Реализация метода минимизации остаточного члена интерполирования в окрестности некоторой точки при заданных узлах интерполирования

Задача:1.1.7(б)

Пусть существует некоторая функция $f(x)$ и дано n узлов, в которых определены значения данной функции. Будем считать, $f(x)$ и ее $(n+1)$ производные непрерывны на промежутке $[a;b]$, на котором берутся n узлов.

По данным n точкам строится интерполяционный полином:

$$P_n(x) = f(x_0) + (x - x_0)f(x_0, x_1) + \cdots$$
$$+ (x - x_0)(x - x_1) \dots (x - x_{n-1})f(x_0, \dots, x_n)$$

Отсюда следует, что $P_n(x_i) = f(x_i), i = 0, \dots, n$

Поскольку $f(x)$ и ее $(n+1)$ производные непрерывны на $[a;b]$, отсюда следует, что для любого x из $[a;b]$ существует $\varepsilon = \varepsilon(x)$из$[a;b]$ такой, что

Остаточный многочлен в форме Лагранжа:

$R_n(x;f) = f(x) - P_n(x) = \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\varepsilon)$, где $\omega(x) = (x - x_0) \dots (x - x_n)$

Далее рассмотрим задачу, которую необходимо решить.

Дано N точек $\big(x_i, f(x_i)\big)$, где $i = 0, \dots, N$, причем $N > n$.

Мы можем оценить остаточный член следующим образом:

$$R_n(x;f) = \frac{\omega(x)}{(n+1)!} f^{(n+1)}(\varepsilon), \text{отсюда следует, что}$$

$$\left|f^{(n+1)}(\varepsilon)\right| \le M, \text{где } M = \max_{x_0,\dots,x_n} |f^{(n+1)}(x)|$$

Отсюда следует, что $|R_n(x;f)| \le \frac{|\omega(x)|}{(n+1)!} M$

Решаемая мною задача – минимизация остаточного члена, то есть $\min|R_n(x;f)|$ равносильно $\min|\omega(x)|$

В моем случае необходимо среди N заданных точек выбрать n точек, чтобы $\mathrm{R}_n(x;f)$ был минимален.

Будем выбирать n ближайших точек вблизи окрестности некоторого выбранного узла x_*, то есть:

$$x_0 : |x_* - x_0| = \min_{k:x_0,\dots,x_N} |x_* - x_k| => x_1 : |x_* - x_1| = \min_{x_k \neq x_0} |x_* - x_k| => \cdots$$

Для удобства в реализации будем рассматривать окрестности каждой из N точек и будем сравнивать полученные результаты.

Программная реализация:

Мною была реализована функция в среде $Wolfram\ Mathematica$.

```
w = Product[#1 - i, {i, #2}] &;

nearestDots[dots_, xSelected_, epsilon_] := Module[
  {chDots = {}},

  Do[AppendTo[chDots, First@Sort[Select[dots, FreeQ[chDots, #] &], Abs[xSelected - #1] < Abs[xSelected - #2] &]],
   {j, epsilon }];
  {chDots, xSelected}
 ]

absW[g_] := Abs[w[g, #[[1]]]] & /@ allCombos;

fd = Sort@RandomReal[ab, nN];
allCombos = nearestDots[fd, #, n] & /@ fd;
maxM = Max[Max[Sequence @@ #] & /@ D[f[g], {g, n + 1}] /. g → # & /@ allCombos];
toPlot = Sort[absW[g], MaxValue[#1, g] < MaxValue[#2, g] &];
Plot[toPlot, {g, ab[[1]], ab[[2]]}]
```

Данный алгоритм строит всевозможные наборы в окрестностях каждой из N точек и выбирает среди всех случаев такую $g(x) = \frac{|\omega(x)|}{(n+1)!}M$, скорость роста которой медленнее всех, но поскольку $\frac{M}{(n+1)!}$ в каждой из функций постоянны, следовательно, достаточно сравнивать скорости роста функций $|\omega(x)|$.

Правильность работы алгоритма нужно проверять на достаточно больших N по сравнению с числом n

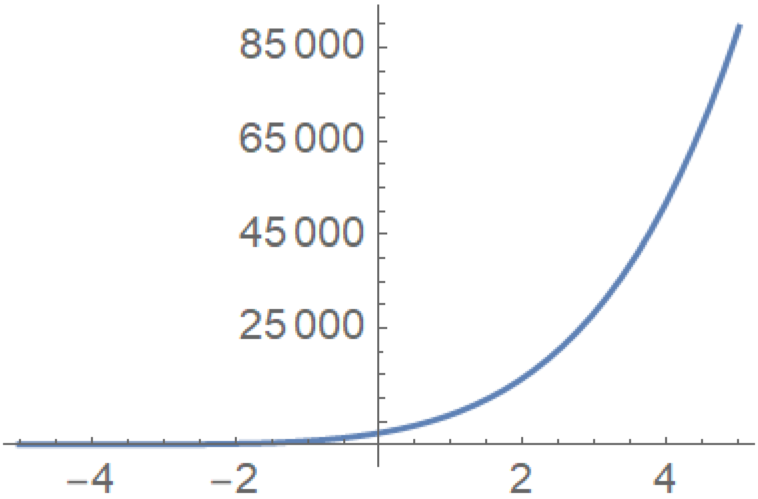
Рассмотрим несколько примеров:

1) Пусть $f(x) = x^{15} - 15x^5 + 10x^3 + 15$ рассматривается на промежутке $[-5; 5]$ и $N = 70$, причем случайным образом выбирается N узлов на промежутке $[-5; 5]$.
   Для начала пусть $n = 5$. Рассмотрим полученный результат – минимальный остаточный многочлен и сравним его с графиками других функций, которые были претендентами среди всех наборов n узлов.
   Пусть N точек выглядят так:

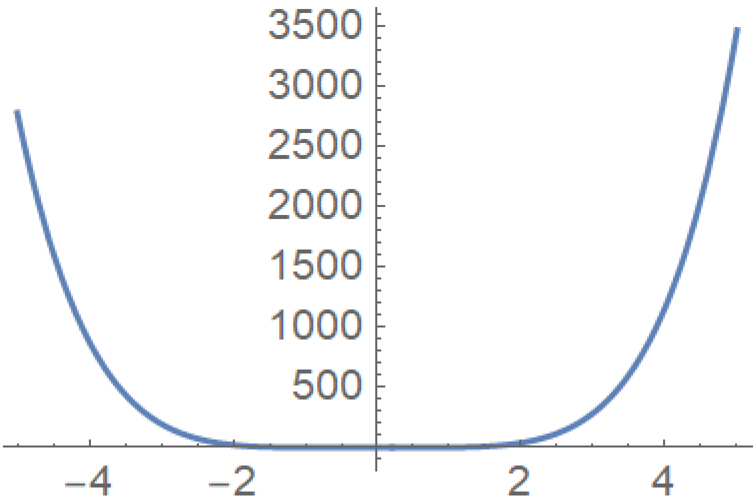
```
{-4.9684, -4.95602, -4.87543, -4.74258, -4.36707, -4.15867, -4.14069, -3.94438, -3.83928,
 -3.51029, -3.47454, -2.89097, -2.67818, -2.5985, -2.51428, -2.51, -2.46656, -2.41459, -2.40123,
 -2.39528, -2.25448, -2.18239, -2.14927, -1.84896, -1.84024, -1.70951, -1.56566, -1.3691, -1.02506,
 -0.977031, -0.627193, -0.612623, -0.568699, -0.377827, -0.250005, 0.00927247, 0.0287433, 0.0383774
 0.0917048, 0.134992, 0.228433, 0.27314, 0.275441, 0.472859, 0.715124, 1.30135, 1.32573, 1.60514,
 1.73234, 2.28559, 2.39096, 2.48367, 2.52156, 2.61396, 2.62311, 2.70531, 2.88078, 2.95458, 3.00637,
 3.09621, 3.14537, 3.38391, 3.47241, 4.08027, 4.16284, 4.20239, 4.21212, 4.33707, 4.6019, 4.9343}
```

   Рассмотрим несколько графиков для различных точек, в окрестности которой выбираются n точек:

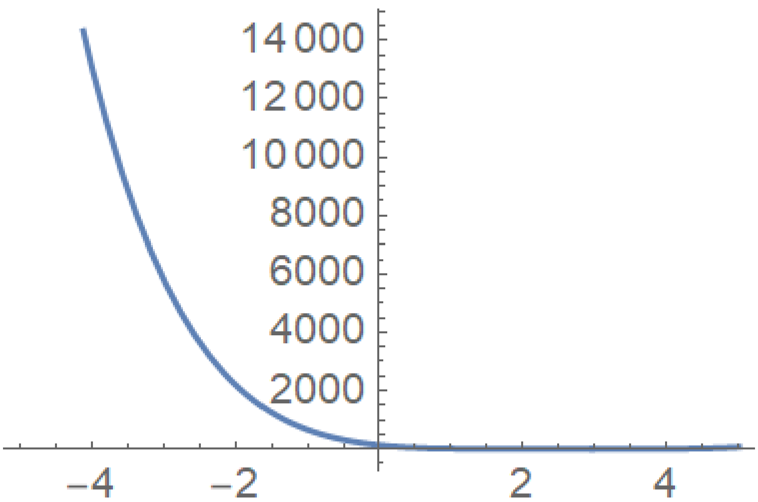
(стоит отметить, что в дальнейших примерах справа от графика располагается точка, в окрестности которой выбирались n узлов)



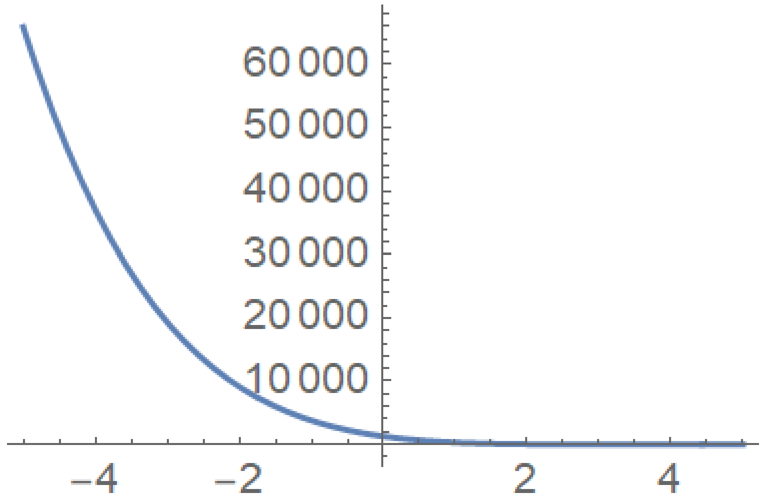
−4.87543



−0.250005

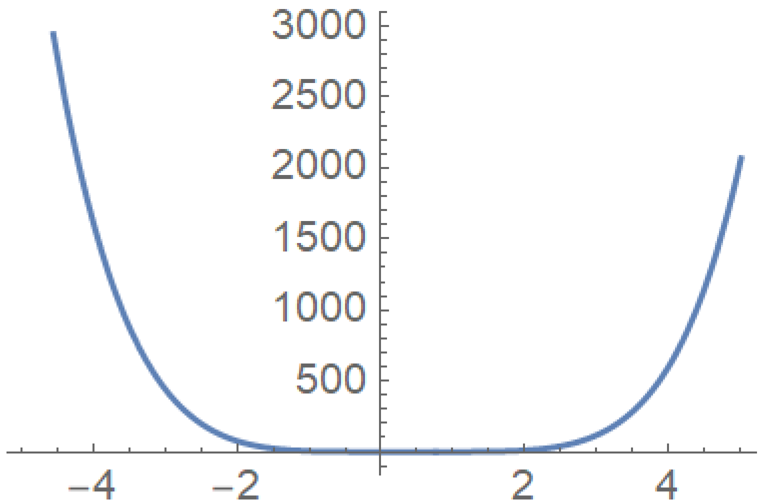


2.70531



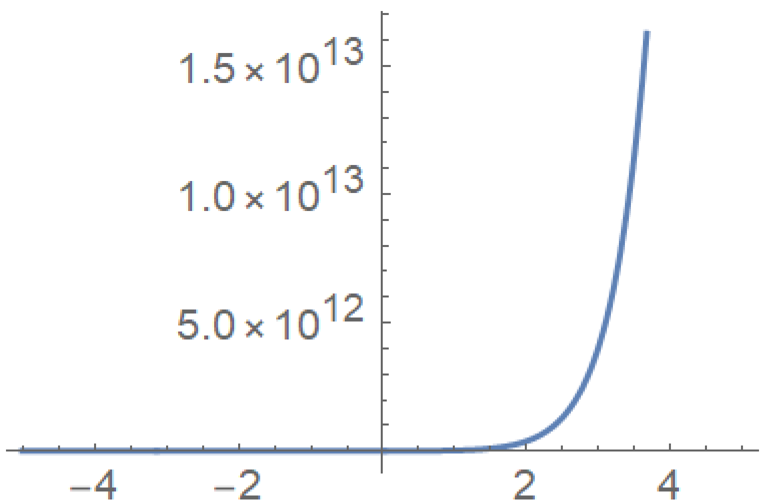
4.21212

Программа посчитала лучшим в окрестности точки $x = 0.715124$:

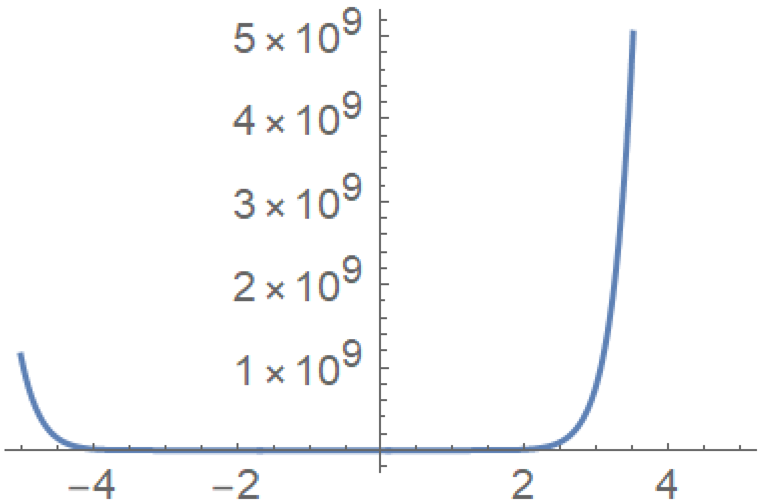


0.715124

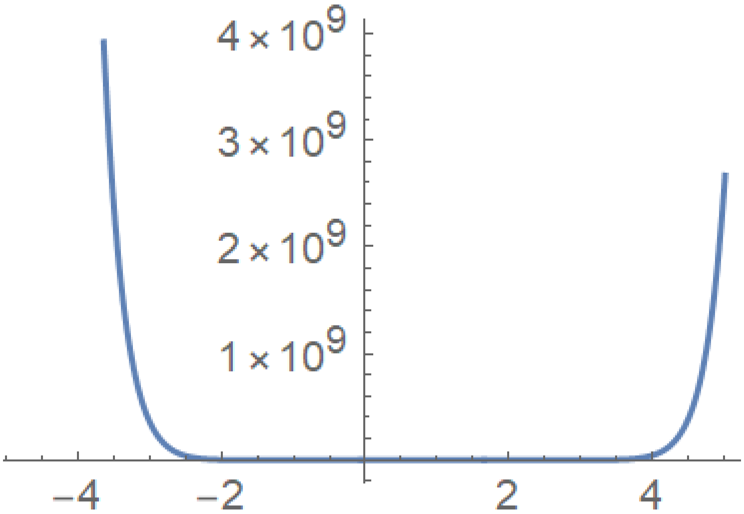
Пусть $n = 15$. Рассмотрим несколько графиков:



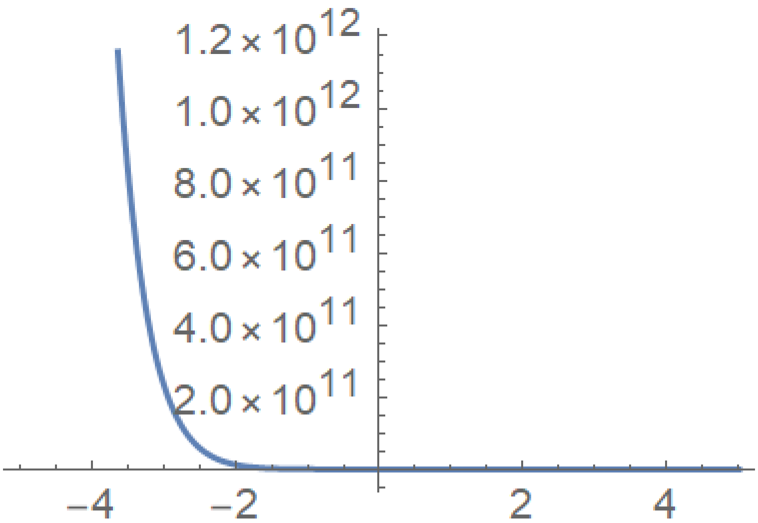
-4.81402



−1.11612

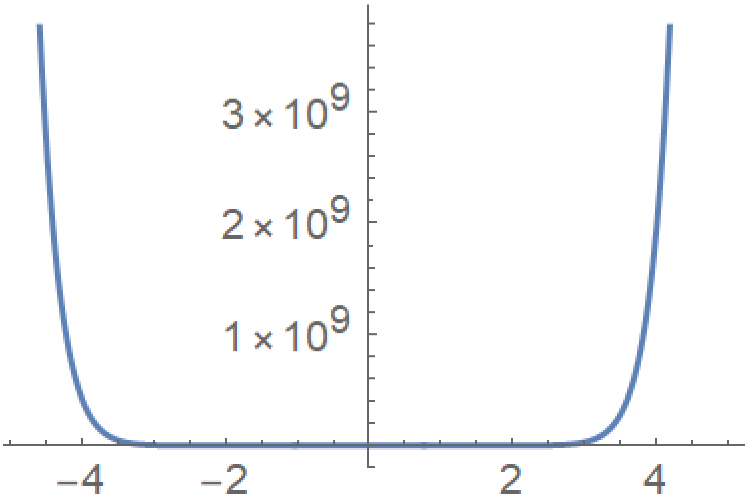


0.777958



2.86422

Программа посчитала результат в окрестности $x = -0.24187$ наилучшим.

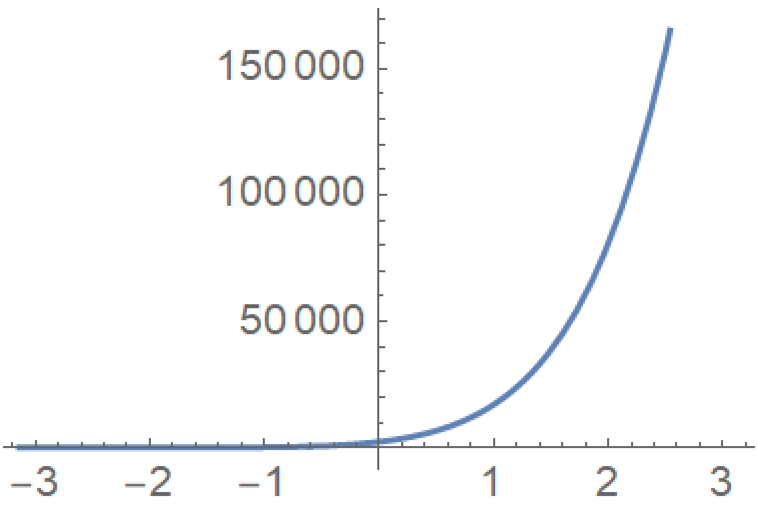


−0.24187

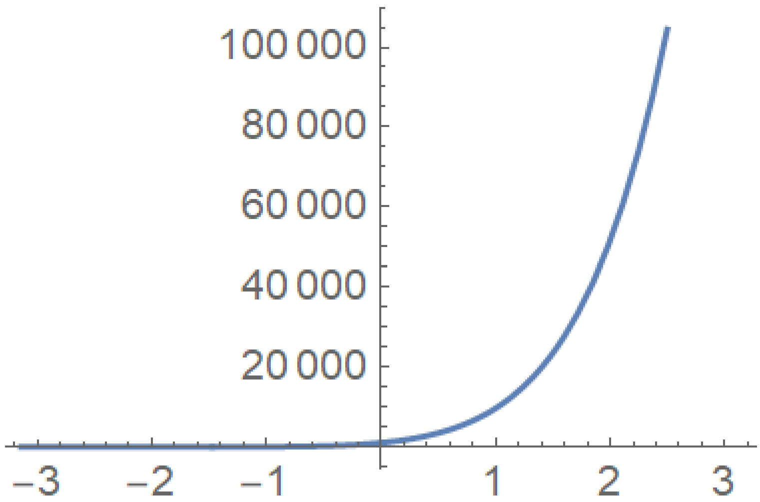
2) Пусть $f(x) = 1.5\cos(300x) + 4\sin(4x) + \sin(25x) + 40 - \frac{x^2}{100}$ и рассмотрим на $[-\pi; \pi]$ и $N = 100$. Данные N узлов также выбраны случайно на промежутке.

```
{-3.13241, -3.09363, -3.06381, -2.9823, -2.91624, -2.90656, -2.88384, -2.84093, -2.77229, -2.77056, -2.73459,
 -2.70749, -2.59089, -2.50737, -2.49078, -2.29597, -2.22899, -2.1474, -2.04234, -1.8977, -1.89694, -1.84303,
 -1.73147, -1.71828, -1.6341, -1.42629, -1.41454, -1.31831, -1.27271, -1.04655, -0.980428, -0.874251, -0.841967,
 -0.8355, -0.795864, -0.795581, -0.785469, -0.76788, -0.765078, -0.758176, -0.584942, -0.495201, -0.447127,
 -0.424769, -0.401843, -0.297206, -0.158699, -0.100556, -0.08912, -0.0822842, -0.0791426, 0.185782, 0.214245,
 0.354376, 0.397673, 0.457682, 0.492345, 0.673201, 0.681596, 0.700483, 0.729606, 0.769102, 0.777609, 0.852768,
 0.927802, 0.966243, 1.00122, 1.06757, 1.14563, 1.18455, 1.19273, 1.25656, 1.26043, 1.52106, 1.53075, 1.6863,
 1.8386, 1.84218, 1.86629, 1.89143, 1.97279, 2.00095, 2.01247, 2.0158, 2.03091, 2.20274, 2.21215, 2.33993,
 2.37993, 2.39439, 2.44896, 2.52543, 2.54681, 2.57789, 2.62444, 2.63692, 2.79558, 2.86045, 2.96683, 3.1128}
```

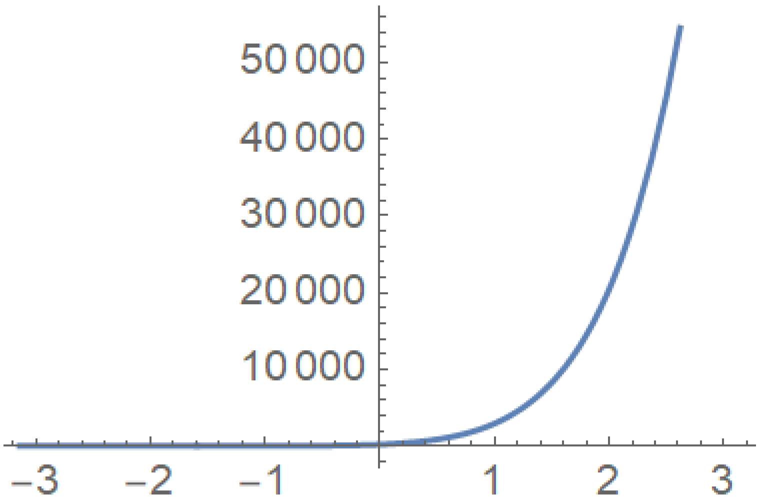
Пусть $n = 7$. Рассмотрим несколько графиков:



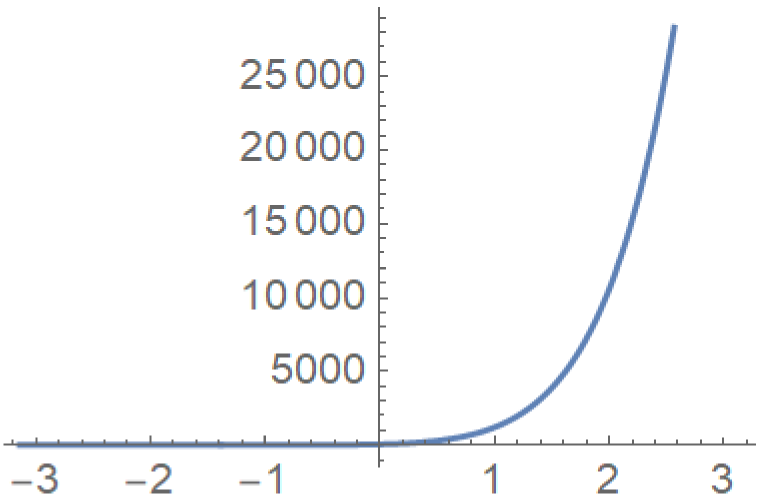
−3.08358



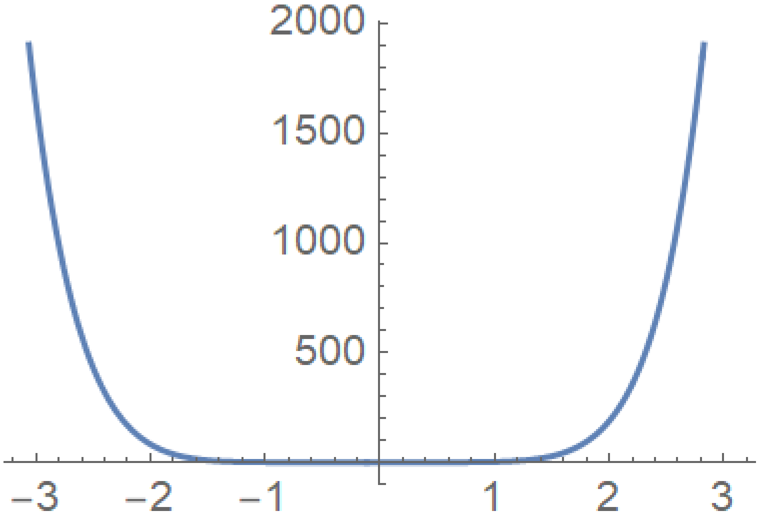
−2.71898



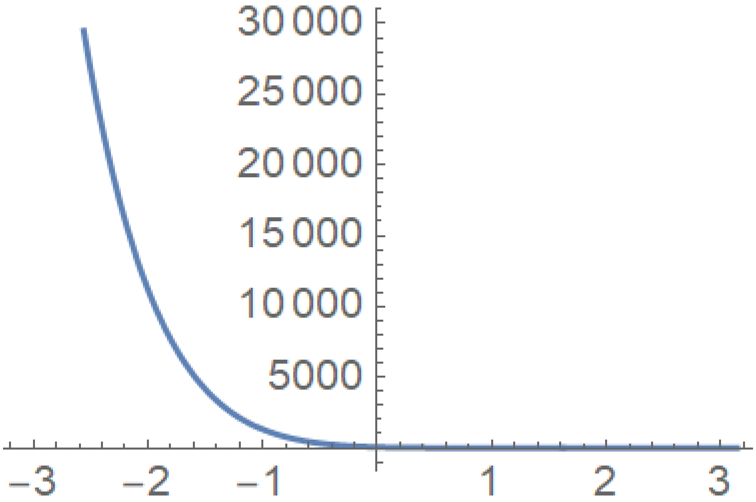
−2.13657



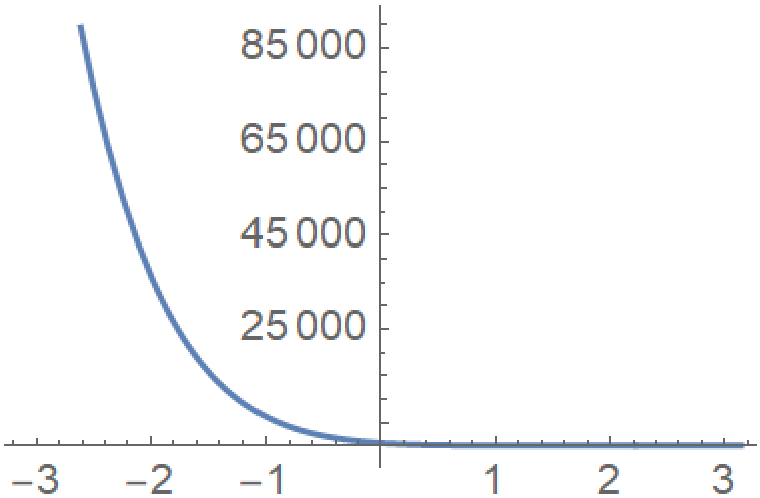
−1.74015



−0.0791656

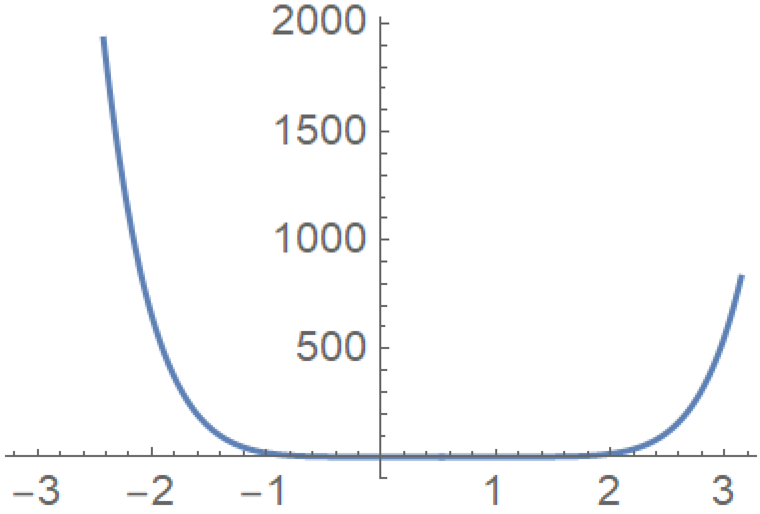


1.77406



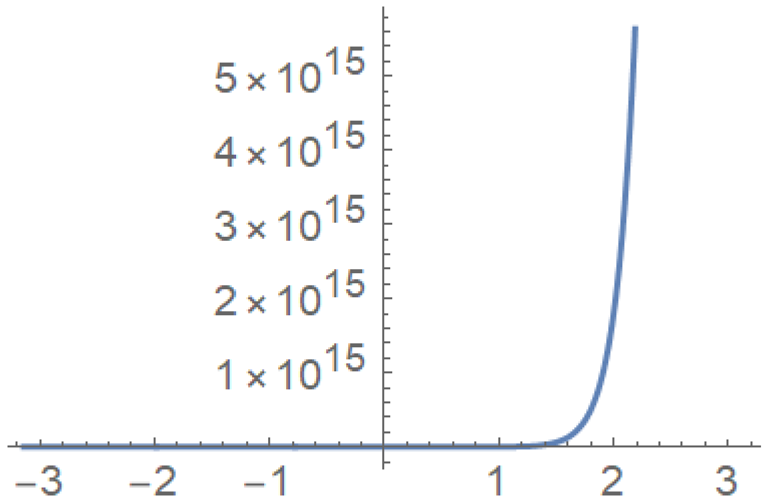
2.52877

Программа посчитала результат в окрестности $x = 0.629839$ наилучшим

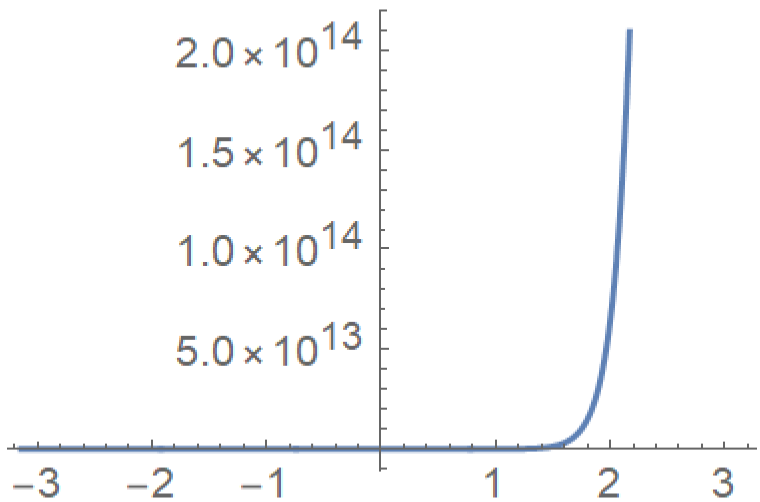


0.629839

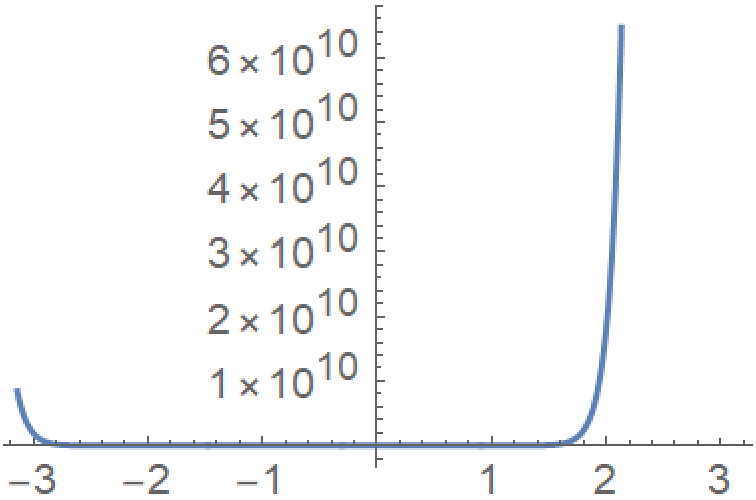
Пусть $n = 25$. Рассмотрим несколько графиков:



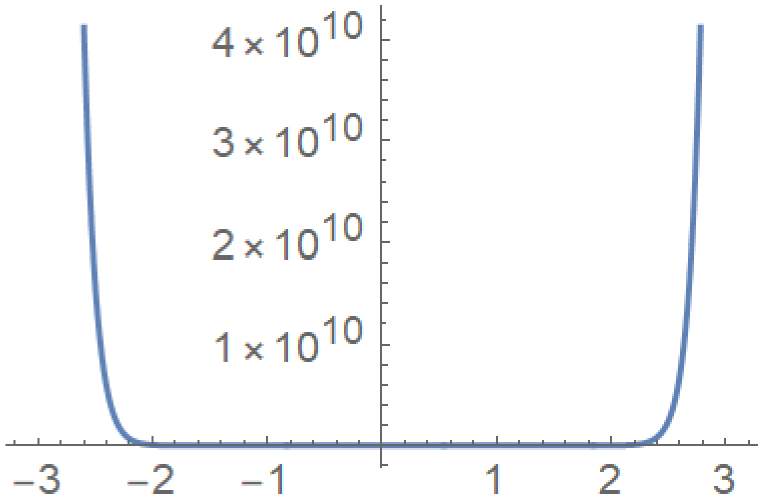
-3.0813



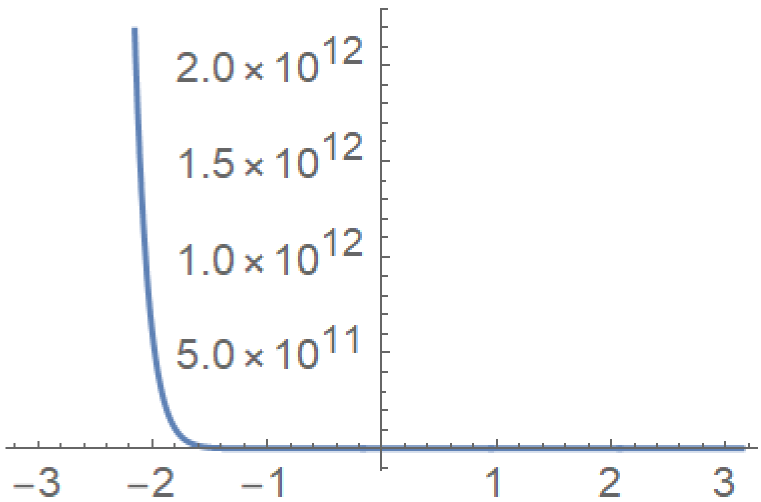
-1.79364



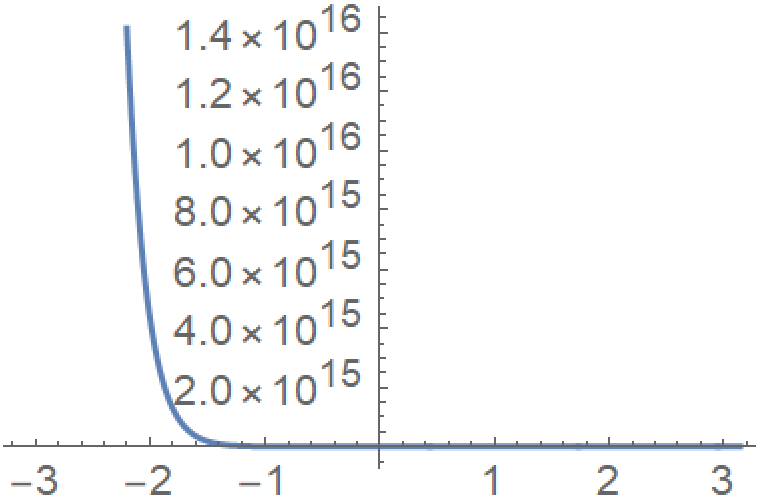
−0.790554



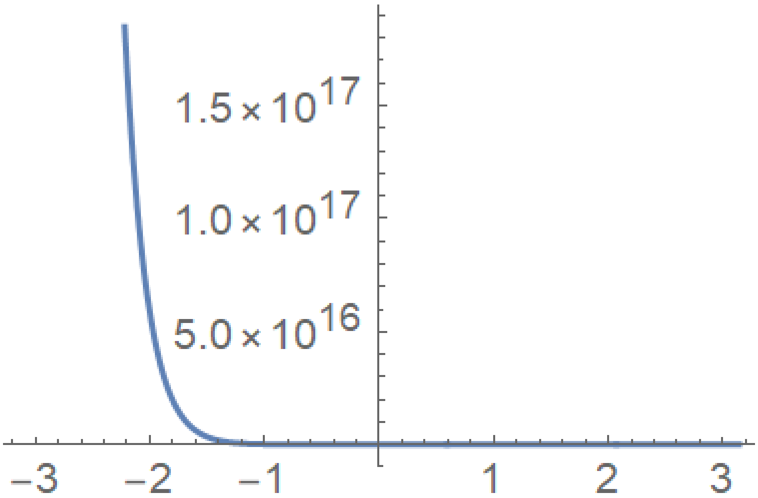
0.0546644



0.998169

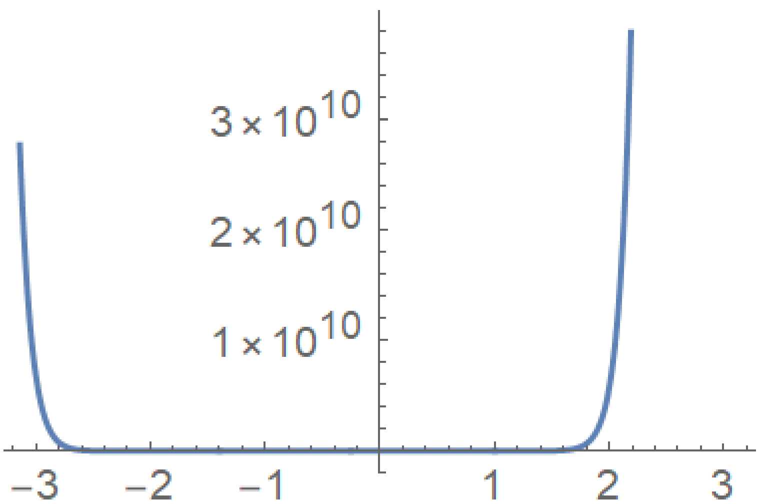


2.24725



3.02151

Наилучшим результатом программа посчитала в окрестности точки $x = -0.541585$



-0.541585

По данным примерам можно сделать вывод, что различный выбор n точек из N напрямую влияет на скорость роста остаточного многочлена n степени, что подтверждает корректность и смысл существования данного алгоритма. Также видно, что не существует одной окрестности, в которой будет наилучший искомый результат. То есть могут существовать несколько точек, в окрестностях которых результат практически не отличается.