



Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего профессионального 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:	Очиров Бадма Юрьевич
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Интерполяция в условиях измерений с неопределенностью

Студент

\_\_\_\_\_

подпись, дата

Очиров Б. Ю.

Фамилия, И.О.

Преподаватель

\_\_\_\_\_

подпись, дата

Соколов А. П.

Фамилия, И.О.

Москва, 2021

# Содержание

<b>Интерполяция в условиях измерений с неопределенностью</b>	<b>3</b>
1    Задание . . . . .	3
2    Цель выполнения лабораторной работы . . . . .	5
3    Базовая часть . . . . .	5
3.1. Разработка функции вычисления коэффициентов естественного кубического сплайна . . . . .	5
3.2. Разработка функций вычисления значения кубического сплайна и его производной в точке $x$ . . . . .	7
3.3. Построение аппроксимации зависимости уровня поверхности жидкости $h(x)$ от координаты $x$ . . . . .	8
4    Продвинутая часть . . . . .	10
4.1. Вычисление $i$ -ого базисного полинома Лагранжа . . . . .	10
4.2. Вычисление значения интерполяционного полинома Лагранжа в точке $x$ . . . . .	11
4.3. Проведение анализа, позволяющего выявить влияние погрешности величины $x$ на интерполяцию . . . . .	11
4.4. Повторение анализа из пункта 3 в предположении, что $x_i$ заданы точно, а $h_i$ заменены на $\tilde{h}_i$ , которые зависят от случайной величины $Z$ . . . . .	17
4.5. Повторение анализов из пунктов 3 и 4, используя метод интерполяции кубическим сплайнам. Сравнение результатов анализа для интерполяции Лагранжа и интерполяции кубическим сплайнам . . . . .	20
5    Заключение . . . . .	25

# Интерполяция в условиях измерений с неопределенностью

## 1 Задание

Базовая часть:

1. Разработать функцию  $qubic\_spline\_coeff(x\_nodes, y\_nodes)$ , которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна. Для простоты, решение матричного уравнения можно производить с помощью вычисления обратной матрицы с использованием функции `numpy.linalg.inv()`.
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  $x$  ( $qs\_coeff$  обозначает матрицу коэффициентов).
3. Используя данные в таблице 1, требуется построить аппроксимацию зависимости уровня поверхности вязкой 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)

$x_i$	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

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) Сгенерировать 1000 векторов значений  $[\tilde{x}_0, \dots, \tilde{x}_{10}]^T$ , 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  $10^{-2}$ .

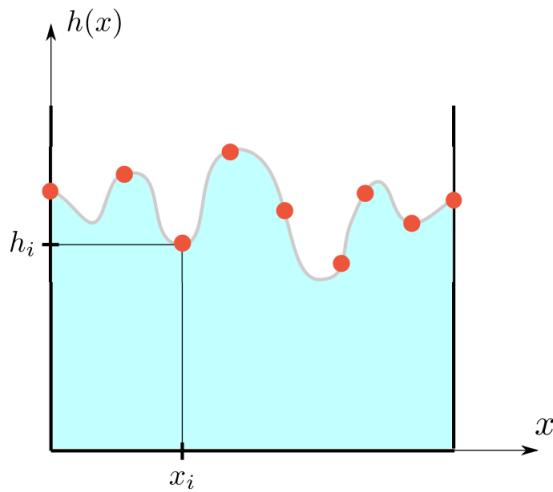


Рис. 1. Поверхность вязкой жидкости (серая кривая), движущейся сквозь некоторую среду (например, пористую). Её значения известны только в нескольких точках (красные узлы).

- (b) Для каждого из полученных векторов построить интерполянт Лагранжа, предполагая, что в качестве абсцисс узлов используются значения  $\tilde{x}_i$ , а ординат –  $h_i$  из таблицы 1. В результате вы должны иметь 1000 различных 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  $[\tilde{h}_l(x); \tilde{h}_u(x)]$  равна 0.9.
  - (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 сплайнами. Какие выводы вы можете сделать, сравнив результаты анализа для интерполяции Лагранжа и интерполяции кубическим сплайном?
  6. Опциональное задание. Изложенный выше анализ позволяет строить доверительные интервалы исключительно для интерполянтов, не оценивая доверительные интервалы с точки зрения предсказаний значений между узлами. Ин-

тересным методом интерполяции, позволяющим получить именно такие вероятностные оценки, является регрессия на основе гауссовых процессов, известная также как кригинг. В этом опциональном задании предлагается провести интерполяцию по данным из таблицы 1, используя кригинг.

## 2 Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – ознакомиться с методами интерполяции кубическим сплайном и полиномом Лагранжа, научиться реализовывать такие методы на языке программирования Python, а также проанализировать, как на результат влияют неопределенности в заданных начальных условиях.

## 3 Базовая часть

1. Разработка функции вычисления коэффициентов естественного кубического сплайна.
2. Разработка функций вычисления значения кубического сплайна и его производной в точке  $x$ .
3. Построение аппроксимации зависимости уровня поверхности жидкости  $h(x)$  от координаты  $x$ .

### 3.1. Разработка функции вычисления коэффициентов естественного кубического сплайна

Допустим, имеются значения в  $n$  узлах некоторой функции  $f(x)$ :  $x_0 = a, x_1, \dots, x_{n-1} = b$  на отрезке  $[a; b]$ , тогда кубическим сплайном для  $f(x)$  называется функция  $S(x)$ , кусочно заданная кубическими многочленами  $S_i(x)$  на каждом отрезке  $[x_i; x_{i+1}]$ , где  $i = 0, \dots, n - 2$ :

$$S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3; \quad (1)$$

где  $a_i, b_i, c_i, d_i$  называются коэффициентами кубического сплайна. Они вычисляются следующим образом:

$$a_i = f(x_i), \quad (2)$$

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

$$d_i = \frac{c_{i+1} - c_i}{3 \cdot h_i}; \quad (4)$$

где  $h_i = x_{i+1} - x_i$ , при  $i = 0, \dots, n - 2$

По теореме об единственности естественного кубического сплайна функция  $f(x)$  имеет такой уникальный кубический сплайн, который будет удовлетворять граничным условиям  $S''(a) = 0$  и  $S''(b) = 0$ .

Далее определена производная второго порядка для формулы (1):

$$S'_i(x) = b_i + 2 \cdot c_i \cdot (x - x_i) + 3 \cdot d_i \cdot (x - x_i)^2,$$

$$S''_i(x) = 2 \cdot c_i + 6 \cdot d_i \cdot (x - x_i);$$

Применяя граничные условия, произведены следующие вычисления:

$$\begin{aligned} S''_0(a) &= 2 \cdot c_0 + 6 \cdot d_0 \cdot (a - x_0) = 0 \\ 2 \cdot c_0 &= 0 \\ c_0 &= 0 \end{aligned}$$

$$\begin{aligned} S''_{n-1}(b) &= 2 \cdot c_{n-1} + 6 \cdot d_{n-1} \cdot (b - x_{n-1}) = 0 \\ 2 \cdot c_{n-1} &= 0 \\ c_{n-1} &= 0 \end{aligned}$$

Таким образом, определено, что  $c_0 = 0, c_{n-1} = 0$ . Остальные  $c_i$  можно найти, составив и решив следующее матричное уравнение:

$$\mathbf{A} \cdot \mathbf{c} = \mathbf{F}$$

$$\begin{aligned} &\begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ h_0 & 2 \cdot (h_1 + h_0) & h_1 & 0 & \dots & 0 \\ 0 & h_1 & 2 \cdot (h_2 + h_1) & h_2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_{n-3} & 2 \cdot (h_{n-2} + h_{n-3}) & h_{n-2} \\ 0 & \dots & \dots & \dots & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \\ &= \begin{pmatrix} 0 \\ \frac{3}{h_1} \cdot (a_2 - a_1) - \frac{3}{h_0} \cdot (a_1 - a_0) \\ \frac{3}{h_2} \cdot (a_3 - a_2) - \frac{3}{h_1} \cdot (a_2 - a_1) \\ \vdots \\ \frac{3}{h_{n-2}} \cdot (a_{n-1} - a_{n-2}) - \frac{3}{h_{n-3}} \cdot (a_{n-2} - a_{n-3}) \\ 0 \end{pmatrix} \end{aligned}$$

Решением этого уравнения является:

$$\mathbf{c} = \mathbf{A}^{-1} \cdot \mathbf{F}$$

Для решения матричного уравнения и нахождения всех коэффициентов разработана функция `qubic_spline_coeff(x_nodes, y_nodes)` на языке Python.

Алгоритм решения данного матричного уравнения таков:

1. Составить три списка с главной и смежными с ней диагоналями, с помощью функции `numpy.diag()` из этих списков составить три матрицы, которые складываются в искомую матрицу  $\mathbf{A}$ .
2. С помощью функции `numpy.linalg.inv(A)` найти обратную ей матрицу  $\mathbf{A}^{-1}$ .
3. Составить матрицу  $\mathbf{F}$ .
4. Получить матрицу  $\mathbf{c}$  с перемножением  $\mathbf{A}^{-1}$  и  $\mathbf{F}$ .

После решения уравнения были получены  $[c_0, \dots, c_{n-1}]^T$ , которые необходимо подставить в формулы коэффициентов (2), (3), (4).

Таким образом, в результате работы функции возвращается матрица  $(n - 1) \times 4$  коэффициентов кубического сплайна:

$$\begin{pmatrix} a_0 & b_0 & c_0 & d_0 \\ a_1 & b_1 & c_1 & d_1 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n-2} & b_{n-2} & c_{n-2} & d_{n-2} \end{pmatrix}$$

Разработанная функция `qubic_spline_coeff(x_nodes, y_nodes)` находится в файле `basic.ipynb`.

### 3.2. Разработка функций вычисления значения кубического сплайна и его производной в точке $x$

Для вычисления значения естественного кубического сплайна  $S_i(x)$  в точке  $x$  проверяется принадлежность  $x$  некоторому отрезку  $[x_i; x_{i+1}]$ , при  $i = 0, \dots, n - 2$ , и подставляется в формулу для этого отрезка:

$$S_i(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3;$$

где  $a_i, b_i, c_i, d_i$  соответствуют коэффициентам кубического сплайна из переменной `qs_coeff` для  $i$ -ого отрезка.

Функция `qubic_spline(x_nodes, x, qs_coeff)`, реализовывающая эти вычисления, представлена в файле `basic.ipynb` и в Листинге 1.

---

Листинг 1. Реализация функции  $qubic\_spline(x\_nodes, x, qs\_coeff)$

---

```
1 def qubic_spline(x_nodes, x, qs_coeff):
2     n = len(x_nodes)
3
4     for i in range(0, n - 1):
5         if x >= x_nodes[i] and x <= x_nodes[i + 1]:
6             a = qs_coeff[i][0]
7             b = qs_coeff[i][1]
8             c = qs_coeff[i][2]
9             d = qs_coeff[i][3]
10            difX = x - x_nodes[i]
11            s_x = a + b * difX + c * difX * difX + d * difX * difX * difX
12
13    return s_x
```

---

То же самое сделано для вычисления производной естественного кубического сплайна:

$$S'_i(x) = b_i + 2 \cdot c_i \cdot (x - x_i) + 3 \cdot d_i \cdot (x - x_i)^2$$

Функция  $d\_qubic\_spline(x\_nodes, x, qs\_coeff)$ , реализовывающая эти вычисления, представлена в файле basic.ipynb и в Листинге 2.

---

Листинг 2. Реализация функции  $d\_qubic\_spline(x\_nodes, x, qs\_coeff)$

---

```
1 def d_qubic_spline(x_nodes, x, qs_coeff):
2     n = len(x_nodes)
3
4     for i in range(0, n - 1):
5         if x >= x_nodes[i] and x <= x_nodes[i + 1]:
6             b = qs_coeff[i][1]
7             c = qs_coeff[i][2]
8             d = qs_coeff[i][3]
9             difX = x - x_nodes[i]
10            d_s_x = b + 2 * c * difX + 3 * d * difX * difX
11
12    return d_s_x
```

---

### 3.3. Построение аппроксимации зависимости уровня поверхности жидкости $h(x)$ от координаты $x$

Для построения приближения зависимости уровня поверхности жидкости  $h(x)$  от координаты  $x$  по точкам из таблицы 1 использован метод интерполяции кубическими сплайнами. С помощью функции  $qubic_spline_coeff(x\_nodes, y\_nodes)$  найдены коэффициенты естественного кубического сплайна после подстановки данных точек.

Далее, была построена система координат, на которой выводятся данные точки и естественный кубический сплайн, проходящий через эти точки, с помощью функции `qubic_spline(x_nodes, x, qs_coeff)`.

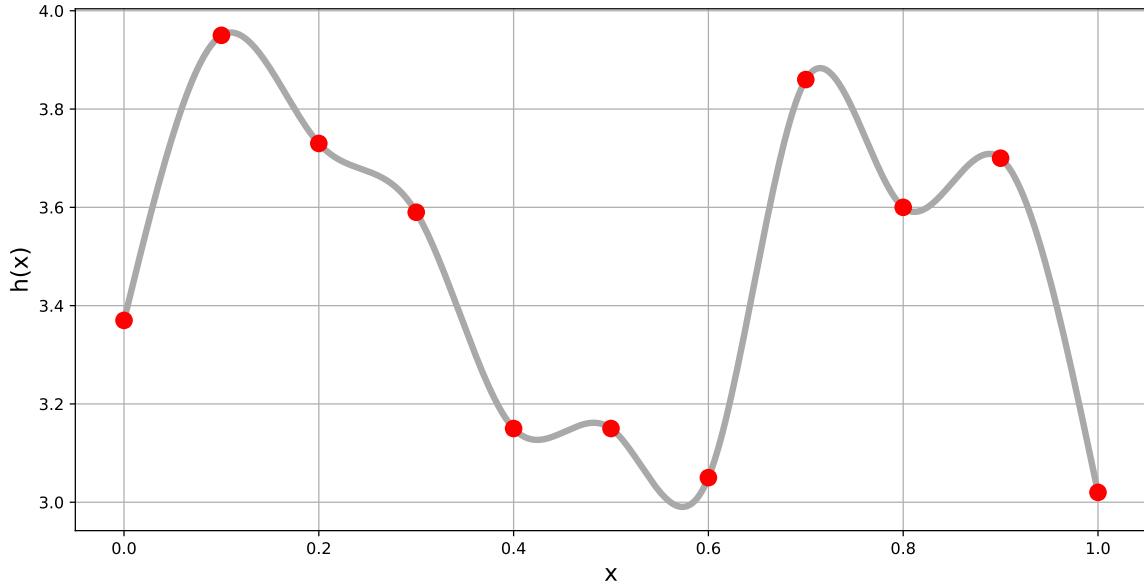


Рис. 2. Аппроксимация зависимости уровня жидкости  $h(x)$  от координаты  $x$  с помощью естественного кубического сплайна вместе с исходными узлами.

Функция `plotDataAndInterpolant(x_nodes, y_nodes)`, реализовывающая построение приближения, представлена в файле basic.ipynb и в Листинге 3.

Листинг 3. Реализация функции `plotDataAndInterpolant(x_nodes, y_nodes)`

---

```

1 def plotDataAndInterpolant(x_nodes, y_nodes):
2     qs_coeff = qubic_spline_coeff(x_nodes, y_nodes)
3
4     fig, ax = plt.subplots(figsize=(12, 6))
5     x_for_plotting = np.linspace(0, 1, 1000)
6
7     ax.set_xlabel('x', fontsize = 'x-large')
8     ax.set_ylabel('h(x)', fontsize = 'x-large')
9     ax.plot(x_for_plotting, [qubic_spline(x_nodes, x, qs_coeff) for x in x_for_plotting],
10             '--', color = 'darkgrey', linewidth = 4)
11    ax.plot(x_nodes, y_nodes, 'ro', markersize = 10)
12
13    ax.grid(True)
14    plt.savefig('basic.pdf')
```

---

## 4 Продвинутая часть

1. Вычисление  $i$ -ого базисного полинома Лагранжа.
2. Вычисление значения интерполяционного полинома Лагранжа в точке  $x$ .
3. Проведение анализа, позволяющего выявить влияние погрешности величины  $x$  на интерполяцию:
  - (a) Генерирование 1000 векторов значений  $\tilde{x}_i$ , зависящих от случайной величины  $Z$ .
  - (b) Построение интерполянта Лагранжа для каждого из 1000 векторов.
  - (c) Построение функций  $\tilde{h}_l, \tilde{h}_u$  таких, что вероятность нахождения значения интерполянта в точке  $x$  в интервале  $[\tilde{h}_l; \tilde{h}_u]$  равна 0.9.
  - (d) Отображение на едином графике функций  $\tilde{h}_l, \tilde{h}_u$ , усреднённого интерполянта и узлов из таблицы 1.
  - (e) Выводы о чувствительности участков интерполянта к погрешности начальных заданных условий.
4. Повторение анализа из пункта 3 в предположении, что  $x_i$  заданы точно, а  $h_i$  заменены на  $\tilde{h}_i$ , которые зависят от случайной величины  $Z$ .
5. Повторение анализов из пунктов 3 и 4, используя метод интерполяции кубического сплайном.

### 4.1. Вычисление $i$ -ого базисного полинома Лагранжа

$$l_i(x) = \prod_{i \neq j} \frac{x - x_j}{x_i - x_j}; \quad (5)$$

где  $j = 0, \dots, n - 1$  ( $n$  - количество узлов  $x\_nodes$ )

Реализация формулы (5) представлена на языке программирования Python в Листинге 4:

Листинг 4. Реализация функции  $l\_i(i, x, x\_nodes)$  вычисления  $i$ -ого базисного полинома Лагранжа

---

```
1 def l_i(i, x, x_nodes):
2     n = len(x_nodes)
3     prod = 1
4     for j in range(0, n):
5         if j != i:
6             prod = prod * (x - x_nodes[j]) / (x_nodes[i] - x_nodes[j])
7     return prod
```

---

## 4.2. Вычисление значения интерполяционного полинома Лагранжа в точке x

$$L_{n-1}(x) = \sum_{i=0}^{n-1} f(x_i) \cdot l_i(x); \quad (6)$$

где  $l_i(x)$  - формула (5), а  $n$  - количество узлов x\_nodes.

Реализация формулы (6) представлена на языке программирования Python в Листинге 5:

Листинг 5. Реализация функции  $L(x, x\_nodes, y\_nodes)$  вычисления значения интерполяционного полинома Лагранжа в точке x

---

```
1 def L(x, x_nodes, y_nodes):
2     n = len(x_nodes)
3     sum = 0
4     for i in range(0, n):
5         sum = sum + y_nodes[i] * l_i(i, x, x_nodes)
6     return sum
```

---

## 4.3. Проведение анализа, позволяющего выявить влияние погрешности величины x на интерполяцию

### Пункт (а)

Для генерирования 1000 векторов значений  $[\tilde{x}_0, \dots, \tilde{x}_{10}]^T$  создан пустой список. В цикле 1000 раз создаётся вектор из  $n$  значений(в нашем случае - 11), которые инициализируем числами  $\tilde{x}_i = x_i + Z$ , где  $x_i$  - значение из таблицы 1,  $Z$  - случайная величина с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением  $10^{-2}$ .

Для создания случайного значения  $Z$  для каждого числа используем модуль *random* языка Python, в котором существует функция *gauss()*, возвращающая случайное число с гауссовым(нормальным) распределением.

Далее, каждый вектор добавляется в список, в итоге получен список из 1000 векторов, в которых по 11 значений  $\tilde{x}_i$  со случайной погрешностью.

Реализация генерирования 1000 векторов значений  $\tilde{x}_i$  или других необходимых значений представлена на языке программирования Python в Листинге 6:

Листинг 6. Реализация функции *p1Calculation(nodesXY)* генерирования 1000 векторов значений, зависящих от случайной величины

---

```
1 def p1Calculation(nodesXY):
2     n = len(nodesXY)
3
4     vectorList = []
5     for i in range(0, 1000):
6         krXY = np.zeros(n)
7         for j in range(0, n):
```

```

8     Z = random.gauss(0, 0.01)
9     krXY[j] = nodesXY[j] + Z
10    vectorList.append(krXY)
11
12 return vectorList

```

---

### Пункт (б)

Для каждого из полученных 1000 векторов был построен интерполянт Лагранжа, воспользовавшись формулой (6).

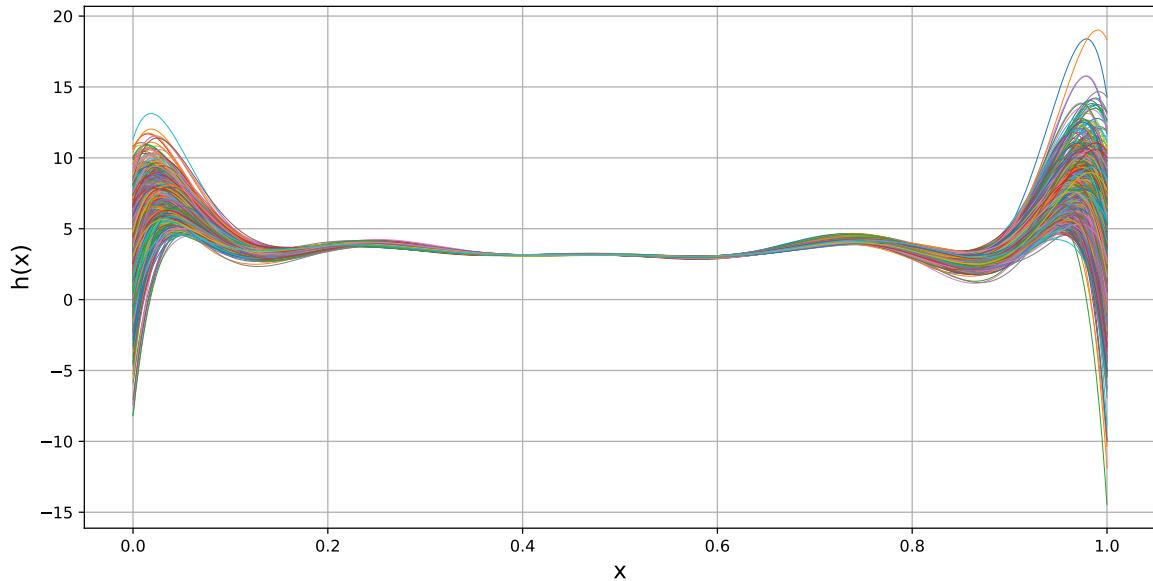


Рис. 3. 1000 различных интерполянтов Лагранжа на основе значений  $\tilde{x}_i$ .

Функция, реализующая построение 1000 таких интерполянтов, представлена на языке программирования Python в Листинге 7:

Листинг 7. Реализация функции *plotParagraph3b(x\_list, y\_nodes)* построения 1000 интерполянтов Лагранжа на основе значений  $\tilde{x}_i$

---

```

1 def plotParagraph3b(x_list, y_nodes):
2     fig, ax = plt.subplots(figsize=(12, 6))
3     x_for_plotting = np.linspace(0, 1, 1000)
4
5     ax.set_xlabel('x', fontsize = 'x-large')
6     ax.set_ylabel('h(x)', fontsize = 'x-large')
7
8     for i in range(0, 1000):
9         ax.plot(x_for_plotting, [L(x, x_list[i], y_nodes) for x in x_for_plotting], '-',
10                 linewidth = 0.5)

```

```

10
11     ax.grid(True)
12     plt.savefig('plotParagraph3b.pdf')

```

---

### Пункт (с)

В математической статистике существует понятие доверительного интервала. Этот термин обозначает интервал, в который попадают измеренные в экспериментах значения с некоторой доверительной вероятностью (процент экспериментальных значений, попавших в доверительный интервал, и есть доверительная вероятность).

То есть, действуя согласно понятию доверительного интервала, можно проанализировать 1000 поставленных экспериментов со значением интерполянта в каждой точке  $x \in [0; 1]$  и определить некоторую доверительную полосу, то есть такой участок между функциями  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$ , что при вычислении значения интерполянта в  $x \in [0; 1]$  эта точка попадёт в участок с вероятностью 90%.

Таким образом, функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$  есть доверительные пределы, то есть граничные точки доверительного интервала на  $x \in [0; 1]$  или же по-другому границы доверительной полосы.

Эти границы необходимо найти, есть два способа, как это можно сделать:

#### Первый способ:

Отсортировать 1000 значений интерполянта в каждой точке  $x \in [0; 1]$ . Далее, найти среди них число, наиболее близкое к среднему значению, а затем от него выбрать 899 точек так, что вместе с ним они определят необходимый доверительный интервал в точке  $x$ . Производя данную операцию на всём отрезке  $[0; 1]$ , будет получено множество доверительных интервалов, которое и составит доверительную полосу, границами которого (функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$ ) будут являться нижние и верхние границы отсортированных и вычисленных 900 точек.

#### Второй способ:

Вычислить в каждой точке  $x \in [0; 1]$  значение усреднённого интерполянта, то есть среднее арифметическое (математическое ожидание) 1000 значений интерполирования в точке  $x$ . Исходя из значений усреднённого интерполянта, можно определить доверительные интервалы в каждой точке по следующей формуле:

$$\bar{h} - z_\alpha \cdot \sigma \leq \mu \leq \bar{h} + z_\alpha \cdot \sigma; \quad (7)$$

где  $\mu$  - некоторое значение интерполянта в точке  $x$ ,  $\bar{h}$  - среднее значение интерполянтов в точке  $x$ ,  $z_\alpha$  - критическое значение стандартного нормального распределения для уровня значимости  $\alpha = 1 - P$ , где  $P$  - доверительная вероятность, а  $\sigma$  - стандартное отклонение от среднего значения, которое можно вычислить по формуле:

$$\sigma = \sqrt{\frac{\sum_{i=0}^{n-1} (h_i - \bar{h})^2}{n}}; \quad (8)$$

где  $n$  - кол-во экспериментов (значений интерполянтов в точке  $x$ ).

В данной работе будет использован **второй способ**, поэтому следует начать с вычисления среднего значения  $\bar{h}$  в точке  $x$ . Данное вычисление реализовано в функции *meanCalculation(x, x\_list, y\_nodes)* на языке Python и представлено в Листинге 8:

Листинг 8. Реализация функции  $meanCalculation(x, x\_list, y\_nodes)$  вычисления среднего значения  $\bar{h}$  в точке  $x$

---

```
1 def meanCalculation(x, x_list, y_nodes):  
2     sum = 0  
3     for i in range(0, 1000):  
4         sum = sum + L(x, x_list[i], y_nodes)  
5     sum = sum / 1000  
6  
7     return sum
```

---

Затем необходимо найти стандартное отклонение от среднего значения  $\sigma$  по формуле (8), а  $z_\alpha$  по таблице для вероятности 0.9 будет равен **1.645**. Таким образом, произведено вычисление доверительного интервала в точке  $x$  по формуле (7).

Реализация этих вычислений представлена в функции  $confIntCalculation(x, x\_list, y\_nodes)$  в Листинге 9:

Листинг 9. Реализация функции  $confIntCalculation(x, x\_list, y\_nodes)$  вычисления доверительного интервала и среднего значения  $\bar{h}$  в точке  $x$

---

```
1 def confIntCalculation(x, x_list, y_nodes):  
2     mean = meanCalculation(x, x_list, y_nodes)  
3     sigma = 0  
4     for i in range(0, 1000):  
5         h_i = L(x, x_list[i], y_nodes)  
6         sigma = sigma + (h_i - mean) * (h_i - mean)  
7     sigma = sqrt(sigma / 1000)  
8     delta = 1.645 * sigma  
9  
10    return mean - delta, mean, mean + delta
```

---

Построены функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$ , а также выделена доверительная полоса между ними.

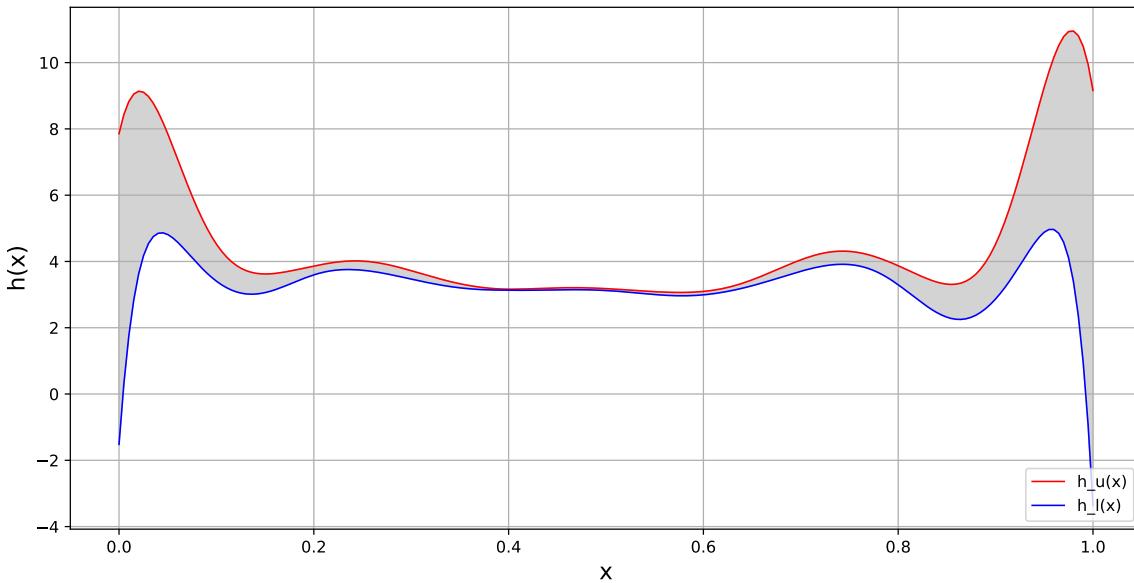


Рис. 4. Функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$

Реализация функции построения представлена в Листинге 10:

Листинг 10. Реализация функции  $plotParagraph3c(x, x\_list, y\_nodes)$  построения функций  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$  и доверительной полосы между ними

```

1 def plotParagraph3c(x_list, x_nodes, y_nodes):
2     fig, ax = plt.subplots(figsize=(12, 6))
3     x_for_plotting = np.linspace(0, 1, 200)
4
5     ax.set_xlabel('x', fontsize = 'x-large')
6     ax.set_ylabel('h(x)', fontsize = 'x-large')
7
8     h_l = []
9     h_u = []
10    for x in x_for_plotting:
11        mean_l, mean, mean_u = conflntCalculation(x, x_list, y_nodes)
12        h_l.append(mean_l)
13        h_u.append(mean_u)
14
15    ax.fill_between(x_for_plotting, h_u, h_l, color = 'lightgray')
16    ax.plot(x_for_plotting, h_u,'-', color = 'r', label = 'h_u(x)', linewidth = 1)
17    ax.plot(x_for_plotting, h_l,'-', color = 'b', label = 'h_l(x)', linewidth = 1)

```

```

18
19     ax.grid(True)
20     ax.legend(loc='lower right')
21     plt.savefig('plotParagraph3c.pdf')

```

#### Пункт (д)

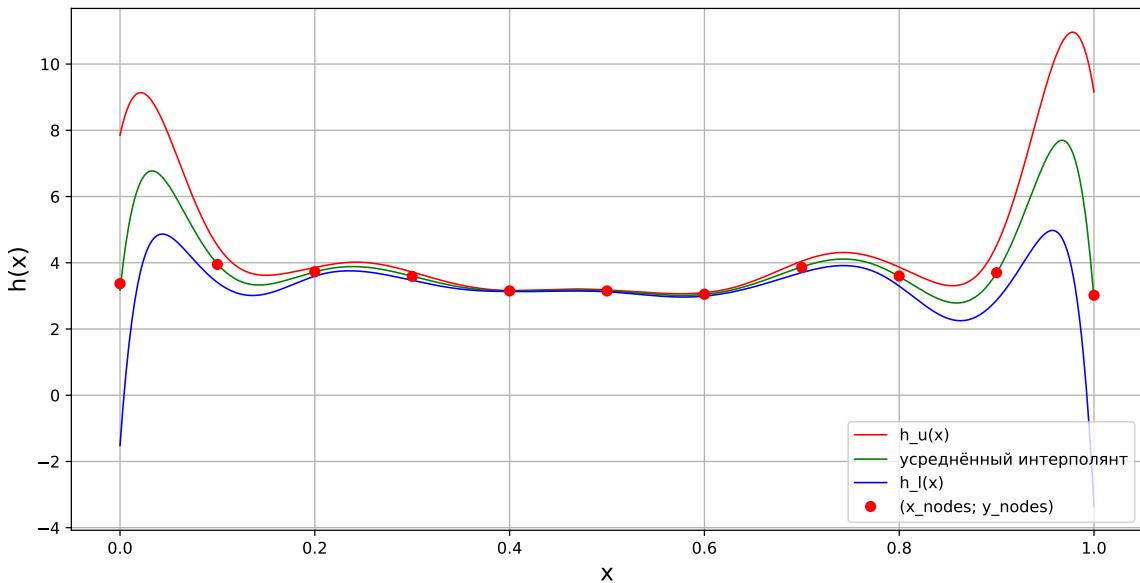


Рис. 5. Построенные функции  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$ , усредненный интерполянт и узлы из таблицы 1

Реализация функции построения этих графиков и все представленные в пункте 3 функции находятся в файле advanced3.ipynb.

#### Пункт (е)

На рисунках 3, 4, 5 заметно, что значительные различия в интерполянтах скапливаются около границ отрезка  $[0; 1]$ . Это связано с тем, что при интерполяции используется полином высокой степени, а именно 10-й степени. Явление возникновения таких нежелательных осциляций называется "феноменом Рунге".

Так как заданные узлы равнодistantны, то согласно исследованию этого феномена погрешность интерполяции с возрастанием степени полинома будет стремиться к бесконечности.

В нашем случае известные узлы измерены с некоторой погрешностью, приближающие и отдаляющие узлы у границ отрезка друг к(от) другу(а). Таким образом, если они отдаляются, то погрешность интерполяции становится значительнее, что видно на рисунках 3, 4, 5. Но если интерполяция затрагивает узлы, находящиеся ближе к центру, то погрешность в измерении  $x_i$  влияет на погрешность интерполяции намного слабее.

**4.4. Повторение анализа из пункта 3 в предположении, что  $x_i$  заданы точно, а  $h_i$  заменены на  $\tilde{h}_i$ , которые зависят от случайной величины Z**

#### Пункт (а)

Так же, как в пункте 4.3, было сгенерировано 1000 векторов значений  $[\tilde{h}_0, \dots, \tilde{h}_{10}]^T$ , где  $\tilde{h}_i = h_i + Z$ , а  $h_i$  - значение из таблицы 1,  $Z$  - случайная величина с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением  $10^{-2}$ .

Реализация генерации 1000 векторов значений  $\tilde{h}_i$  представлена на языке программирования Python в Листинге 6.

#### Пункт (б)

Для каждого из полученных 1000 векторов был построен интерполянт Лагранжа, воспользовавшись формулой (6).

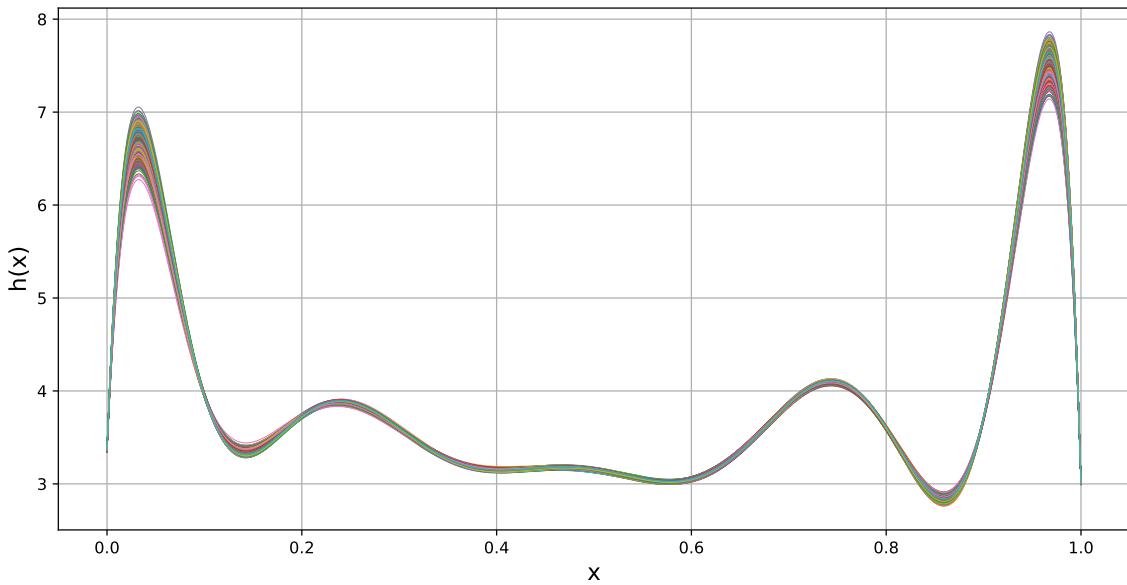


Рис. 6. 1000 различных интерполянтов Лагранжа на основе значений  $\tilde{h}_i$ .

Функция, реализующая построение 1000 таких интерполянтов, практически не отличается от функции, представленной в Листинге 7, с той лишь разницей, что здесь используется 1000 векторов значений  $\tilde{h}_i$ .

#### Пункт (с)

Повторив метод из анализа в пункте №3, были построены функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$ , но в этом случае используются точные значения  $x_i$ , а  $\tilde{h}_i$  зависят от случайной величины.

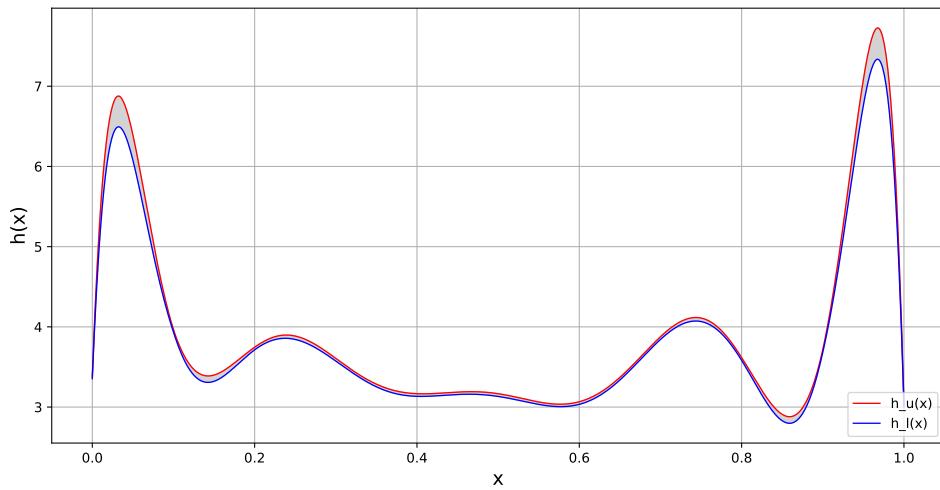


Рис. 7. Функции  $\tilde{h}_l(x)$  и  $\tilde{h}_u(x)$

Функции, реализующие вычисления и построение, практически не отличаются от функций, представленных в Листингах 8, 9, 10, с той лишь разницей, что используются 1000 векторов значений  $\tilde{h}_i$ .

#### Пункт (d)

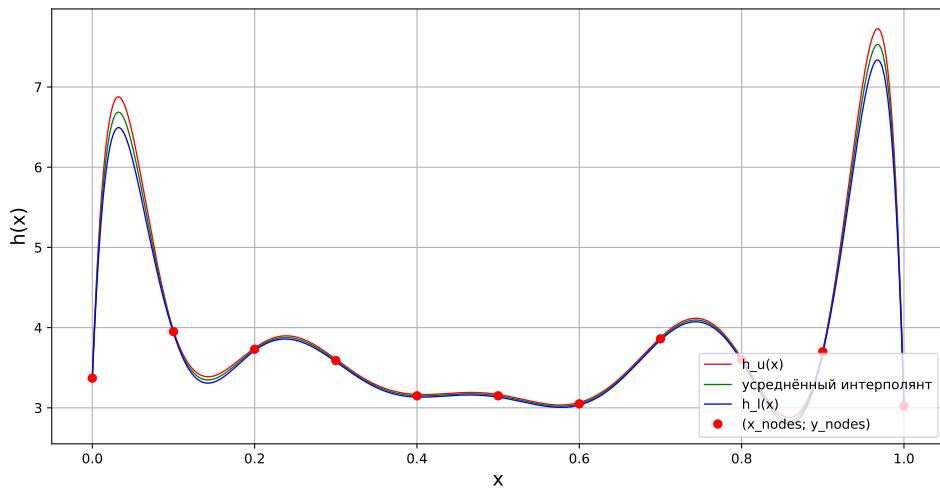


Рис. 8. Построенные функции  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$ , усредненный интерполянт и узлы из таблицы 1

Реализация функции построения этих графиков и все представленные в пункте 4 функции находятся в файле advanced4.ipynb.

### Пункт (е)

Рассматривая рисунки 6, 7, 8, можно заметить, что между интерполянтами нет различий, кроме как отклонения по оси Y ( $h(x)$ ). Это связано с тем, что у всех 1000 интерполянтов значения базисного полинома Лагранжа вычисляются одинаково точно без погрешностей. Соответственно, при поиске производной различие будет только в коэффициенте перед ними, зависящие от погрешности измерения значения  $\tilde{h}_i$ .

Повторив рассуждения о 'феномене Рунге', можно сделать вывод, что осцилляции, возникающие у границ всего отрезка, значительно отличаться не будут. Разница лишь в том, что в участках около экстремумов интерполирующей функции, погрешность измерения влияет чуть сильнее остальных участков.

Сравнивая с анализом, выполненным в пункте 4.3, можно сказать, что в целом графики и результаты похожи, но погрешность в  $x_i$  оказывается и влияет гораздо сильнее, чем погрешность в  $h_i$ . В целом, выводы изменились.

#### 4.5. Повторение анализов из пунктов 3 и 4, используя метод интерполяции кубическим сплайнам. Сравнение результатов анализа для интерполяции Лагранжа и интерполяции кубическим сплайнам

##### Пункт (3а)

Сгенерировано 1000 векторов значений  $[\tilde{x}_0, \dots, \tilde{x}_{10}]^T$ , где  $\tilde{x}_i = x_i + Z$ , а  $x_i$  - значение из таблицы 1,  $Z$  - случайная величина с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением  $10^{-2}$ .

Реализация генерации 1000 векторов значений  $\tilde{x}_i$  представлена на языке программирования Python в Листинге 6:

##### Пункт (3б)

Для каждого из полученных 1000 векторов выполнено интерполирование кубическими сплайнами, использовав функции и формулы из базовой части (1).

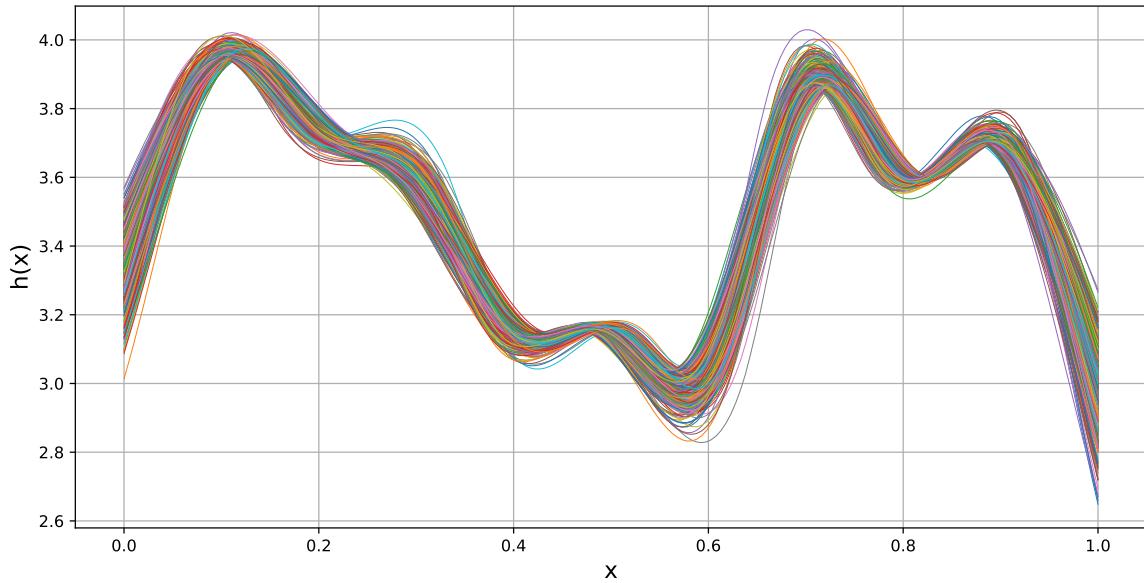


Рис. 9. 1000 различных кубических сплайнов на основе значений  $\tilde{x}_i$

Разница в реализации состоит в том, что при интерполировании кубическими сплайнами используются отрезки  $[x_i; x_{i+1}]$ , которые подвергаются некоторой погрешностью. Соответственно, происходит смещение относительно 0 и 1 на концах отрезка, что делает невозможным в некоторых случаях построение в этих точках. Поэтому в тех случаях, когда наши отрезки попадают внутрь отрезка  $[0; 1]$ , в котором происходит построение, то используем коэффициенты, вычисленные для граничных отрезков  $[x_0; x_1]$  и  $[x_9; x_{10}]$ .

Реализация функции построения  $plotParagraph53b(x\_list, y\_nodes)$  и вспомогательных функций находится в файле advanced53.ipynb.

### Пункт (3с)

Построены функции  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$ , а также выделена доверительная полоса между ними точно так же, как это было сделано в пункте 4.3.

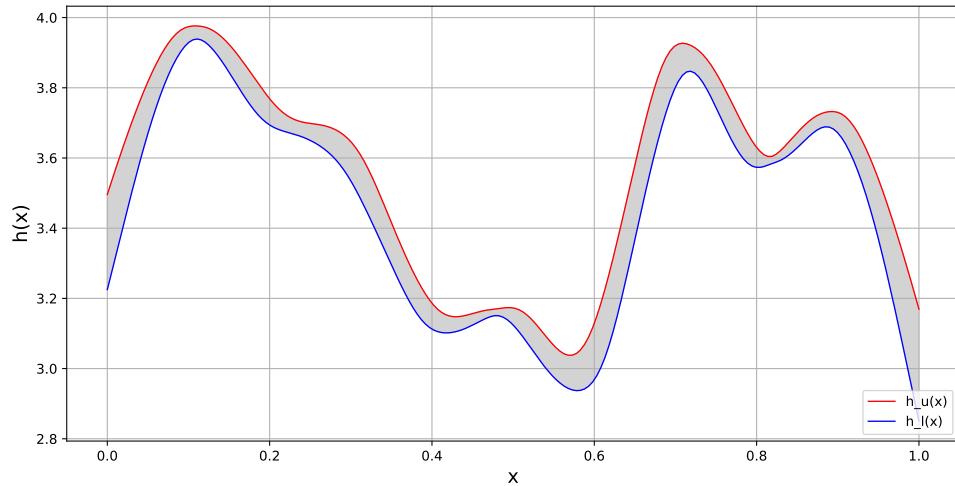


Рис. 10. Функции  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$

Реализация функции построения `plotParagraph53c(x_list, x_nodes, y_nodes)` и вспомогательных функций находится в файле advanced53.ipynb.

### Пункт (3д)

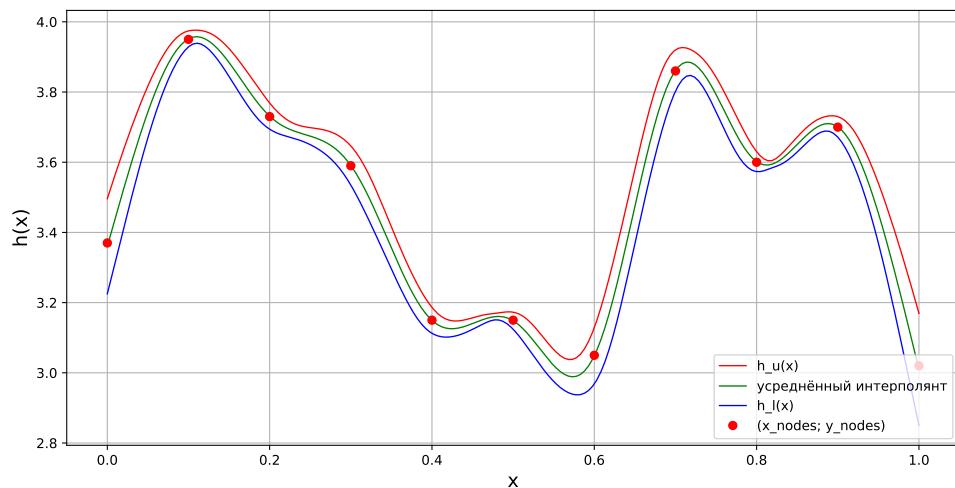


Рис. 11. Построение функций  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$ , усредненного интерполянта и узлов из таблицы 1

Реализация функции построения  $plotParagraph53d(x\_list, x\_nodes, y\_nodes)$  и вспомогательных функций находится в файле advanced53.ipynb.

#### Пункт (3e)

Из рисунков 9, 10, 11 понятно, что подобных осциляций, как при приближении интерполяционным многочленом Лагранжа, нет. Прежде всего, это связано с оптимальной степенью многочленов, а именно третьей, а также с применением не глобальной интерполяции, а локальной для отдельно взятых соседних узлов.

Вычисление кубического сплайна зависит от его коэффициентов, которые в свою очередь зависят от расстояния между узлами по координате X, а также от самих значений в этих узлах. На рисунках 9, 10, 11 заметно, что на протяжении всего отрезка погрешности в измерении влияют примерно одинаково. Там, где значения чуть-чуть выбираются, большее расстояние между соседними узлами, что значительно изменяет коэффициенты, а также вычисление кубического сплайна.

#### Пункт (4a)

Сгенерировано 1000 векторов значений  $[\tilde{h}_0, \dots, \tilde{h}_{10}]^T$ , где  $\tilde{h}_i = h_i + Z$ , а  $h_i$  - значение из таблицы 1,  $Z$  - случайная величина с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением  $10^{-2}$ .

Реализация генерации 1000 векторов значений  $\tilde{h}_i$  представлена на языке программирования Python в Листинге 6:

#### Пункт (4b)

Для каждого из полученных 1000 векторов выполнено интерполирование кубическими сплайнами, использовав функции и формулы из базовой части (1).

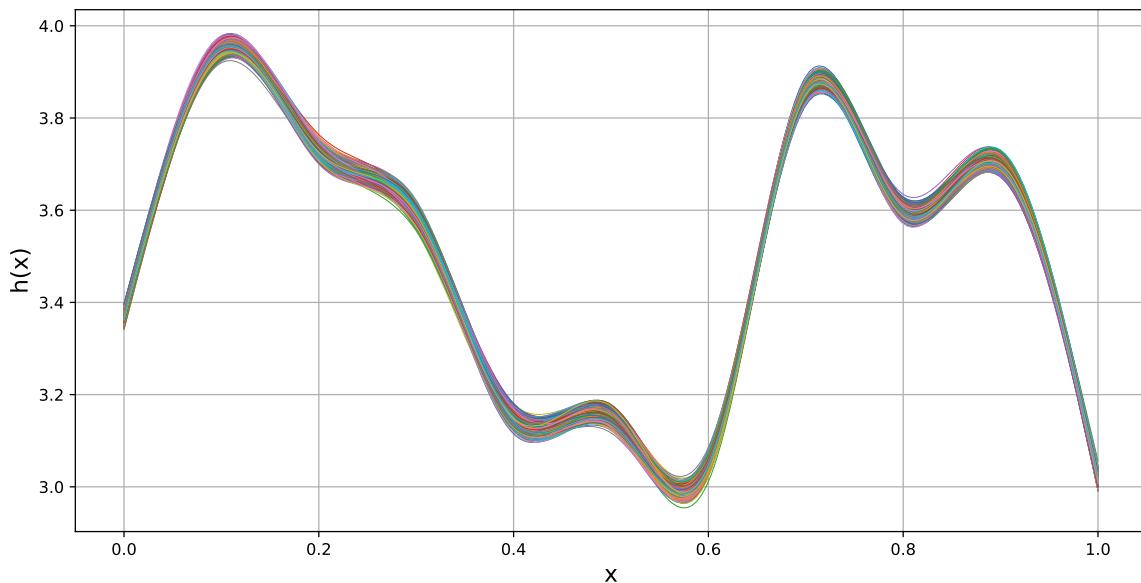


Рис. 12. 1000 различных кубических сплайнов на основе значений  $\tilde{h}_i$

В отличие от пункта 4.5.3 погрешность (случайная величина в значении) затрагивает не  $x_i$ , а  $h_i$ . Соответственно, действия при построении совпадают с пунктом 4.4, но уже для интерполяции кубическими сплайнами.

Реализация функции построения  $plotParagraph54b(x\_nodes, y\_list)$  и вспомогательных функций находится в файле advanced54.ipynb.

#### Пункт (4c)

Построены функции  $\tilde{h}_l(x), \tilde{h}_u(x)$ , выделена доверительная полоса между ними точно так же, как было сделано в пункте 4.4.

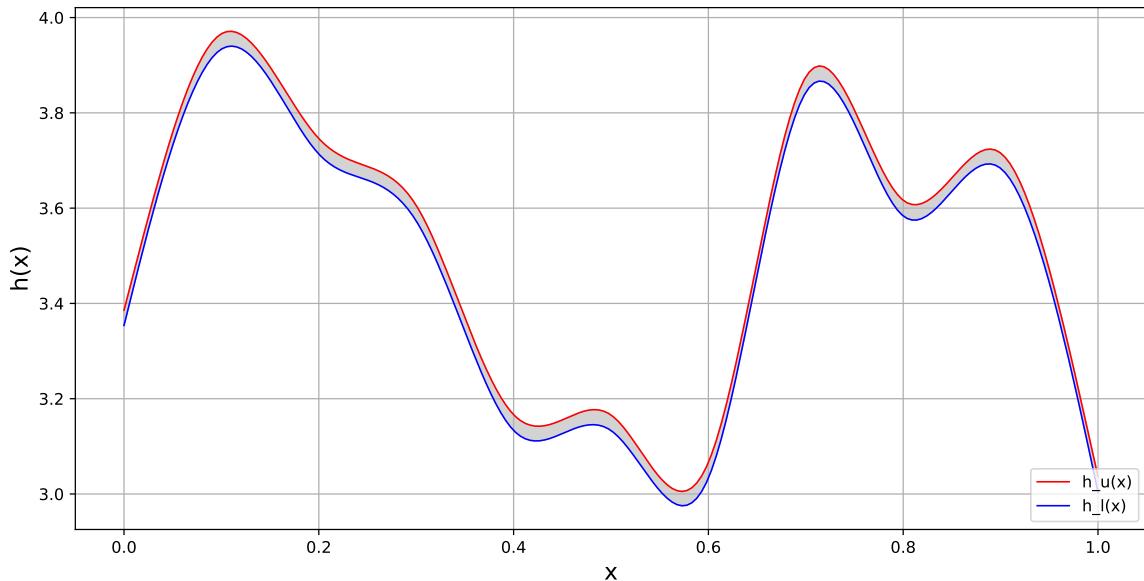


Рис. 13. Функции  $\tilde{h}_l(x), \tilde{h}_u(x)$

Реализация функции построения  $plotParagraph54c(y\_list, x\_nodes, y\_nodes)$  и вспомогательных функций находится в файле advanced54.ipynb.

#### Пункт (4d)

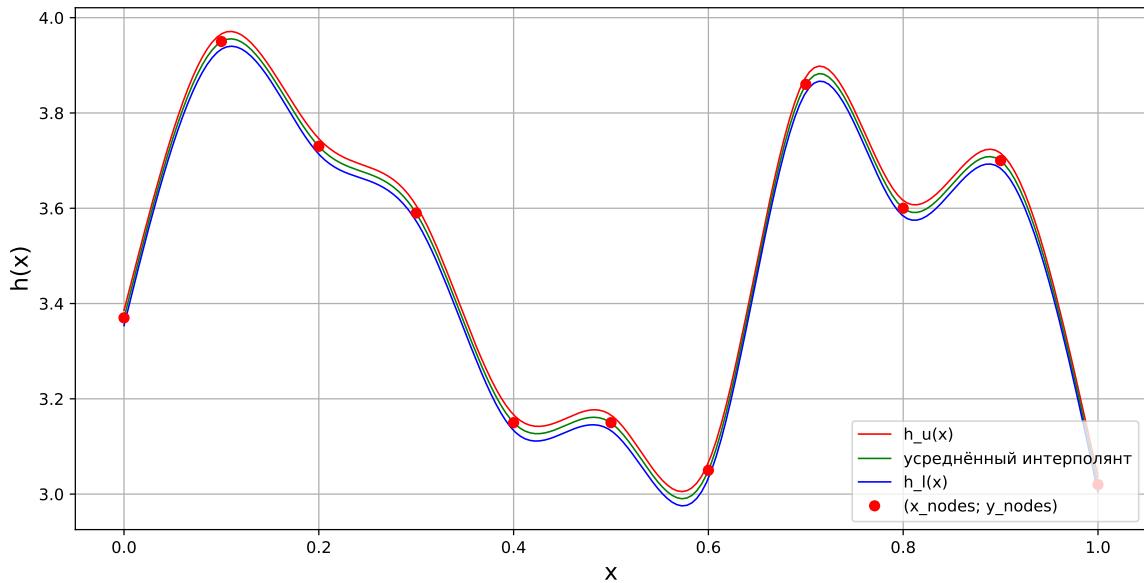


Рис. 14. Построение функций  $\tilde{h}_l(x)$ ,  $\tilde{h}_u(x)$ , усредненного интерполянта и узлов из таблицы 1

Реализация функции построения `plotParagraph54d(y_list, x_nodes, y_nodes)` и вспомогательных функций находится в файле advanced54.ipynb.

#### Пункт (4e)

По аналогии с пунктом Зе из рисунков 12, 13, 14 понятно, что погрешность в значении  $h_i$  не сильно изменяет значение построенного кубического сплайна. В данном случае доверительная полоса гораздо уже, чем была в пункте 4.5.3, что говорит о том, что при вычислении коэффициентов кубического сплайна и его значения гораздо большую роль играют именно точки  $x_i$ , чем  $h_i$ .

Сравнивая результаты анализов интерполяции Лагранжа и интерполяции кубическим сплайнами, необходимо отметить, что методы значительно различаются. В случае интерполяции кубическими сплайнами отсутствуют нежелательные осцилляции, а также, в целом, значения близки к исходным узлам. Погрешности, которые накладываются на значения  $x_i$ ,  $h_i$ , влияют сильнее на результаты в случае интерполяции многочленом Лагранжа и гораздо меньше эти изменения можно увидеть в случае интерполяции кубическими сплайнами.

## 5 Заключение

1. По завершении базовой части был изучен метод интерполяции кубическими сплайнами, были разработаны функция вычисления коэффициентов кубического сплайна с естественными граничными условиями посредством решения матричного уравнения, функции вычисления значения естественного кубического сплайна и его производной в некоторой точке  $x$ , а также эти функции были применены для построения аппроксимации уровня жесткости  $h(x)$  от координаты  $x$ .
2. По завершении продвинутой части был изучен метод интерполяции многочленом Лагранжа. Ознакомились с методами получения случайных значений с нормальным распределением. Проанализировали влияние погрешностей в изначально заданных параметрах на приближение функции методами интерполяции многочленом Лагранжа и кубическими сплайнами. Были сравнены оба метода интерполяции, а также были разработаны функции по реализации метода интерполяции многочленом Лагранжа, вычисления доверительного интервала и доверительной полосы на языке программирования Python.

## Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.

## Выходные данные

Очиров Б. Ю.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] – Москва: 2021. – 25 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка: © ассистент кафедры РК-6, PhD А.Ю. Першин  
Решение и вёрстка: © студент группы РКб-55Б, Очиров Б. Ю.

2021, осенний семестр