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:	Степанов Никита Николаевич
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Интерполяция в условиях измерений с неопределенностью (вариант 5)

Студент _____ Степанов Н. Н.
подпись, дата Фамилия, И.О.

Преподаватель _____ _____
подпись, дата Фамилия, И.О.

Москва, 2021

Содержание

Интерполяция в условиях измерений с неопределенностью (вариант 5)

Задание

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 (*qs_coeff* обозначает матрицу коэффициентов).

3. Используя данные в таблице (Рис. 2), требуется построить аппроксимацию зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна и продемонстрировать ее на графике вместе с исходными узлами.

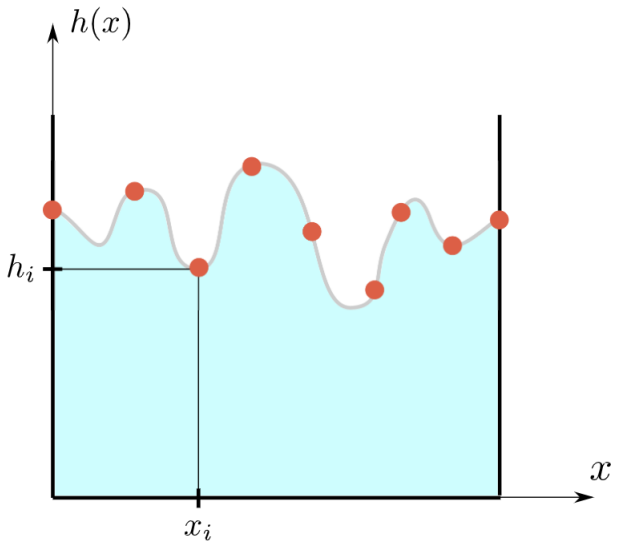

Рис. 1. Поверхность вязкой жидкости (серая кривая), движущейся сквозь некоторую среду (например, пористую). Её значения известны только в нескольких точках(красные узлы)

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

Рис. 2. Значения уровня поверхности вязкой жидкости

Продвинутая часть

1. Разработать функцию *l_ i(i, x, x_ nodes)*, которая возвращает значение i – 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 $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$, предполагая, что $\tilde{x_i} = x_i + Z$ 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_i из таблицы 1. В результате вы должны иметь 1000 различных интерполянтов.

(c) Предполагая, что все интерполянты представляют собой равновероятные события, построить такие функции $\tilde{h}_l(x)\tilde{h}_u(x)$, где $\tilde{h}_u(x) < \tilde{h}_u(x)$ 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 сплайном. Какие выводы вы можете сделать, сравнив результаты анализа для интерполяции Лагранжа и интерполяции кубическим сплайном?

1 Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – знакомство с интерполяцией в целом (базовая часть), анализ влияния неопределенностей на ее результат (продвинутая часть)

2 Знакомство с интерполяцией

1. Коэффициенты естественного кубического сплайна

Разработаем функцию *qubic_spline_coeff(x_nodes, y_nodes)*, которая посредством решения матричного уравнения вычисляет коэффициенты естественного кубического сплайна. Однако, для большего удобства будем возвращать матрицу размерности $(N-1)*4$, где N - кол-во узлов интерполяции. Составление и решение матричного уравнения $Ac = b$ и дальнейшее использование коэффицента кубического сплайна c для нахождения других коэффициентов: a, b, d происходит согласно материалу и формулам, предсталенных в лекциях.

```
1  def qubic_spline_coeff(x_nodes, y_nodes):
2      N = len(x_nodes)
3      A = [[0] * N for i in range (0,N)]
4      A[0][0] = 1
5      A[N-1][N-1] = 1
6      a = [y_nodes[i] for i in range(0, N)]
7      h = np.array([x_nodes[i + 1] - x_nodes[i] for i in range(0, N-1)])
8      b = np.array([0] + [((3 / h[i] * (a[i + 1] - a[i]) - (3 / h[i - 1] * (a[i] - a[i - 1])))) for
           i in range (1, N-1)] + [0])
9      for i in range (1, N-1):
10             A[i][i-1] = h[0]
11             A[i][i] = 2*(h[0] + h[1])
12             A[i][i+1] = h[1]
13
14     A_inv = np.linalg.inv(A)
15     c = A_inv @ b
16
17     res = np.zeros((N-1 , 4))
18
19     for i in range (0, N-1):
20         res[i][0] = a[i]
21         res[i][1] = ((1 / h[i]) * (a[i+1] - a[i]) - (h[i] / 3) * (c[i+1] + 2*c[i]))
22         res[i][2] = c[i]
23         res[i][3] = (c[i+1] - c[i]) / (3 * h[i])
24
25     return res
```

2. Значение кубического сплайна и его производной в точке

Напишем 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). Аналогично п. 1, формула для кубического сплайна берется из лекций, формула для производной кубического сплайна получается, соответственно, посредством дифференцирования формулы для кубического сплайна.

$S_i(x_i) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ - формула кубического сплайна

$S_i(x_i)' = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$ - формула производной кубического сплайна

```
def qubic_spline(x_nodes, x, qs_coeff):

  for i in range (0, len(x_nodes)-1):
      if (x_nodes [i] <= x <= x_nodes[i+1]):
        k = i
        x_wk = x
        break

  S_i = qs_coeff[k][0] + qs_coeff[k][1] * (x_wk - x_nodes[k]) + qs_coeff[k][2] * ((x_wk
      - x_nodes[k])**2) + qs_coeff[k][3] * ((x_wk - x_nodes[k])**3)
  return S_i


def d_qubic_spline(x_nodes, x, qs_coeff):
    for i in range (0, len(x_nodes)-1):
      if (x_nodes [i] <= x <= x_nodes[i+1]):
        k = i
        break

    Sd_i = qs_coeff[k][1] + 2 * qs_coeff[k][2] * (x - x_nodes[k]) + 3 * qs_coeff[k][3] *
        ((x - x_nodes[k])**2)
    return Sd_i
```

3. График аппроксимации зависимости уровня поверхности жидкости $h(x)$ от координаты x

В данном задании необходимо, используя данные в таблице (Рис. 2), построить аппроксимацию зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна и продемонстрировать ее на графике вместе с исходными узлами. Осуществим все это с помощью раннее созданных функций. При этом сгенерируем 1000 точек на отрезке [0;1] с помощью функции бибилиотеки *numpy* языка *Python*.

```
x = np.linspace(x_nodes[0], x_nodes[10], 1000)
y = [qubic_spline(x_nodes, x[i], res) for i in range (0, len(x))]
plt.subplots(figsize = (10, 10))
```

```
plt.plot(x, y, color = 'black')
plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

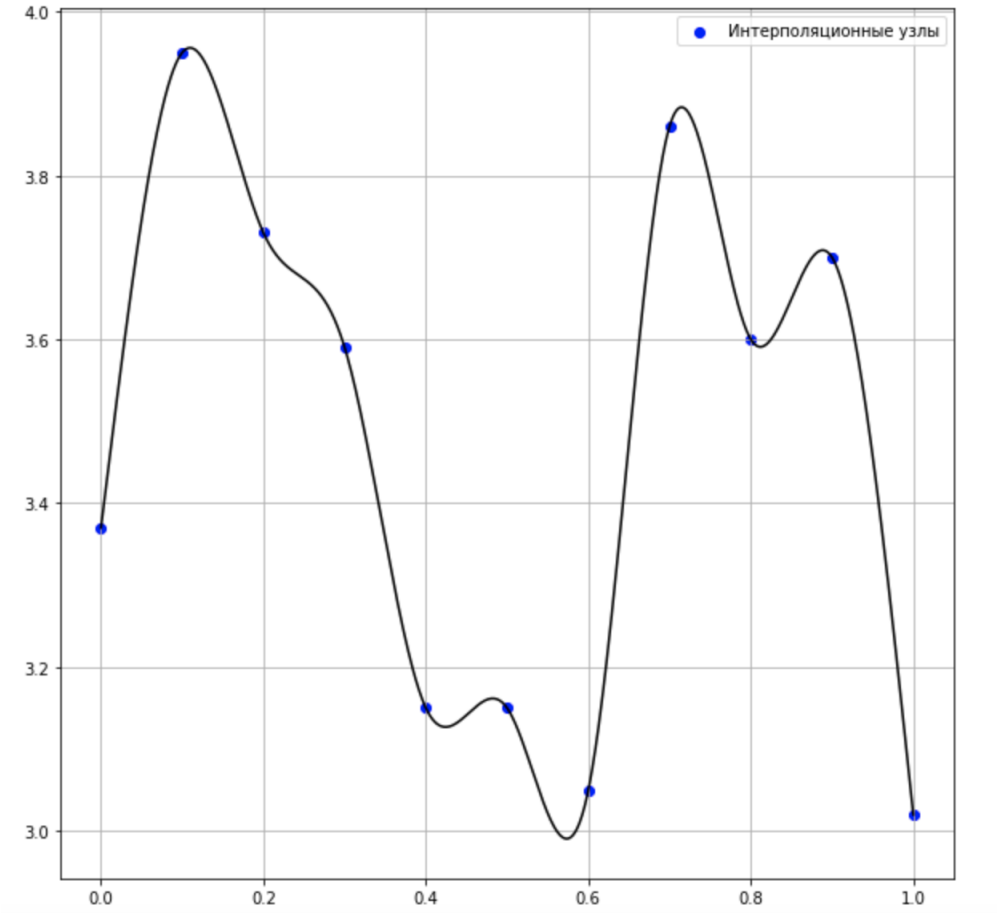

Рис. 3. Аппроксимация зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна. Исходные интерполяционные узлы отмечены синим цветом.

Для большей наглядности, приложу график производной кубического сплайна, построенного на том же наборе точек, что и график кубического сплайна.

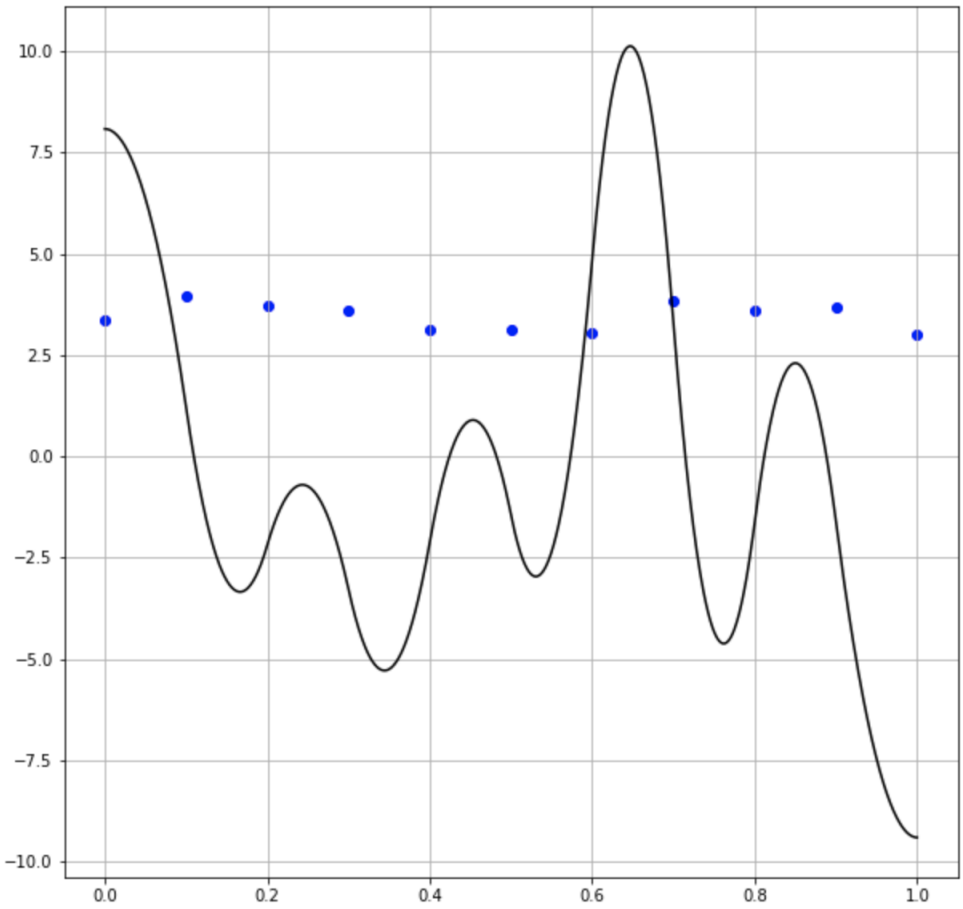

Рис. 4. Производная аппроксимации зависимости уровня поверхности жидкости $h(x)$ от координаты x с помощью кубического сплайна. Исходные интерполяционные узлы отмечены синим цветом.

3 Анализ влияния неопределенностей на результат интерполяции

1. Базисный полином Лагранжа

Согласно заданию, разработаем 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. Формула для $i-$ базисного полинома Лагранжа представлена в лекциях.

```
def l_i(i, x, x_nodes):
    num = 1
    den = 1
    for k in range (0, len(x_nodes)):
        if k == i:
          pass
        else:
          num = num * (x - x_nodes[k])

    for k in range (0, len(x_nodes)):
        if k == i:
          pass
        else:
          den = den * (x_nodes[i] - x_nodes[k])
    l = num / den
    return l
```

2. Интерполяционный полином Лагранжа

Согласно заданию, разработаем 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. Формула для интерполяционного полинома Лагранжа представлена в лекциях.

```
def L(x, x_nodes, y_nodes):
  L_a = 0
  for i in range (0,len(x_nodes)):
    L_a = L_a + l_i(i, x, x_nodes) * y_nodes[i]
  return L_a
```

3. Анализ влияния погрешности на интерполяцию (абсциссы узлов)

а. Генерация 1000 векторов значений $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$

Согласно заданию, необходимо: сгенерировать 1000 векторов значений $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$, предполагая, что $\tilde{x_i} = x_i + Z$ соответствует значению в таблице 1 и Z является

случайной величиной с нормальным распределением с нулевым математическим ожиданием и стандартным отклонением 10^{-2}.
Предварительно создадим матрицу необходимой размерности, затем заполним ее, таким образом, что каждый элемент будет иметь вид: *vecs[i][j] = x_ nodes[j] + np.random.normal(0, 0.01)* После этого, транспонируем матрицу "иксов" с погрешностями, т. к. в условии каждый вектор имеет именно столбец координат узлов по оси абсцисс, а не строку.
Библиотечная функция *np.random.normal* модуля *numpy* возвращает образцы нормального распределения, согласно заданным параметрам.

```
vecs = [[0] * 11 for i in range (0,1000)]
for i in range (0,1000):
  for j in range (0, 11):
    vecs[i][j] = x_nodes[j] + np.random.normal(0, 0.01)
vecs = np.transpose(vecs)
vecs = np.array(vecs)
```

b. Интерполянты Лагранжа для каждого из векторов

Необходимо для каждого из раннее полученных векторов (координаты абсцисс с погрешностью, координаты ординат - нет) построить интерполянт Лагранжа. В результате необходимо получить 1000 графиков. Будем это осуществлять с помощью ранее разработанной функции *L(x, x_ nodes, y_ nodes)*.

```
x_int = np.linspace(0, 1, 1000)
interp_lagr = [[L(x_int[i], vecs[: , j], y_nodes) for i in range (0, len(x_int))] for j in range
    (0, 1000)]
interp_lagr = np.array(interp_lagr)
plt.subplots(figsize = (10, 10))
for i in range (0, 100):
  plt.plot(x_int, interp_lagr[i , :])

plt.grid()
```

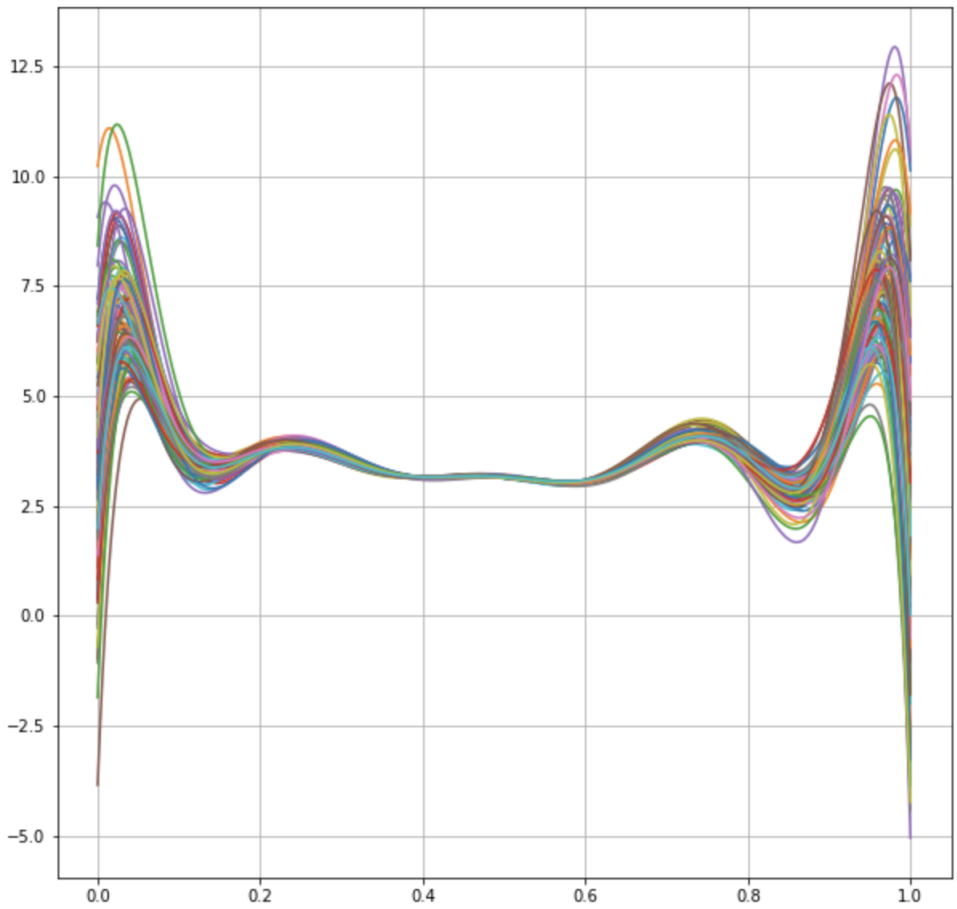

Рис. 5. 1000 интерполянтов Лагранжа, построенных на узлах $[\tilde{x_1}, ..., \tilde{x_{11}}]_i, y\_nodes$

c. Построение функций $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ (аналитический вид)

Необходимо построить функции $\tilde{h}_l(x)\tilde{h}_u(x)$, где $\tilde{h}_u(x) < \tilde{h}_u(x)$ 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.
Осуществим это с помощью функции *percentile* модуля *numpy*.
Алгоритм следующий:

1. *Находим срез значений ординат интерполянтов в конкретной точке*
2. *Находим заданный перцентиль*
3. *Повторяем для каждой точки*

В итоге получаем набор значений ординат интерполянтов, которые соответствуют заданному перцентилю. Для решения данной задачи я выбрал перцентиль 5 и, соответственно, 95.

```
p_95 = [np.percentile(interp_lagr[:, i], 95) for i in range (0, 100)]
p_5 = [np.percentile(interp_lagr[:, i], 5) for i in range (0, 100)]
```

d. Отображение функций $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненного интерполянта, а также интерполяционных узлов

Для решения данной задачи, необходимо отобразить функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненный интерполянт,и, кроме того, интерполяционные узлы. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ были найдены в прошлом пункте, а усредненный интерполянт находится путем нахождения перцентиля 50. Алгоритм его нахождения аналогичен описанному раннее.

```
mean = [np.percentile(interp_lagr[:, i], 50) for i in range (0, 1000)]
plt.subplots(figsize = (10, 10))

plt.plot(x_int, p_95, label = "h_u(x)")
plt.plot(x_int, p_5, label = "h_l(x)")
plt.plot(x_int, mean, label = "Усредненный интерполянт")
plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

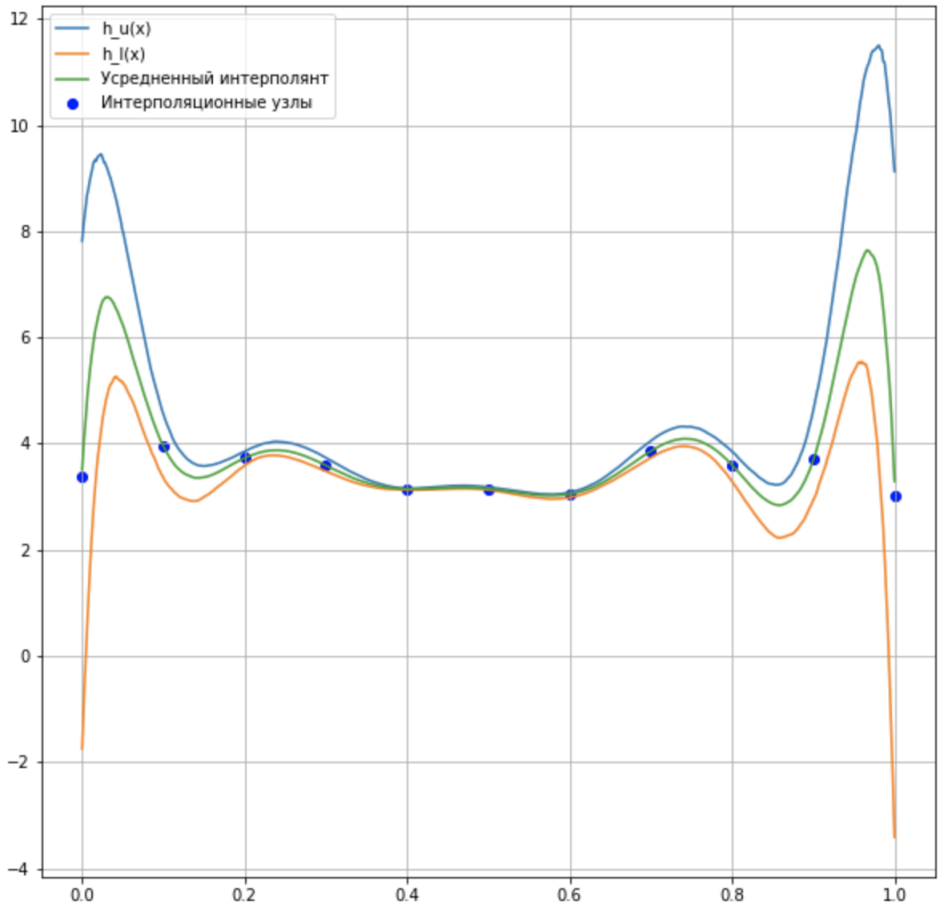

Рис. 6. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт. Интерполяционные узлы отмечены синим.

е. Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям?

Паразитные осцилляции, возникающие при интерполяции полиномом Лагранжа, очень сильно проявляются на концах отрезка и слабо в середине. Проблема такого поведения, заключается в базисных полиномах Лагранжа, именно ближе к концам отрезка интерполирования *"группируются"* их экстремумы. Что, собственно, приводит к сильному проявлению паразитных осцилляций.

4. Анализ влияния погрешности на интерполяцию (ординаты узлов)

а. Генерация 1000 векторов значений $[\tilde{y_1}, ..., \tilde{y_{11}}]^T$

Пункт выполняется аналогично пункту *3а*

```
vecs1 = [[0] * 11 for i in range (0,1000)]
for i in range (0,1000):
  for j in range (0, 11):
    vecs1[i][j] = y_nodes[j] + np.random.normal(0, 0.01)
vecs1 = np.transpose(vecs1)
vecs1 = np.array(vecs1)
```

b. Интерполянты Лагранжа для каждого из векторов

Необходимо для каждого из раннее полученных векторов (координаты абсцисс - нет, координаты ординат с погрешностью) построить интерполянт Лагранжа. В результате необходимо получить 1000 графиков. Решение аналогично пункту *3b*.

```
x_int = np.linspace(0, 1, 100)
interp_lagr = [[L(x_int[i], x_nodes, vecs1[: , j]) for i in range (0, len(x_int))] for j in
    range (0, 1000)]
interp_lagr = np.array(interp_lagr)
plt.subplots(figsize = (10, 10))
for i in range (0, 1000):
  plt.plot(x_int, interp_lagr[i , :])

plt.grid()
```

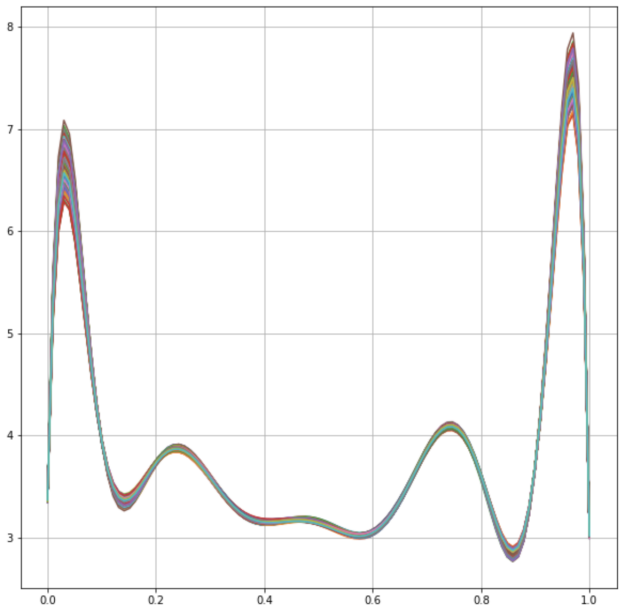

Рис. 7. 1000 интерполянтов Лагранжа, построенных на узлах x_nodes, $[\tilde{y_1}, ..., \tilde{y_{11}}]_i$

c. Построение функций $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ (аналитический вид)

Необходимо построить функции $\tilde{h}_l(x)\tilde{h}_u(x)$, где $\tilde{h}_u(x) < \tilde{h}_u(x)$ 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.
Осуществим это с помощью функции *percentile* модуля *numpy*.
Алгоритм дальнейшего нахождения данных фукнций совпадает с алгоритмом *3c*

```
p_95 = [np.percentile(interp_lagr[:, i], 95) for i in range (0, 100)]
p_5 = [np.percentile(interp_lagr[:, i], 5) for i in range (0, 100)]
```

d. Отображение функций $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненного интерполянта, а также интерполяционных узлов

Для решения данной задачи, необходимо отобразить функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненный интерполянт, и, кроме того, интерполяционные узлы. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ были найдены в прошлом пункте, а усредненный интерполянт находится путем нахождения перцентиля 50. Алгоритм его нахождения аналогичен описанному раннее.

```
mean = [np.percentile(interp_lagr[:, i], 50) for i in range (0, 100)]
plt.plot(x_int, p_95, label = "h_u(x)")
plt.plot(x_int, p_5, label = "h_l(x)")
plt.plot(x_int, mean, label = "Усредненный интерполянт")
plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

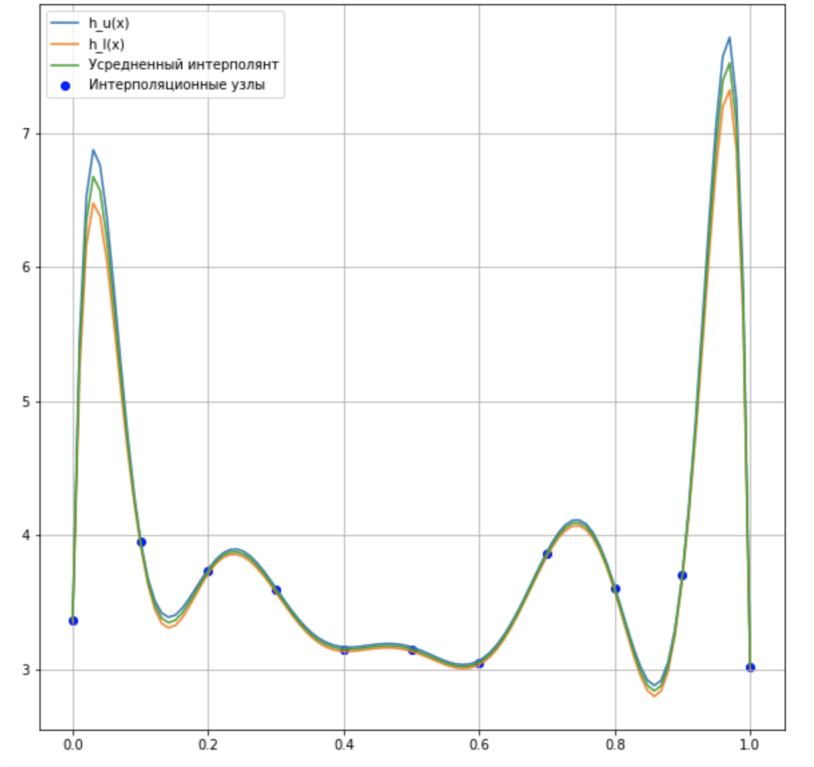

Рис. 8. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт. Интерполяционные узлы отмечены синим.

e. Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям? *Изменились ли выводы вашего анализа?

Аналогично пункту *3e*, наибольшие паразитные осцилляции наблюдаются ближе к концам отрезка интерполяции. Причина этого - аналогична пункту *3e* и заключается в базисных полиномах Лагранжа.
В целом, мои выводы не изменились, однако в данном случае, паразитные осцилляции имеют даже большую видимость, нежели в предыдущем пункте *3e*. Это связано с тем, что значения ординат узлов, на которые умножаются базисные полиномы лагранжа, имееют некоторую погрешность из нормального распределения.

5. Анализ влияния погрешности на интерполяцию (кубический сплайн)

1.a. Генерация 1000 векторов значений $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$

Пункт выполняется аналогично пункту *3a*

```
vecs_x_spl = [[0] * 11 for i in range (0,1000)]
for i in range (0,1000):
  for j in range (0, 11):
    vecs_x_spl[i][j] = x_nodes[j] + np.random.normal(0, 0.01) #x_nodes with errors
vecs_x_spl = np.transpose(vecs_x_spl)
vecs_x_spl = np.array(vecs_x_spl)
```

1.b. Кубические интерполянты для каждого из векторов

Идейно данный пункт совпадает с пунктом *3 b*, однако в данном пункте используется другой способ интерполяции - кубический сплайн. Кроме того, в контексте данной задачи, необходимо модифицировать функцию *qubic_spline(x, qs_coeff)* и добавить дополнительный аргумент - *x_nodes*, т. к. каждый из сплайнов строится на своем отрезке $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$ Кроме того, т. к. потенциально значения $[\tilde{x_1}, ..., \tilde{x_{11}}]^T$ могут располагаться вне отрезка [0;1] или же, не покрывать его полностью, было принято решение в теле функции *qubic_spline(x_nodes, x, qs_coeff)* учесть этот момент и ввести превентивные меры:

```
def qubic_spline(x_nodes, x, qs_coeff):
  x_wk = 0 # if x < x_nodes[0]
  k = 0 # if x < x_nodes[0]
  for i in range (0, len(x_nodes)-1):
      if (x_nodes [i] <= x <= x_nodes[i+1]):
        k = i
        x_wk = x
        break
  if (x>1):
    x_wk = 1 # if x > x_nodes[len(x_nodes) - 1]
    k = len(x_nodes) - 1 # if x > x_nodes[len(x_nodes) - 1]
  S_i = qs_coeff[k][0] + qs_coeff[k][1] * (x_wk - x_nodes[k]) + qs_coeff[k][2] * ((x_wk
      - x_nodes[k])**2) + qs_coeff[k][3] * ((x_wk - x_nodes[k])**3)
  return S_i
```

Алгоритм нахождения значений кубического сплайна в каждой точке:

1. *Для каждого сплайна строится своя матрица коэффициентов, по своему набору узлов абсцисс, узлы ординат - общие для каждого сплайна*

2. *Для каждого сплайна в каждой промежуточной точке отрезка находится свое значение i-го интерполянта*

3. *После нахождения промежуточных значений для каждого слпайна, строится график, где отображаются все 1000 сплайнов на заданном отрезке*

```
plt.subplots(figsize = (10, 10))

x_nod = np.linspace(x_nodes[0], x_nodes[10], 1000)
for i in range (0, 1000):

  matrix = qubic_spline_coeff(vecs_x_spl[: , i], y_nodes)
  S = [qubic_spline(vecs_x_spl[: , i], x_nod[j], matrix) for j in range (0, len(x_nod))]
  plt.plot (x, S)

plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

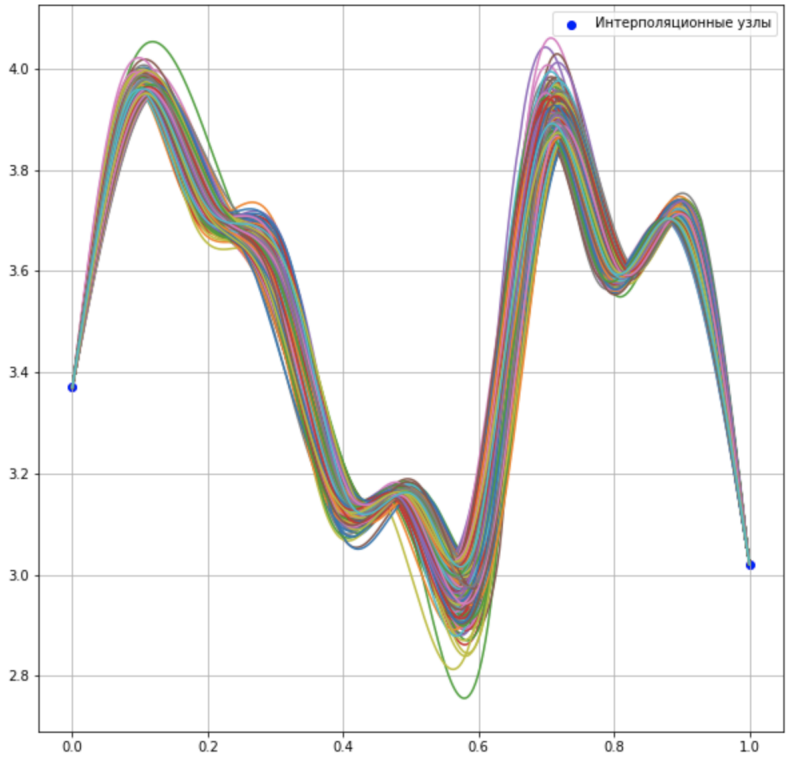

Рис. 9. 1000 графиков для Интерполяции кубическим сплайном, построенных на узлах $[\tilde{x_1}, ..., \tilde{x_{11}}]_i$, y_nodes

1.c. Построение функций $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ (аналитический вид)

Необходимо построить функции $\tilde{h}_l(x)\tilde{h}_u(x)$, где $\tilde{h}_u(x) < \tilde{h}_u(x)$ 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.

Идейно данный пункт схож с пунктом *3.c* (аналогичное использование функции *np.percentile*, однако, т. к. используется другой способ интерполяции, то имеются свои корректировки.
Алгоритм:

1. *Создаем пустую матрицу, размерности 1000 x N, где N - кол-во промежуточных точек*
2. *Для каждого сплайна находим свою матрицу коэффициентов*
3. *Заполняем матрицу значений сплайна в точке для каждой промежуточной точки и сплайна. В результате получится матрица, содержащая значение каждого сплайна в каждой промежуточной точке*
4. *Процесс нахождения необходимых значений перцентилей аналогичен пункту 3.c*

```
x_int = np.linspace(0, 1, 1000)
S_res = np.zeros((1000, len(x_int)))
for i in range (0, 1000):
  matrix = qubic_spline_coeff(vecs_x_spl[: , i], y_nodes)
  for j in range (0, len(x_int)):
    S_res[i, j]= qubic_spline(vecs_x_spl[:, i], x_int[j], matrix)
print(np.shape(S_res))
```

1.d. Отображение функций $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненного интерполянта, а также интерполяционных узлов

Отобразим на едином графике необходимые функции

```
p_95 = [np.percentile(S_res[:, i], 95) for i in range (0, 1000)]
p_5 = [np.percentile(S_res[:, i], 5) for i in range (0, 1000)]
mean = [np.percentile(S_res[:, i], 50) for i in range (0, 1000)]
plt.subplots(figsize = (10, 10))

plt.plot(x_int, p_95)
plt.plot(x_int, p_5)
plt.plot(x_int, mean)
plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

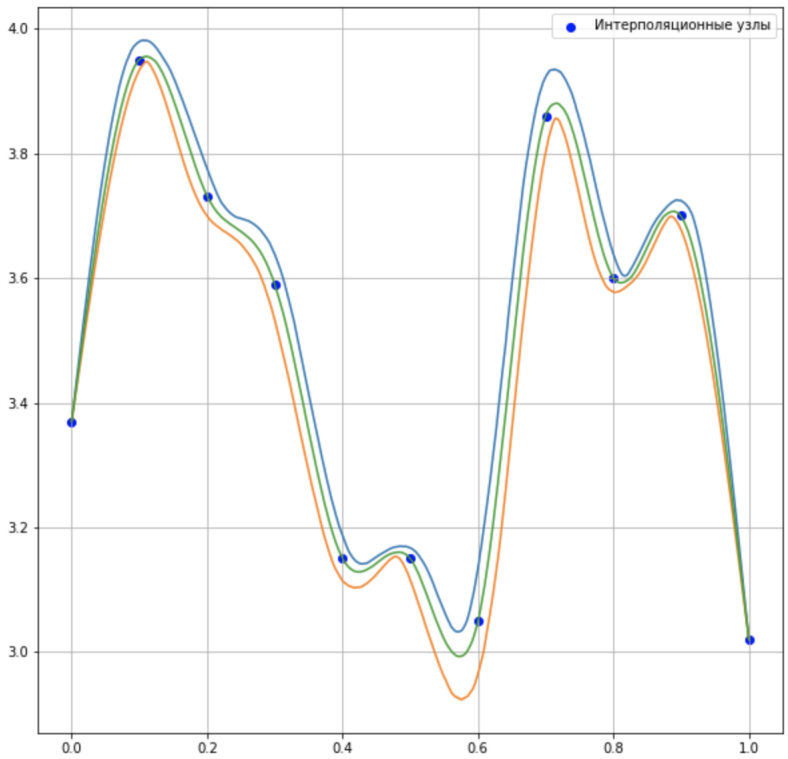

Рис. 10. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт. Интерполяционные узлы отмечены синим.

1.e. Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям? *Изменились ли выводы вашего анализа?

Паразитных осцилляций практически не наблюдается, кроме того, нет тенденции их появления ближе к концам отрезка интерполяции. Причина, почему при интерполяции кубическими сплайнами, паразитных осцилляций меньше чем при интерполяции полиномом Лагранжа, заключается в самом методе аппроксимации - при интерполяции Лагранжа, мы пытаемся аппроксимировать функцию многочленом высокой степени, причем всю функцию сразу. При интерполяции кубическими сплайнами, мы аппроксимируем полиномом заданной степени (не большой, например - в нашем случае: 3) и кроме того, мы аппроксимируем этим полиномом не весь рассматриваемый отрезок, а лишь его часть. То есть вся интерполяция в данном случае получается путем "склеивания"достаточно гладких "кусочков"сплайна, что и приводит к существенно меньшим осцилляциям, ошибкам.

2.a Генерация 1000 векторов значений $[\tilde{y_1}, ..., \tilde{y_{11}}]^T$

Пункт выполняется аналогично пункту *5.1.a*, за тем исключением, что погрешность

добавляется не к абсциссам, а к ординатам узлов.

```
vecs_y_spl = [[0] * 11 for i in range (0,1000)]
for i in range (0,1000):
  for j in range (0, 11):
    vecs_y_spl[i][j] = y_nodes[j] + np.random.normal(0, 0.01) #y_ nodes with errors
vecs_y_spl = np.transpose(vecs_y_spl)
vecs_y_spl = np.array(vecs_y_spl)
print(np.shape(vecs_y_spl)
```

2.b. Кубические интерполянты для каждого из векторов

Пункт аналогичен пункту *5.1.b*.

```
plt.subplots(figsize = (10, 10))

x_nod = np.linspace(x_nodes[0], x_nodes[10], 1000)
for i in range (0, 1000):
  matrix = qubic_spline_coeff(x_nodes, vecs_y_spl[:, i])
  S = [qubic_spline(x_nodes, x_nod[j], matrix) for j in range (0, len(x_nod))]
  plt.plot (x, S)

plt.scatter(x_nodes, y_nodes, color = 'blue', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

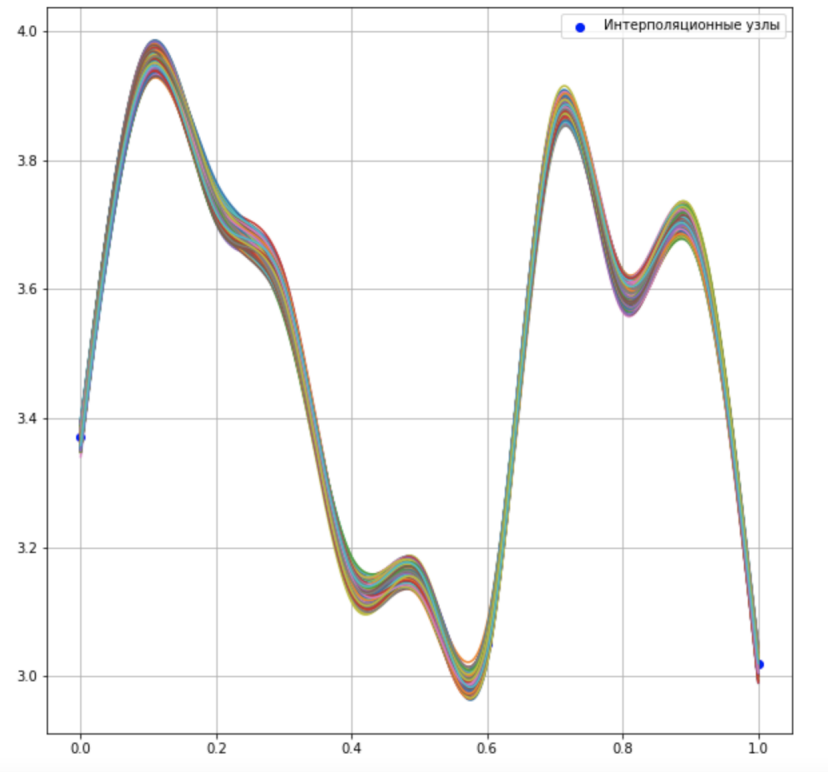

Рис. 11. 1000 графиков для Интерполяции кубическим сплайном, построенных на узлах (x_nodes), $[\tilde{y_1}, ..., \tilde{y_{11}}]^T$

2.c. Построение функций $\tilde{h}_l(x)$ и $\tilde{h}_u(x)$ (аналитический вид)

Пункт аналогичен пункту *5.1.c.*
Отличие лишь в том, что в пункте *5.1.c* абсциссы узлов с погрешностями, а в этом пунтке - ординаты узлов с погрешностями.

```
x_int = np.linspace(0, 1, 1000)
S_res1 = np.zeros((1000, len(x_int)))
for i in range (0, 1000):
  matrix = qubic_spline_coeff(x_nodes, vecs_y_spl[:, i])
  for j in range (0, len(x_int)):
    S_res1[i, j] = qubic_spline(x_nodes, x_int[j], matrix)
```

2.d. Отображение функций $\tilde{h}_l(x)$, $\tilde{h}_u(x)$ и усредненного интерполянта, а также интерполяционных узлов

Отобразим на едином графике необходимые функции

```
p_95 = [np.percentile(S_res1[:, i], 95) for i in range (0, 1000)]
p_5 = [np.percentile(S_res1[:, i], 5) for i in range (0, 1000)]
mean = [np.percentile(S_res1[:, i], 50) for i in range (0, 1000)]
plt.subplots(figsize = (10, 10))

plt.plot(x_int, p_95)
plt.plot(x_int, p_5)
plt.plot(x_int, mean)
plt.scatter(x_nodes, y_nodes, color = 'red', label = 'Интерполяционные узлы')
plt.legend()
plt.grid()
```

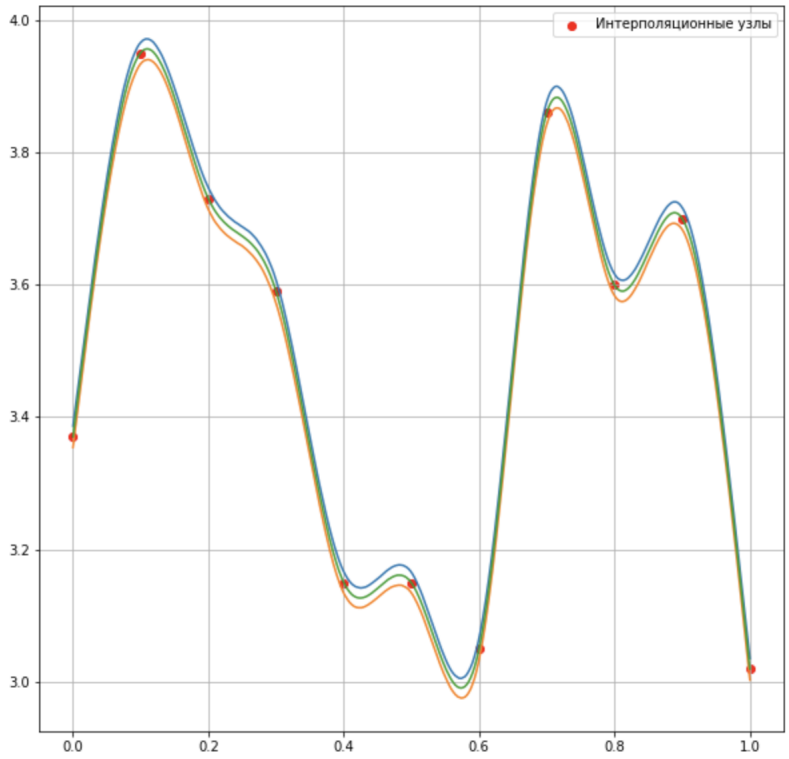

Рис. 12. Функции $\tilde{h}_l(x)$, $\tilde{h}_u(x)$, усредненный интерполянт. Интерполяционные узлы отмечены красным.

2.e. Какие участки интерполянта и почему являются наиболее чувствительными к погрешностям? *Изменились ли выводы вашего анализа?

Выводы аналогично пункту *5.1.e*, кроме того, в данном случае паразитных осцилляций еще меньше, что свидельствует о том, что погрешность ординат узлов

менее существенно влияет на итоговую интерполяцию, чем погрешность абсцисс узлов.

4 Заключение

В ходе данной лабораторной работы было сделано и изучено, а так же проанализировано:

1. *Интерполяция в простейшем случае, применение интерполяции к реальным практическим задачам (ассоциация с вязкой жидкостью)*
2. *Осуществление интерполяции полиномом Лагранжа и кубическими сплайнами*
3. *Сравнение и анализ одного вида интерполяции при погрешностях разных координат узлов (абсцисс, ординат)*
4. *Сравнение и анализ разных видов интерполяции при погрешностях разных координат узлов (абсцисс, ординат)*

В совокупности, данная лабораторная работа позволила лучше проработать материал, связанный с понятиями: аппроксимация, интерполяция, экстраполяция, полином Лагранжа, базисный полином Лагранжа, кубический сплайн, паразитные осцилляции, нормальное распределение, доверительный интервал, перцентиль.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.
2. Першин А. Ю. Видео-лекции по курсу "Вычислительная математика". Москва, 2021 *https://www.youtube.com/channel/UC69GDhPVLY_7IXn3EhmcH2w*
3. [Электронный ресурс] Wikipedia *https://ru.wikipedia.org/wiki/Квантиль*

Выходные данные

Степанов Н. Н.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 26 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 РК6-55Б, Степанов Н. Н.*

2021, осенний семестр