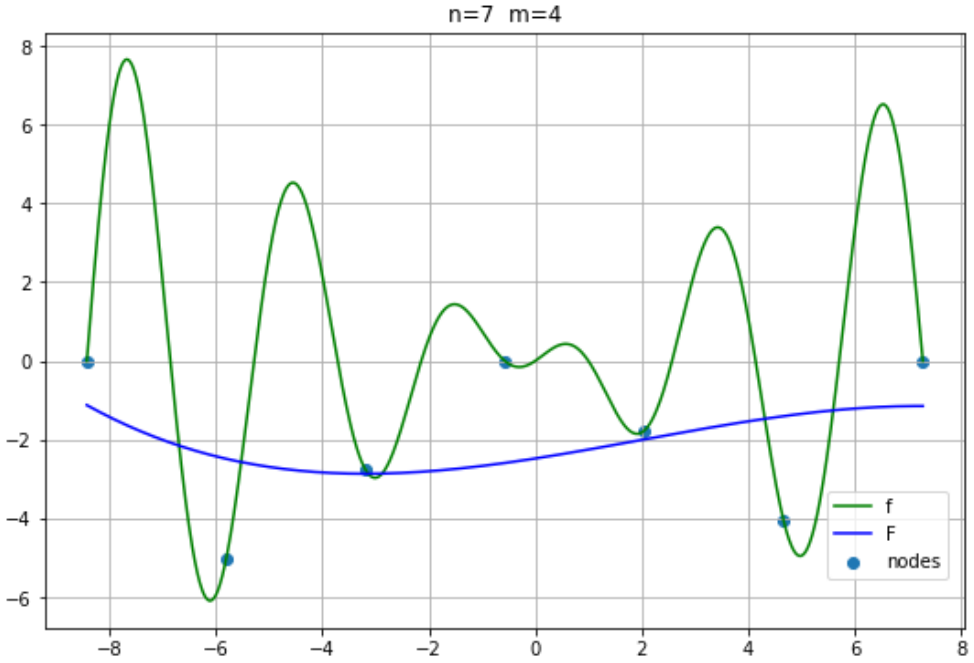
Dla m stałego równego 5 zauważamy, że zbyt stopień wielomianu jest zbyt mały, by dokładnie aproksymować funkcję, dlatego sprawdzamy przypadki w których m jest zależne od n.

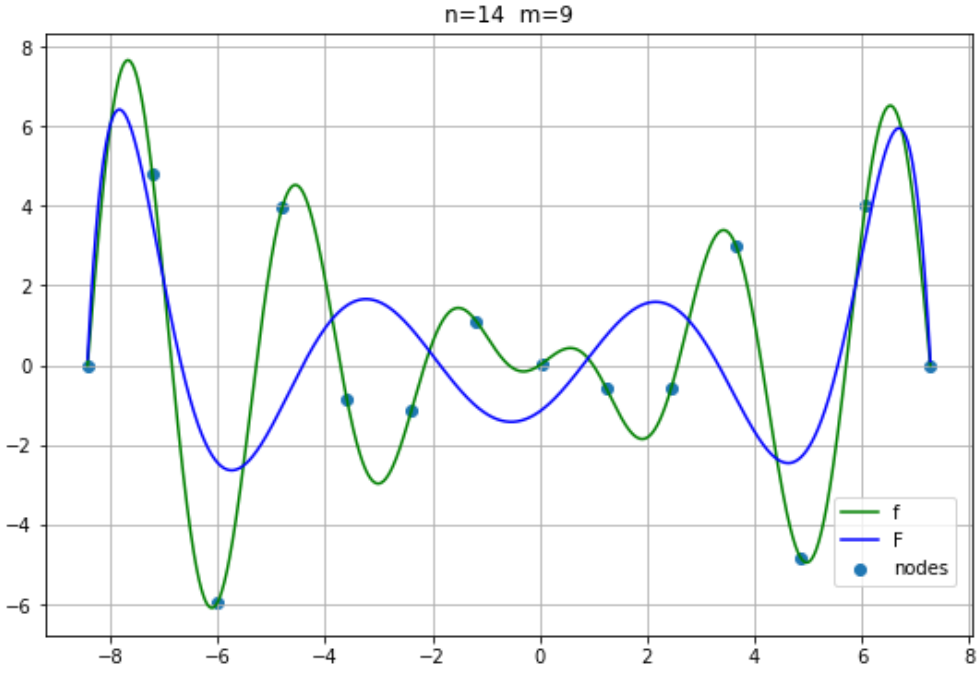
Dla m = floor(0.7 * n)

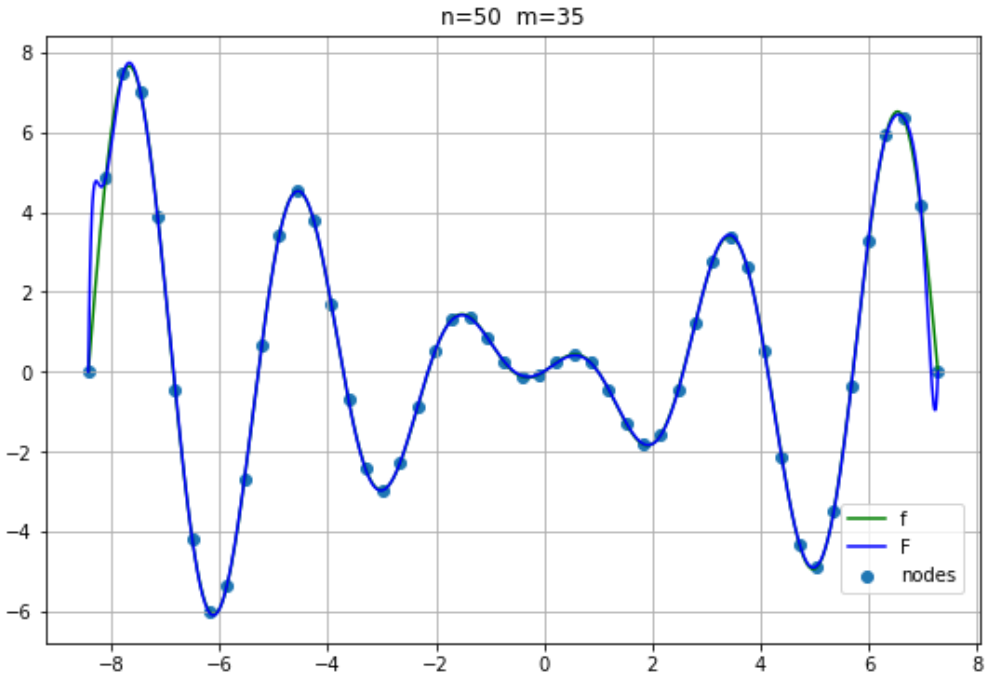
Liczba węzłów	stopień wielomianu	sse	max
5	3	14094.671	7.830
7	4	16321.011	9.295
10	7	13261.349	9.688
12	8	9630.386	6.611
14	9	5617.683	5.080
20	14	1341.577	3.053
25	17	4964.192	12.763
30	21	1561.103	7.988
50	35	111.044	2.960
70	49	24.908	1.780
100	70	106545.253	154.662

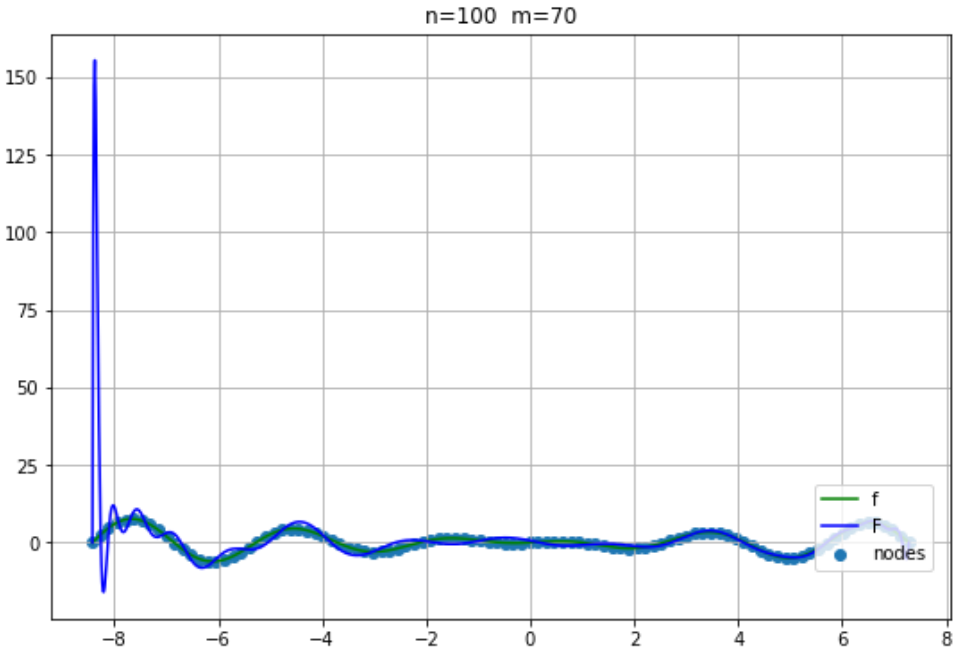



n=12 m=8
f
F
nodes


n=7 m=4
f
F
nodes


n=14 m=9
f
F
nodes

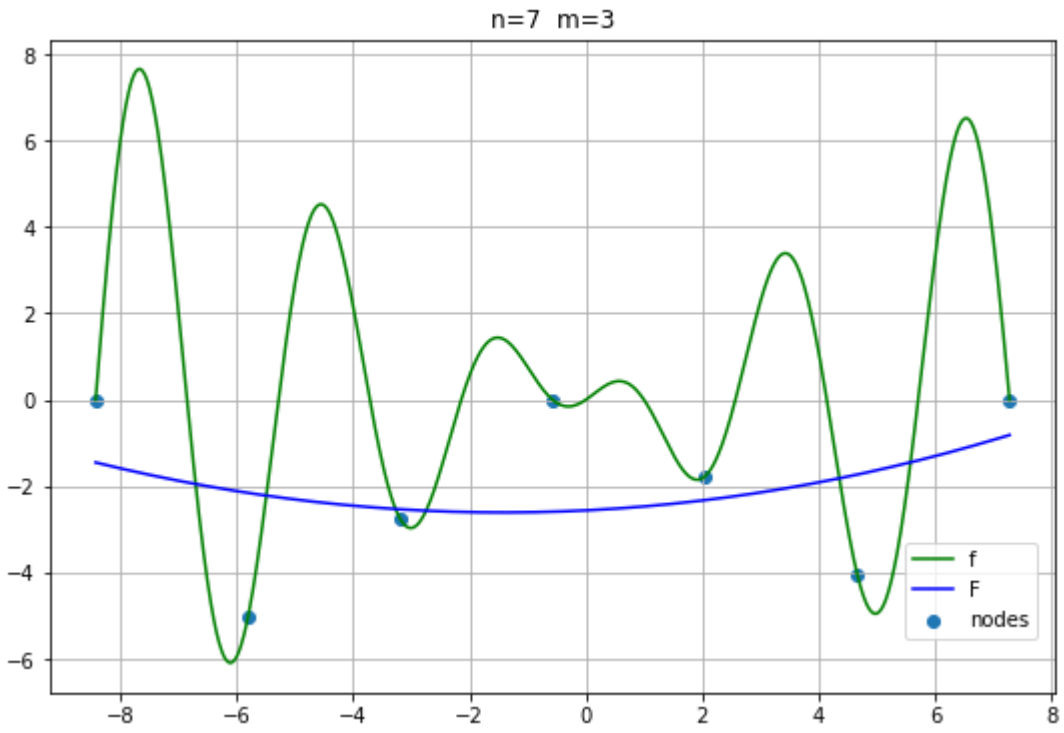


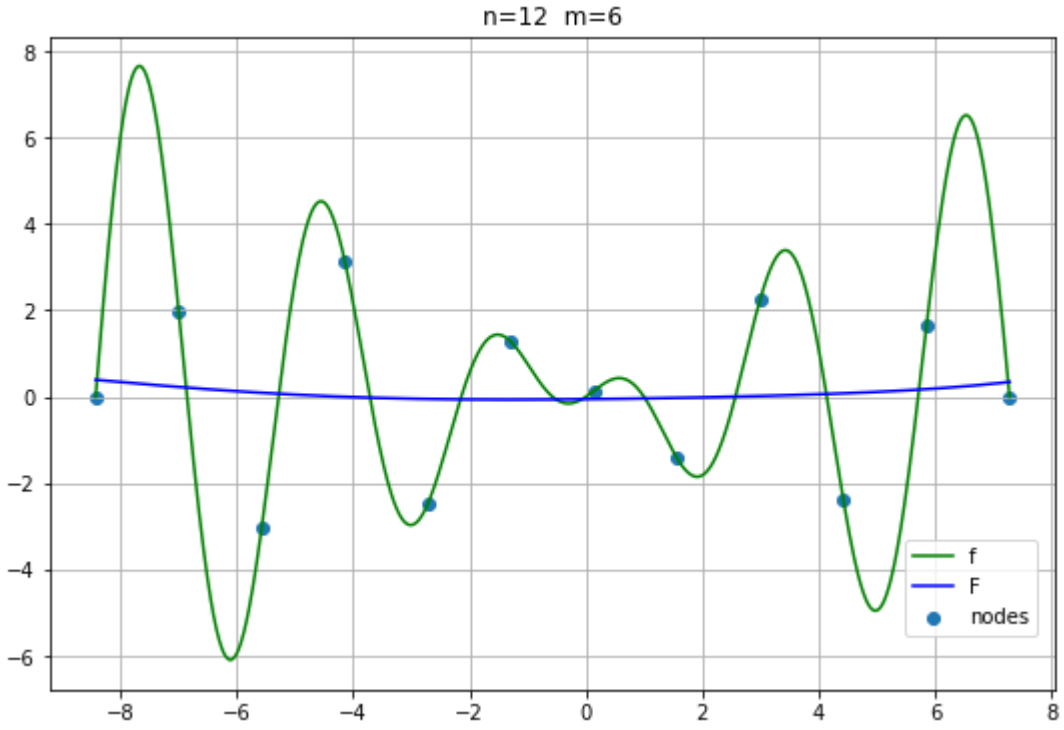


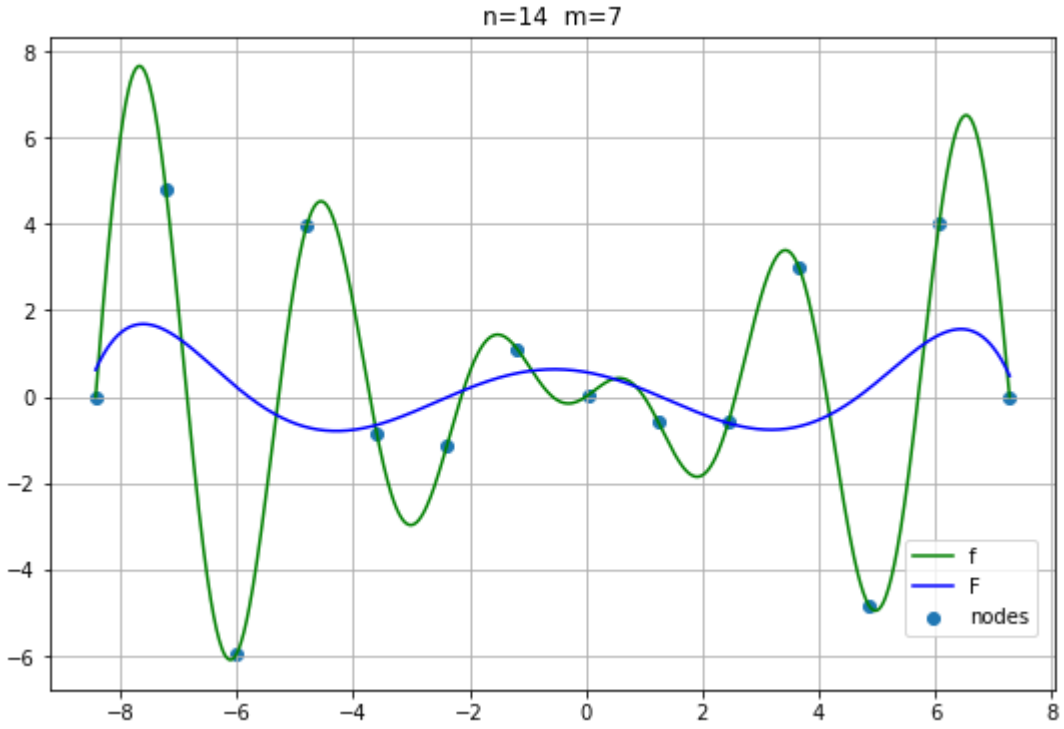
W przypadku m = floor(0.7 * n), widzimy znaczną poprawę w dokładności interpolowanej funkcji w porównaniu do poprzedniego przypadku, natomiast problematyczne stają się przypadki dla dużego n (ok 100), wtedy występuje efekt Runge'go.
Najdokładniejsze wyniki otrzymujemy dla n = 50 oraz m = 35.

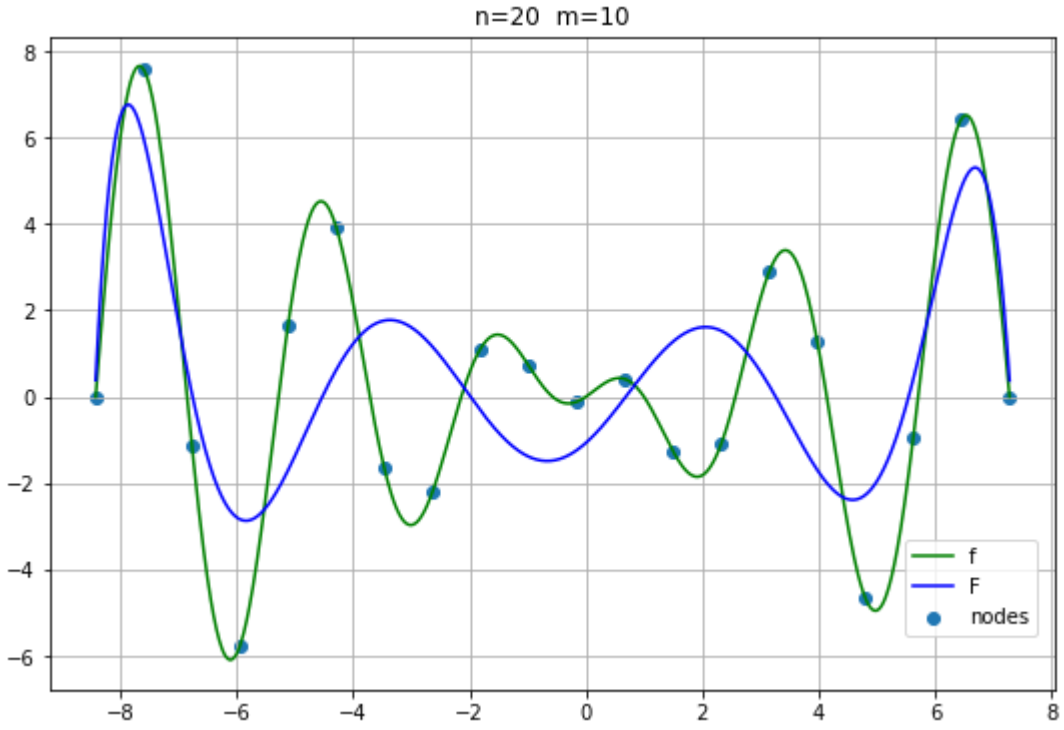
Dla m = floor(0.5 * n)

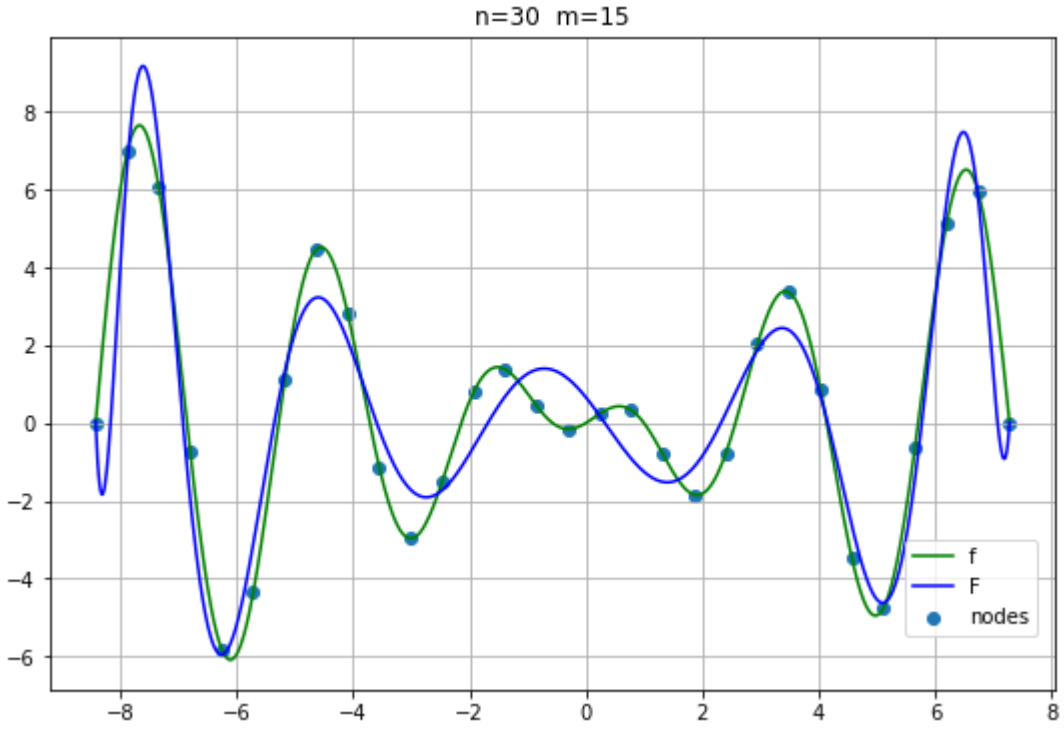
Liczba węzłów	stopień wielomianu	sse	max
5	2	11270.002	7.821
7	3	16213.258	9.343
10	5	10843.750	8.083
12	6	10037.456	7.353
14	7	9221.092	6.460
20	10	5551.987	4.810
25	12	3559.723	5.115
30	15	1170.034	4.278
50	25	22.001	1.113
70	35	1276.521	12.630
100	50	4.915	0.775

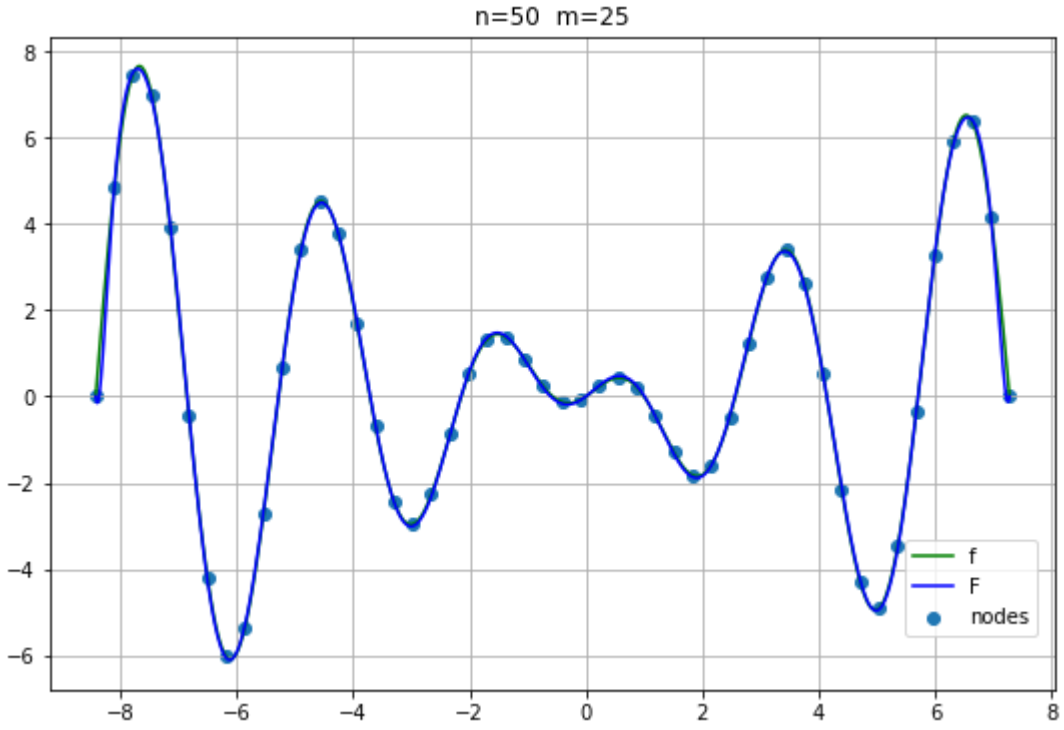


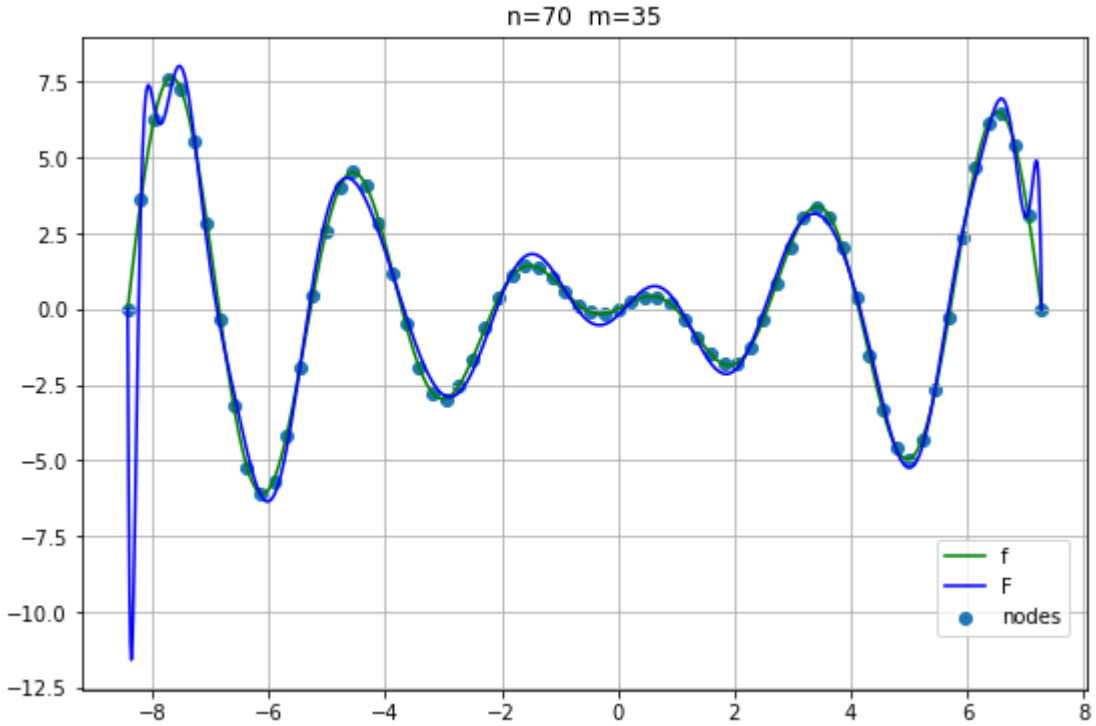

n=12 m=6
f
F
nodes

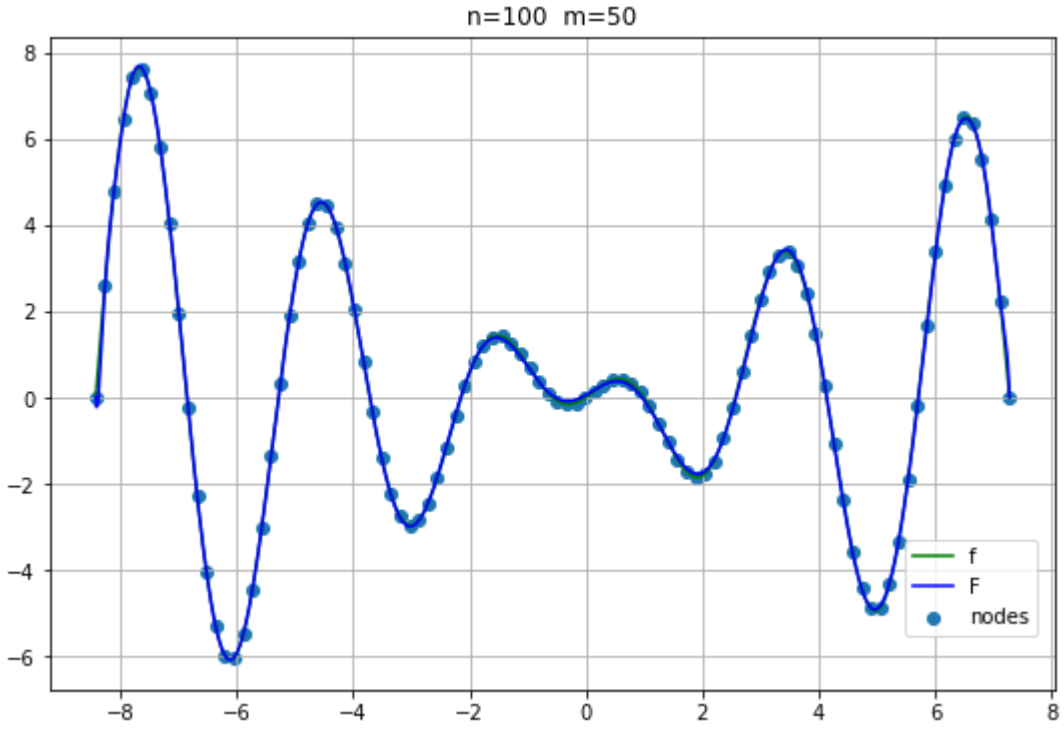

n=14 m=7
f
F
nodes


n=20 m=10
f
F
nodes


n=30 m=15
f
F
nodes


n=50 m=25
f
F
nodes

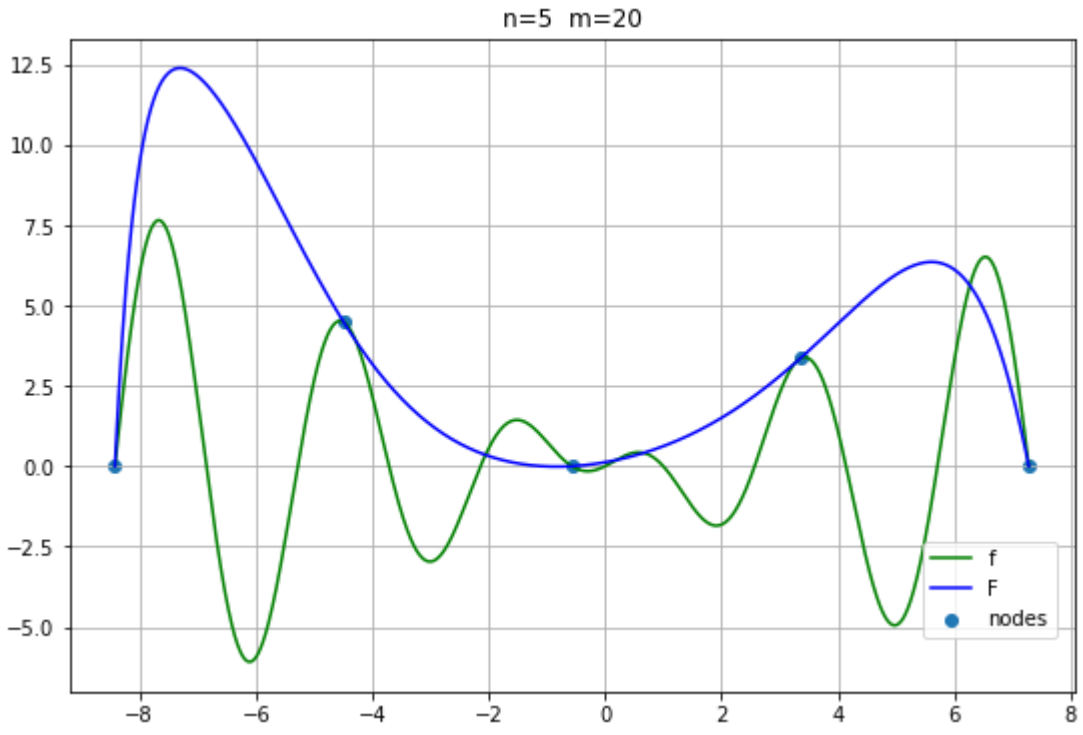

n=70 m=35
f
F
nodes

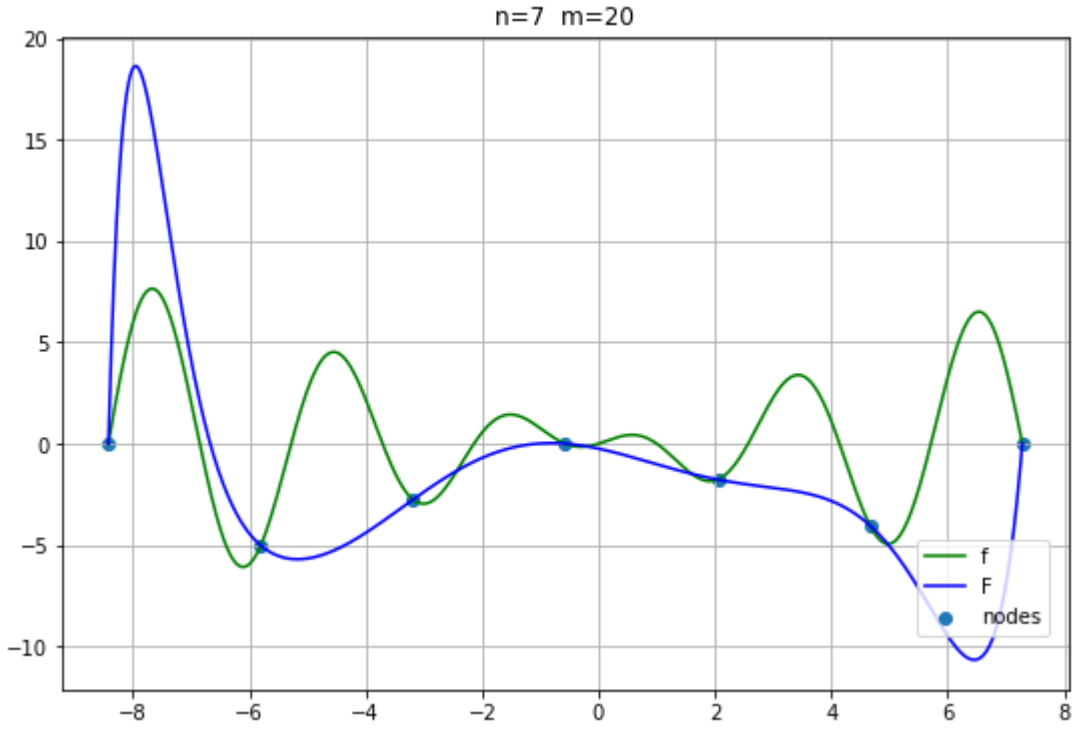


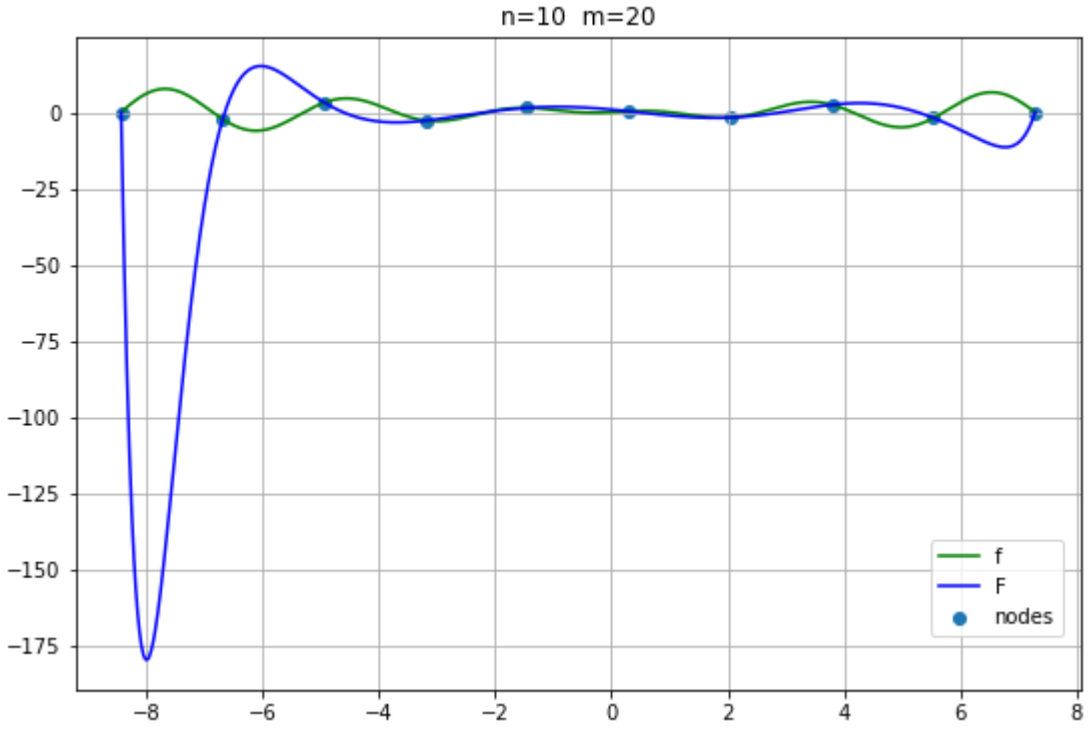
Podobnie jak w poprzednim przypadku, dla m = floor(0.5 * n), efekt Runge'go występuje, lecz szybko znika dla n = około 70, a dla większych n aproksymacja jest dokładna, a zwiększenie n jest równoważne ze zwiększeniem dokładności aproksymacji.

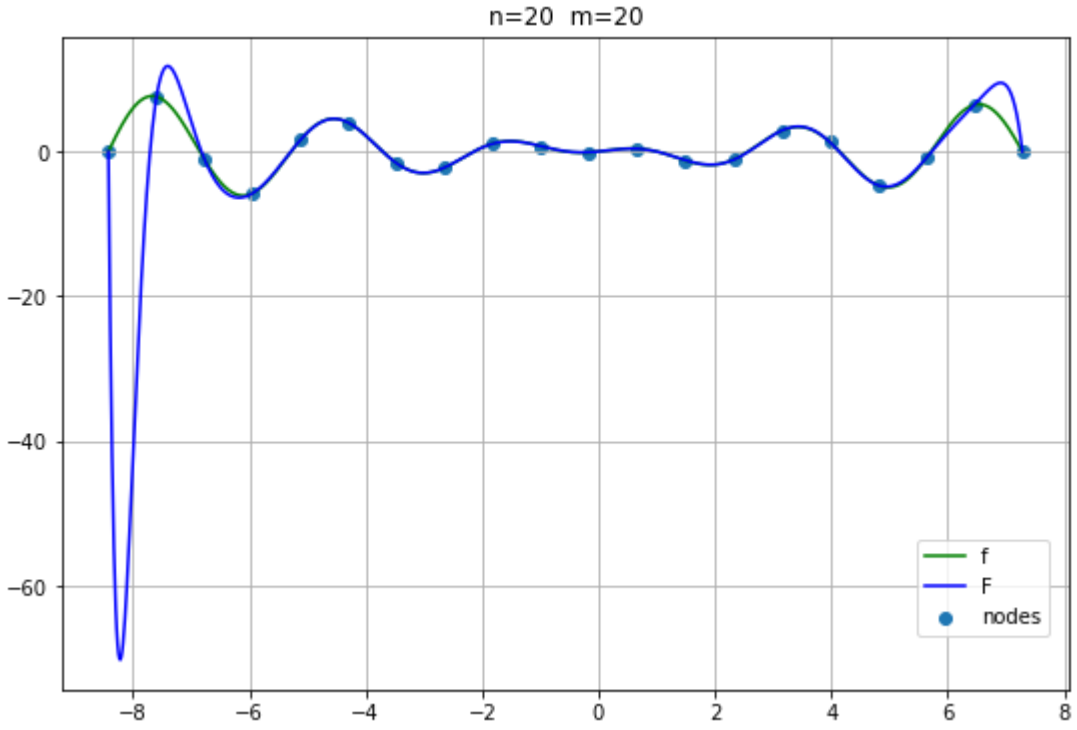
Dla m stałego równego 20

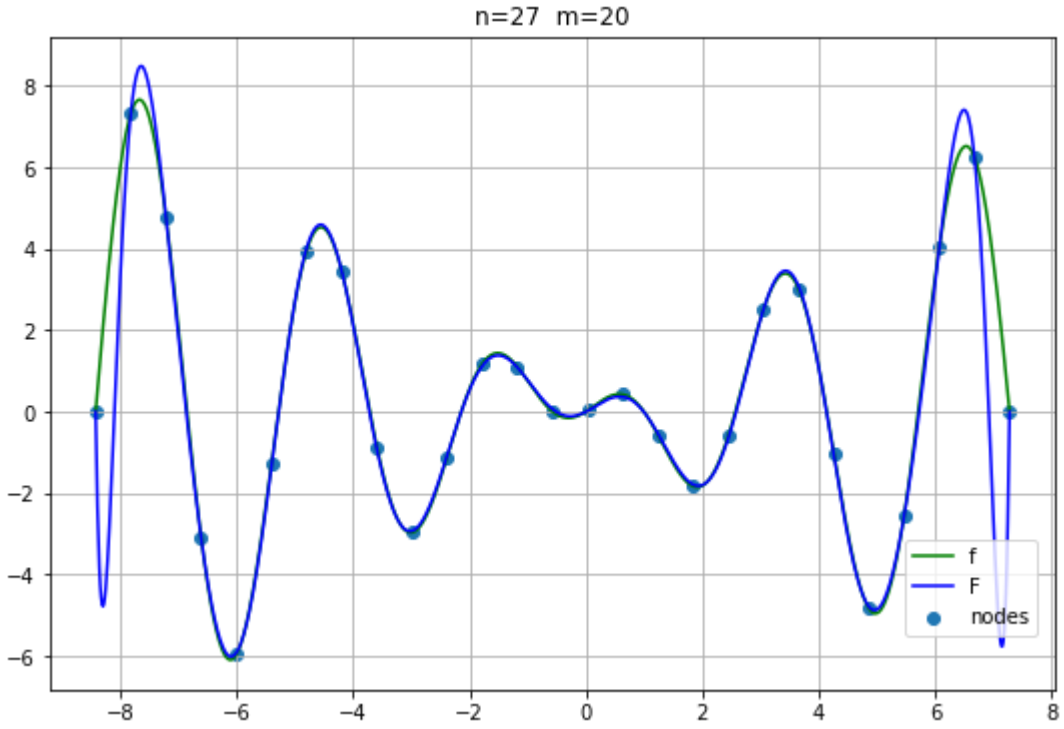
Liczba węzłów	stopień wielomianu	sse	max
21	20	636332.812	166.500
25	20	5939.053	16.653
30	20	550.013	4.356
50	20	24.110	0.771
70	20	18.340	0.453
100	20	17.567	0.354

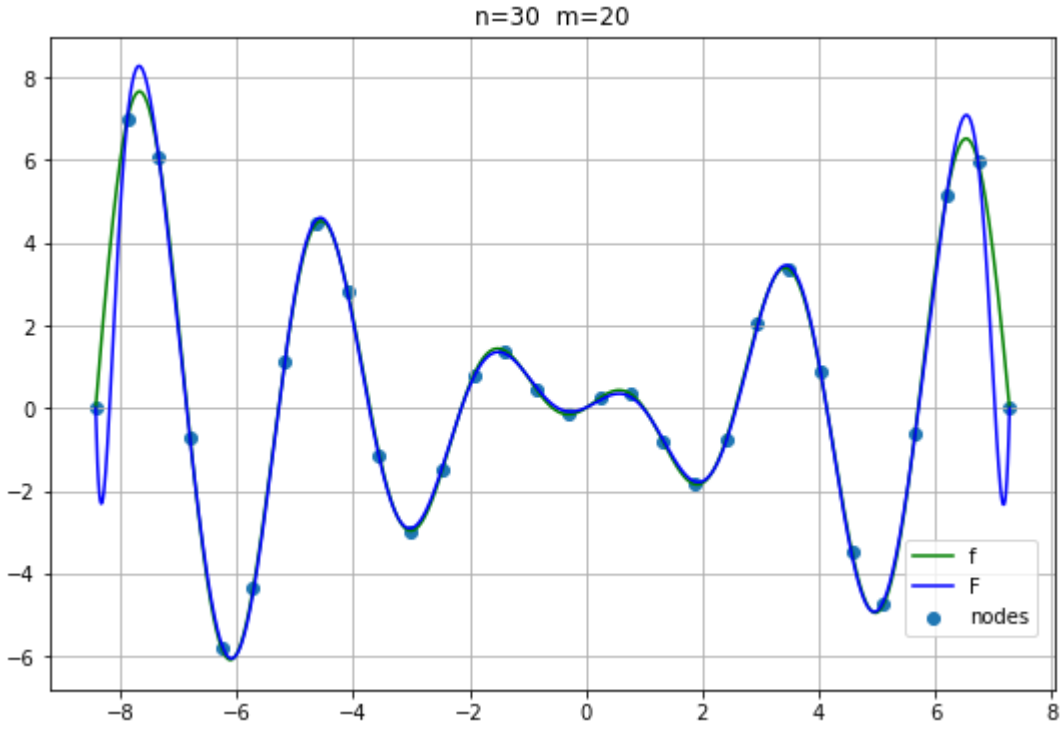


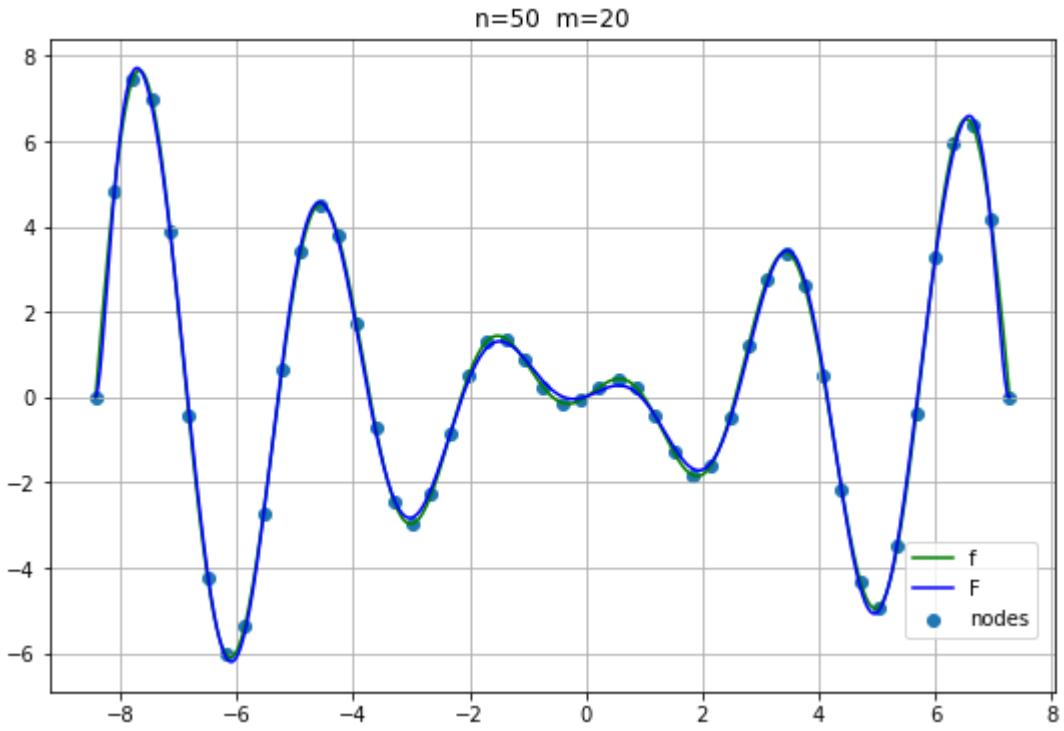

n=7 m=20
20
15
10
5
0
−5
−10
−8
−6
−4
−2
0
2
4
6
8
f
F
nodes

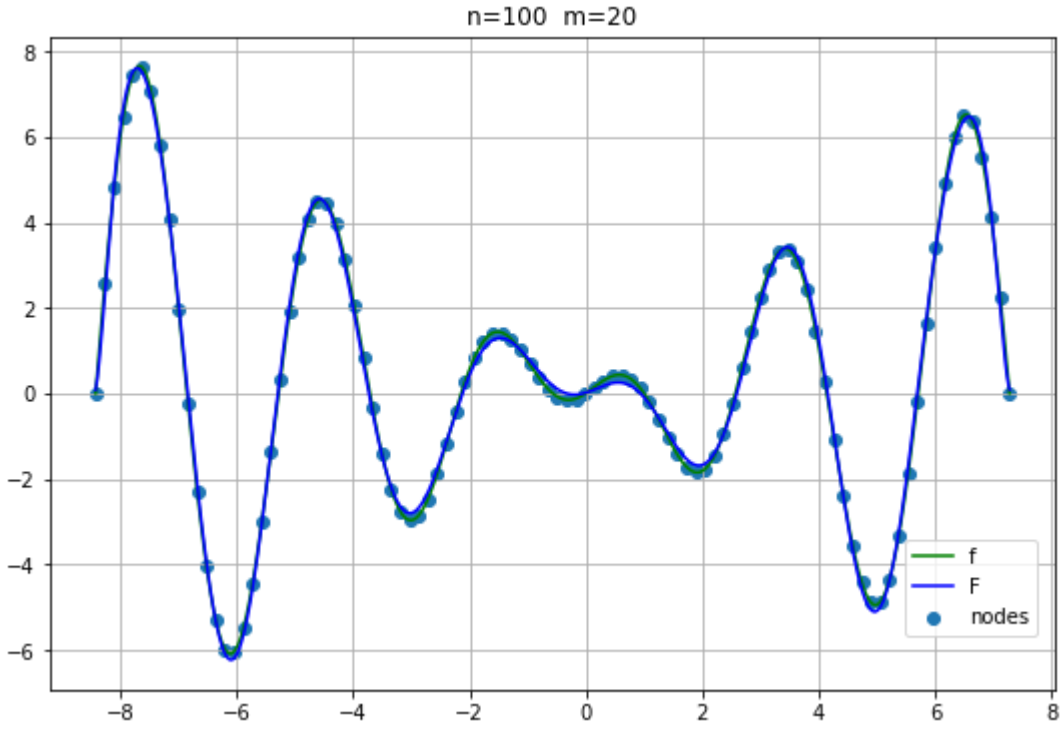

n=10 m=20
0
−25
−50
−75
−100
−125
−150
−175
−8
−6
−4
−2
0
2
4
6
8
f
F
nodes


n=20 m=20
0
−20
−40
−60
−8
−6
−4
−2
0
2
4
6
8
f
F
nodes


n=27 m=20
8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8
f
F
nodes

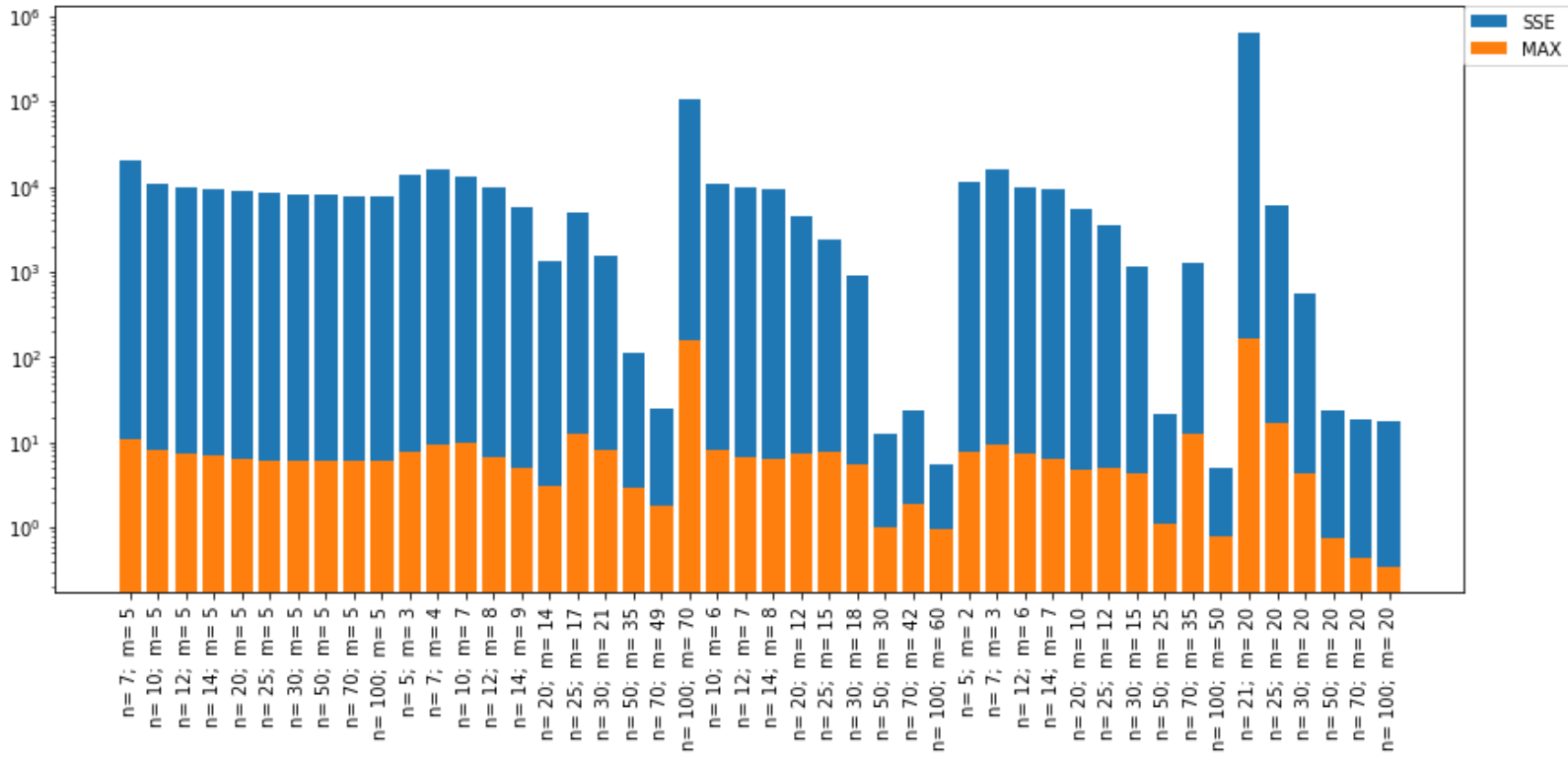

n=30 m=20
f
F
nodes


n=50 m=20
f
F
nodes



Przypadek m stałe równe 20, jest o tyle ciekawy, że wielomian aproksymacyjny jest dokładny dla dużych n, a dla małych występują duże różnice spowodowane efektem Runge'go. Zaczyna on być dokładny mniej więcej w tym samym czasie co poprzedni przypadek m = floor(0.5 * n) natomiast dla wcześniejszych jest on niedokładny przez efekt Runge'go.

Zebrane statystyki dla testowanych m oraz n



Wnioski:

- Używając aproksymacji, musimy uważnie analizować wpływ hiperparametrów (n oraz m) na dokładność aproksymacji, aby dobrać najlepsze dla posiadanych danych wyniki.
- Dla takicih samych wartości n zwiększenie m , oraz odwrotnie dla jednakowych m zwiększenie n, często niekorzystnie wpływa na wynik aproksymacji.
- Dobierając odpowiednie m, stałe dla zadanej funkcji, jesteśmy w stanie otrzymać dokładnie wyniki (m stałe równe 20) dopiero od pewnego momentu, od pewnej wartości n.
- Dobierając odpowiedni współczynnik liniowej transofmacji n na m, czyli np. m = floor(0.5 * n), jesteśmy w stanie uzyskać poziom błędu SSE który jest odwrotnie proporcjonalny do przyrostu n.
- Aproksymacja, dla punktów dyskretyzacji rozłożonych symetrycznie względem osi X, da nam wynik przybliżony do funkcji **f(x) = 0**