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	Караф Сармат Майк
Группа	РК6-52Б
Тип задания	лабораторная работа
Тема лабораторной работы	Интерполяция в условиях измерений с неопределенностью

Студент _____ **Караф С.М.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Першин А.Ю**
подпись, дата *фамилия, и.о.*

Москва, 2021 г.

Оглавление

Задание на лабораторную работу

Задача 5 (интерполяция кубическими сплайнами)

Требуется (базовая часть):

1. Разработать функцию qubic_spline_coeff(x_nodes, y_nodes), которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна.
2. Написать функции qubic_spline(x, qs_coeff) и d_qubic_spline(x, qs_coeff), которые вычисляют соответственно значение кубического сплайна и его производной в точке x
3. Используя данные в таблице 1, требуется построить аппроксимацию зависимости уровня поверхности жидкости $h(x)$ от координаты x(см.рисунок 1) с помощью кубического сплайна и продемонстрировать ее на графике вместе с исходными узлами.

Таблица 1: Значения уровня поверхности вязкой жидкости (рис.1)

i	1	2	3	4	5	6	7	8	9	10	11
x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
h_i	3.37	3.95	3.73	3.59	3.15	3.15	3.05	3.86	3.60	3.70	3.02

Требуется (продвинутая 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. Известно, что при измерении координаты x_i всегда возникает погрешность, которая моделируется случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным

отклонением 10^{-2} .Требуется провести следующий анализ, позволяющий выявить влияние этой погрешности на интерполяцию:

a. Сгенерировать1000 векторов значений $[\tilde{x}_1, \dots, \tilde{x}_{11}]^T$, предполагая, что $\tilde{x}_i = x_i + Z$, где x_i соответствует значению в таблице 1 и Z является случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением 10^{-2}.
b. Для каждого из полученных векторов построить интерполянт Лагранжа, предполагая, что в качестве абсцисс узлов используются значения $\tilde{x}_i$, а ординат h_iиз таблицы 1. В результате вы должны иметь1000различных интерполянтов
c. Предполагая, что все интерполянты представляют собой равновероятные события, построить такие функции $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$, где $\tilde{h}_l(x) < \tilde{h}_u(x)$для любого $x \in [0; 1]$, что вероятность того, что значение интерполянта в точке x будет лежать в интервале $[\tilde{h}_l(x); \tilde{h}_u(x)]$ равна0.9.
d. Отобразить на едином графике функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт и узлы из таблицы 1.
e. Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям?

4. Повторить анализ, описанный в предыдущем пункте, в предположении, что координаты x_i вам известны точно, в то время как измерения уровня поверхности h_i имеют ту же погрешность, что и в предыдущем пункте. Изменились ли выводы вашего анализа?

5. Повторить два предыдущие пункта для случая интерполяции кубическим сплайном. Какие выводы вы можете сделать, сравнив результаты анализа для интерполяции Лагранжа и интерполяции кубическим сплайном?

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – Изучение метода интерполяции кубическим сплайном (базовая часть) и Лагранжа (продвинутая часть). Проанализировать работу этих методов при варьировании исходных параметров, разработав программное обеспечение на языке Python.

Выполненные задачи

1. Разработана функция qubic_spline_coeff(x_nodes, y_nodes), которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна.

2. Разработана функция 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. Используя данные из таблицы 1, была построена аппроксимирующая зависимость уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна и продемонстрирована на графике вместе с исходными узлами.

4. 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.

5. Разработана функция 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.

6. Формированы 1000 векторов значений $[\tilde{x}_1, \dots, \tilde{x}_{11}]^T$, предполагая, что $\tilde{x}_i = x_i + Z$, где x_i соответствует значению в таблице 1 и Z является случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением 10^{-2}.

7. Были построены для каждого из полученных векторов интерполянт Лагранжа.

8. Показали на едином графике функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт и узлы из таблицы 1.

9. Повторили анализы, описанный в 3 пункте, в предположении, что координаты x_i вам известны точно, в то время как измерения уровня поверхности h_i имеют ту же погрешность, что и в предыдущем 3 пункте.

10. Повторили 3 и 4 пункты для случая интерполяции кубическим сплайном.

1. Интерполяция кубическими сплайнами (Базовая часть)

1.1. Функция коэффициентов кубического сплайна

Рассмотрим формулу кубического сплайна:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad (1.1)$$

Нам нужно найти коэффициенты b, c, d, для этого разработаем функцию qubic_spline_coeff(x_nodes, y_nodes) , которая по средству решении матричного уравнения вычислит коэффициенты кубического сплайна.

$$\begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ h_1 & 2(h_2 + h_1) & h_2 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_3 + h_2) & 2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_{n-2} \end{bmatrix} =$$

$$\begin{bmatrix} 0 \\ \frac{3}{h_2}(a_3 - a_2) - \frac{3}{h_1}(a_2 - a_1) \\ \frac{3}{h_3}(a_4 - a_3) - \frac{3}{h_2}(a_3 - a_2) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix} \quad (1.2)$$

где $h_i = x_{i+1} - x_i$ и a_iравна значениям функции в узлах (Таблица 1)

Из матрицы (1.2) можно найти коэффициенты c (см. листинг 1.1)

Листинг 1.1 (Код решения матричного уравнения, нахождение коэффициентов c)

```
for i in range(0, len(x_I) - 2):  # подсчёт главной диагонали
    k_ser.append(2 * (h_i[i + 1] + h_i[i]))
k_ser.append(1)
k_ser = np.diag(k_ser, k=0)  # правая диагональ
k_max = np.diag(h_i, k=1)  # главная диагональ
k_min = np.diag(h_i, k=-1)  # левая диагональ
x_matr = k_ser + k_max + k_min  # объединение диагоналей в матрицу
for i in range(1, 10):
    x_matr[0][i] = 0
    x_matr[10][i] = 0

for i in range(0, len(x_I) - 2):  # матрица a= F(x)
    a_matr.append(((3 / h_i[i + 1]) * (h_I[i + 2] - h_I[i + 1])) - ((3 / h_i[i])
* (h_I[i + 1] - h_I[i])))
a_matr.append(0)

x_matr = np.linalg.inv(x_matr)  # инвертирование матрицы a
c_koef = np.dot(x_matr, a_matr # Перемножение матрицы, получение коэффициентов c
c_koef = c_koef.tolist()
```

В реализации кода используем c_koef = c_koef.tolist(), чтобы перезаписать c_koef из матрицы в список.

После нахождения коэффициентов c, высчитаем остальные коэффициенты по формулам (1.3), (1.4). (см. листинг 1.2)

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \tag{1.3}$$

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} - 2c_i), \tag{1.4}$$

Листинг 1.2 (Код расчёта коэффициентов b, d)

```
for i in range(0, len(x_I) - 1):     # расчёт коэффициентов b, d
    d_koef.append(((c_koef[i + 1] - c_koef[i]) / (3 * h_i[i])))
    b_koef.append(((h_I[i + 1] - h_I[i]) / h_i[i]) - ((h_i[i] * (c_koef[i + 1] +
2 * c_koef[i])) / 3))
```

Результатом функции qubic_spline_coeff(x_nodes, y_nodes) должен быть объединение всех коэффициентов в одну матрицу, в итоге функция возвращает $(N-1) \times 3$ матрицу, где N количество узлов интерполяции. (см. листинг 1.3)

Листинг 1.3 (Результат функции qubic_spline_coeff(x_nodes, y_nodes))

```
coef = np.r_["0,2", b_koef, c_koef, d_koef] # объединение всех коэффициентов в
одну матрицу
return coef # результат функции
```

1.2. Расчёт кубического сплайна и производная кубического сплайна

Для расчёта кубического сплайна воспользуемся формулой (1.1) и так же найдем производную кубического сплайна, получим:

$$S'(x) = S_i'(x) = b_i x + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \tag{1.5}$$

Разработаем функции вычисления кубического сплайна qubic_spline(x, qs_coeff) и производную кубического сплайна d_qubic_spline(x, qs_coeff). (см. листинг1.5 и 1.6) добавив проверку на индексацию, чтобы принятый x в функции находились в промежутке $[x_0, \dots, x_i]$, где значение $[x_0, \dots, x_i]$ значение с таблицы 1. (см. листинг1.4)

Листинг 1.4 (Проверка индексации для указанных x)

```
for i in range(0, len(x_I) - 1):     # x в промежутке от x_0 до x_i
    if x_I[i] <= x < x_I[i + 1]:
        ind = i
        break
if x >= x_I[len(x_I) - 1]:  # x больше x_i
    ind = len(x_I) - 2
if x <= x_I[0]:      # x меньше x_0
    ind = 0
```

Листинг 1.5 (Код функции qubic_spline(x, qs_coeff))

```
def qubic_spline(x, qs_coeff, x_I, h_I):
    for i in range(0, len(x_I) - 1):     # x в промежутке от x_0 до x_i
        if x_I[i] <= x < x_I[i + 1]:
            ind = i
            break
    if x >= x_I[len(x_I) - 1]:  # x больше x_i
        ind = len(x_I) - 2
    if x <= x_I[0]:      # x меньше x_0
        ind = 0
        # расчёт кубического сплайна
    return h_I[ind] + gs_coeff[0][ind] * (x - x_I[ind]) + (gs_coeff[1][ind] * (x
- x_I[ind]) ** 2) + (gs_coeff[2][ind] * (x - x_I[ind]) ** 3)
```

Листинг 1.6 (Код функции qubic_spline(x, qs_coeff))

```
def d_qubic_spline(x, gs_coeff, x_I):
    for i in range(0, len(x_I) - 1):      # x в промежутке от x_0 до x_i
        if x_I[i] <= x < x_I[i + 1]:
            index = i
            break
    if x >= x_I[len(x_I) - 1]:        # x больше x_i
        index = len(x_I) - 2
    if x <= x_I[0]:      # x меньше x_0
        index = 0
        # расчёт производной кубического сплайна
    return gs_coeff[0][index] + 2 * gs_coeff[1][index] * (x - x_I[i]) + 3 *
gs_coeff[2][index] * (x - x_I[index]) ** 2
```

Результаты функций qubic_spline(x, qs_coeff) и d_qubic_spline(x, qs_coeff) являются значение кубического сплайна в переданной точке x

1.3. Построение аппроксимирующую зависимость уровня поверхности

Для четкого построение аппроксимирующей функции нужно взять больше точек. Поэтому воспользуемся операцией o = np.arange(x_I[0], x_I[-1] + x_I[-1] / 100, x_I[-1] / 100), что создает нам массив из 101 элементов, которые находятся в промежутке [0;1] и с шагом 0.01, затем мы этот массив передаем в функцию qubic_spline(x, qs_coeff). (см. листинг1.7)

При помощи полученных результатов кубического сплайна в точках можно построить аппроксимирующую функцию используя библиотеку matplotlib. (см. листинг1.8)

Листинг 1.7 (Расчёты кубического сплайна)

```
o = np.arange(x_I[0], x_I[-1] + x_I[-1] / 100, x_I[-1] / 100) # массив от 0 до 1
for i in range(101):          # получение расчётов кубического сплайна
    s.append(qubic_spline(o[i], gs_coeff, x_I, h_I))
```

Листинг 1.8 (Отображение аппроксимирующей функции с узлами из таблицы)

```
plt.scatter(x_I, h_I)    # построение узлов со знач. из таблицы
plt.plot(o, s)       # построение аппроксимирующей функции
plt.grid()
plt.show()  # отображение аппроксимирующей функции с узлами из таблицы
```

На рисунке 1 изображён результат работы метода кубического сплайна, проходящего через узлы из таблицы 1.

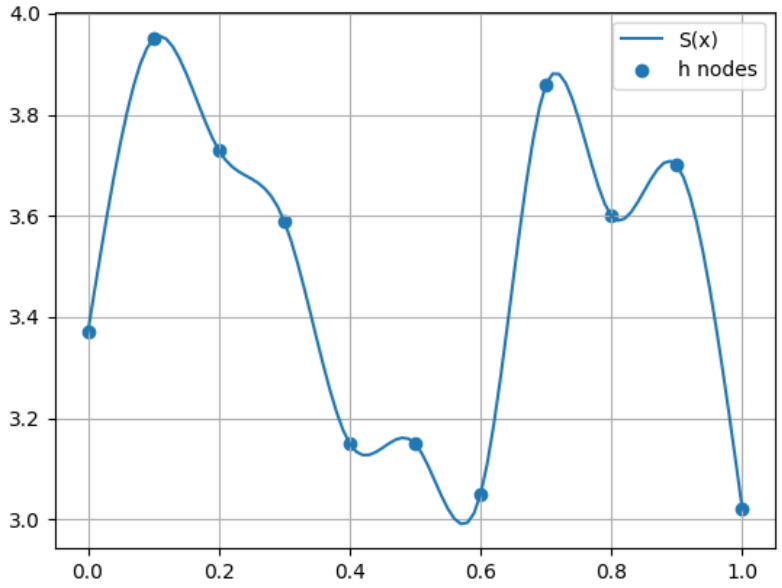

Рис. 1: кубический сплайн, проходящий через узлы из таблицы 1.

2. Влияние погрешностей на интерполяцию (Продвинутая часть)

2.1. Функция возвращение значение i –го базисного полинома Лагранжа

Для разработки функции l_i(i, x, x_nodes), которая возвращает значение i-го базисного полинома Лагранжа, на заданных узлах с абсциссами x_nodes, в точке x., воспользуемся формулой (2.1). (см. листинг2.1)

$$l_i(x) = \prod_{j=0, i \neq j}^{n} \frac{x - x_j}{x_i - x_j} \tag{2.1}$$

Листинг 2.1 (Код функции l_i(i, x, x_nodes))

```
def l_i(i, x, x_I):
    p = 1
    for j in range(0, len(x_I)):
        if i != j:
            p *= (x - x_I[j]) / (x_I[i] - x_I[j]) #расчёт i-го базисного полинома Лагранжа
    return p
```

Результатом функции является вычисленный i-ый базис полинома Лагранжа.

2.2.Функция интерполяционного полинома Лагранжа

Для разработки функции L(x, x_nodes, y_nodes), которая возвращает значение интерполяционного полинома Лагранжа n-1 степени, заданного на узлах с абсциссами x_nodes и ординатами y_nodes, в точке, воспользуемся формулой (2.2). (см. листинг2.2)

$$L(x) = \sum_{i=0}^{n} y_i l_i(x), \tag{2.2}$$

Листинг 2.2 (Код функции L(x, x_nodes, y_nodes))

```
def L(x, x_I, h_I):
    L = 0
    for i in range(0, len(x_I)):
        L += h_I[i] * l_i(i, x, x_I)     #расчёт значение интерполяционного полинома Лагранжа n-1 степени
    return L
```

Результатом функции является вычисленный значение интерполяционного полинома Лагранжа n-1 степени

2.3.Анализ выявление влияния погрешности на интерполяцию

a. Генерация 1000 векторов со случайными величинами

Исходя из условия было сформирована 1000 векторов значений $[\tilde{x}_1, \dots, \tilde{x}_{11}]^T$ так что $\tilde{x}_i = x_i + Z$, где x_i значению из таблице 1 и Z случайная величина с нормальным распределением и нулевым мат ожиданием и отклонением 10^{-2}. Для создание погрешности воспользовались операцией z=[random.normalvariate(mu=0, sigma=0.01)], задавая характеристики указанных в условии, что величина с нормальным распределением и нулевым мат ожиданием и отклонением 10^{-2} .Затем заполняем нашу матрицу векторов $\tilde{x}_i = x_i + Z$ (см.листинг2.3). Для удобства пользование округлим наше случайное число.

Листинг 2.3 (Код формирование 1000 векторов со случённым величиной)

```
for i in range(1000):  # формирование 1000 векторов
    x_pog = []
#создание случайной величины
    z = [random.normalvariate(mu=0, sigma=0.01) for j in range(len(x_I))]
    for k in range(len(x_I)):
        z[k] = round(z[k], 5)
        x_pog.append(x_I[k] + z[k]) # формирование 1000 векторов со случённым
величиной
    vec.insert(i, x_pog)
```

В результате будет матрица 1000×11 с исходными $x_i + Z$

b. Интерполяция Лагранжа со случайной величиной

В предыдущем пункте (a) формировали 1000 векторов со случайными величинами, передадим эту матрицу векторов в функцию интерполяционного полинома Лагранжа L(x, x_nodes, y_nodes) , с дополнительными иксами суммированных с погрешностью, получим матрицу со значениями интерполяционного полинома Лагранжа n-1 степени для каждого x_i (см. листинг2.4)

Листинг 2.4 (Код интерполяции Лагранжа со случайной величиной)

```
for i in range(1000):
    temp = []
    for j in range(101):
        temp.append(L(1 / 101 * j, vec[i], h_I)) # передача x в функция
интерполяции Лагранжа
    L_I.insert(i, temp) # Запись значений интерполяции в матрицу
```

Выведем полиномы Лагранжа для 1000 векторов (Рис 2), что абсцисс узлов используются значения x_i, а ординат h_i из таблицы 1

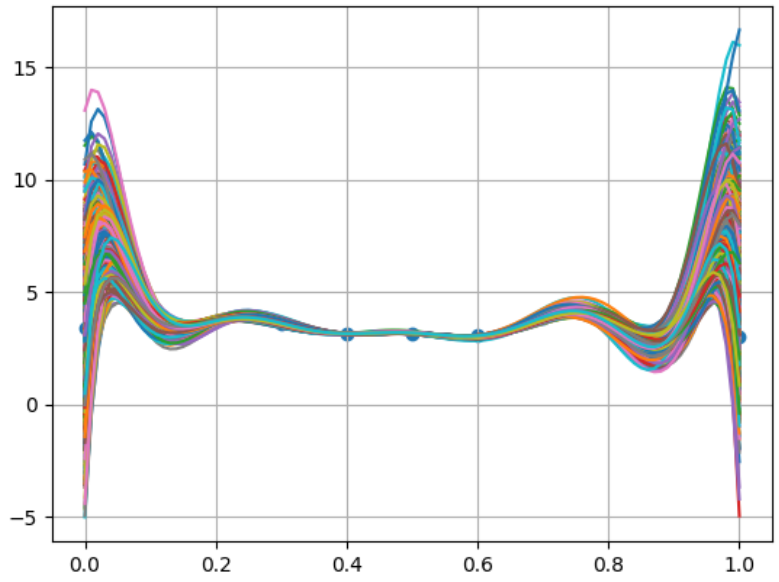

Рис. 2: 1000 интерполлянтов Лагранжа со случайной величиной

c. Построение добротной полосы

Исходя из условий задания понимаем что функции $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ границы доверительного интервала, чтобы значение интерполлянта в точке x лежать на интервале $[\tilde{h}_l(x);\tilde{h}_u(x)]$, с вероятность 0.9,надо откинуть первые и последние 50 интерполлянтов. Для этого надо отсортировать массив 1000 интерполлянтов так, чтобы для любого $x \in [0;1]$ выполнялось условие $\tilde{h}_l(x) < \tilde{h}_u(x)$. После сортировки $\tilde{h}_l(x)$ соответствует 49 элементу массива, а $\tilde{h}_u(x)$ соответствует 949 элементу массива (см. листинг2.5), затем выведем интерполлянта на (Рис 3)

Листинг 2.5 (Сортировка и построение $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$)

```
L_I = np.sort(L_I, 0)  #сортировка массива
plt.figure()
for i in range(1000):
    if i == 49 or i == 949: # построение графиков h̃_l(x) и h̃_u(x)
        plt.plot(o, L_I[i])
plt.grid()
plt.show()
```

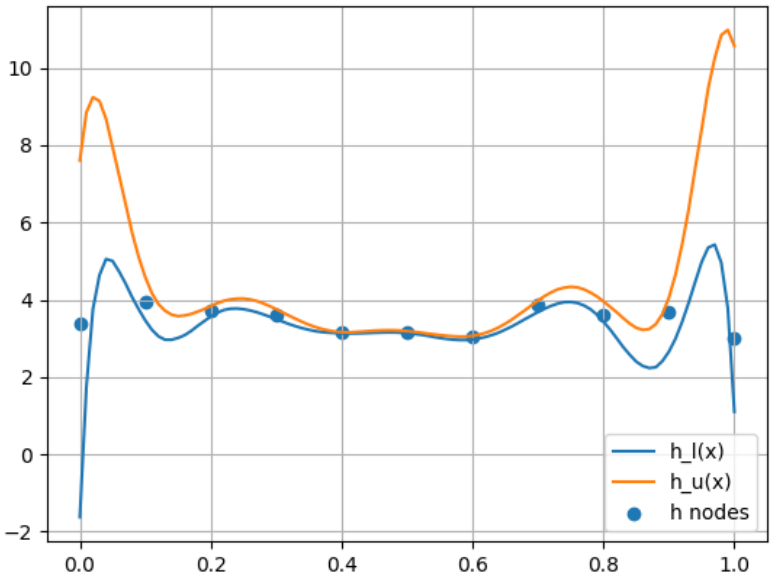

Рис. 3: Интерполянты $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$.

d. Усреднённый интерполлянт

Из предыдущего пункта мы отсортировали наш массив интерполлянтов Лагранжа по условию $\tilde{h}_l(x) < \tilde{h}_u(x)$, т.к. у нас отсортированный массив то усреднённый интерполлянт соответствует 499 элементу массива (см.листинг2.6). Выведем интерполлянта на (Рис 4)

Листинг 2.5 (построение $\tilde{h}_l(x)$и $\tilde{h}_{ser}(x)$ и $\tilde{h}_u(x)$)

```
plt.figure()
for i in range(1000):
    if i == 49 or i == 499 or i == 949: #построение графиков h̃_l(x) и h̃_u(x) и h̃_ser(x)
        plt.plot(o, L_I[i])
plt.grid()
plt.show()
```

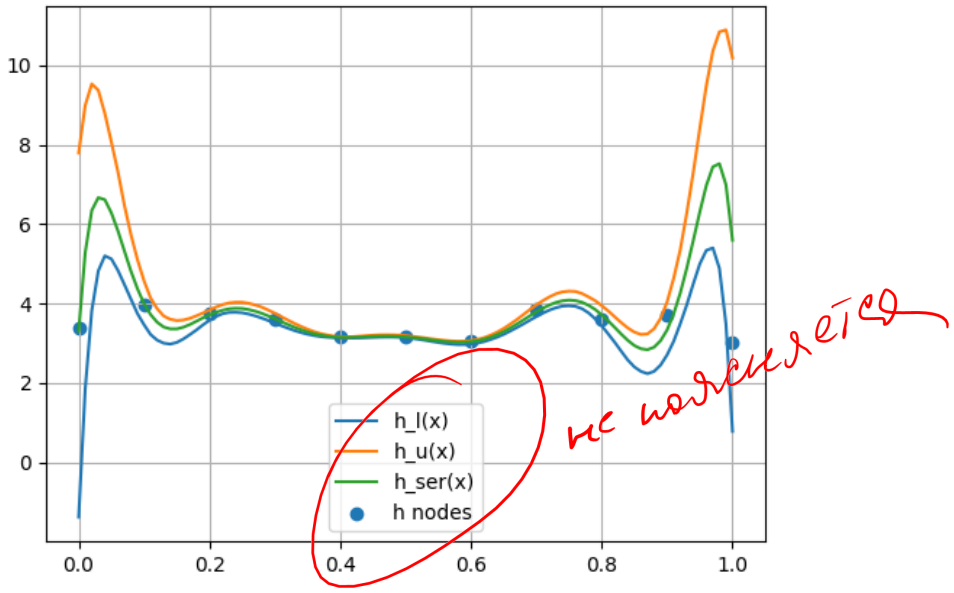

Рис. 4: Интерполянты $\tilde{h}_l(x)$и $\tilde{h}_{ser}(x)$ и $\tilde{h}_u(x)$

e. Чувствительные участки к погрешностям.

Проанализировав (Рис 2) можно понять что более чувствительным к погрешностям являются граничные отрезки, т.к. в начале и в конце интерполлянтов Лагранжа базовой полиномы накапливаются паразитные осцилляции. С увеличением точек этот эффект будет увеличиваться из-за накопление паразитных осцилляций. Данную проблему можно решить с помощью интерполяционного узла Чебышева

2.4.Погрешность значений ординат h_i

Как и в предыдущем пункте формируем матрицу из 1000 векторов поменяв значение $\left[\tilde{h}_1, \dots, \tilde{h}_{11}\right]^T$, так что $\tilde{h}_i = h_i + Z$, где h_i значению из таблице 1 и Z случайная величина с нормальным распределением и нулевым мат ожиданием и отклонением 10^{-2}(см.листинг2.7).

Листинг 2.3 (Код формирование 1000 векторов со случённым величиной для h_i)

```
for i in range(1000):
    x_pog = []
#создание случайной величины
    z = [random.normalvariate(mu=0, sigma=0.01) for j in range(len(x_I))]
    for k in range(len(x_I)):
        z[k] = round(z[k], 5)
        x_pog.append(h_I[k] + z[k])
    vec.insert(i, x_pog) # формирование 1000 векторов со случённым величиной
```

Сформированную матрицу 1000×11 передадим в функцию итерполяции Лагранжа, добавив дополнительные точки для более грамотного вывода интерполянтов (см. листинг2.8).

Листинг 2.8 (Код интерполяции Лагранжа со случайной величиной для h_i)

```
for i in range(1000):
    temp = []
    for j in range(101):
        temp.append(L(1 / 101 * j, vec[i], x_I)) # передача hi в функция интерполяции Лагранжа
    L_I.insert(i, temp) # Запись значений интерполяции в матрицу
```

Построим графики интерполянта Лагранжа (Рис 5)

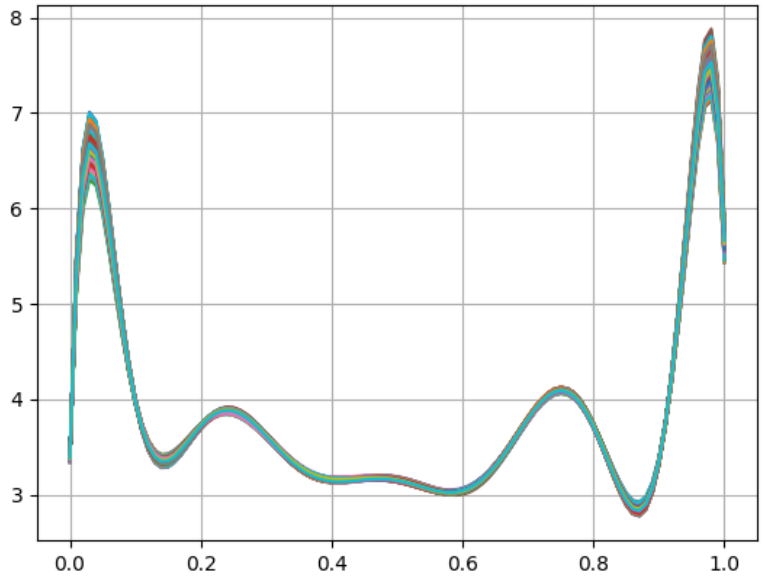

Рис. 5: интерполянты Лагранжа со случайной величиной для случайной величины h_i

Отсортируем матрицу интерполянтов Лагранжа чтобы наша точка h_i) попадала в промежуток $[\tilde{h}_l(x)\,;\,\tilde{h}_u(x)\,]$, с вероятностью 0.9 , после $\tilde{h}_l$ соответствует 49 элементу массива, а $\tilde{h}_u$ соответствует 949 элементу массива(см. листинг2.9), затем выведем интерполлянты на (Рис 6)

Листинг 2.9 (Сортировка и построение $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$)

```
plt.figure()
for i in range(1000):
    if i == 49 or i == 949:
        plt.scatter(x_I, h_I)
        plt.plot(o, L_I[i])    # построение графиков h̃l(x) и h̃u(x)
```

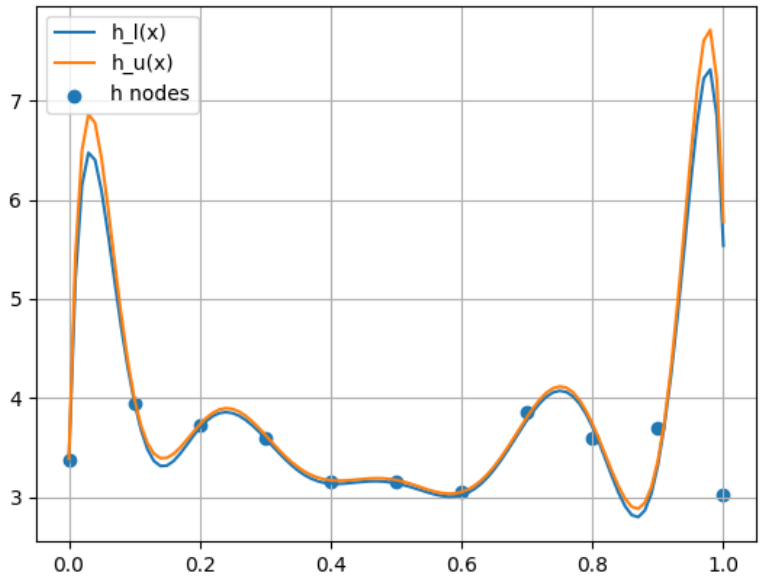

Рис. 6: Интерполянты $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ для случайной величины h_i.

Из предыдущего пункта мы отсортировали наш массив интерполлянтов Лагранжа по условию $\tilde{h}_l(x) < \tilde{h}_u(x)$, т.к. у нас отсортированный массив то усреднённый итнтерполлянт соответствует 499 элементу массива.Выведем интерполлянта на (Рис 7)

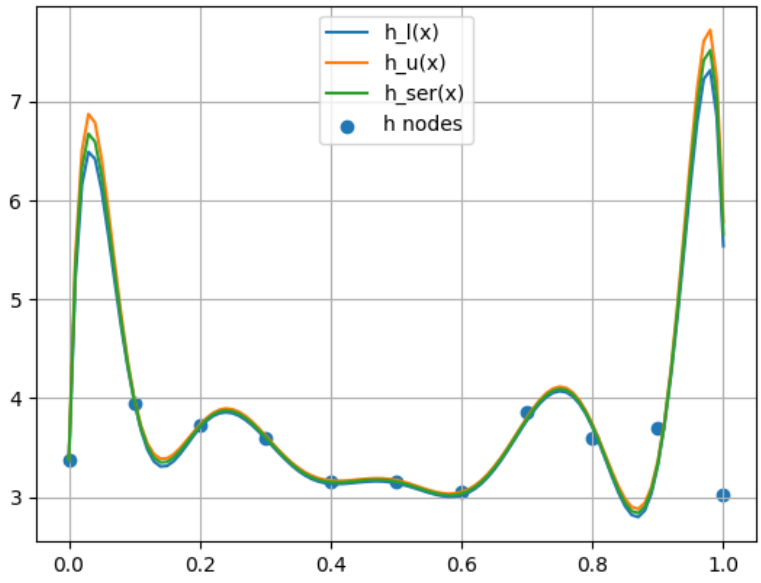

Рис. 7: Интерполянты $\tilde{h}_l(x)$и $\tilde{h}_{ser}(x)$ и $\tilde{h}_u(x)$ для случайной величины h_i

Проанализировав (Рис 5) можно что как и на (Рис 2) более чувствительным к погрешностям являются граничные отрезки, т.к. в начале и в конце интерполлянтов Лагранжа базовой полиномы накапливаются паразитные осцилляции.

2.5.Влияние погрешности на интерполяцию кубическим сплайном

По аналогии с 3 пунктом построим графики интерполяции кубическим сплайном, предполагая что абсцисса узлов используются $\tilde{x}_i$ где $\tilde{x}_i = x_i + Z$, Z- Случайная величина, а $\tilde{h}_i$взят из таблицы 1(Рис 8)

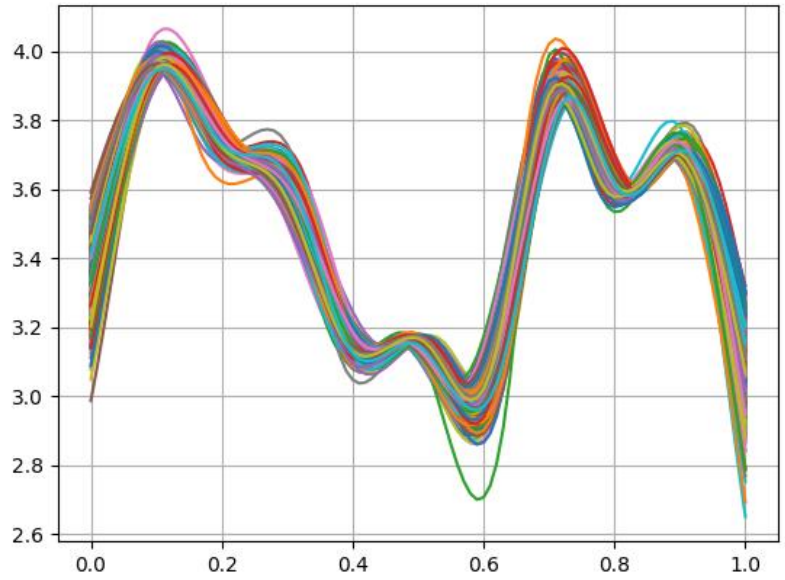

Рис. 8: график кубических сплайнов со случайной величиной x_i

Снова построим график интерполяции но теперь предлагая что абсцисса $\tilde{x}_i$ будет взята из таблицы 1, а ордината $\tilde{h}_i$, где $\tilde{h}_i = h_i + Z$, Z- Случайная величина(Рис 9)

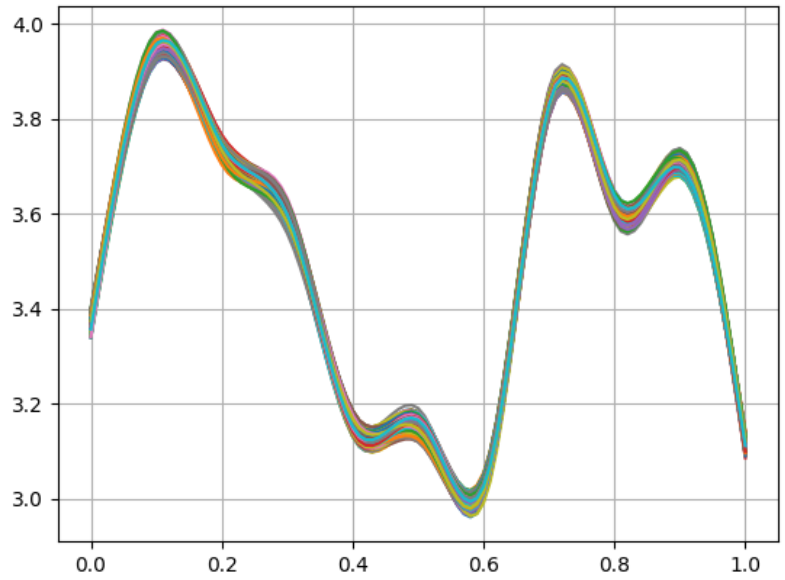

Рис. 9: график кубических сплайнов со случайной величинойh_i

Построим графики интерполяции с функциями $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$, с усредненным кубическим сплайном , предполагая что абсцисса узлов используются $\tilde{x}_i$ где $\tilde{x}_i = x_i + Z$, Z- Случайная величина, а $\tilde{h}_i$взят из таблицы 1(Рис 10)

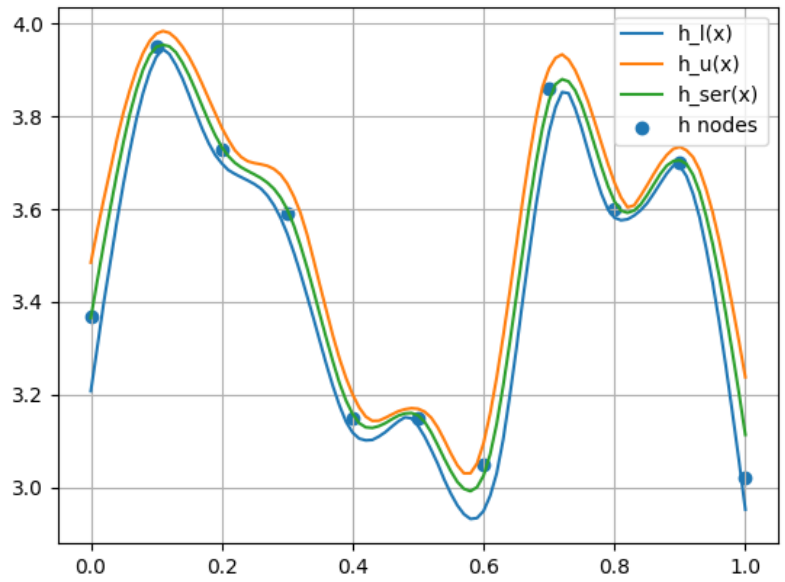

Рис. 10: Интерполянты $\tilde{h}_l(x)$и $\tilde{h}_{ser}(x)$ и $\tilde{h}_u(x)$ для случайной величиныx_i.

Построим графики интерполяции с функциями $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$, с усредненным кубическим сплайном , предлагая что абсцисса $\tilde{x}_i$ будет взята из таблицы 1, а ордината $\tilde{h}_i$, где $\tilde{h}_i = h_i + Z$, Z- Случайная величина(Рис119)

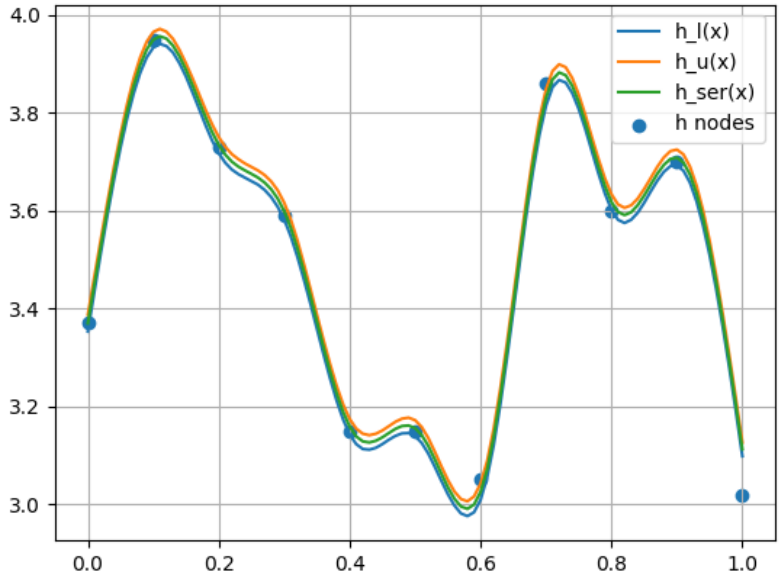

Рис. 11: Интерполянты $\tilde{h}_l(x)$и $\tilde{h}_{ser}(x)$ и $\tilde{h}_u(x)$ для случайной величины$\tilde{h}_i$.

Проанализировав (Рис 8) и (Рис 9) заметим, что паразитные осцилляции не накапливается в кубическом сплайне, от сюда следует что для большого количества точек подходит интерполяция методом кубического сплайна.

Заключение

Заключение:

В данной работе мы рассмотрели два метода интерполяции кубическим сплайном и полином Лагранжа, рассмотрели их плюсы и минусы в определённых областях.

При работе с не точными измерениями интерполяция кубическим сплайнам лучше справляется чем интерполяция полинома Лагранжа, т.к можно получить разницу в результате, при незначительных изменений в измерениях

Анализируя интерполянты полинома Лагранжа и интерполяция кубическим сплайнам, заметим, что интерполянты полинома Лагранжа хоть и работает быстрее, но если нужна будет численная точность, то лучше использовать интерполяция кубическим сплайнам, а если нужно отсортировывать в реальном времени, то лучше использовать интерполянты полинома Лагранжа

Исходя из выполненной работы модно понять, что для каждого метода интерполяции есть своя наиболее подходящая область

Список использованных источников

1. **Першин А.Ю.** Лекции по курсу «Вычислительная математика». Москва, 2018- 2021. С. 140. [1].
2. **Соколов А. П., Першин А. Ю.** Инструкция по выполнению лабораторных работ. Москва-2021
3. **Документация Python [Электронный ресурс]** /. — Электрон. текстовые дан. — 2020. — Режим доступа: www.python.org/doc/ (Дата обращения: 25.05.2020);

[1]Оформляется согласно ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления», и ГОСТ 7.82-2001 «Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления»