|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

по дисциплине: «Вычислительная математика»

|  |  |  |
| --- | --- | --- |
| Студент |  | Шашко Олег Владимирович |
| Группа |  | РК6-51Б |
| Тип задания |  | Лабораторная работа |
| Тема лабораторной работы |  | Интерполяция Лагранжа |

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шашко О.В.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соколов А.П.**

*подпись, дата фамилия, и.о.*

*Москва, 2022 г.*

Оглавление

[Задание на лабораторную работу 3](#_Toc115818757)

[Цель выполнения лабораторной работы 5](#_Toc115818758)

[Выполненные задачи 5](#_Toc115818759)

[Выполнение задач 6](#_Toc115818760)

[Заключение 13](#_Toc115818760)

[Список использованных источников 13](#_Toc115818761)

# Задание на лабораторную работу

1.1 Требования к знаниям для выполнения

Для выполнения лабораторной̆ работы обучающийся должен обладать знаниями:  
– владеть навыками разработки программного обеспечения на языке Python (рекомендуется) или С++ на базовом уровне;  
– владеть навыками использования программных инструментов: numpy, matplotlib; – знать понятия: интерполяция, интерполяционный̆ полином Лагранжа, принципы интерполяции кубическими сплайн-функциями.

1.2 Интерполяция Лагранжа (вариант 1)

Интерполяция Лагранжа является одним из самых важных численных методов и лежит в основе многих методов численного дифференцирования и интегрирования. Точность интерполяции полиномами Лагранжа зависит не только от максимальной̆ степени выбранного подмножества полиномов, но и от расположения узлов. Очевидный̆, казалось бы, выбор равномерно расположенных узлов может приводить к неожиданным проблемам. Одним из примеров является так называемый̆ эффект Рунге, который̆ выражается в большой̆ осцилляции аппроксимированного полинома вблизи конечных узлов отрезка интерполирования и который̆ предлагается исследовать в базовой̆ части. В продвинутой̆ части предлагается исследовать влияние расположения узлов и их количества на интерполяцию Лагранжа более систематически, используя случайные функции, сгенерированные с помощью аппроксимации Паде. [ссылку на лекции или методичку следует добавить]

Задача 1 (интерполирование полиномами Лагранжа)

(1) ,

где 𝑥 ∈ [−1; 1]. Также дана рациональная функция, известная как аппроксимация Паде:

, где 𝑥 ∈ [−1; 1].

Требуется (базовая часть):  
1. Разработать функцию l\_i(i, x, x\_nodes), которая возвращает значение 𝑖-го базисного полинома Лагранжа, заданного на узлах с абсциссами x\_nodes, в точке 𝑥.

2. Написать функцию L(x, x\_nodes, y\_nodes), которая возвращает значение интерполяционного полинома Лагранжа, заданного на узлах с абсциссами x\_nodes и ординатами y\_nodes, в точке 𝑥.

3. Для равномерно расположенных узлов вывести на экран одновременно графики 𝑓(𝑥) и полученного интерполяционного полинома 𝐿(𝑥) для следующих количеств узлов: 5, 8, 11, 14, 17, 20, 23. В результате это должно дать 7 пар графиков. Опишите, что наблюдается при увеличении количества узлов?

4. Повторить предыдущий̆ пункт для чебышевских узлов. В чем разница между интерполяцией̆ Лагранжа функции 𝑓(𝑥) на основе равномерно расположенных узлов и чебышевских? Сделать выводы.

Требуется (продвинутая часть):

1. Сгенерировать 100 функции 𝑓𝑛,𝑚(𝑥), где целые степени 𝑛,𝑚 ∈ [7;15] и вещественные коэффициенты 𝑎𝑗 , 𝑏𝑘 ∈ [0; 1] генерируются случайным образом для каждой̆ из функции.

2. Для нескольких из сгенерированных функций вывести на экран одновременно графики 𝑓𝑛,𝑚(𝑥) и соответствующего интерполяционного полинома 𝐿(𝑥), построенного по 𝑁 равномерно расположенным узлам, где 𝑁 выбирается по собственному усмотрению, но должно быть не меньше 5. На том же графике выведите 𝐿(𝑥), построенного по 𝑁 чебышевским узлам.

3. Для каждой̆ из функции, сгенерированных в предыдущем пункте, найдите интерполяционные полиномы 𝐿(𝑥), построенные по 𝑁 ∈ {1,2,...,30} равномерно расположенным узлам и чебышевским узлам. Для каждого 𝑁 рассчитаете расстояние между и 𝐿(𝑥) в лебеговом пространстве 𝐿∞ .2 Рассмотрите несколько графиков зависимости этого расстояния для равномерных и чебышевских узлов от 𝑁 и сделаете по ним вывод.3 Добавьте в отчет один характерный̆ график, который̆ наглядно демонстрирует верность вашего вывода.

4. Объясните, что такое аппроксимация Паде и до какой̆ степени предложенный̆ метод генерации случайных функций позволяет обобщить выводы предыдущего пункта на произвольные функции.

# Цель выполнения лабораторной работы

**Цель выполнения лабораторной работы** – реализовать алгоритм интерполяции произвольной функции методом Лагранжа в зависимости от заданного числа узлов; рассмотреть два вида узлов − чебышевские и равномерно расположенные, на примере функции Рунге оценить степень точности приближения в обоих случаях; рассмотреть влияние выбора числа точек интерполяции и двух типов на примере функции Паде.

# Выполненные задачи

* **Задача 1 (интерполирование полиномами Лагранжа)**

1. Была разработана функция l\_i(i, x, x\_nodes), которая возвращает значение 𝑖-го базисного полинома Лагранжа, заданного на узлах с абсциссами x\_nodes, в точке 𝑥.
2. Была написана функция L(x, x\_nodes, y\_nodes), которая возвращает значение ин- терполяционного полинома Лагранжа, заданного на узлах с абсциссами x\_nodes и ординатами y\_nodes, в точке 𝑥.
3. Для равномерно расположенных узлов было выведено на экран одновременно графики 𝑓(𝑥) и полученного интерполяционного полинома 𝐿(𝑥) для следующих количеств узлов: 5, 8, 11, 14, 17, 20, 23 и описано что наблюдалось при росте количества узлов.
4. Был повторен предыдущий̆ пункт для чебышевских узлов и сделаны выводы, в чем разница между интерполяцией̆ Лагранжа функции 𝑓(𝑥) на основе равномерно расположенных узлов и чебышевских.
5. Было сгенерировано 100 функции 𝑓𝑛,𝑚(𝑥), где целые степени 𝑛,𝑚 ∈ [7;15] и веще- ственные коэффициенты 𝑎𝑗 , 𝑏𝑘 ∈ [0; 1] генерируются случайным образом для каждой из функции
6. Для нескольких из сгенерированных функций выведены на экран одновременно графики 𝑓𝑛,𝑚(𝑥) и соответствующего интерполяционного полинома 𝐿(𝑥), построенного по 𝑁 равномерно расположенным узлам, где 𝑁 выбирается по собственному усмотрению, но должно быть не меньше 5. На том же графике выведено 𝐿(𝑥), построенного по 𝑁 чебышевским узлам
7. Для каждой из функции, сгенерированных в предыдущем пункте, найдены интерполяционные полиномы 𝐿(𝑥), построенные по 𝑁 ∈ {1,2,...,30} равномерно расположенным узлам и чебышевским узлам. Для каждого 𝑁 рассчитано расстояние между и 𝑁 в лебеговом пространстве . Рассмотрено несколько графиков зависимости этого расстояния для равномерных и чебышевских узлов от N и сделаны по ним вывод.

**1. Интерполирование полиномами Лагранжа**

1. Задача 1 - Разработать функцию l\_i(i, x, x\_nodes)

Для начала заметим, что эта функция нам понадобится для двух видов сетки: для равномерного распределения и для чебышевских узлов, соответственно целесообразно будет передавать в аргументах функции еще один параметр – список, содержащий узлы интерполяции (назовем этот параметр values). В таком случае наша реализация будет более универсальной и позволит избежать дублирования кода.

Тогда разрабатываемая функция принимает вид: l\_i(i, x, x\_nodes, values).

i-й полином Лагранжа находится по формуле (1.1):

(1.1)

Программная реализация функции представлена на листинге 1.

def l\_i(x, n, j, values: list):  
 p1 = 1  
 p2 = 1  
 for i in range(0, n + 1):  
 if i != j:  
 p1 \*= x - values[i]  
 p2 \*= values[j] - values[i]  
 return p1 / p2

*Листинг 1 – реализация функции l\_i*

Чтобы вызвать представленную на листинге 1 функцию, нам необходимо инициализировать список значений узлов интерполяции. При этом не забываем что по условию требуется равномерное распределение точек, лежащее внутри отрезка [a; b]. Число точек будет целесообразно вынести в отдельный параметр (назовем его number\_of\_points) (см. Листинг 2).

x\_grid\_values = [a + (b - a) / number\_of\_points \* i for i in range(0, number\_of\_points + 1)]

*Листинг 2 – один из возможных вариантов заполнения списка точек равномерного распределения*

1. Задача 2 - Написать функцию L(x, x\_nodes, y\_nodes)

Аналогично задаче 1, добавим дополнительный параметр, содержащий список узлов интерполяции в аргументы разрабатываемой функции (также назовем его values).

Наша функция принимает вид: L(x, x\_nodes, y\_nodes, values)

Формула, по которой вычисляется интерполяционный полином Лагранжа:

(1.2)

Программная реализация функции представлена на листинге 3:

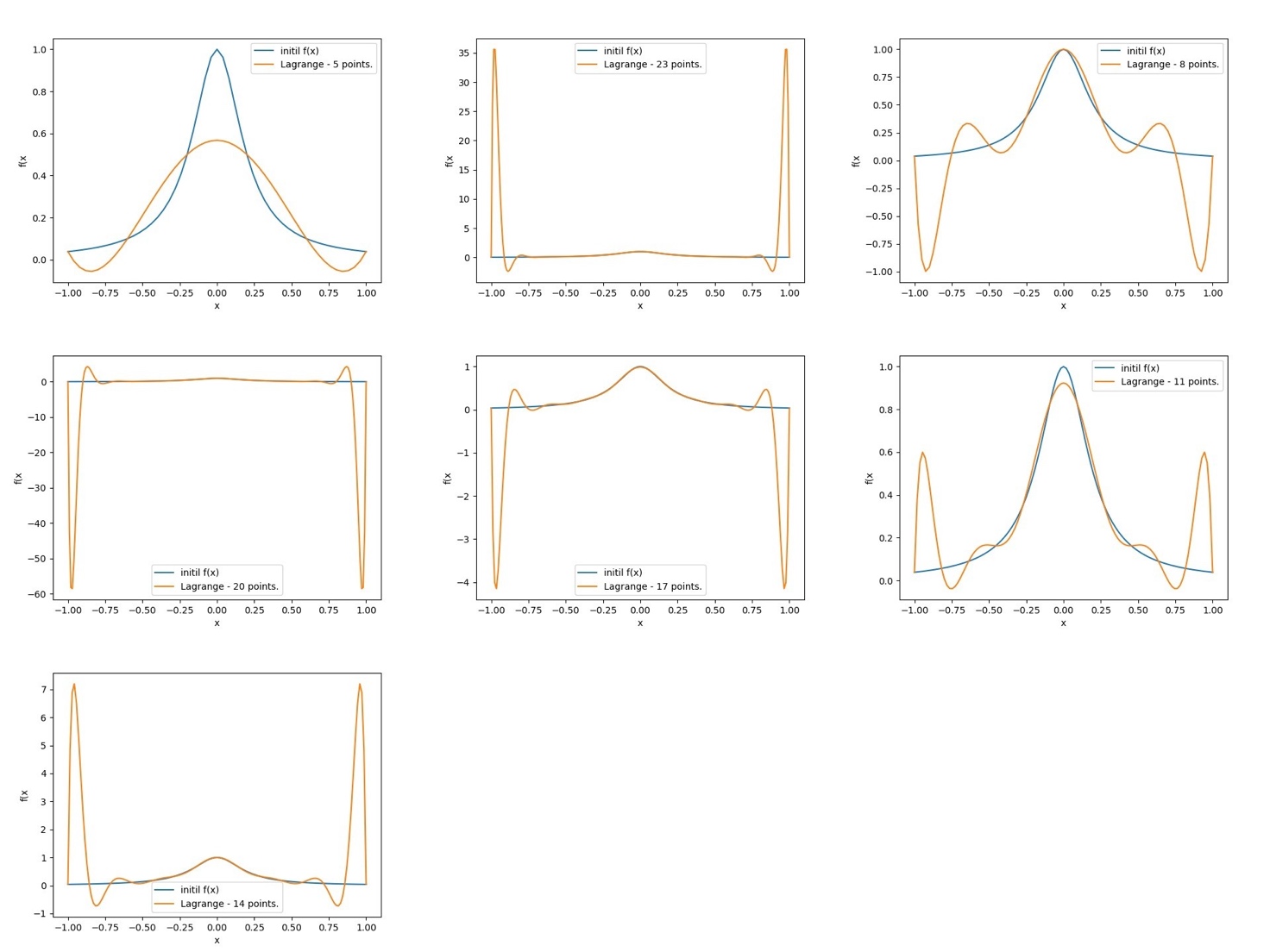
def L\_n(x, n, values: list):  
 s = 0  
 for i in range(0, n + 1):  
 s += f(values[i]) \* l\_i(x, n, i, values)  
 return s

*Листинг 3 – реализация функции L*

При вызове реализованной функции в качестве параметра values передадим список x\_grid\_values из задачи 1. Один из возможных способов его инициализации представлен на листинге 2.

3. Задача 3 - вывести на экран одновременно графики исходной функции и ее аппроксимации.

Построим 7 пар графиков функций, каждая из которых будет содержать график исходной функции и график аппроксимации Лагранжа (рис 1) для равномерно распределенных на отрезке узлов. Число точек для построения графиков возьмём в 10 раз больше, чем число узлов интерполяции (50, 80 и т.д.)

*Рисунок 1 – график функции и ее аппроксимации при равномерном распределении*

Как мы видим, при увеличении числа узлов увеличивается и точность интерполяции, однако все равно присутствует большая погрешность метода, связанная с выбором равномерно распределенных узлов. Из этого мы можем сделать вывод, что для данной функции равномерная сетка является не лучшим вариантом.

Эти скачки по бокам (так называемая осцилляция) – феномен Рунге. Возникает при [интерполяции](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F) [полиномами](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC) высоких степеней. Иначе говоря, при росте степени полинома погрешность интерполяции стремится к бесконечности.

Задача 4 - повторить предыдущий пункт для чебышевских узлов.

Рассмотрим чебышевскую сетку, 7 пар графиков представлены на рисунке 2. Сделать это возможно поскольку изначально в Задаче 1 и Задаче 2 реализация не опиралась на выбор того или иного типа узлов.

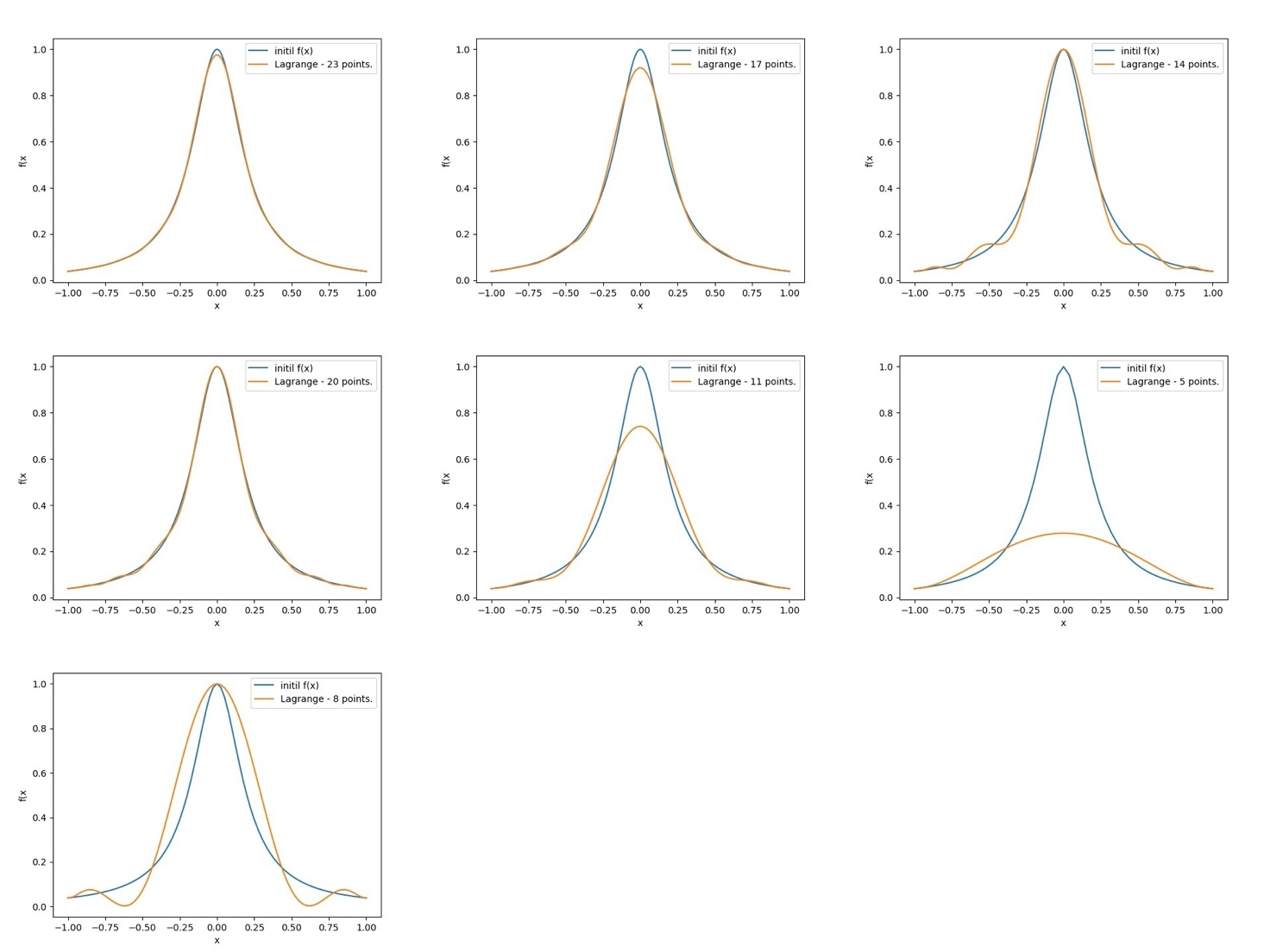
Значение чебышевских узлов вычисляются по формуле:

*(1.3)*

Для вычисления чебышевских узлов реализуем вспомогательную функцию (листинг 4)

def x\_grid\_ch(n):  
 ch = [-1]  
 for i in range(1, n):  
 ch.append(np.cos((2 \* i - 1) / (2 \* (n - 1)) \* np.pi))  
 ch.append(1)  
 ch.sort()  
 return ch

*Листинг 4 – функция для вычисления чебышевских узлов*



*Рисунок 2 - график функции и ее аппроксимации по чебышевским узлам*

Мы наблюдаем аналогичную картину, как и с интерполяций по равномерному распределению – при увеличении числа узлов возрастает точность. Однако в случае с чебышевскими узлами погрешность много меньше чем в случае с использованием равномерного распределения.

Задача 5 – сгенерировать 100 функций.

Для генерации рандомных вещественных чисел воспользуемся функцией из библиотеки random – random(). При вызове этой функции без аргументов будет сгенерировано одно псевдорандомное вещественное число внутри отрезка [0; 1].

Для генерации целых чисел опять же воспользуемся библиотекой random и методом randint(a, b), где a,b – границы отрезка, на котором генерируются псевдорандомные числа.

Программная реализация представена на листинге 5.

def pade(n, m, x):  
 sum1 = 0  
 for j in range(0, m):  
 sum1 += random.random() \* (x \*\* j)  
 sum2 = 0  
 for k in range(1, n):  
 sum2 += random.random() \* (x \*\* k)  
 return sum1 / (sum2 + 1)

for i in range(0, 100):  
 n = random.randint(7, 15)  
 m = random.randint(7, 15)  
 rnd.append(pade(n, m, x\_points[i]))

*Листинг 5 – генерация функций*

Задача 6 - для нескольких из сгенерированных функций вывести на экран одновременно графики

Для начала нам необходимо выбрать узлы N, по которым мы будем строить график. Возьмем 4 точки - [3, 8, 13, 18, 23] и построим по ним график(рисунок 3).

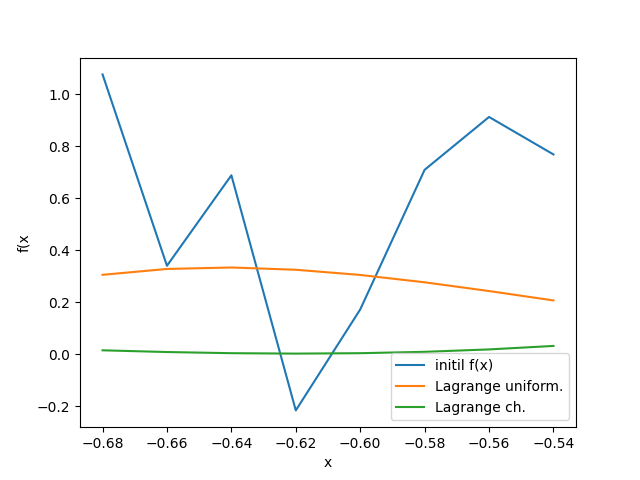


Рисунок 3

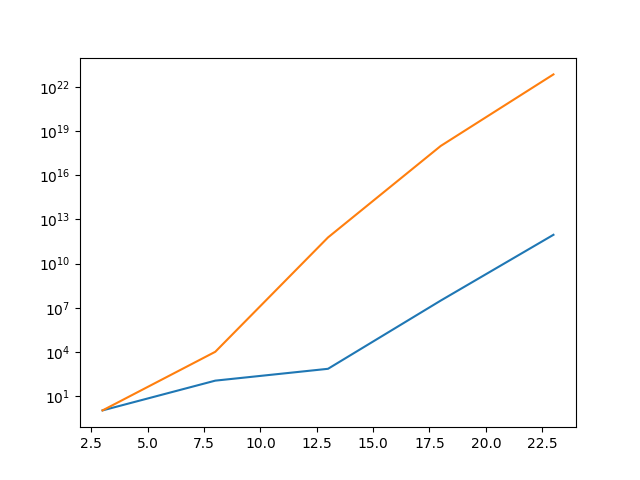
**7. Нахождение интерполяционных полиномов 𝐿(𝑥).**

В лебеговых пространствах под расстоянием подразумевается норма ||𝑔1 − 𝑔2||. В пространстве нормой является равномерная норма, которую можно определить как , предполагая, что функция 𝑔(𝑥) рассматривается на интервале 𝑥 ∈ [𝑎; 𝑏].

Для построения графиков необходимо инициализировать 2 списка точек – для равномерного распределения (diff\_points) и для чебышевских узлов (diff\_points\_ch). Программная реализация представлена на листинге 6.

polynom = []  
max\_diffs = []  
for N in range(1, 31):  
 max\_diff\_uniform = 0  
 max\_diff\_ch = 0  
 for i in range(0, 100):  
 uniform = L\_n(x\_points[i], N, x\_points)  
 ch = L\_n(x\_points[i], N, x\_points\_ch)  
 diff\_uniform = abs(x\_points[i] - uniform)  
 diff\_ch = abs(x\_points[i] - ch)  
 polynom.append([x\_points[i], uniform, ch, N, diff\_uniform, diff\_ch])  
 if diff\_uniform > max\_diff\_uniform:  
 max\_diff\_uniform = diff\_uniform  
 if diff\_ch > max\_diff\_ch:  
 max\_diff\_ch = diff\_ch  
 max\_diffs.append([N, max\_diff\_uniform, max\_diff\_ch])  
  
n\_points = [3, 8, 13, 18, 23]  
diff\_points = []  
diff\_points\_ch = []  
n\_index = []  
for point in n\_points:  
 n\_index.append(point - 1)  
 diff\_points.append(max\_diffs[point-1][1])  
 diff\_points\_ch.append(max\_diffs[point-1][2])

*Листинг 7 – реализация алгоритма заполнения списков точек*

На рисунке 4 изображен один из характерных графиков, демонстрирующий рост разности между исходной функцией и ее аппроксимаций в зависимости от числа узлов. Из этого можно сделать вывод, что при увеличении числа узлов мы получаем значительный рост погрешности приближения. 

*Рисунок 4*

# Заключение

В ходе выполнения лабораторной работы были получены навыки нахождения интерполяционного многочлена Лагранжа по сетке с равномерным распределением и чебышевским узлам, изучены и протестированы методы библиотеки языка программирования Python numpy.

# Список использованных источников

1. **Першин А.Ю.** *Лекции по курсу “Вычислительная математика”.* // Москва, 2018-2021. С. 140.
2. *Официальная документация numpy* [Электронный ресурс] // Режим доступа: <https://numpy.org/>, своюодный. Яз. англ.

*Шашко О.В. Отчет о выполнении лабораторной работы по курсу “Вычислительная математика”. [Электронный ресурс] – Москва: 2022 – 13 с. URL:* [*https://sa2systems.ru:88*](https://sa2systems.ru:88) *(система контроля версий кафедры РК6)*

*2022, осенний семестр*