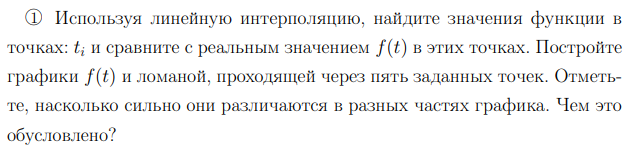
# ЛР3, В15

image.png

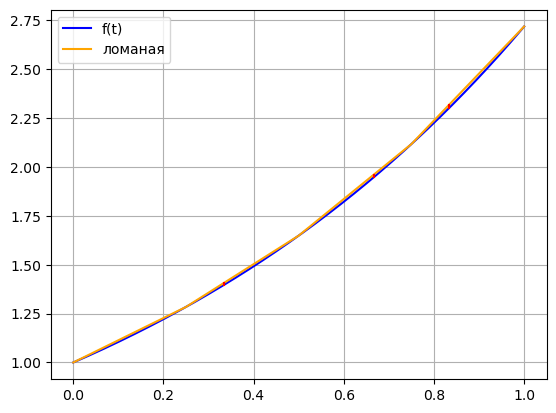
import numpy as np  
import sympy as sp  
from sympy import \*  
import matplotlib.pyplot as plt  
from scipy.optimize import root\_scalar  
import math

## 1.1

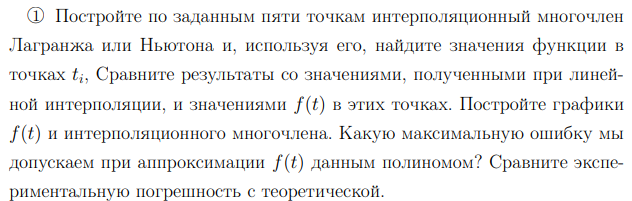


def f(x):  
 return np.e\*\*x  
  
def LinearInterpolation(f, t):  
 x = np.linspace(0, 1, 1000)  
 plt.plot(x, f(x), color = 'blue', label = 'f(t)')  
   
 # fx = []  
 # for i in range(len(t2)):  
 # fx.append(f(t2[i]))  
 # plt.plot(t2, fx, color = 'orange', label = 'ломаная')  
  
 x2 = np.linspace(0, 1, 5)  
 plt.plot(x2, f(x2), color = 'orange', label = 'ломаная')  
   
 fxList = []  
 flList = []  
 for i in range(len(t) - 1):  
 for j in range(len(t)):  
 if x2[i] < t[j] <= x2[i+1]:  
 plt.plot([t[j], t[j]],   
 [f(t[j]), f(x2[i]) + (t[j] - x2[i]) \* ((f(x2[i+1]) - f(x2[i]))/(x2[i+1] - x2[i]))],   
 color = 'red')  
 print("Точка: ", t[j])  
 fx = f(t[j])  
 fxList.append(fx)  
 print(" Значение ф-ии: ", fx)  
 fl = f(x2[i]) + (t[j] - x2[i]) \* ((f(x2[i+1]) - f(x2[i]))/(x2[i+1] - x2[i]))  
 flList.append(fl)  
 print(" Значение ломаной: ", fl)  
 print(" Разница: ", np.abs(fx - fl))  
 print("")  
   
 plt.legend()  
 plt.grid()  
 plt.show()  
 return flList, fxList  
  
t = []  
for i in range(5):  
 t.append((i+2)/6)  
  
# t2 = []  
# for i in range(5):  
# t2.append(i/4)  
   
flList, fxList = LinearInterpolation(f, t)

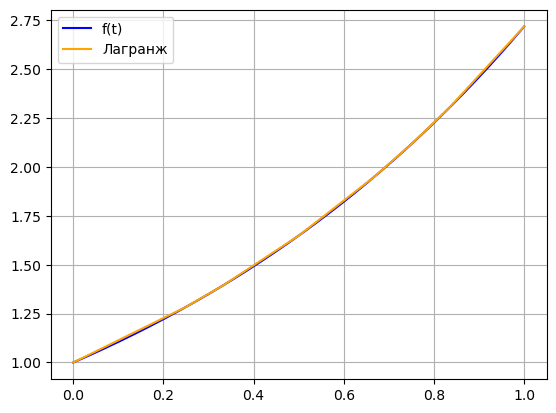
Точка: 0.3333333333333333  
 Значение ф-ии: 1.3956124250860895  
 Значение ломаной: 1.405590701358537  
 Разница: 0.009978276272447406  
  
Точка: 0.5  
 Значение ф-ии: 1.6487212707001282  
 Значение ломаной: 1.6487212707001282  
 Разница: 0.0  
  
Точка: 0.6666666666666666  
 Значение ф-ии: 1.9477340410546757  
 Значение ломаной: 1.9609071013084924  
 Разница: 0.013173060253816704  
  
Точка: 0.8333333333333334  
 Значение ф-ии: 2.300975890892825  
 Значение ломаной: 2.317427287228132  
 Разница: 0.01645139633530679  
  
Точка: 1.0  
 Значение ф-ии: 2.718281828459045  
 Значение ломаной: 2.718281828459045  
 Разница: 0.0



## 1.2



def f(x):  
 return np.e\*\*x  
  
def Lagrange(t, x, fx):  
 result = 0  
 for i in range(len(x)):  
 y = fx[i]  
 for j in range(len(x)):  
 if i != j:  
 y = y \* ((t - x[j])/(x[i] - x[j]))  
 result += y  
 return result  
  
t = []  
for i in range(5):  
 t.append((i+2)/6)  
  
x = np.linspace(0, 1, 5)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
ftList = []  
for ti in t:  
 ftList.append(Lagrange(ti, x, fx))  
  
  
xt = x.tolist() + t  
xt.sort()  
  
  
ff = fx + ftList  
ff.sort()  
  
z = np.linspace(0, 1, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)')  
plt.plot(xt, ff, color = 'orange', label = 'Лагранж')  
plt.legend()  
plt.grid()  
plt.show()  
  
for i in range(len(t)):  
 print("Точка: ", t[i])  
 print("\tЗначение ф-ии: ", fxList[i])  
 print("\tЗначение ломаной: ", flList[i])  
 print("\tЗначение Лагранжа: ", ftList[i])  
 print(f"\tРазница c функцией: {np.abs(fxList[i] - ftList[i]):f}")  
 print(f"\tРазница c ломаной: {np.abs(flList[i] - ftList[i]):.16f}")  
 print("")



Точка: 0.3333333333333333  
 Значение ф-ии: 1.3956124250860895  
 Значение ломаной: 1.405590701358537  
 Значение Лагранжа: 1.395629743887834  
 Разница c функцией: 0.000017  
 Разница c ломаной: 0.0099609574707029  
  
Точка: 0.5  
 Значение ф-ии: 1.6487212707001282  
 Значение ломаной: 1.6487212707001282  
 Значение Лагранжа: 1.6487212707001282  
 Разница c функцией: 0.000000  
 Разница c ломаной: 0.0000000000000000  
  
Точка: 0.6666666666666666  
 Значение ф-ии: 1.9477340410546757  
 Значение ломаной: 1.9609071013084924  
 Значение Лагранжа: 1.9477157346643834  
 Разница c функцией: 0.000018  
 Разница c ломаной: 0.0131913666441090  
  
Точка: 0.8333333333333334  
 Значение ф-ии: 2.300975890892825  
 Значение ломаной: 2.317427287228132  
 Значение Лагранжа: 2.301008855521036  
 Разница c функцией: 0.000033  
 Разница c ломаной: 0.0164184317070957  
  
Точка: 1.0  
 Значение ф-ии: 2.718281828459045  
 Значение ломаной: 2.718281828459045  
 Значение Лагранжа: 2.718281828459045  
 Разница c функцией: 0.000000  
 Разница c ломаной: 0.0000000000000000

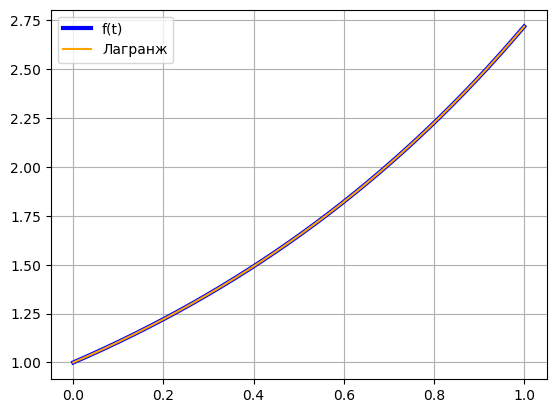
def R(t, x, f, z):  
 p = 1  
 d = f  
 for i in range(len(x)-1):  
 p \*= (t - x[i])  
 d = sp.diff(d, z)  
 d1 = np.abs(d.subs(z, x[0]))  
 d2 = np.abs(d.subs(z, x[len(x)-1]))  
 max\_d = max(d1, d2)  
 return max\_d \* np.abs(p) / math.factorial(len(x))  
  
t = []  
for i in range(5):  
 t.append((i+2)/6)  
  
x = np.linspace(0, 1, 5)  
z = sp.symbols('z')  
f = np.e\*\*z  
for i in range(len(t)):  
 print("Точка: ", t[i])  
 th = R(t[i], x, f, z)  
 print("\tТеоретическая погрешность:", th)  
 ex = np.abs(fxList[i] - ftList[i])  
 print(f"\tЭкспериментальная погрешность: {ex:f}")  
 print(f"\tРазница погрешностей: {np.abs(ex - th)}")  
 print("")

Точка: 0.3333333333333333  
 Теоретическая погрешность: 4.36966600511035e-5  
 Экспериментальная погрешность: 0.000017  
 Разница погрешностей: 0.0000263778583066110  
  
Точка: 0.5  
 Теоретическая погрешность: 0  
 Экспериментальная погрешность: 0.000000  
 Разница погрешностей: 0  
  
Точка: 0.6666666666666666  
 Теоретическая погрешность: 8.73933201022070e-5  
 Экспериментальная погрешность: 0.000018  
 Разница погрешностей: 0.0000690869298098850  
  
Точка: 0.8333333333333334  
 Теоретическая погрешность: 0.000305876620357725  
 Экспериментальная погрешность: 0.000033  
 Разница погрешностей: 0.000272911992146606  
  
Точка: 1.0  
 Теоретическая погрешность: 0.00212365767848363  
 Экспериментальная погрешность: 0.000000  
 Разница погрешностей: 0.00212365767848363

## 2.1

image.png

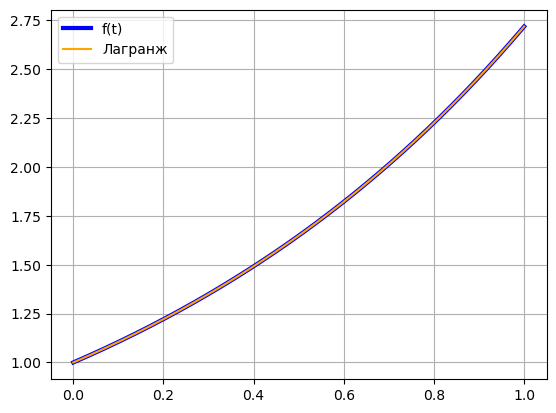
def f(x):  
 return np.e\*\*x  
  
def Lagrange(t, x, fx):  
 result = 0  
 for i in range(len(x)):  
 y = fx[i]  
 for j in range(len(x)):  
 if i != j:  
 y = y \* ((t - x[j])/(x[i] - x[j]))  
 result += y  
 return result  
  
x = np.linspace(0, 1, 5)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(0, 1, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(0, 1, 1000)  
xt = x.tolist() + t.tolist()  
xt.sort()  
plt.plot(xt, Lagrange(xt, x, fx), color = 'orange', label = 'Лагранж')  
plt.legend()  
plt.grid()  
plt.show()



## 3.1

image.png

x = np.linspace(0, 1, 10)  
x.tolist()  
f = np.e\*\*x  
#print(x)  
n = len(x)  
  
#a = x[:, np.newaxis]  
#print(a)  
x = np.tile(x[:, np.newaxis], (1, n)) #продублирование столбца из x\_0, ..., x\_n n раз  
#print(x)  
#print(temp)  
degree = np.tile(np.arange(n), (n, 1)) #продублирование строки из 0, ..., 9 n раз  
#print(degree)  
X = x \*\* degree #каждый элемент возводится в степень  
A = np.linalg.inv(X).dot(f) #нахождение коэффициентов   
#(решение системы уравнений для коэффициентов A путем умножения обратной матрицы X на вектор значений f)  
  
  
def P(x, A):  
 n = len(A)  
 result = 0  
 for i in range(n):  
 result += A[i] \* x\*\*i  
 return result  
  
def f(x):  
 return np.e\*\*x  
   
z = np.linspace(0, 1, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
#plt.plot(x, f, 'bo', label='Исходные точки')  
t = np.linspace(0, 1, 1000)  
plt.plot(t, P(t, A), color = 'orange', label = 'Лагранж')  
plt.legend()  
plt.grid()  
plt.show()



## 1.3

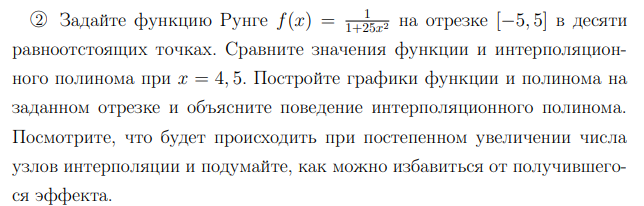
image.png

f2 = f(2)  
print("Значение ф-ии:", f2)  
L = Lagrange(2, x, fx)  
print("Значение Лагранжа:", L)  
print("Разница:", np.abs(L - f2))

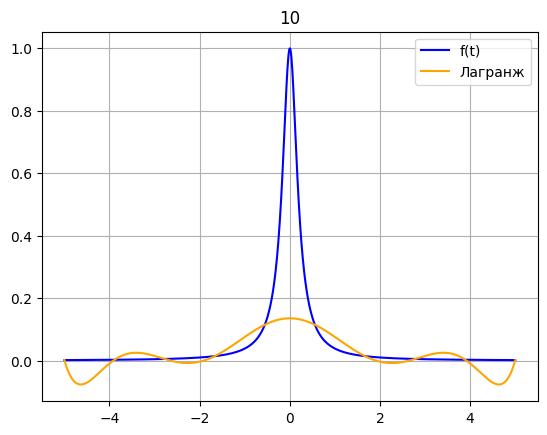
Значение ф-ии: 7.3890560989306495  
Значение Лагранжа: 7.269613396891231  
Разница: 0.11944270203941887

Точка t = 2 находится далеко от узлов и не принадлежит отрезку между двумя узлами

## 2.2



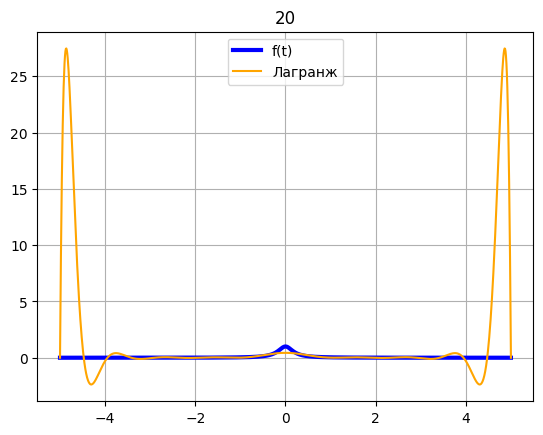
def f(x):  
 return 1 / (1 + 25 \* x\*\*2)  
  
def Lagrange(t, x, fx):  
 result = 0  
 for i in range(len(x)):  
 y = fx[i]  
 for j in range(len(x)):  
 if i != j:  
 y = y \* ((t - x[j])/(x[i] - x[j]))  
 result += y  
 return result  
  
x = np.linspace(-5, 5, 10)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)')  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(10)  
plt.legend()  
plt.grid()  
plt.show()



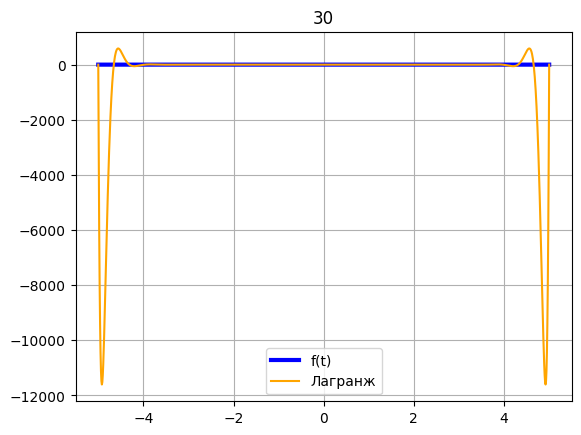
f2 = f(4.5)  
print("Значение ф-ии:", f2)  
L = Lagrange(4.5, x, fx)  
print("Значение Лагранжа:", L)  
print("Разница:", np.abs(L - f2))

Значение ф-ии: 0.001971414489896501  
Значение Лагранжа: -0.0704934541952141  
Разница: 0.07246486868511061

x = np.linspace(-5, 5, 20)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(20)  
plt.legend()  
plt.grid()  
plt.show()



x = np.linspace(-5, 5, 30)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(30)  
plt.legend()  
plt.grid()  
plt.show()



Поведение интерполяционного полинома, особенно в случае интерполяции функции Рунге, обусловлено явлением, известным как "эффект Рунге".

Эффект Рунге проявляется при использовании высокостепенных интерполяционных многочленов на равномерной сетке узлов. В случае функции Рунге, у которой производные быстро изменяются вблизи краев интерполяционного отрезка, высокостепенной интерполяционный многочлен может привести к резким осцилляциям (колебаниям) вблизи краев интерполяции.

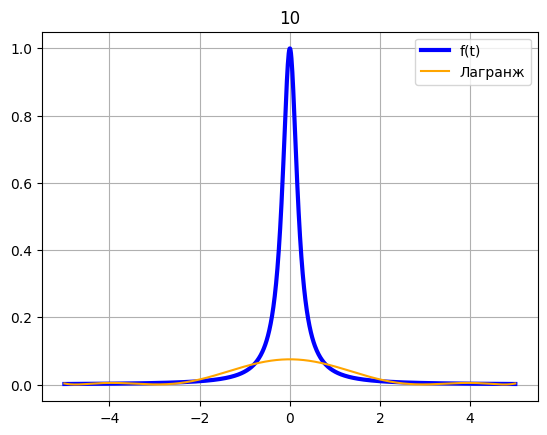
Это происходит из-за того, что при использовании равномерной сетки узлов интерполяции на отрезке, расстояние между узлами становится слишком малым на краях отрезка, где функция имеет большие значения производных. Это приводит к тому, что интерполяционный полином начинает "перекачивать" значения функции в попытке аппроксимировать ее, что приводит к нежелательным осцилляциям.

Для избежания эффекта Рунге рекомендуется использовать методы регуляризации, такие как интерполяция сплайнами или выбор оптимальных узлов интерполяции, например, узлов Чебышева. Эти методы позволяют уменьшить осцилляции и обеспечить более точную интерполяцию функции на заданном отрезке.

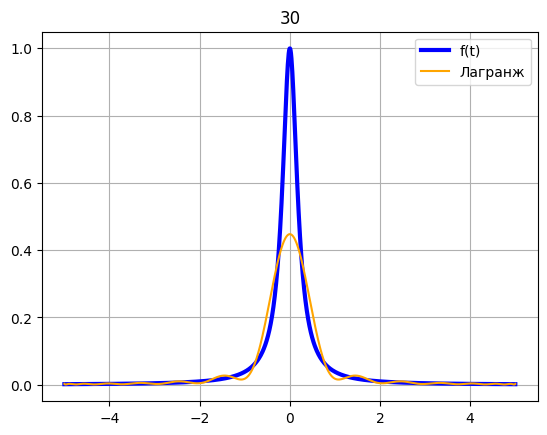
# 3.2

image.png

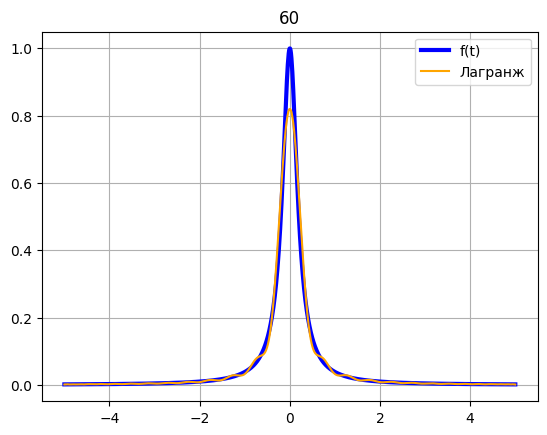
# x = []  
# for i in range(0, 19, 2):  
# x.append(np.cos((19-i)\*np.pi/20))  
# print(x)  
# fx = []  
# for xi in x:  
# fx.append(f(xi))  
  
def f(x):  
 return 1 / (1 + 25 \* x\*\*2)  
  
def chebyshev(a, b, n):  
 k = np.arange(1, n+1)  
 x = 0.5 \* (a + b) + 0.5 \* (b - a) \* np.cos((2\*k - 1) \* np.pi / (2\*n))  
 return x  
  
c = 10  
  
x = chebyshev(-5, 5, c)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(c)  
plt.legend()  
plt.grid()  
plt.show()



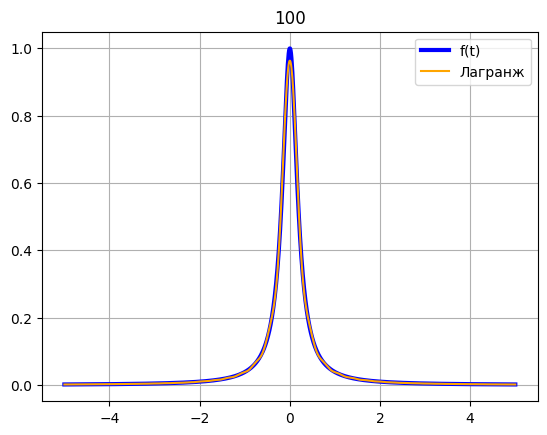
c = 30  
  
x = chebyshev(-5, 5, c)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(c)  
plt.legend()  
plt.grid()  
plt.show()



c = 60  
  
x = chebyshev(-5, 5, c)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(c)  
plt.legend()  
plt.grid()  
plt.show()

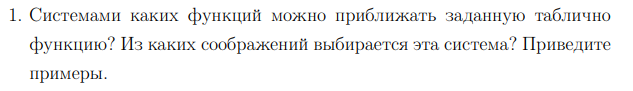


c = 100  
  
x = chebyshev(-5, 5, c)  
fx = []  
for xi in x:  
 fx.append(f(xi))  
  
  
z = np.linspace(-5, 5, 1000)  
plt.plot(z, f(z), color = 'blue', label = 'f(t)', linewidth = 3)  
t = np.linspace(-5, 5, 1000)  
plt.plot(t, Lagrange(t, x, fx), color = 'orange', label = 'Лагранж')  
plt.title(c)  
plt.legend()  
plt.grid()  
plt.show()

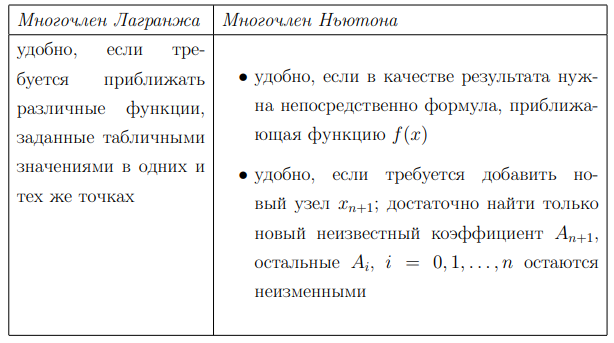


## Контрольные вопросы

### 1



Многочлен Лагранжа, многочлен Ньютона.



## 2

image.png

Коэффициенты Лагранжа Pn(x) зависят от выбора узлов xi и точки, но не зависят от вида функции. Это удобно, когда по заданной системе узлов надо интерполировать несколько различных функций.

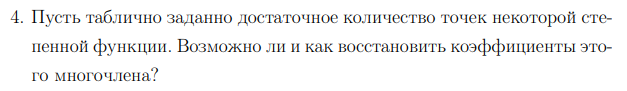
Во многих случаях интерполяционный многочлен Ньютона более удобен, чем интерполяционный многочлен Лагранжа. Особенность этого многочлена заключается в том, что при переходе от многочлена -ой степени к многочлен(n+1) -й степени первыn+1е членов не меняются, а только добавляется новый члта. Формула Лагранжа этого делать не позволяет, так как в ней добавление нового узла заставляет заново пересчитывать все коэффициенты .

## 3

image.png

Степени n-1 только один, степени n - много

## 4



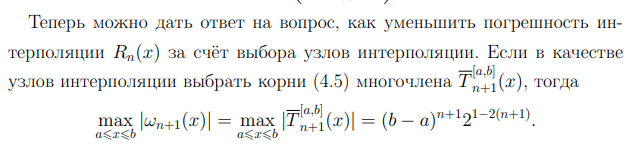
A = np.linalg.inv(X).dot(f) #нахождение коэффициентов

#(решение системы уравнений для коэффициентов A путем умножения обратной матрицы X на вектор значений f)

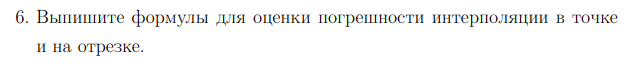
## 5

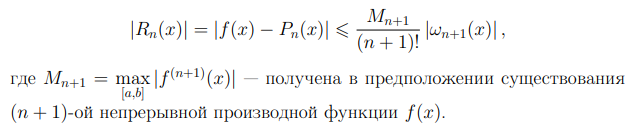
image.png

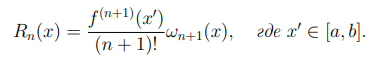
Нужно выбрать такие узлы, чтобы расстояние между ними на концах отрезка были мало, а ближе к середине отрезка увеличивалось



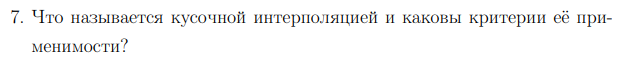
## 6



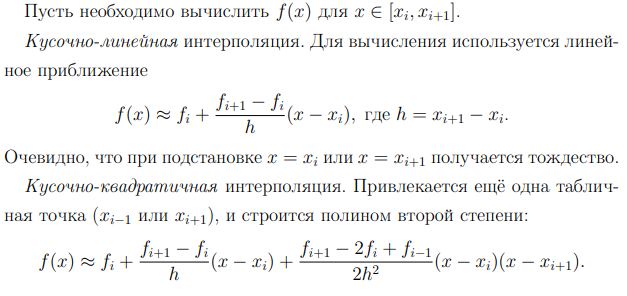


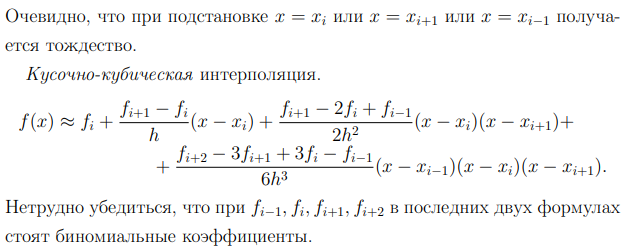


## 7



Соседние два узла соединяются отрезком прямой. Применяется в случаях, когда использование одного общего интерполяционного многочлена приводит к сложным вычислениям





## 8

