МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №3
«Численные методы»
Вариант 1

Бобовоза Владислава
Сергеевича
студента 3 курса, 6 группы
специальность «прикладная математика»

Преподаватель:
Репников В.И.

Минск, 2024

Постановка задачи 1

На отрезке $[0;p]$ задана таблица значений функции $f(x) = x\sin x$ с шагом $h = \frac{p}{20}$. Погрешность каждого заданного значения не превышает $eps = 10^{-5}$. Используя интерполирование Ньютона для начала и конца таблицы, с помощью многочленов минимальной степени построить таблицу значений функции f(x) с шагом 0.5h. Погрешность каждого нового значения также не должна превышать заданной величины eps.

В отчет включить обе таблицы значений, а также подробное описание выбора степени интерполирующих многочленов.

Решение задачи 1

Положим $h = \frac{1}{20}$, $n = \frac{1}{h} + 1 = 21$.

При интерполировании в начале таблицы, остаток интерполирования оценивается так:

$$|r_k(x)| = \left|h^{k+1}\frac{t(t-1)...(t-k)}{(k+1)!}f^{(k+1)}(\xi)\right| \le \left|h^{k+1}\frac{t(t-1)...(t-k)}{(k+1)!}\right| \max_{x\in[a;\,b]}\left|f^{(k+1)}(x)\right|,$$

где $\xi\in[x_0;\, x_0 + kh]$, $t = \frac{x-x_0}{h}$. В нашем случае $t = 0.5$.

Тогда используя вышесказанное, получим:

$$\left|\left(\frac{1}{20}\right)^{k+1}\frac{0.5(0.5-1)...(0.5-k)}{(k+1)!}\right| \max_{x\in[0;\,1]}\left|(x\sin x)^{(k+1)}(x)\right| < 10^{-5}.$$

Теперь будем оценивать значение k подбором:

Пусть $k = 2$, тогда $f^{(3)}(x) = -x\cos x - 3\sin x$. Оценим значение 3-ей производной на нашем отрезке $[0;\,1]$: $\max\limits_{x\in[0;\,1]}\left|f^{(3)}(x)\right| = 3$. Тогда $\left|\left(\frac{1}{20}\right)^3\frac{0.5(0.5-1)(0.5-2)}{3!}\right| = 0.0078125\cdot10^{-3}$. Отсюда имеем: $|r_2(x)| = 3\cdot0.0078125\cdot10^{-3} = 2.34375\cdot10^{-5} > 10^{-5}$.

Пусть $k = 3$, тогда $f^{(4)}(x) = x\sin x - 4\cos x$. Оценим значение 4-ой производной на нашем отрезке $[0;\,1]$: $\max\limits_{x\in[0;\,1]}\left|f^{(4)}(x)\right| \le 5$. Тогда $\left|\left(\frac{1}{20}\right)^4\frac{0.5(0.5-1)(0.5-2)(0.5-3)}{4!}\right| = 2.44140625\cdot10^{-7}$. Отсюда имеем: $|r_3(x)| \le 5\cdot2.44140625\cdot10^{-7} < 10^{-5}$. Это говорит о достижении необходимой погрешности.

При интерполировании в конце таблицы, остаток интерполирования оценивается так:

$$|r_k(x)| = \left| h^{k+1} \frac{t(t+1)...(t+k)}{(k+1)!} f^{(k+1)}(\xi) \right| \le \left| h^{k+1} \frac{t(t+1)...(t+k)}{(k+1)!} \right| \max_{x\in[a;\,b]} \left| f^{(k+1)}(x) \right|,$$

где $\xi \in [x_0; x_0 + kh]$, $t = \frac{x - x_0}{h}$. В нашем случае $t = 0.5$.

Тогда используя вышесказанное, получим:

$$\left| \left(\frac{1}{20}\right)^{k+1} \frac{0.5(0.5+1)...(0.5+k)}{(k+1)!} \right| \max_{x\in[0;\,1]} \left| (x \sin x)^{(k+1)}(x) \right| < 10^{-5}.$$

Теперь будем оценивать значение k подбором:

Пусть $k = 2$, тогда $f^{(3)}(x) = -x\cos x - 3\sin x$. Оценим значение 3-ей производной на нашем отрезке $[0; 1]$: $\max\limits_{x\in[0;\,1]} \left| f^{(3)}(x) \right| = 3$. Тогда $\left| \left(\frac{1}{20}\right)^3 \frac{0.5(0.5+1)(0.5+2)}{3!} \right| = 3.90625 \cdot 10^{-5}$. Отсюда имеем: $|r_2(x)| = 3 \cdot 3.90625 \cdot 10^{-5} > 10^{-5}$.

Пусть $k = 3$, тогда $f^{(4)}(x) = x\sin x - 4\cos x$. Оценим значение 4-ой производной на нашем отрезке $[0; 1]$: $\max\limits_{x\in[0;\,1]} \left| f^{(4)}(x) \right| \le 5$. Тогда $\left| \left(\frac{1}{20}\right)^4 \frac{0.5(0.5+1)(0.5+2)(0.5+3)}{4!} \right| = 1.70898438 \cdot 10^{-6}$. Отсюда имеем: $|r_3(x)| \le 5 \cdot 1.70898438 \cdot 10^{-6} < 10^{-5}$. Это говорит о достижении необходимой погрешности.

Построим интерполяционный многочлен Ньютона заданной степени по следующим формулам в начале и в конце таблицы:

$$P_k(x) = P_k(x + th) = y_0 + \frac{t}{1!}\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + ... + \frac{t(t-1)...(t-k+1)}{k!}\Delta^k y_0,$$

$$P_k(x) = P_k(x_n - th) =$$
$$= y_n + \frac{t}{1!}\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + ... + \frac{t(t+1)...(t+k-1)}{k!}\Delta^k y_{n-k}.$$

В нашем случае, т.к. $k = 3$, формулы примут следующий вид:

$$P_3(x) = y_0 + \frac{1}{2}\Delta y_0 - \frac{1}{8}\Delta^2 y_0 + \frac{1}{16}\Delta^3 y_0,$$

$$P_3(x) = y_n + \frac{1}{2}\Delta y_{n-1} + \frac{3}{8}\Delta^2 y_{n-2} + \frac{5}{16}\Delta^3 y_{n-3}.$$

Но в данном случае есть небольшая проблема: из первой формулы можем получить значения в узлах $x \in [x_0; x_1]$, а из второй, значения в узлах $x \in [x_{n-1}; x_n]$. Таким образом, чтобы получить значения в остальных узлах, нужно

перестраивать таблицу конечных разностей после каждого нового полученного значения.

Тогда получим следующий алгоритм:

```
import random
import math
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sb

def f(x):
    return x*np.sin(x)

eps = 1e-5
h = 1 / 20
n = 21
a, b = 0, 1

interpolation_nodes = np.around(np.array([a + i*h for i in range(n)]), 5)

def find_diff(nodes, k, i):
    result = 0
    for j in range(k+1):
        result += (-1)**j * math.comb(k, j) * f(nodes[i+k-j])
    return result

def newton_inteprolation_eqsp_nodes(t, k, h, nodes):
    n = nodes.shape[0] - 1
    new_nodes = np.around(np.array([nodes[i] + t * h for i in range(n)]), 5)
    P_from_start = np.zeros(n)
    P_from_end = np.zeros(n)

    for step in range(n - k):
        P_k = 0
        for i in range(k + 1):
            mul = 1
            for j in range(i):
                mul *= (t - j) / (j + 1)
            P_k += mul * find_diff(interpolation_nodes, i, step)
        P_from_start[step] = P_k

    for step in range(n - 1, k - 1, -1):
        P_k = 0
        for i in range(k + 1):
            mul = 1
            for j in range(i):
                mul *= (t + j) / (j + 1)
            P_k += mul * find_diff(interpolation_nodes, i, step-i)
```

```
        P_from_end[step] = P_k

    return pd.DataFrame({'x' : new_nodes, 'P_from_start' : P_from_start, 'P_from_end' : P_from_end}).transpose()

if __name__ == '__main__':
    newton_inteprolation_eqsp_nodes(t=0.5, k=3, h=h, nodes=interpolation_nodes)
```

В итоге, получим следующую таблицу:

	0	1	2	3	4	5	6
x	0.025000	0.075000	0.125000	0.175000	0.225000	0.275000	0.325000
P_from_start	0.000624	0.005619	0.015583	0.030468	0.050198	0.074675	0.103774
P_from_end	0.000000	0.000000	0.000000	0.030476	0.050206	0.074682	0.103782

	7	8	9	10	11	12	13
x	0.375000	0.425000	0.475000	0.525000	0.575000	0.625000	0.675000
P_from_start	0.137351	0.175236	0.217235	0.263136	0.312704	0.365685	0.421805
P_from_end	0.137359	0.175243	0.217242	0.263143	0.312711	0.365691	0.421811

	14	15	16	17	18	19
x	0.725000	0.775000	0.825000	0.875000	0.925000	0.975000
P_from_start	0.480773	0.542279	0.606001	0.000000	0.000000	0.000000
P_from_end	0.480778	0.542284	0.606006	0.671604	0.738728	0.807012

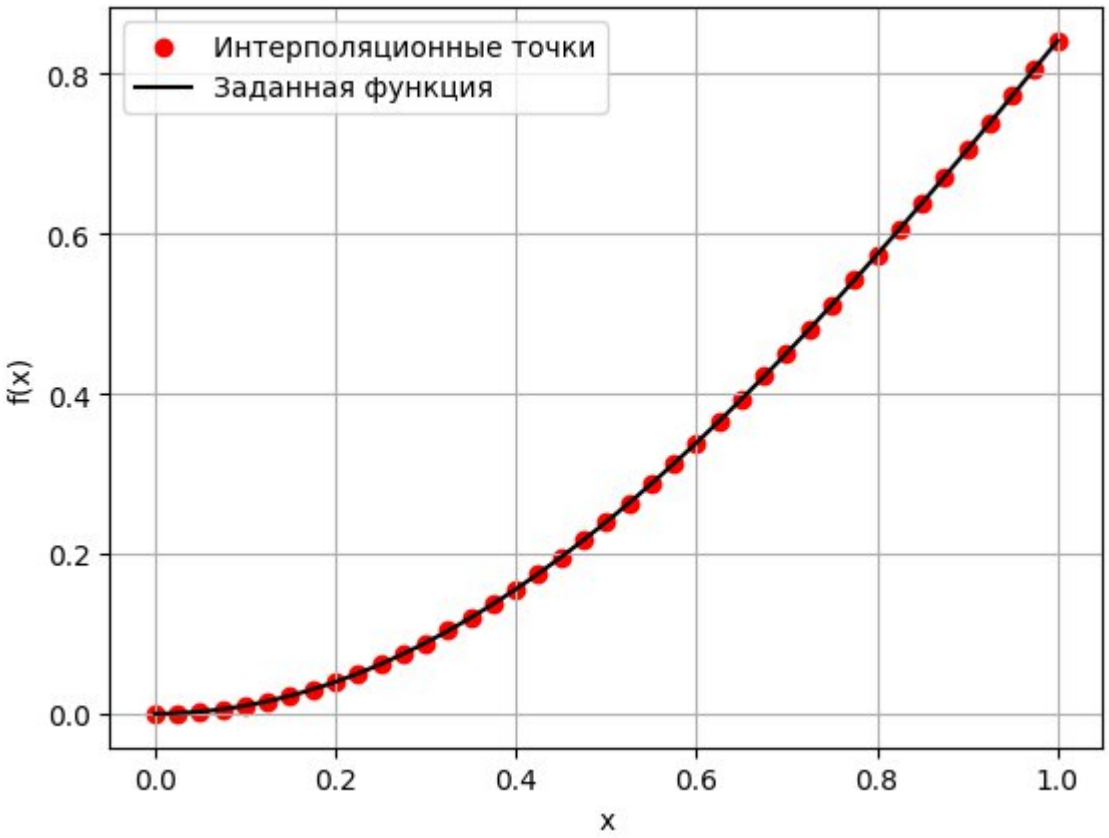
Для получение итоговой таблицы, совместим строки P_from_start и P_from_end путем выбора максимального из двух значений. Тогда получим:

	0	1	2	3	4	5	6
x	0.025000	0.075000	0.125000	0.175000	0.225000	0.275000	0.325000
f(x)	0.000624	0.005619	0.015583	0.030476	0.050206	0.074682	0.103782

	7	8	9	10	11	12	13
x	0.375000	0.425000	0.475000	0.525000	0.575000	0.625000	0.675000
f(x)	0.137359	0.175243	0.217242	0.263143	0.312711	0.365691	0.421811

	14	15	16	17	18	19
x	0.725000	0.775000	0.825000	0.875000	0.925000	0.975000
f(x)	0.480778	0.542284	0.606006	0.671604	0.738728	0.807012

Тогда объединим эту таблицу, и изначальную, а результат визуализируем в виде графика:



Постановка задачи 2

На отрезке $[-3; 3]$ заданы функции $f_1(x) = \sin x$ и $f_2(x) = \frac{1}{1+5x^2}$. Построить многочлены степени n = 3, 5, 7, 10, 15, 20, интерполирующие каждую из них по узлам:

a) равномерно расположенным на указанном отрезке;
b) расположенным на указанном отрезке оптимальным (минимизирующим погрешность) образом.

В отчет включить графики полученных интерполяционных многочленов, представление, использованное при их построении, а также способ выбора узлов.

Решение задачи 2

В качестве отрезка [a; b] возьмем отрезок [-3; 3]. Разобьем его на n=3,5,7,10,15,20 равномерных частей.

	3 узла	5 узлов	7 узлов	10 узлов	15 узлов	20 узлов
0	-3.0	-3.0	-3.0	-3.0	-3.0	-3.0
1	0.0	-1.5	-2.0	-2.333333	-2.571429	-2.684211
2	3.0	0.0	-1.0	-1.666667	-2.142857	-2.368421
3		1.5	0.0	-1.0	-1.714286	-2.052632
4		3.0	1.0	-0.333333	-1.285714	-1.736842
5			2.0	0.333333	-0.857143	-1.421053
6			3.0	1.0	-0.428571	-1.105263
7				1.666667	0.0	-0.789474
8				2.333333	0.428571	-0.473684
9				3.0	0.857143	-0.157895
10					1.285714	0.157895
11					1.714286	0.473684
12					2.142857	0.789474
13					2.571429	1.105263
14					3.0	1.421053
15						1.736842
16						2.052632
17						2.368421
18						2.684211
19						3.0

Рассмотрим построение интерполяционного многочлена Лагранжа.

$$c_i = \frac{1}{(x_i - x_0)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)},$$

$$w_{n+1}(x) = (x - x_0)...(x - x_n),$$

$$c_i = \frac{1}{w'_{n+1}(x_i)},$$

тогда можем выписать формулу Лагранжа для интерполяционного многочлена P_n:

$$P_n(x) = \sum_{i=0}^{n} \frac{w_{n+1}(x)}{(x - x_i)w'_{n+1}(x_i)} f(x_i)$$

Реализуем это на Python:

```
def c(x_i, nodes):
    product = 1
    for node in nodes:
        if node == x_i:
            continue
        product *= (x_i - node)
```

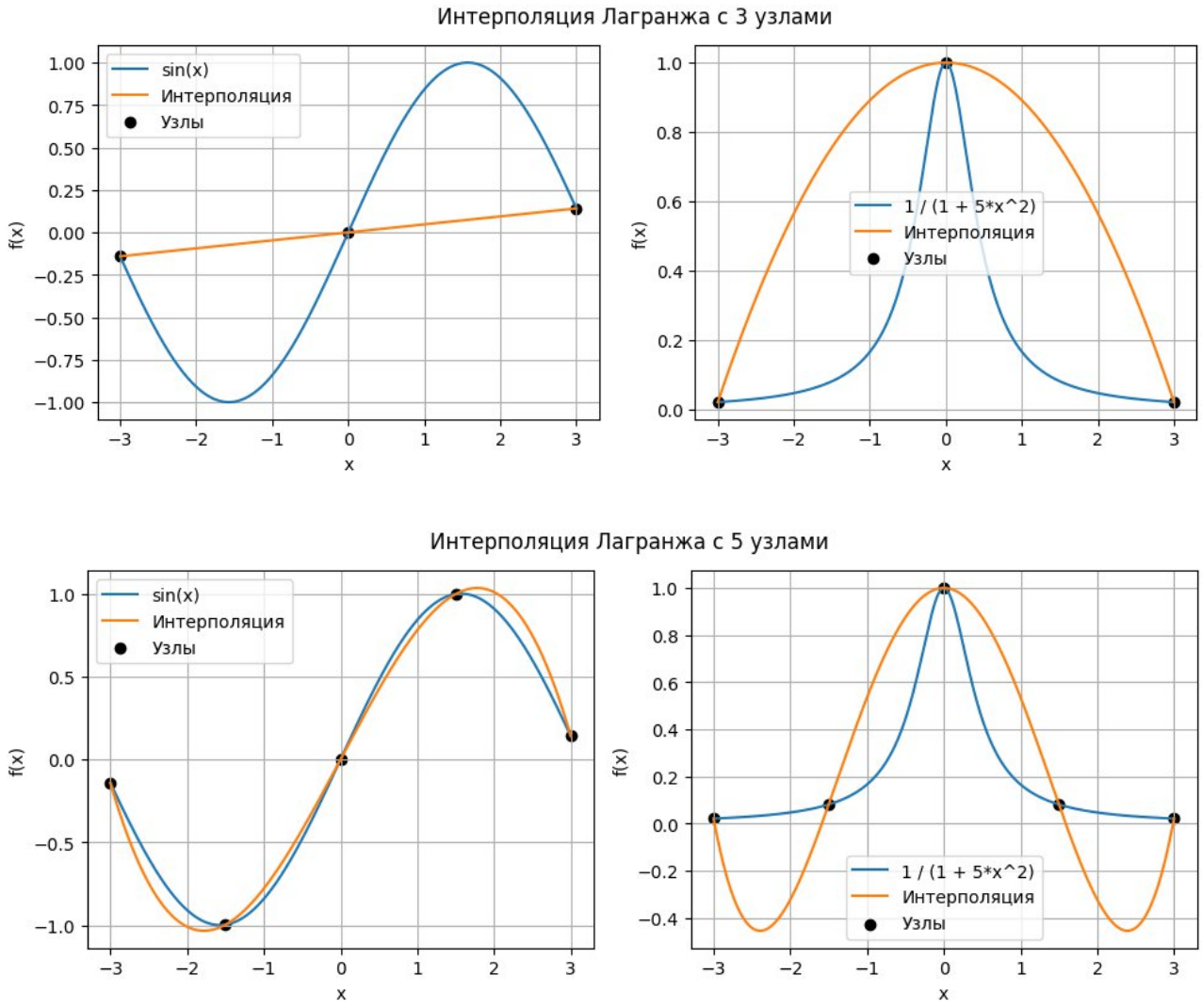
```
    return 1 / product


def w(x, nodes):
    product = 1
    for node in nodes:
        product *= (x - node)
    return product


def lagrange_interpolation(x, nodes, function):
    summary = 0
    for node in nodes:
        summary += w(x, nodes) * function(node) * c(node, nodes) / (x - node)
    return summary
```

Теперь напишем цикл, который для каждых n узлов будет строить график функций $f_1(x)$ и $f_2(x)$ на отрезке [-3; 3]:

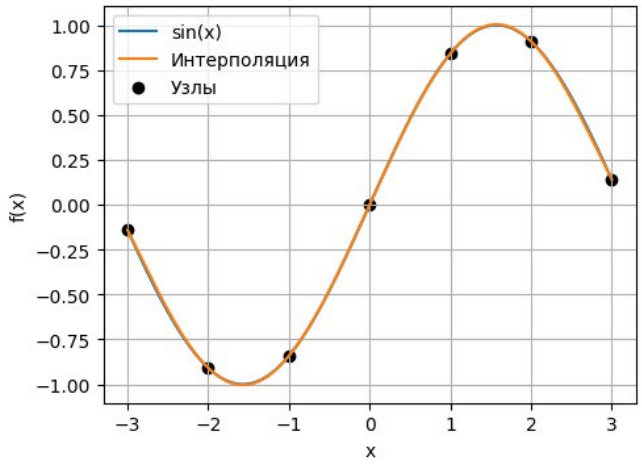
```
x = np.linspace(a, b, 100000)


def f_1(x):
    return np.sin(x)


def f_2(x):
    return 1 / (1 + 5 * x**2)


for nodes_num in [3, 5, 7, 10, 15, 20]:
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))
    fig.tight_layout(pad=2.5)
    interpolation_nodes = np.linspace(a, b, nodes_num)
    plot_dots = np.setdiff1d(x, interpolation_nodes)
    ax1.plot(x, f_1(x), label='sin(x)')
    ax1.plot(plot_dots, lagrange_interpolation(x=plot_dots, nodes=interpolation_nodes,
function=f_1), label='Интерполяция')
    ax1.scatter(interpolation_nodes, f_1(interpolation_nodes), label='Узлы', color='black')
    ax1.set_xlabel('x')
    ax1.set_ylabel('f(x)')
    ax1.legend()
    ax1.grid()
    ax2.plot(x, f_2(x), label='1 / (1 + 5*x^2)')
    ax2.plot(plot_dots, lagrange_interpolation(x=plot_dots, nodes=interpolation_nodes,
function=f_2), label='Интерполяция')
    ax2.scatter(interpolation_nodes, f_2(interpolation_nodes), label='Узлы', color='black')
    ax2.set_xlabel('x')
    ax2.set_ylabel('f(x)')
```

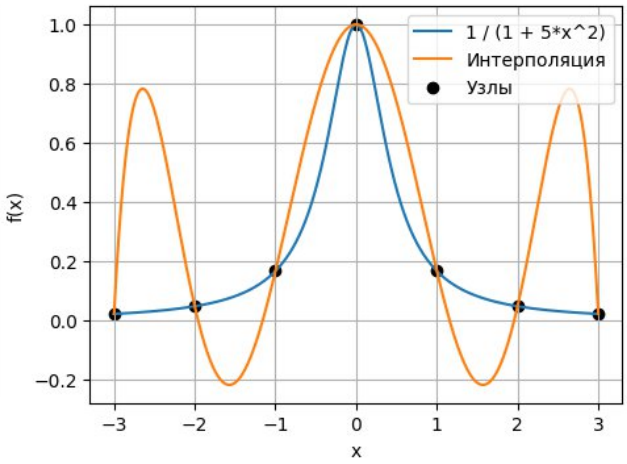
```
ax2.legend()
ax2.grid()
fig.suptitle(f'Интерполяция Лагранжа с {nodes_num} узлами')
plt.show()
```

В итоге получим следующие графики:

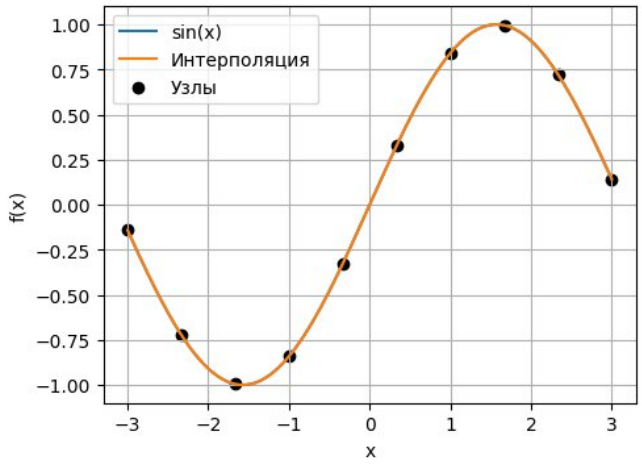


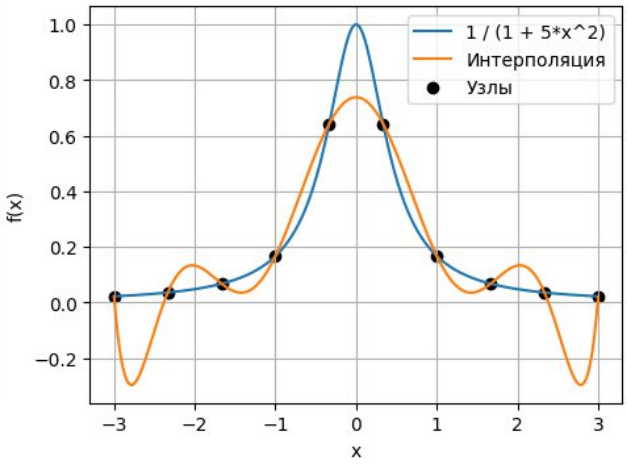
Интерполяция Лагранжа с 7 узлами



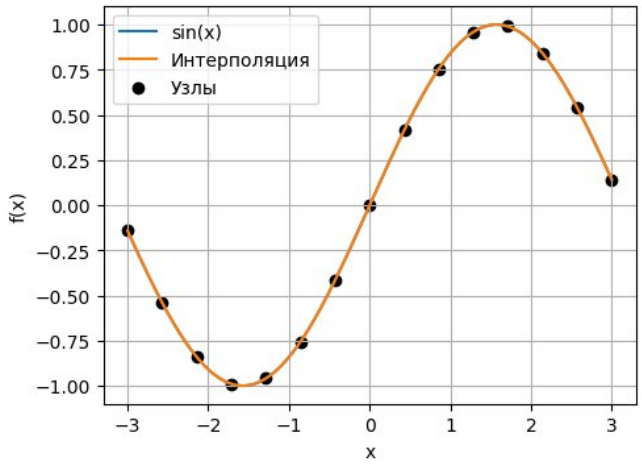


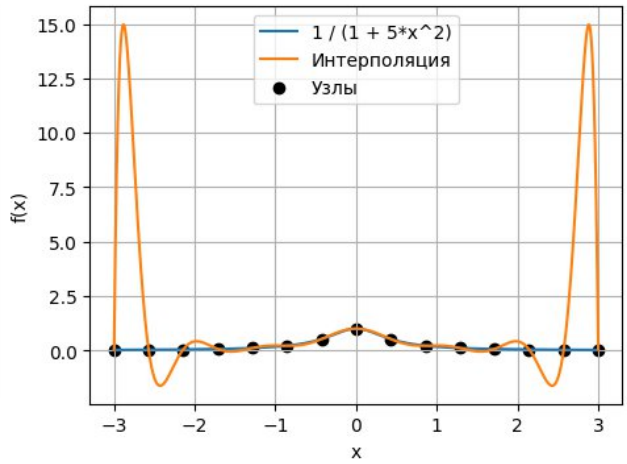
Интерполяция Лагранжа с 10 узлами

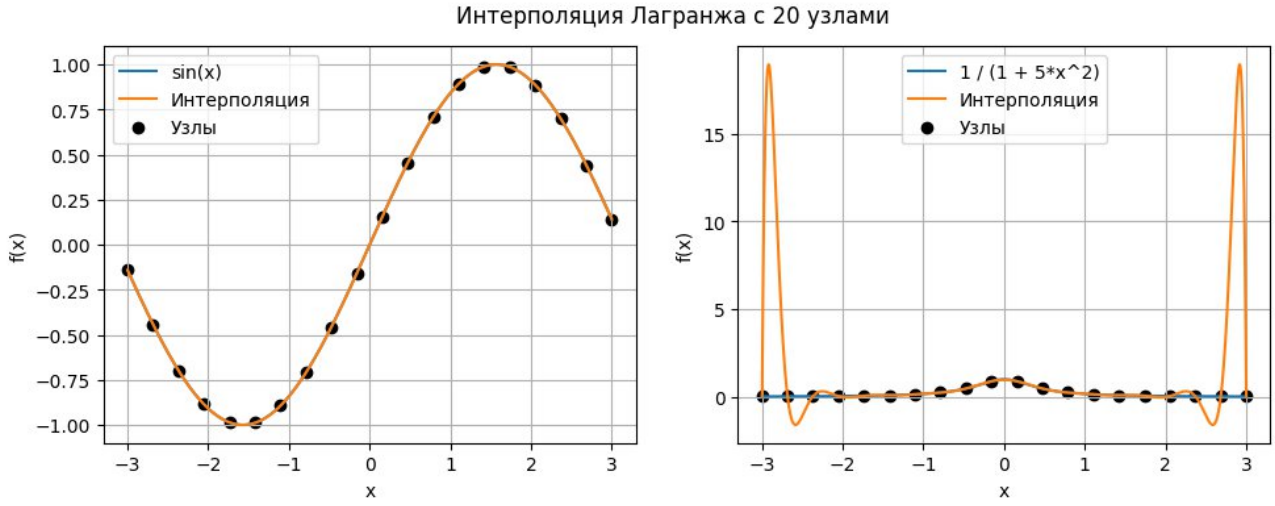




Интерполяция Лагранжа с 15 узлами







Рассмотрим построение интерполяционного многочлена Ньютона.

Напишем функцию, которая будет считать таблицу разделенных разностей:

$$P_n(x) =$$
$$= f(x_0) + (x - x_0)f(x_0; x_1) + (x - x_0)(x - x_1)f(x_0; x_1; x_2) \ + \ ... \ + \ (x - x_0)...(x - \\ - x_{n-1})f(x_0; ...; x_n).$$

Эта формула называется формулой Ньютона для интерполяционного многочлена P_n.

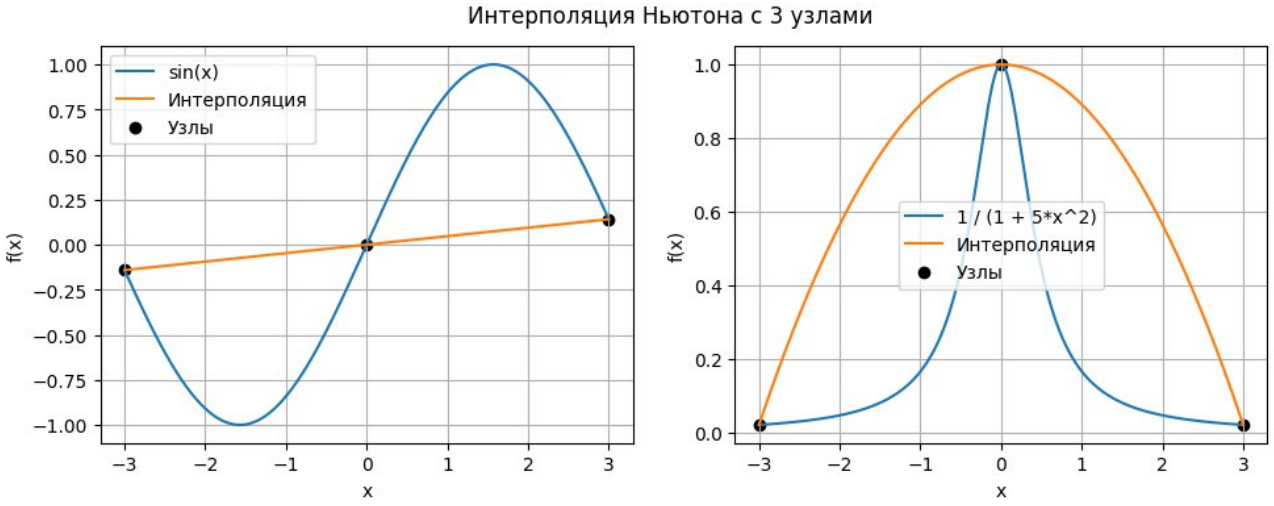
```
def div_diff_table(nodes, function, as_table=False):
    if isinstance(nodes, list):
        nodes = np.array(nodes)
    n = nodes.shape[0]
    table = np.zeros((n, n+1))
    for i in range(n):
        table[i, 0] = nodes[i]
        table[i, 1] = function(nodes[i])
    for i in range(2, n+1):
        for j in range(n-i+1):
            table[j,i] = (table[j+1,i-1] - table[j, i-1]) / (nodes[j+i-1] - nodes[j])
    if as_table:
        return table
    else:
        return table[0, n]



def newton_interpolation(x, nodes, function):
    P_n = 0
    nodes_list = []
    for node in nodes:
        nodes_list.append(node)
```

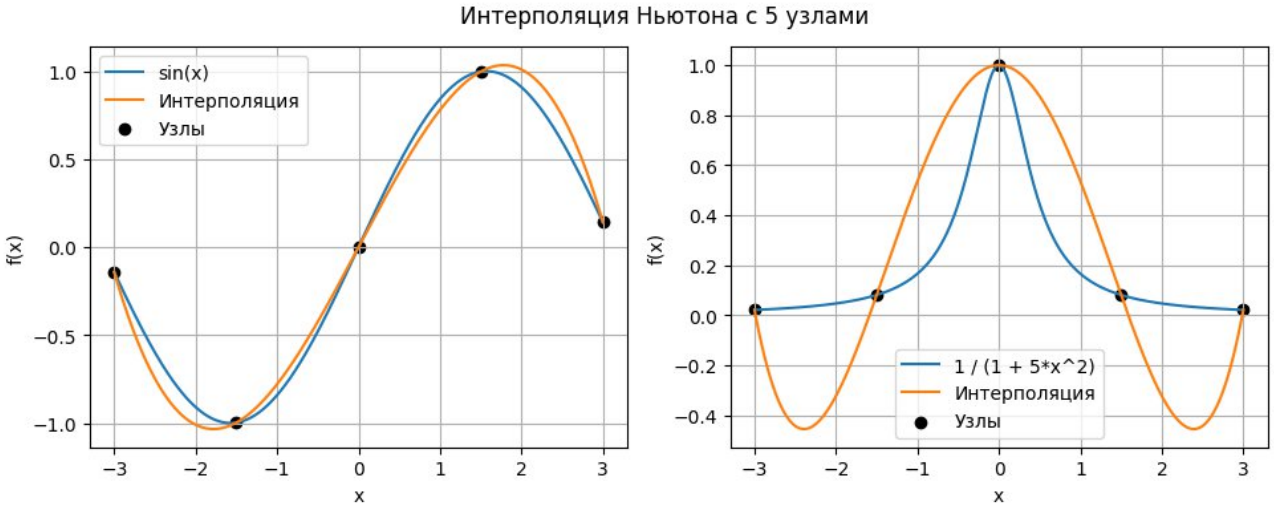
```
        P_n += div_diff_table(nodes_list, function)*w(x, nodes_list[0:-1])
    return P_n
```

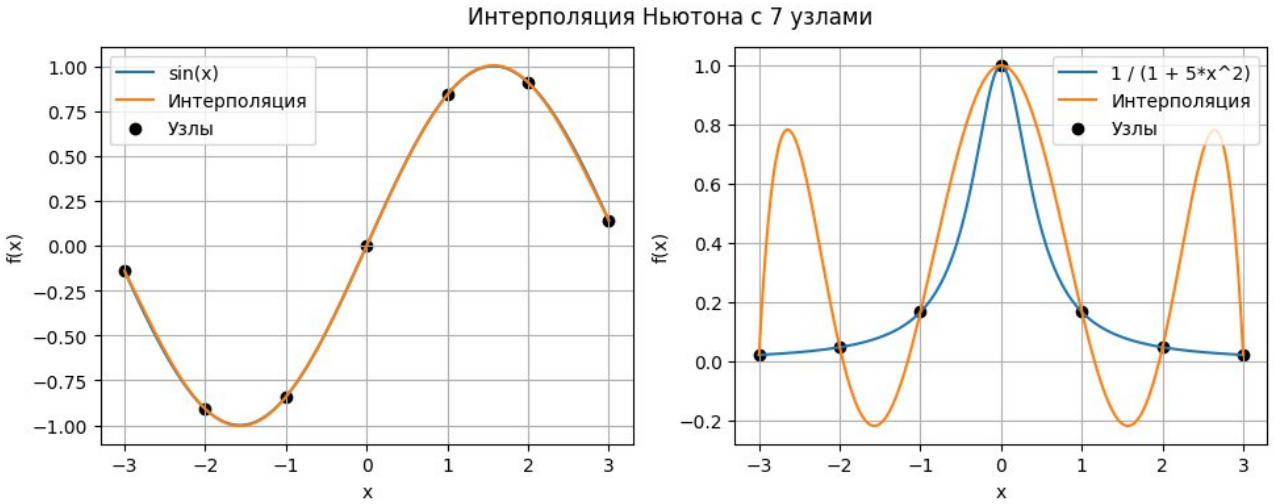
Теперь напишем цикл, который для каждых n узлов будет строить график функций $f_1(x)$ и $f_2(x)$ на отрезке [-3; 3]:

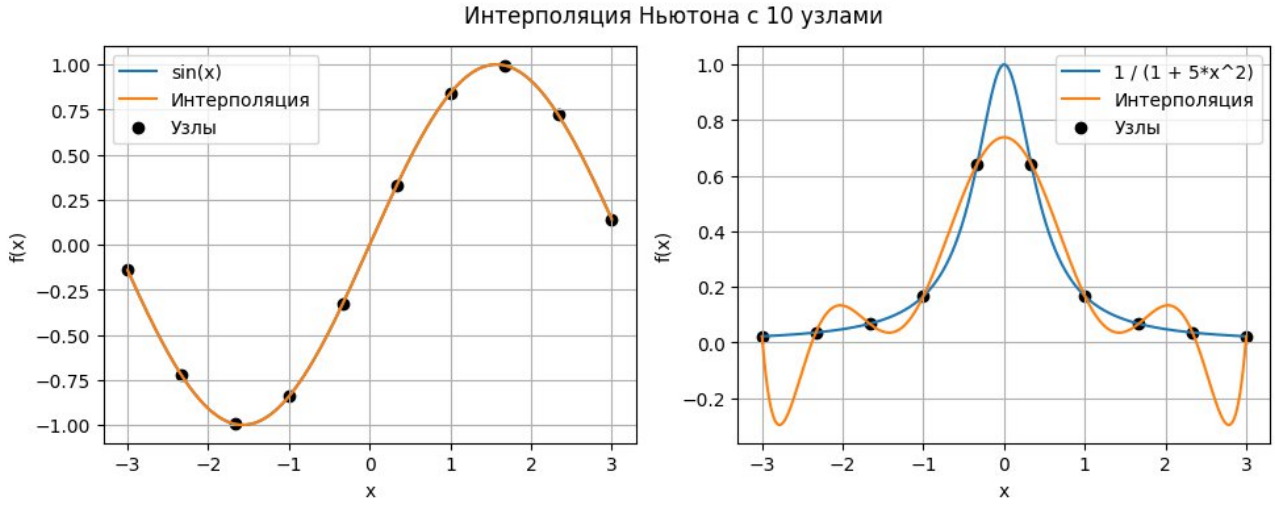
```
x = np.linspace(a, b, 100000)


def f_1(x):
    return np.sin(x)


def f_2(x):
    return 1 / (1 + 5 * x**2)


for nodes_num in [3, 5, 7, 10, 15, 20]:
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))
    fig.tight_layout(pad=2.5)
    interpolation_nodes = np.linspace(a, b, nodes_num)
    plot_dots = np.setdiff1d(x, interpolation_nodes)
    ax1.plot(x, f_1(x), label='sin(x)')
    ax1.plot(plot_dots, newton_interpolation(x=plot_dots, nodes=interpolation_nodes,
function=f_1), label='Интерполяция')
    ax1.scatter(interpolation_nodes, f_1(interpolation_nodes), label='Узлы', color='black')
    ax1.set_xlabel('x')
    ax1.set_ylabel('f(x)')
    ax1.legend()
    ax1.grid()
    ax2.plot(x, f_2(x), label='1 / (1 + 5*x^2)')
    ax2.plot(plot_dots, newton_interpolation(x=plot_dots, nodes=interpolation_nodes,
function=f_2), label='Интерполяция')
    ax2.scatter(interpolation_nodes, f_2(interpolation_nodes), label='Узлы', color='black')
    ax2.set_xlabel('x')
    ax2.set_ylabel('f(x)')
    ax2.legend()
    ax2.grid()
    fig.suptitle(f'Интерполяция Ньютона с {nodes_num} узлами')
    plt.show()
```

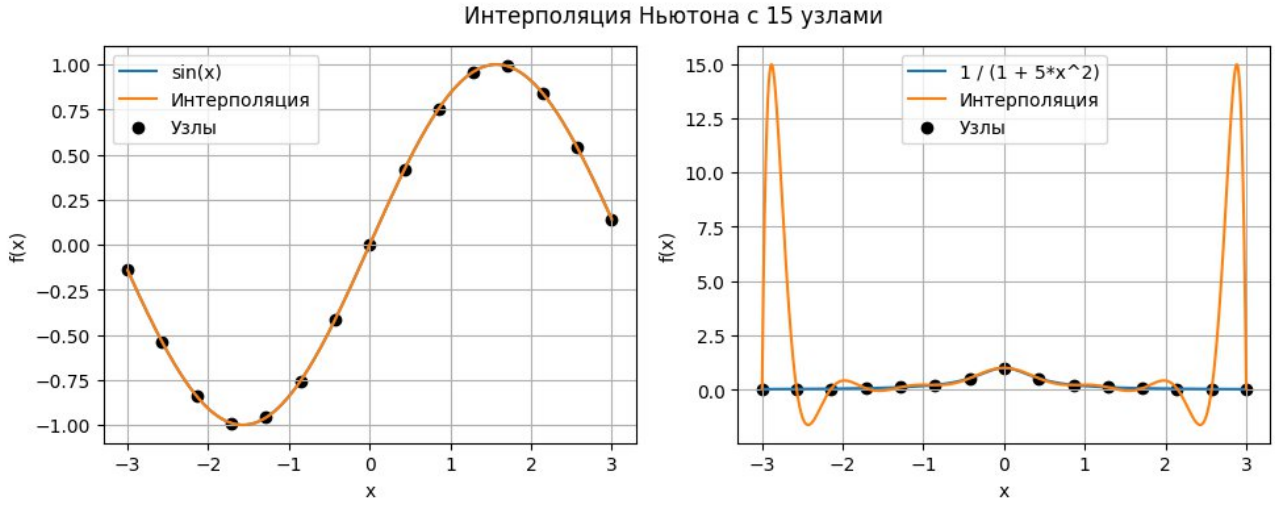
В итоге получим следующие графики:

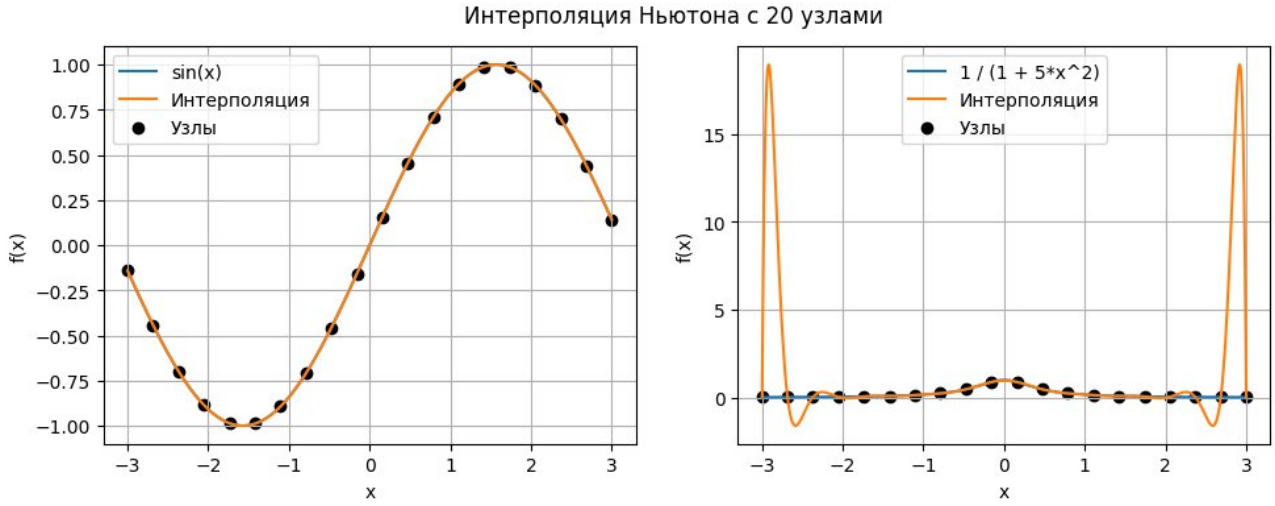

Интерполяция Ньютона с 3 узлами
sin(x)
Интерполяция
Узлы
1 / (1 + 5*x^2)
Интерполяция
Узлы
f(x)
x


Интерполяция Ньютона с 5 узлами
sin(x)
Интерполяция
Узлы
1 / (1 + 5*x^2)
Интерполяция
Узлы
f(x)
x


Интерполяция Ньютона с 7 узлами
sin(x)
Интерполяция
Узлы
1 / (1 + 5*x^2)
Интерполяция
Узлы
f(x)
x







Из графиков можно сделать вывод, что решение единственно, т.е. оба интерполяционных многочлена дали одинаковые результаты. Первая функция уже на 7 узлах дает довольно хорошую интерполяцию. Со второй функцией не все так однозначно: на 10 узлах интерполяция достаточно хороша, но видно, что при повышении числа узлов, увеличиваются и отклонения.

Интерполирование многочленами по узлам, расположенным на указанном отрезке оптимальным образом.

Для минимизации остатка, узлы будем выбирать следующим образом:

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\frac{\pi(2k+1)}{2(n+1)},\ k = \overline{0,n}.$$

Таким образом, новые узлы интерполирования x_0, ..., x_n дадут минимальную величину отклонения $w_{n+1}(x)$ от нуля.

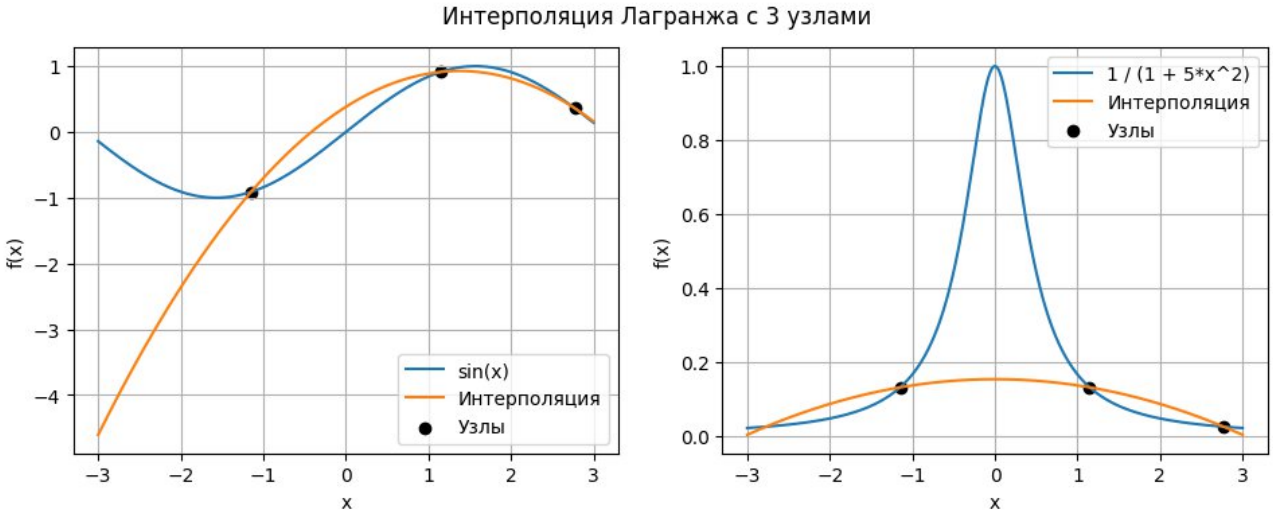
Построим новую таблицу узлов по указанной выше формуле, а узлы соответственно перенумеруем.

```
interpolation_3_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(4))) for k in range(3)][::-1]
interpolation_5_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(6))) for k in range(5)][::-1]
interpolation_7_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(8))) for k in range(7)][::-1]
interpolation_10_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(11))) for k in range(10)][::-1]
interpolation_15_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(15))) for k in range(15)][::-1]
interpolation_20_nodes_opt = [(a+b)/2 + (b-a)/2 * np.cos(((2*k+1)*np.pi)/(2*(20))) for k in range(20)][::-1]
interpolation_nodes_table_opt = pd.DataFrame.from_dict({
   '3 узла' : interpolation_3_nodes_opt,
   '5 узлов' : interpolation_5_nodes_opt,
   '7 узлов' : interpolation_7_nodes_opt,
   '10 узлов' : interpolation_10_nodes_opt,
   '15 узлов' : interpolation_15_nodes_opt,
   '20 узлов' : interpolation_20_nodes_opt}, orient='index').T.fillna(' ')


interpolation_nodes_table_opt
```

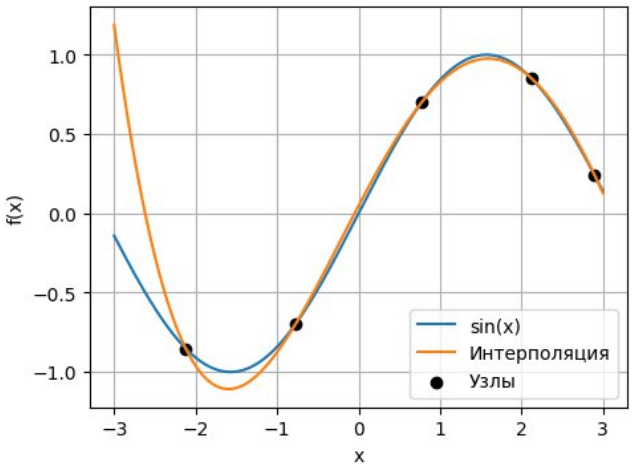
Тогда получим следующую таблицу:

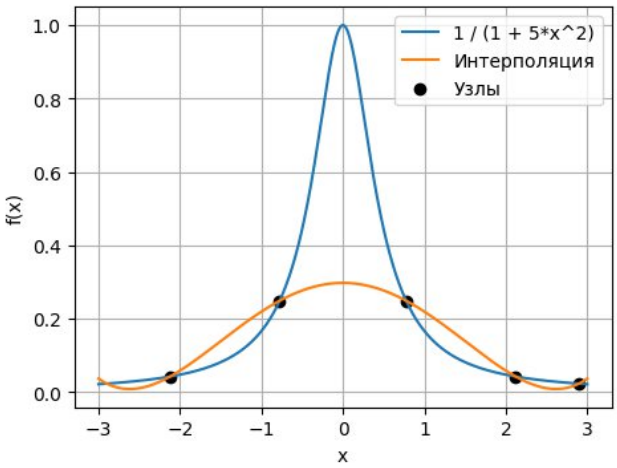
	3 узла	5 узлов	7 узлов	10 узлов	15 узлов	20 узлов
0	-1.14805	-2.12132	-2.494409	-2.728896	-2.983566	-2.990752
1	1.14805	-0.776457	-1.666711	-2.267249	-2.85317	-2.917110
2	2.771639	0.776457	-0.585271	-1.621922	-2.598076	-2.771639
3		2.12132	0.585271	-0.845198	-2.229434	-2.557920
4		2.897777	1.666711	0.0	-1.763356	-2.281218
5			2.494409	0.845198	-1.22021	-1.948344
6			2.942356	1.621922	-0.623735	-1.567496
7				2.267249	0.0	-1.148050
8				2.728896	0.623735	-0.700336
9				2.969464	1.22021	-0.235377
10					1.763356	0.235377
11					2.229434	0.700336
12					2.598076	1.148050
13					2.85317	1.567496
14					2.983566	1.948344
15						2.281218
16						2.557920
17						2.771639
18						2.917110
19						2.990752

Теперь проведем интерполяцию многочленом Лагранжа:

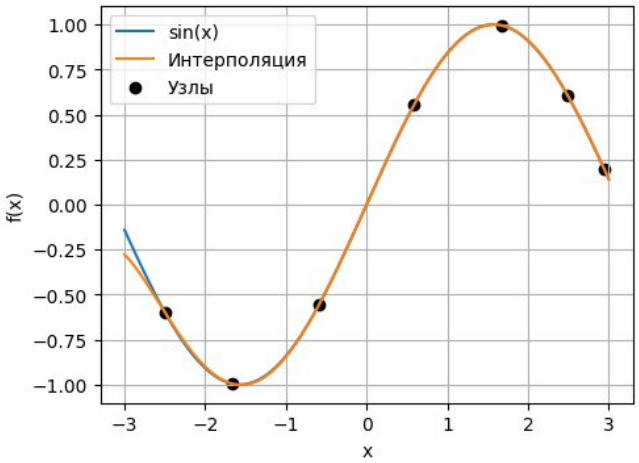


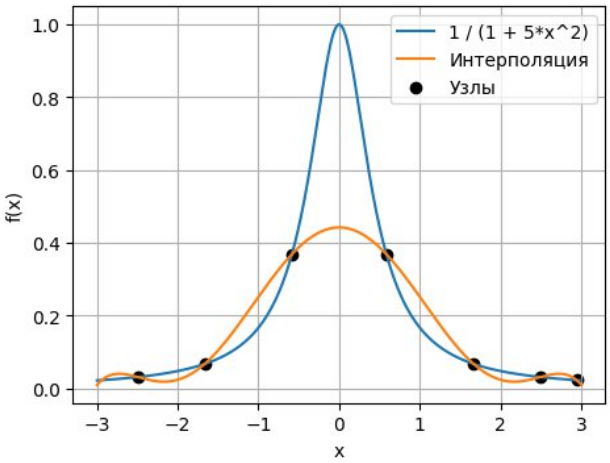
Интерполяция Лагранжа с 5 узлами



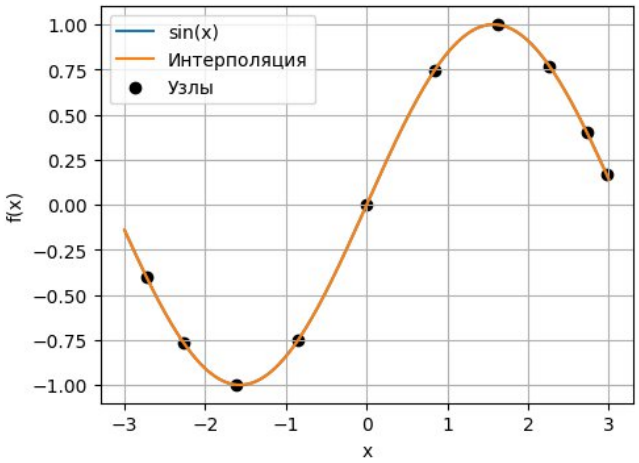


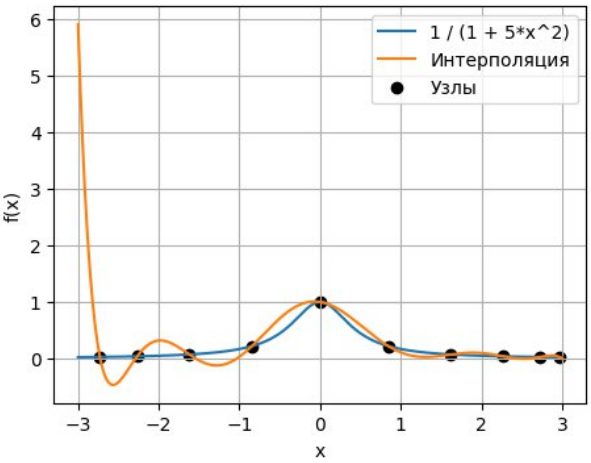
Интерполяция Лагранжа с 7 узлами

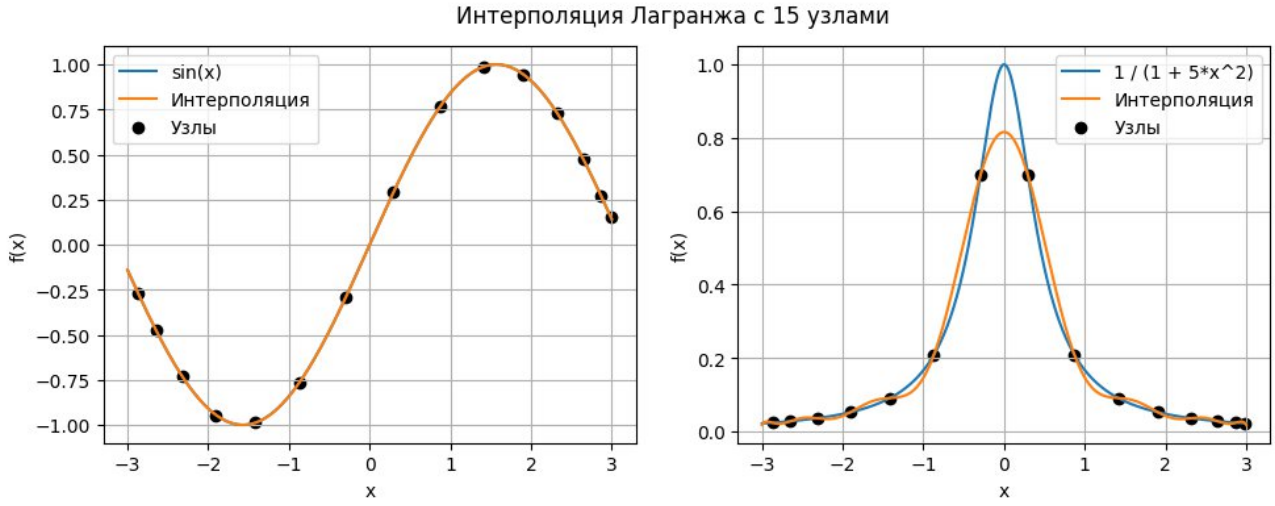


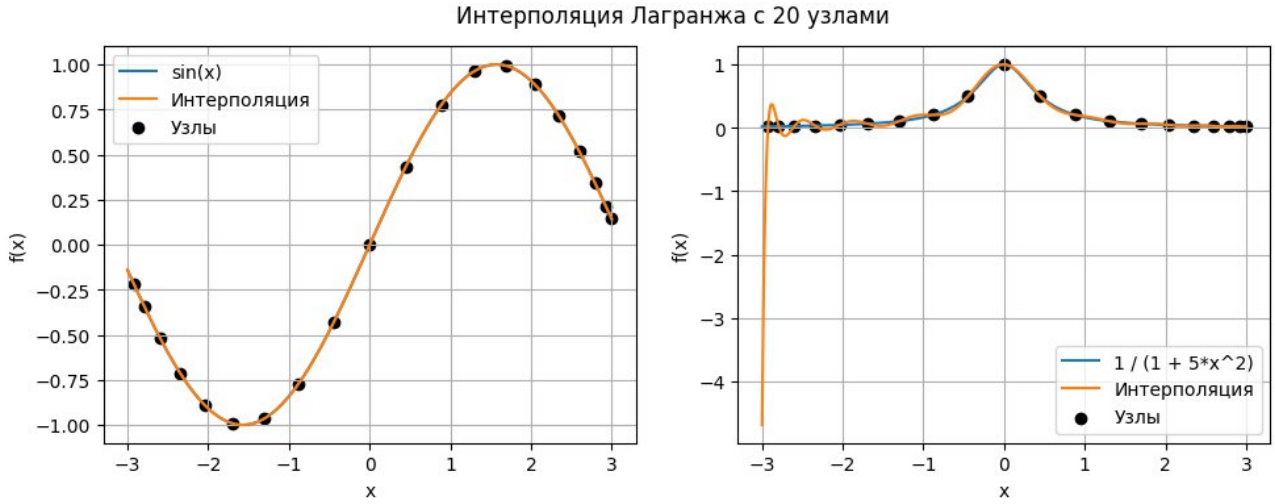


Интерполяция Лагранжа с 10 узлами

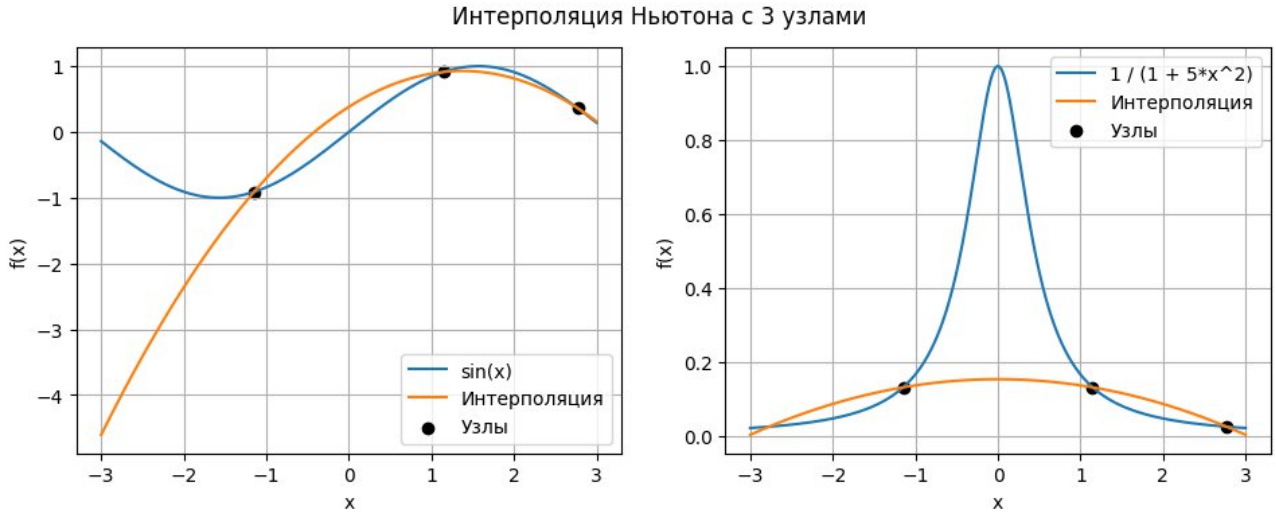




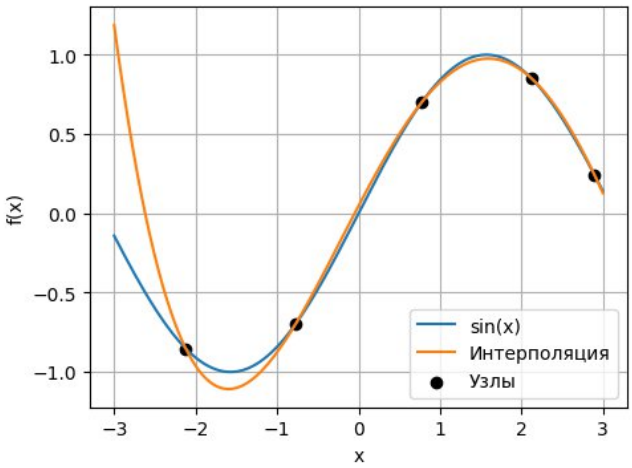


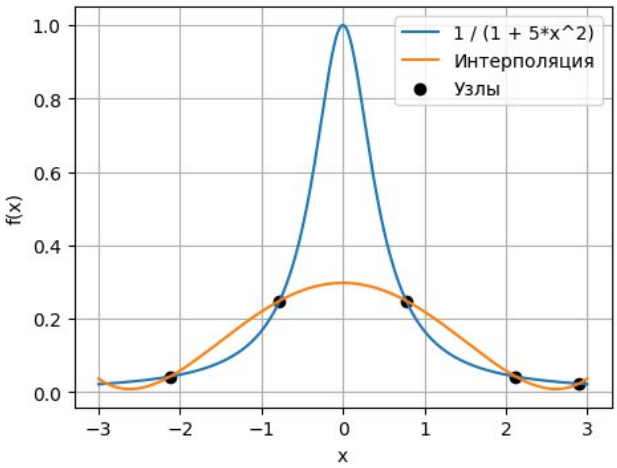


Теперь проведем интерполяцию многочленом Ньютона:

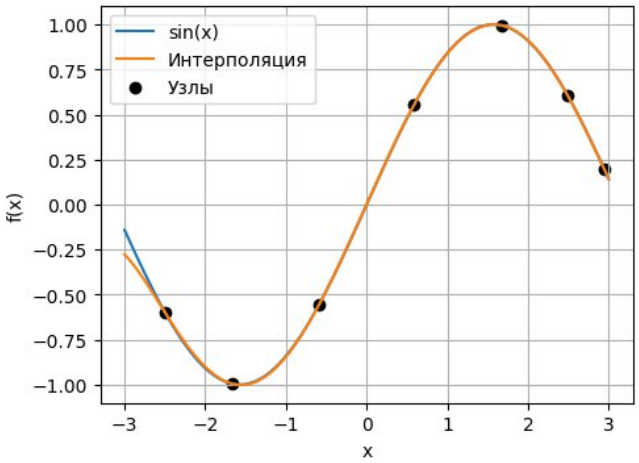


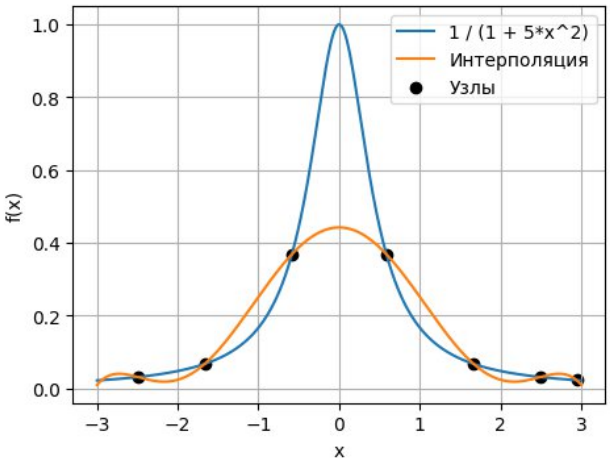
Интерполяция Ньютона с 5 узлами



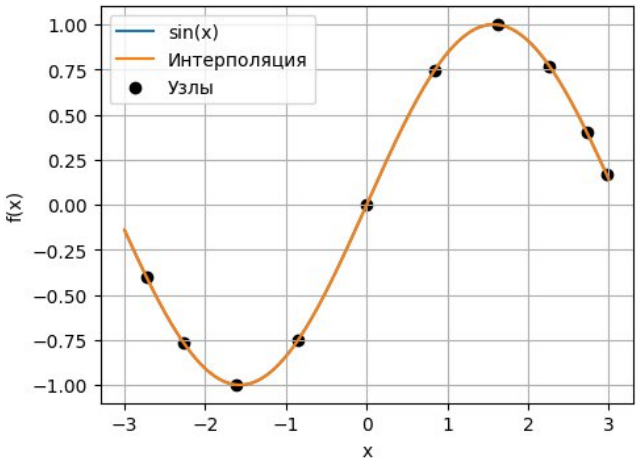


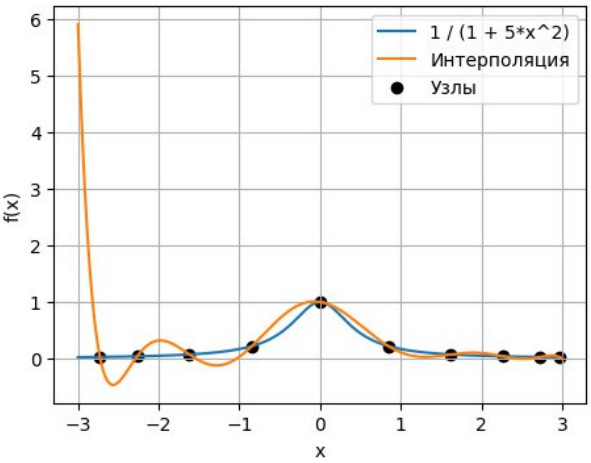
Интерполяция Ньютона с 7 узлами

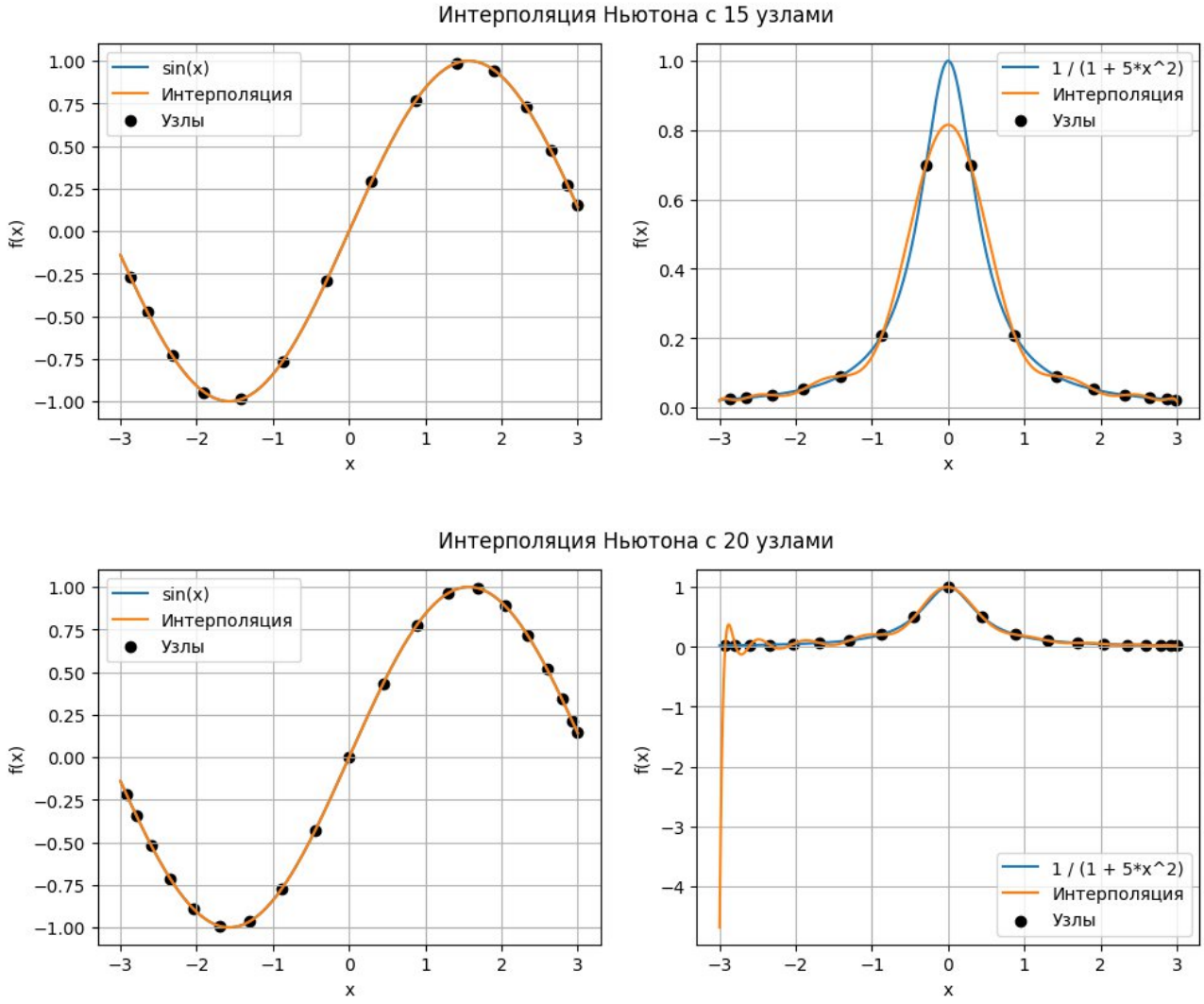




Интерполяция Ньютона с 10 узлами







Тогда можно сделать вывод, что для первой функции уже начиная с интерполяции с 5 узлами все почти хорошо, а с 7 узлами почти идеально. Для второй функции, интерполяцию с 15 узлами дала лучший результат, т.к. при 10 и 20 узлах явно видны серьезные отклонения.