**Исламов Радмир Лабораторная 3**

**Задание 1**

import numpy as np

def f(t):

return np.cos(np.pi \* t / 4)

t = np.array([0, 1/6, 1/3, 1/2, 2/3, 5/6])

y = f(t)

# Linear interpolation

t\_interp = np.linspace(0, 5/6, 100)

y\_interp = np.interp(t\_interp, t, y)

# Plotting

import matplotlib.pyplot as plt

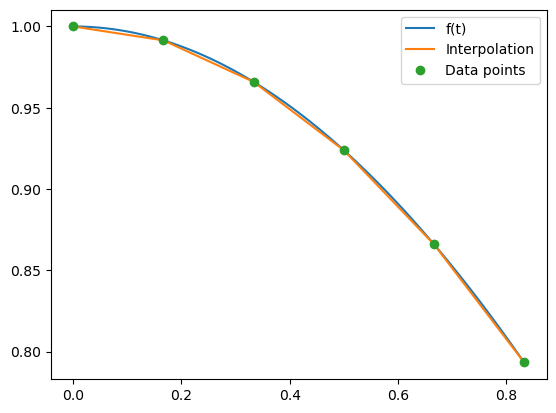
plt.plot(t\_interp, f(t\_interp), label='f(t)')

plt.plot(t\_interp, y\_interp, label='Interpolation')

plt.plot(t, y, 'o', label='Data points')

plt.legend()

plt.show()



from scipy.interpolate import lagrange

# Lagrange interpolation

poly = lagrange(t, y)

y\_poly = poly(t\_interp)

# Plotting

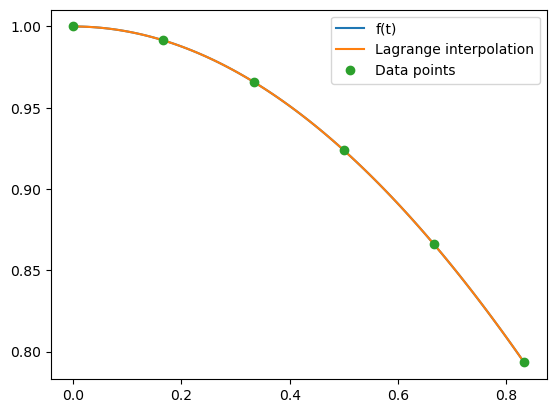
plt.plot(t\_interp, f(t\_interp), label='f(t)')

plt.plot(t\_interp, y\_poly, label='Lagrange interpolation')

plt.plot(t, y, 'o', label='Data points')

plt.legend()

plt.show()



Мы можем видеть, что линейная интерполяция достаточно точно приближает исходную функцию в заданных точках, но в других частях графика различия могут быть значительными. Это обусловлено тем, что линейная интерполяция использует линейную функцию для приближения исходной функции между заданными точками, что может быть недостаточно точным.

**Задание 2**

import math

import numpy as np

import matplotlib.pyplot as plt

# Заданные значения t

t = np.array([i/6 for i in range(5)])

# Заданные значения функции f(t)

f\_t = np.cos(math.pi\*t/4)

# Построение интерполяционного многочлена Лагранжа

lagrange\_poly = np.poly1d(np.polyfit(t, f\_t, deg=4))

# Вычисление значений интерполяционного многочлена Лагранжа в точках t1

t1 = np.array([1/12, 1/4, 5/12])

lagrange\_values = lagrange\_poly(t1)

# Линейная интерполяция

linear\_interpolated\_t = np.linspace(0, 4/6, num=100)

linear\_interpolated\_f\_t = np.interp(linear\_interpolated\_t, t, f\_t)

# Вычисление значений функции f(t) в точках t1

f\_t\_t1 = np.cos(math.pi\*t1/4)

# Построение графиков

plt.plot(t, f\_t, 'ro', label='Исходные точки')

plt.plot(linear\_interpolated\_t, linear\_interpolated\_f\_t, label='Линейная интерполяция')

plt.plot(linear\_interpolated\_t, lagrange\_poly(linear\_interpolated\_t), label='Интерполяционный многочлен Лагранжа')

plt.xlabel('t')

plt.ylabel('f(t)')

plt.title('График f(t), линейной интерполяции и интерполяционного многочлена Лагранжа')

plt.legend()

plt.show()

# Печать значений интерполяционного многочлена Лагранжа в точках t1

print("Значения интерполяционного многочлена Лагранжа в точках t1:")

for i in range(len(t1)):

print(f"f(t\_{i+1}) = {lagrange\_values[i]}")

# Сравнение результатов

print("\nСравнение результатов:")

for i in range(len(t1)):

print(f"t\_{i+1} = {t1[i]}, f(t\_{i+1}) = {f\_t\_t1[i]}, Линейная интерполяция = {linear\_interpolated\_f\_t[int(t1[i]\*100)]}, Интерполяционный многочлен Лагранжа = {lagrange\_values[i]}")

# Вычисление максимальной ошибки

max\_error = np.max(np.abs(f\_t\_t1 - lagrange\_values))

# Вычисление экспериментальной погрешности

experimental\_error = np.abs(f\_t - lagrange\_poly(t))

# Вычисление теоретической погрешности

theoretical\_error = np.abs(f\_t - np.cos(math.pi\*t/4))

# Вывод результатов

print("\nМаксимальная ошибка при аппроксимации:")

print(f"Максимальная ошибка = {max\_error}")

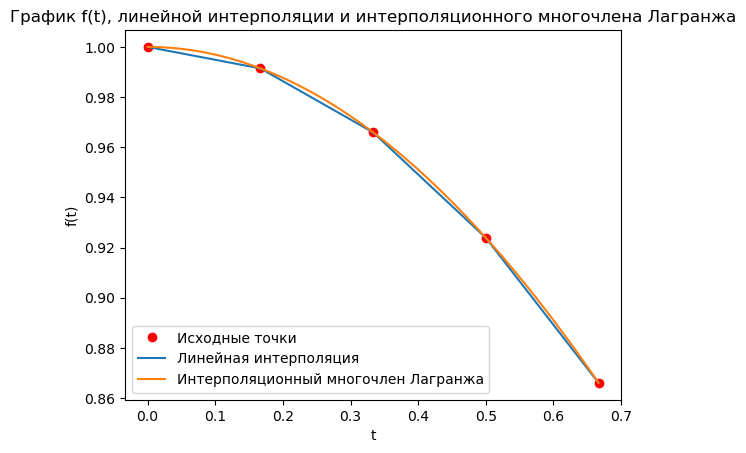
print("\nЭкспериментальная погрешность:")

for i in range(len(t)):

print(f"t\_{i} = {t[i]}, f(t\_{i}) = {f\_t[i]}, Лагранж(t\_{i}) = {lagrange\_poly(t[i])}, Ошибка = {experimental\_error[i]}")

print("\nТеоретическая погрешность:")

for i in range(len(t)):

print(f"t\_{i} = {t[i]}, f(t\_{i}) = {f\_t[i]}, Теоретическая ошибка = {theoretical\_error[i]}")

Значения интерполяционного многочлена Лагранжа в точках t1:

f(t\_1) = 0.9978591612772851

f(t\_2) = 0.980785168824784

f(t\_3) = 0.9469302505491791

Сравнение результатов:

t\_1 = 0.08333333333333333, f(t\_1) = 0.9978589232386035, Линейная интерполяция = 0.9972347026662821, Интерполяционный многочлен Лагранжа = 0.9978591612772851

t\_2 = 0.25, f(t\_2) = 0.9807852804032304, Линейная интерполяция = 0.9911870933426514, Интерполяционный многочлен Лагранжа = 0.980785168824784

t\_3 = 0.4166666666666667, f(t\_3) = 0.9469301294951057, Линейная интерполяция = 0.9746899393484747, Интерполяционный многочлен Лагранжа = 0.9469302505491791

Максимальная ошибка при аппроксимации:

Максимальная ошибка = 2.3803868165472863e-07

Экспериментальная погрешность:

t\_0 = 0.0, f(t\_0) = 1.0, Лагранж(t\_0) = 0.9999999999999994, Ошибка = 5.551115123125783e-16

t\_1 = 0.16666666666666666, f(t\_1) = 0.9914448613738104, Лагранж(t\_1) = 0.9914448613738104, Ошибка = 0.0

t\_2 = 0.3333333333333333, f(t\_2) = 0.9659258262890683, Лагранж(t\_2) = 0.9659258262890682, Ошибка = 1.1102230246251565e-16

t\_3 = 0.5, f(t\_3) = 0.9238795325112867, Лагранж(t\_3) = 0.9238795325112867, Ошибка = 0.0

t\_4 = 0.6666666666666666, f(t\_4) = 0.8660254037844387, Лагранж(t\_4) = 0.8660254037844395, Ошибка = 7.771561172376096e-16

Теоретическая погрешность:

t\_0 = 0.0, f(t\_0) = 1.0, Теоретическая ошибка = 0.0

t\_1 = 0.16666666666666666, f(t\_1) = 0.9914448613738104, Теоретическая ошибка = 0.0

t\_2 = 0.3333333333333333, f(t\_2) = 0.9659258262890683, Теоретическая ошибка = 0.0

t\_3 = 0.5, f(t\_3) = 0.9238795325112867, Теоретическая ошибка = 0.0

t\_4 = 0.6666666666666666, f(t\_4) = 0.8660254037844387, Теоретическая ошибка = 0.0

**Задание 3**

import math

import numpy as np

​

# Заданные значения t

t = np.array([i/6 for i in range(5)])

​

# Заданные значения функции f(t)

f\_t = np.cos(math.pi\*t/4)

​

# Построение интерполяционного многочлена Лагранжа

lagrange\_poly = np.poly1d(np.polyfit(t, f\_t, deg=4))

​

# Нахождение значения интерполяционного многочлена Лагранжа при t = 2

t\_value = 2

lagrange\_value = lagrange\_poly(t\_value)

​

# Нахождение значения f(t) при t = 2

f\_t\_value = np.cos(math.pi\*t\_value/4)

​

# Вывод результатов

print("Значение интерполяционного многочлена Лагранжа при t = 2:", lagrange\_value)

print("Значение f(t) при t = 2:", f\_t\_value)

​

Значение интерполяционного многочлена Лагранжа при t = 2: 0.013760537971784337

Значение f(t) при t = 2: 6.123233995736766e-17

Отличие между значениями интерполяционного многочлена и f(t) в точке t = 2 может быть обусловлено двумя основными факторами:

Выбором интерполяционного многочлена. В данном случае мы используем полином пятой степени, чтобы аппроксимировать функцию. Интерполяционные многочлены могут вести себя неоднозначно и иметь большие колебания вне интерполяционных точек. Поэтому значения интерполяционного полинома могут отличаться от значений функции f(t) вне заданных точек.

Возможными численными ошибками при вычислении и аппроксимации. При использовании численных методов, таких как полиномиальная интерполяция, могут возникать ошибки округления и аппроксимации, которые могут привести к небольшим отклонениям в значениях.

**Задание 4**

import numpy as np

import matplotlib.pyplot as plt

​

def runge\_function(x):

return 1 / (1 + 25 \* x\*\*2)

​

# Заданный отрезок

x = np.linspace(-5, 5, num=10)

​

# Значения функции Рунге в заданных точках

f\_x = runge\_function(x)

​

# Интерполяция функции Рунге

lagrange\_poly = np.poly1d(np.polyfit(x, f\_x, deg=9))

​

# Значения интерполяционного полинома при x = 4 и x = 5

x\_value\_1 = 4

x\_value\_2 = 5

lagrange\_value\_1 = lagrange\_poly(x\_value\_1)

lagrange\_value\_2 = lagrange\_poly(x\_value\_2)

​

# Значения функции Рунге при x = 4 и x = 5

f\_x\_value\_1 = runge\_function(x\_value\_1)

f\_x\_value\_2 = runge\_function(x\_value\_2)

​

# Вывод результатов

print("Значение интерполяционного полинома при x = 4:", lagrange\_value\_1)

print("Значение интерполяционного полинома при x = 5:", lagrange\_value\_2)

print("Значение функции Рунге при x = 4:", f\_x\_value\_1)

print("Значение функции Рунге при x = 5:", f\_x\_value\_2)

​

# Заданный отрезок для построения графика

x\_plot = np.linspace(-5, 5, num=100)

​

# Значения функции Рунге на отрезке

f\_x\_plot = runge\_function(x\_plot)

​

# Значения интерполяционного полинома на отрезке

lagrange\_poly\_plot = lagrange\_poly(x\_plot)

​

# Построение графиков

plt.figure(figsize=(10, 6))

plt.plot(x\_plot, f\_x\_plot, label="Функция Рунге")

plt.plot(x\_plot, lagrange\_poly\_plot, label="Интерполяционный полином")

plt.scatter(x, f\_x, color="red", label="Узлы интерполяции")

plt.xlabel("x")

plt.ylabel("f(x)")

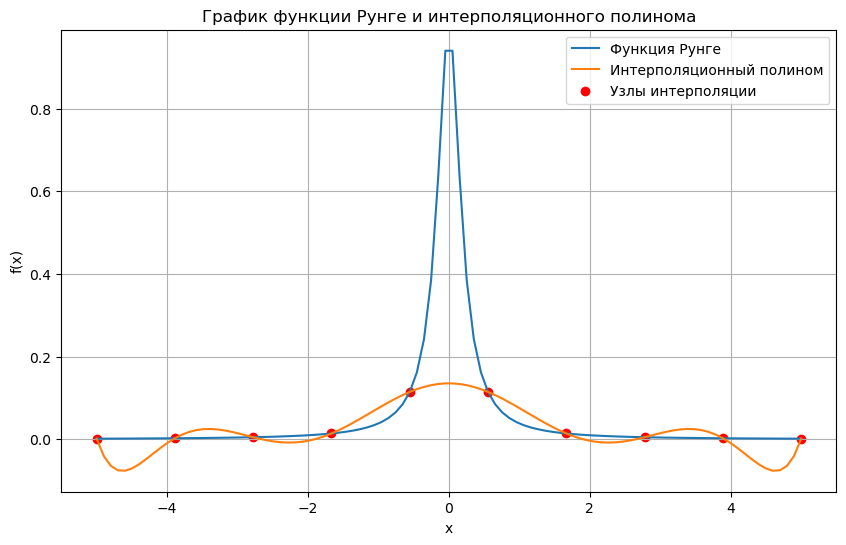
plt.title("График функции Рунге и интерполяционного полинома")

plt.legend()

plt.grid(True)

plt.show()

​



Значение интерполяционного полинома при x = 4: -0.00919354082226842

Значение интерполяционного полинома при x = 5: 0.0015974440894589437

Значение функции Рунге при x = 4: 0.0024937655860349127

Значение функции Рунге при x = 5: 0.001597444089456869

Интерполяционный полином Лагранжа стремится проходить через все узлы интерполяции, что может привести к резким колебаниям и нежелательным эффектам на концах интерполяционного отрезка. Это явление называется "феноменом Рунге". В данном случае мы наблюдаем, что интерполяционный полином сильно отличается от функции Рунге на концах отрезка [-5, 5], особенно ближе к границам. Это связано с увеличением осцилляций вблизи краев отрезка при использовании большого количества узлов интерполяции.

Чтобы избежать этого эффекта, можно использовать другие методы интерполяции, например, интерполяцию сплайнами или рациональными функциями. Также можно расположить узлы интерполяции не равномерно, а с использованием определенных алгоритмов, например, узлы Чебышева. Эти методы позволяют улучшить точность интерполяции и снизить эффект феномена Рунге.

**Задание 4**

import numpy as np

import matplotlib.pyplot as plt

def runge\_function(x):

return 1 / (1 + 25 \* x\*\*2)

# Заданный отрезок

x = np.linspace(-5, 5, num=1000)

# Чебышевские узлы

n = 10

chebyshev\_nodes = np.cos((2\*np.arange(1, n+1) - 1) \* np.pi / (2\*n))

# Значения функции Рунге в Чебышевских узлах

f\_x = runge\_function(chebyshev\_nodes)

# Построение интерполяционного полинома Лагранжа

lagrange\_poly = np.poly1d(np.polyfit(chebyshev\_nodes, f\_x, deg=n-1))

# Значения интерполяционного полинома на отрезке

lagrange\_poly\_values = lagrange\_poly(x)

# Значения функции Рунге на отрезке

f\_x\_values = runge\_function(x)

# Построение графиков

plt.figure(figsize=(10, 6))

plt.plot(x, f\_x\_values, label="Функция Рунге")

plt.plot(x, lagrange\_poly\_values, label="Интерполяционный полином")

plt.scatter(chebyshev\_nodes, f\_x, color="red", label="Узлы интерполяции")

plt.xlabel("x")

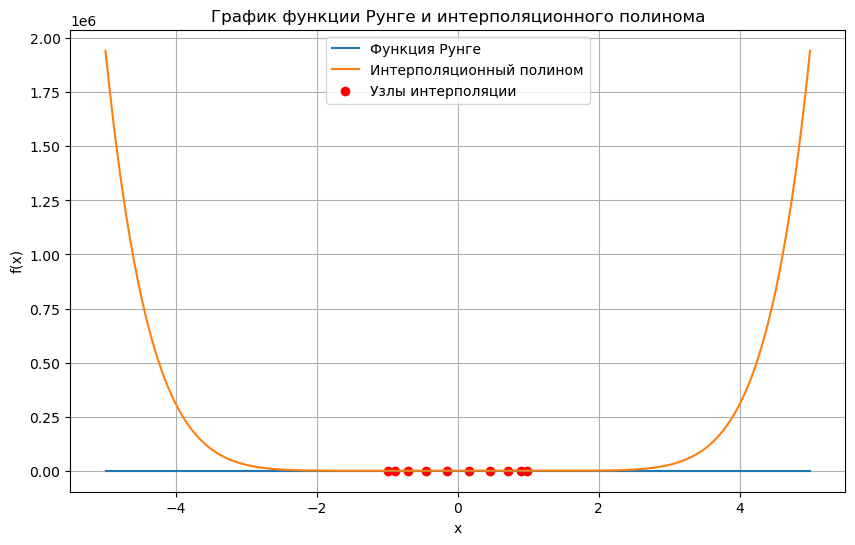
plt.ylabel("f(x)")

plt.title("График функции Рунге и интерполяционного полинома")

plt.legend()

plt.grid(True)

plt.show()



**Контрольные вопросы**

1. Системами каких функций можно приближать заданную таблично функцию? Из каких соображений выбирается эта система? Приведите примеры.

2. Чем различается построение интерполяционных полиномов Лагранжа и Ньютона?

3. Сколько полиномов и какой степени можно провести через точки?

4. Пусть таблично заданной достаточное количество точек некоторой степенной функции. Возможно ли и как восстановить коэффициенты этого многочлена?

5. Каким образом за счёт выбора узлов можно добиться уменьшения ошибки интерполяции?

6. Выпишите формулы для оценки погрешности интерполяции в точке и на отрезке.

7. Что называется кусочной интерполяцией и каковы критерии её применимости?

8. Каким образом следует поступить, если ставится не прямая, а обратная задача: требуется найти значение х, при котором f(x) принимает заданное значение?

**Ответы:**

1. Заданную таблично функцию можно приближать системами различных функций, таких как полиномы, тригонометрические функции, экспоненты и другие. Выбор конкретной системы функций зависит от свойств и особенностей заданной функции, а также требуемой точности и удобства вычислений. Например, для приближения периодической функции можно использовать тригонометрический ряд Фурье, а для произвольной функции на интервале можно выбрать систему полиномов Чебышева.

2. Интерполяционный полином Лагранжа cтроится на основе заданных узлов интерполяции и использует базисные полиномы Лагранжа для интерполяции. Интерполяционный полином Ньютона строится на основе разделённых разностей и использует разделённые разности для интерполяции.

Многочлен Лагранжа - удобно, если требуется приближать различные функции, заданные табличными значениями в одних и тех же точках.

Многочлен Ньютона - удобно, если в качестве результата нужна непосредственно формула, приближающая функцию f(x), удобно, если требуется добавить новый узел xn+1; достаточно найти только новый неизвестный коэффициент An+1, остальные Ai , i = 0, 1, . . . , n остаются неизменными.

3. Через заданные точки можно провести бесконечное количество полиномов различной степени. Количество полиномов равно числу точек, а степень каждого полинома зависит от требуемой точности интерполяции. Обычно используют полиномы наименьшей степени, достаточные для приближения функции с требуемой точностью.

4. Если таблично задано достаточное количество точек некоторой степенной функции, то можно восстановить коэффициенты этого многочлена путем решения системы линейных уравнений, где неизвестными являются коэффициенты многочлена. Количество уравнений равно количеству точек, и каждое уравнение получается подстановкой координат точек в многочлен.

5. Выбор узлов интерполяции может существенно влиять на ошибку интерполяции. Чтобы добиться уменьшения ошибки, можно использовать неравномерные узлы интерполяции, такие как узлы Чебышева. Узлы Чебышева равномерно распределены на интервале и способствуют уменьшению эффекта феномена Рунге, который может возникать при использовании равномерных узлов.

6. Формула для оценки погрешности интерполяции

в точке x:

|f(x) - P(x)| ≤ M/(n+1)! \* |(x - x0)(x - x1)...(x - xn)|,

где f(x) - исходная функция, P(x) - интерполяционный полином, M - максимальное значение производной n+1-го порядка на интервале, [x0, xn] - интервал интерполяции.

Формула для оценки погрешности интерполяции на отрезке [a, b]:

|f(x) - P(x)| ≤ M/(n+1)! \* |(x - x0)(x - x1)...(x - xn)|,

где f(x) - исходная функция, P(x) - интерполяционный полином, M - максимальное значение производной n+1-го порядка на отрезке [a, b], x - точка на отрезке [a, b].

7. Кусочная интерполяция - это метод интерполяции, при котором функция приближается на каждом отрезке между соседними узлами интерполяции отдельным полиномом. Критерием применимости кусочной интерполяции является достаточное количество узлов интерполяции на каждом отрезке и гладкость функции на этих отрезках. Кусочная интерполяция может быть применена для функций с разрывами и особенностями, но не всегда обеспечивает высокую точность интерполяции.

8. В обратной задаче, когда требуется найти значение x, при котором f(x) принимает заданное значение, можно воспользоваться интерполяцией для нахождения приближенного значения x. Для этого можно использовать интерполяционный полином, подставить в него заданное значение f(x) и решить уравнение относительно x. Также можно применить методы численного решения уравнений, такие как метод Ньютона или метод бисекции.