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:	Кутепов Никита Сергеевич
Группа:	РК6-55Б
Тип задания:	лабораторная работа
Тема:	Интерполяция в условиях измерений с неопределенностью

Студент _____ (подпись, дата) Кутепов Н.С. (Фамилия, И.О.)

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

Москва, 2021

Содержание

Интерполяция в условиях измерений с неопределенностью **3**

1 Расчет коэффициентов естественного кубического сплайна. 6

2 Вычисление значения кубического сплайна и его производной в точке x. Построение графика интерполяции кубическим сплайном. 7

3 Вычисление значения i-го базисного полинома Лагранжа. Вычисление значения интерполяционного полинома Лагранжа в заданной точке x. . 9

4 Анализ, позволяющий выявить влияние погрешности на интерполяцию полиномом Лагранжа. 9

5 Анализ из предыдущего пункта, учитывая, что h_i имеет погрешность, а координаты x_i нам известны точно. 13

6 Анализ из предыдущих пунктов для кубического сплайна. 15

7 Заключение. 17

Интерполяция в условиях измерений с неопределенностью

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 жидкости h(x) от координаты x (см. рисунок 11) с помощью кубического сплайна и продемонстрировать ее на графике вместе с исходными узлами.*

Таблица 1. Значения уровня поверхности вязкой жидкости (рис. 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

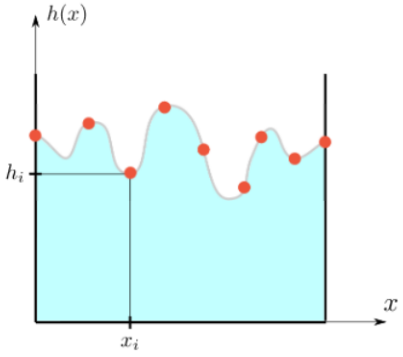

Рис. 1. Поверхность вязкой жидкости (серая кривая), движущейся сквозь некоторую среду (например, пористую). Её значения известны только в нескольких точках (красные узлы).

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 $[\tilde{x}_0, ..., \tilde{x}_{10}]^T$, 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}.*

   (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 сплайном. 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. Интересным методом интерполяции, позволяющим получить именно такие веро-*

ятностные оценки, является регрессия на основе гауссовских процессов, известная также как кригинг. В этом опциональном задании предлагается провести интерполяцию по данным из таблицы 1, используя кригинг.

Цель лабораторной работы.

Цель выполнения лабораторной работы - познакомиться с методами интерполяции кубическими сплайнами и Лагранжа, написать эти алгоритмы на языке программирования Python. Оценить устойчивость интерполяции в условиях измерений с неопределенностью.

Выполненные задачи.

1. Расчет коэффициентов естественного кубического сплайна.
2. Вычисление значения кубического сплайна и его производной в точке x. Построение графика интерполяции кубическим сплайном.
3. Вычисление значения i-го базисного полинома Лагранжа. Вычисление значения интерполяционного полинома Лагранжа в заданной точке x.
4. Анализ, позволяющий выявитьься влияние погрешности на интерполяцию полиномом Лагранжа.
5. Анализ из предыдущего пункта, учитывая, что h_i имеет ту же погрешность, что и в пункте 4, а координаты x_i нам известны точно.
6. Анализ из предыдущих пунктов для кубического сплайна.

1 Расчет коэффициентов естественного кубического сплайна.

Пусть функция f(x) задана в n интерполяционных узлах a = x_1, x_2,..., x_n = b на отрезке [a;b]. Тогда кубическим сплайном для функции f(x) называется функция S(x), для которой верно:

1. S(x) кусочно задана кубическими многочленами S_i(x) на каждом отрезке $[x_i; x_{i+1}]$, i = 1,...,n - 1;
2. S_i(x_i) = f(x_i) и S(x_{i+1}) = f(x_{i+1}), i = 1,...,n - 1;
3. значения смежных многочленов совпадают в общих узлах: $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$, i = 1,...,n - 2;
4. значения первых производных смежных многочленов совпадают в общих узлах: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$, i = 1,...,n - 2;
5. значения вторых производных смежных многочленов совпадают в общих узлах: $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$, i = 1,...,n - 2;
6. заданы граничные условия:
   (a) естественные граничные условия: $S''(x_1) = S''(x_n) = 0$
   (b) граничные условия на касательную: $S'(x_1) = f'(x_1)$ и $S'(x_n) = f'(x_n)$

Чтобы найти коэффициенты естественного кубического сплайна, решим матричное уравнение: $Ac = b$, где $c = [c_1, ..., c_n]^T$

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

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

При этом $h_i = x_{i+1} - x_i a_i = f(x_i)$. Определим вектор c через матричное уравнение, после чего найдем значения векторов b и d:

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

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

Так как по условию нам заданы узлы x_nodes и значения функции в этих узлах y_nodes, вектор a мы считаем известным. Эти 4 параметра позволяют однозначно описать кубический сплайн на заданном интервале. Функция cubic_spline_coeff(x_nodes, y_nodes) создает матрицы A и b, используя функции create_A() и create_b(), которые находятся в директории tools в файле tools.py. После чего, инвертировав матрицу A, находим вектор c:

Листинг 1. Нахождение вектора c

```
1       h = np.array([(x_nodes[i + 1] − x_nodes[i]) for i in range(0, n − 1)])
2       a = np.array(y_nodes)
3
4         A = create_A(x_nodes, h)
5         A_ob = np.linalg.inv(A)
6         b = create_b(n, h, a)
7         c = A_ob @ b
```

Зная вектор c, найдем коэффициенты b и d, после чего сформируем матрицу $3\times(n-1)$ с помощью функции np.c_().

Листинг 2. Формирование матрицы коэффициентов

```
1       b_i = lambda i: (a[i + 1] − a[i]) / h[i] − (h[i] / 3) * (c[i + 1] + 2 * c[i])
2       d_i = lambda i: (c[i + 1] − c[i]) / (3 * h[i])
3
4       b = np.array([b_i(i) for i in range(0, n − 1)])
5       d = np.array([d_i(i) for i in range(0, n − 1)])
6
7       c = np.delete(c, n − 1)
8
9       return np.c_[b, c, d], a
```

2 Вычисление значения кубического сплайна и его производной в точке x. Построение графика интерполяции кубическим сплайном.

Для нахождения значения кубического сплайна в точке x, воспользуемся формулой:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \tag{4}$$

при этом $x \in [x_1; x_n]$. Формула для нахождения производной выглядит следующим образом:

$$S_i'(x) = b_i x + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \tag{5}$$

Так как нам известна матрица параметров, найденная в прошлом пункте, мы можем получить значение сплайна и его производной в заданной точке. Воспользуемся функцией get_i() для определения индекса(интервала, в котором находится заданный x):

Листинг 3. Нахождение индекса

```
def get_i():
    n = len(x_nodes)
    for i in range(0, len(x_nodes) - 1):
        if x >= x_nodes[i] and x <= x_nodes[i + 1]:
            return i
    if x > x_nodes[n - 1]:
        return n - 2
    if x < x_nodes[0]:
        return 0
```

Далее подставляем соответствующие элементы матрицы для нахождения значения функции. Построение сплайна осуществляет функция spline_plot(x_nodes, y_nodes, qs_coeff, interpolate), в которой мы создаем 1000 точек x с помощью функции np.linspace(), после чего находим значения функции в этих точках. Отрисовку графика же осуществляем с помощью пакета matplotlib:

Листинг 4. Отрисовка графика

```
plt.plot(new_x_nodes, new_y_nodes, color='green')
plt.plot(x_nodes, y_nodes, 'o', color='blue')
plt.grid(True)
plt.show()
```

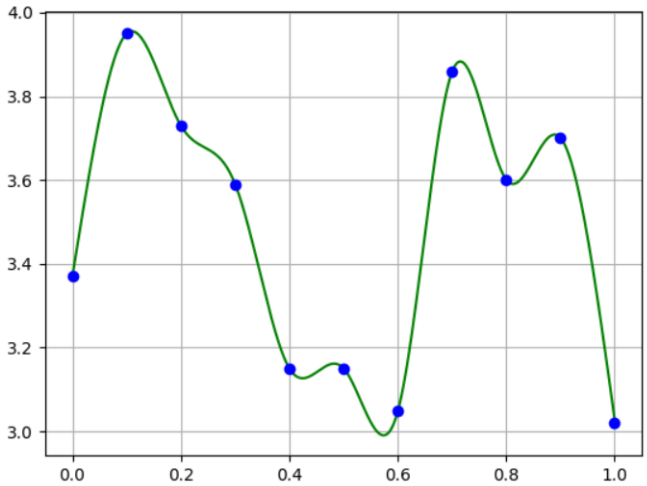

Рис. 2. Интерполяция кубическим сплайном

3 Вычисление значения i-го базисного полинома Лагранжа. Вычисление значения интерполяционного полинома Лагранжа в заданной точке x.

Для вычисления значения i-го базисного полинома Лагранжа воспользуемся формулой:

$$l_i(x) = \prod_{i \neq j} \frac{x - x_j}{x_i - x_j} \tag{6}$$

В функции l_i(i, x, x_nodes) итеративно вычисляем базисный полином. Процедура L_i(x, x_nodes, y_nodes, nothing=None) принимает точку, в которой мы хотим вычислить значения полинома, узлы, значения в этих узлах и параметр nothing (необходим для оптимизации последующих функций, в данном случае он не играет роли). Значение полинома вычисляется как линейная комбинация базисных полиномов и значений функции:

$$L(x) = \sum_{i=0}^{n} l_i(x) f(x_i) \tag{7}$$

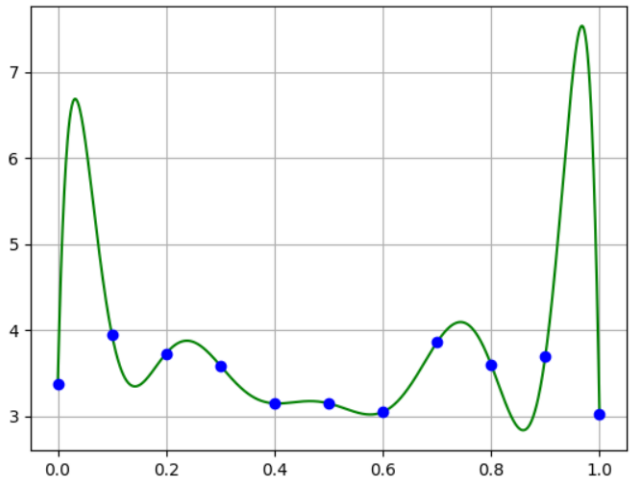

Рис. 3. Интерполяция полиномом Лагранжа

На рис. 3 изображена аппроксимация интерполяционным полиномом Лагранжа, вблизи границ интервала заметны ярко выраженные осцилляции, что объясняется довольно высокой степенью многочлена.

4 Анализ, позволяющий выявить влияние погрешности на интерполяцию полиномом Лагранжа.

Для генерации 1000 векторов значений $[\tilde{x}_0, ..., \tilde{x}_1 0]^T$, учитывая, что $\tilde{x}_i = x_i + Z$, необходимо сгенерировать Z, величину с нормальным распределением и нулевым математи-

ческим ожиданием и стандартным отклонением 10^{-2}. Для этого используем функцию из библиотеки scipy - sps.norm(loc=0, scale=scale).rvs(size=n):

scipy.stats.norm - осуществляет формирует случайную величину с нормальным распределением и заданными параметрами(loc - параметр сдвига, scale - параметр масштаба), rvs же сгенерирует выборку заданного размера size. Весь этот процесс осуществляет функция create_fault_i(f, n, scale), которая принимает вектор величин, которым необходимо задать погрешность, размер выборки и значение отклонения. Данная функция находится в директории tools в файле tools.py.

Листинг 5. Создание вектора значений с погрешностью

```
1  def create_fault_i(f, n, scale):
2      f_i = []
3          Z = sps.norm(loc=0, scale=scale).rvs(size=n)
4
5          for i in range(0, n):
6              f_i.append(f[i] + Z[i])
7
8          return f_i
```

После чего нам необходимо построить интерполянты Лагранжа. Выполним следующий алгоритм:

1. Создание 1000 точек для построения с помощью функции np.linspace(0, 1, current_points)
2. Генерация вектора значений x с погрешностью для каждого вектора
3. Вычисление значения полинома Лагранжа, учитывая новые параметры при создании многочлена
4. На данном этапе мы возвращаем полученные значения x_plot и y_plot для построения 1000 векторов

Данный процесс осуществляется в функции inaccuracy_x(x_nodes, y_nodes, interpolate, scale, vec_count, current_points), которая учитывает метод интерполирования(принимает параметр interpolate):

Листинг 6. Функция для создания точек, по которым будет строиться график

```
1  def inaccuracy_x(x_nodes, y_nodes, interpolate, scale, vec_count, current_points):
2      plot_x_nodes = np.linspace(0, 1, current_points)
3      plot_y_nodes = []
4      qs_coeff = []
5
6      for i in range(0, vec_count):
7          x_i = create_fault_i(x_nodes, len(x_nodes), scale)
8
9          if interpolate == qubic_spline:
```

```
10              qs_coeff, a = qubic_spline_coeff(x_i, y_nodes)
11
12          plot_y_nodes.append(get_new_y_nodes(x_i, plot_x_nodes, current_points,
                y_nodes, interpolate, qs_coeff))
13
14      return plot_x_nodes, plot_y_nodes
```

Построение графиков осуществляет функция plot_thousand_splines(plot_x, plot_y, vec_count):

Листинг 7. Функция построения 1000 графиков

```
1 def plot_thousand_splines(plot_x, plot_y, vec_count):
2     for i in range(0, vec_count):
3         plt.plot(plot_x, plot_y[i])
4
5     plt.grid(True)
6     plt.show()
```

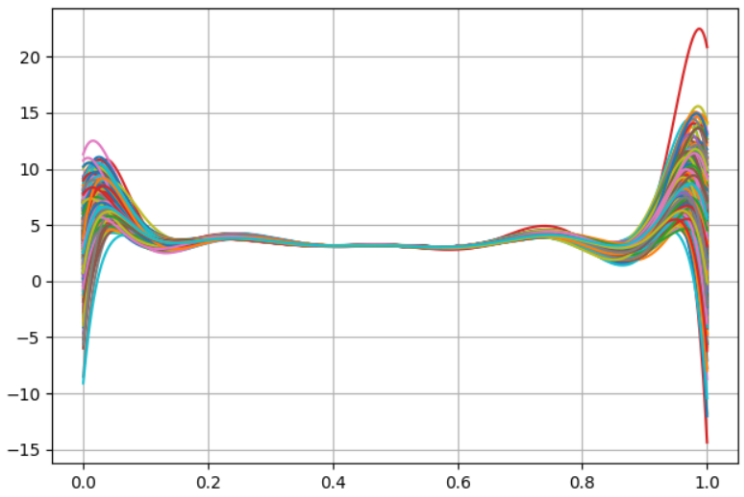

Рис. 4. 1000 графиков интерполяции полиномом Лагранжа для случая погрешности в x

Далее предположим, что все интерполянты представляют собой равновероятные события, рассмотрим функции $\tilde{h}_l(x)\tilde{h}_u(x)$, где $\tilde{h}_l(x) < \tilde{h}_u(x)$ для любого $x \in [0;1]$, при этом вероятность того, что значение интерполянта в точке x будет лежать в интервале $[\tilde{h}_l(x);\tilde{h}_u(x)]$ равна 0.9. Рассчитаем доверительный интервал с помощью функции confidence_interval(interval), которая принимает вертикаль значений функции в каждой из точек построения, соответственно вызываем эту функцию итеративно для

каждой точки, полученной с помощью np.linspace(). Далее функция сортирует полученный вектор, берет медиану и два значения, которые образуют доверительный интервал, при этом 900 из 1000 точек лежат в нем:

Листинг 8. Функция нахождения медианы и доверительного интервала

```
def confidence_interval(interval):
    interval = sorted(interval)
    av_num = interval[499]
    dov = []
    dov.append(interval[49])
    dov.append(interval[948])

    return dov[0], av_num, dov[1]
```

Осуществим построение графика с помощью функции plot_stat(x, y, current_points, vec_counts), которая формирует 3 вектора значений функции в точках построения, после чего осуществляется отрисовка графиков:

Листинг 9. Функция построения графика с доверительным интервалом

```
def plot_stat(x, y, current_points, vec_counts):
    new_y_1 = []
    new_y_2= []
    new_y_3 = []

    for i in range(0, current_points):
        _y = []
        for j in range(0, vec_counts):
            _y.append(y[j][i])

        y_1, y_2, y_3 = confidence_interval(_y)
        new_y_1.append(y_1)
        new_y_2.append(y_2)
        new_y_3.append(y_3)

    plt.plot(x, new_y_1, ':', color='red')
    plt.plot(x, new_y_2, color='green')
    plt.plot(x, new_y_3, ':', color='red')
    plt.fill_between(x, new_y_1, new_y_3, color='yellow')
    plt.grid(True)
    plt.show()
```

Заметим, что именно концы интервала являются наиболее чувствительным, что объясняется феноменом Рунге, т. е. с возрастанием степени полинома погрешность интерполяции стремится к бесконечности. Такой эффект роста уклонения при росте степени

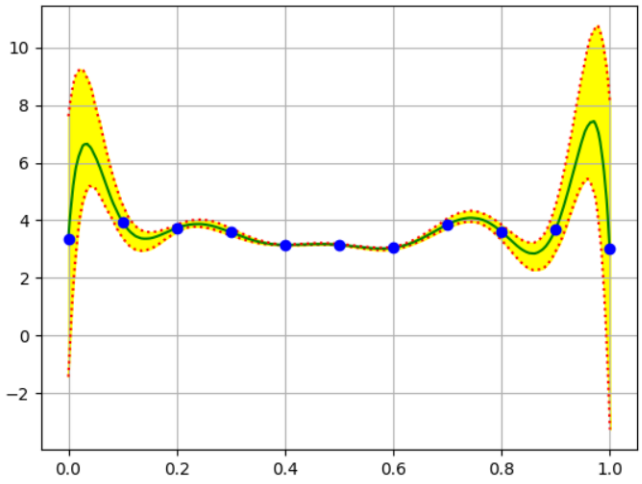

Рис. 5. График усредненного значения(обозначена зеленой линией) и доверительного интервала(область, закрашенная желтым цветом) для случая погрешности в x

многочлена зависит как от выбираемой последовательности узлов, так и от интерполируемой функции. Соответственно, для любой последовательности узлов можно подобрать такую непрерывную функцию, что ошибка ее интерполяции по этим конкретным узлам будет неограниченно расти. Компромиссом можно считать узлы Чебышёва, погрешность интерполяции по ним равномерно убывает для любой абсолютно непрерывной функции. Именно это объясняет сильные осцилляции при небольшом смещении одной из точек.

5 Анализ из предыдущего пункта, учитывая, что h_i имеет погрешность, а координаты x_i нам известны точно.

Аналогично предыдущему пункту воспользуемся функциями: inaccuracy_y(x_nodes, y_nodes, interpolate, scale, vec_count, current_points), plot_thousand_splines(plot_x, plot_y, vec_count), get_new_y_nodes(x, new_x_nodes, current_points, y_nodes, interpolate, qs_coeff), confidence_interval(interval), plot_stat(x, y, current_points, vec_counts). Единственное отличие заключается в функции inaccuracy_y(x_nodes, y_nodes, interpolate, scale, vec_count, current_points), так как мы генерируем векторы y_i, а не x_i как в прошлом пункте:

Листинг 10. Единственное отличие кода от предыдущего пункта

```
1    for i in range(0, vec_count):
2        y_i = create_fault_i(y_nodes, len(y_nodes), scale)
3
4        if interpolate == qubic_spline:
5            qs_coeff, a = qubic_spline_coeff(x_nodes, y_i)
6
```

```
7        plot_y_nodes.append(get_new_y_nodes(x_nodes, plot_x_nodes, current_points,
             y_i, interpolate, qs_coeff))
```

Кажется, что осцилляции стали менее ярким, но скорее всего дело именно в масштабе графика. Рассмотрим полученные графики:

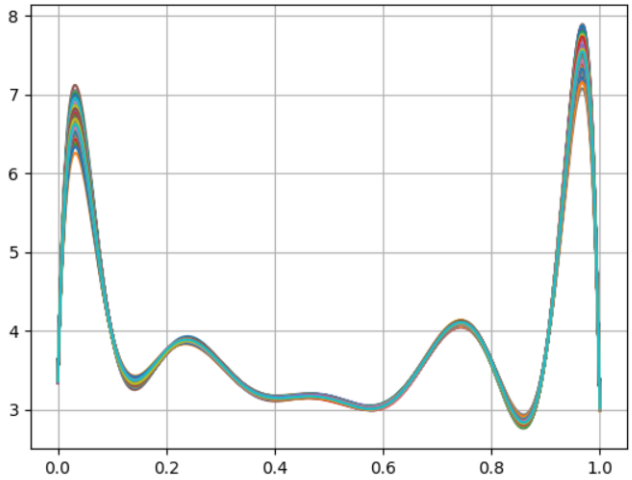

Рис. 6. 1000 графиков интерполяции полиномом Лагранжа для случая погрешности в y

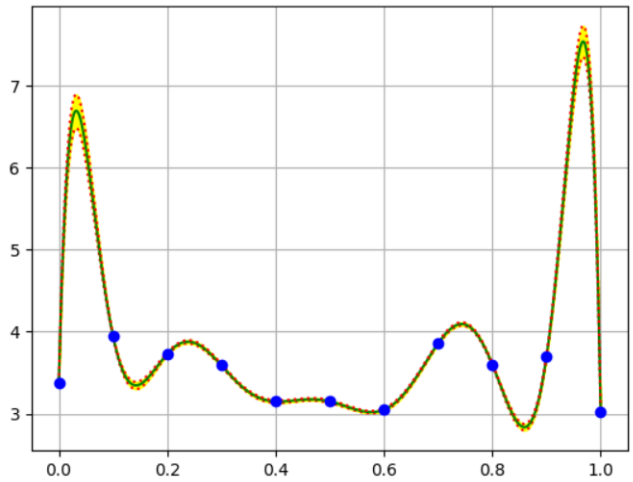

Рис. 7. График усредненного значения(обозначена зеленой линией) и доверительного интервала(область, закрашенная желтым цветом) для случая погрешности в y

Можно заметить, что доверительный интервал стал заметно уже на краях и на всех участках интервала в целом, соответственно результат интерполяции не так чувствителен к значениям ординат, так как небольшое изменение абсциссы имеет высокую

степень при вычислении базисного полинома, а погрешность в ординате всегда остается в первой степени, что приводит к меньшему доверительному интервалу.

6 Анализ из предыдущих пунктов для кубического сплайна.

Проведем все вычисления и построения для случая с естественным кубическим сплайном. Заметим, что интерполяция оказалась гораздо точнее, так как весь интервал мы разбиваем на участки, при этом полином имеет 3 степень, что значительно снижает вероятность появления паразитных осцилляций на краях. Стоит отметить, что небольшая погрешность абсциссы не так сильно влияет на итог интерполяции, то же можно заметить и в случае с погрешностью в ординате, из этого можно сделать вывод, что данный способ интерполяции подходит, если измерения имеют некую погрешность.

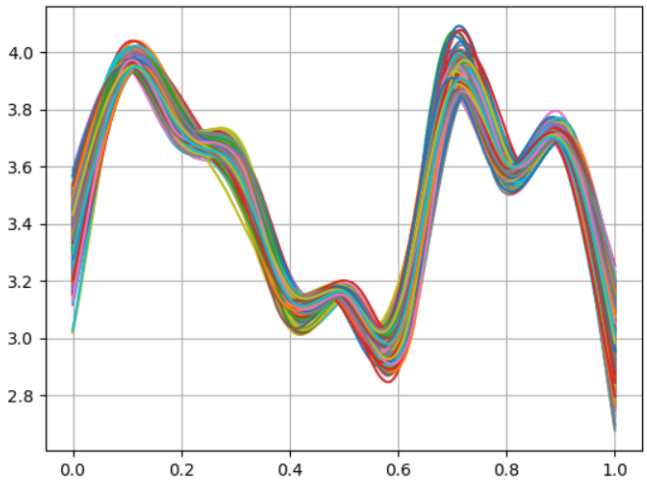

Рис. 8. 1000 графиков интерполяции кубическим сплайном для случая погрешности в x

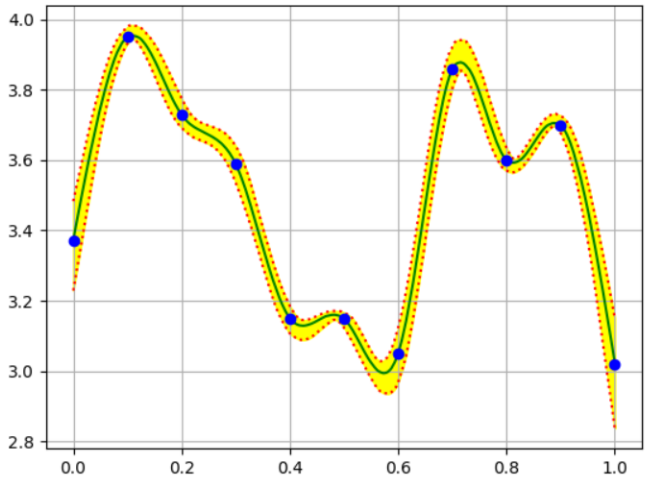

Рис. 9. График усредненного значения(обозначена зеленой линией) и доверительного интервала(область, закрашенная желтым цветом) для случая погрешности в x

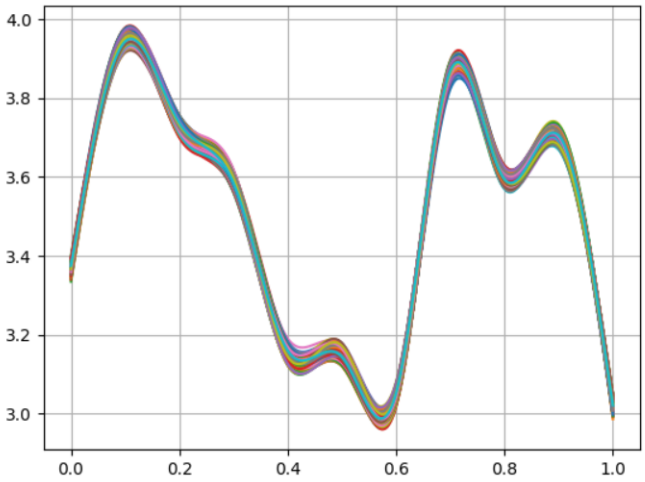

Рис. 10. 1000 графиков интерполяции кубическим сплайном для случая погрешности в y

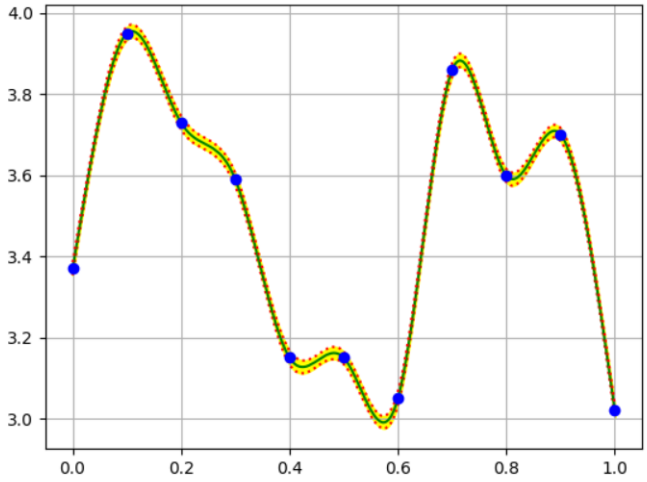

Рис. 11. График усредненного значения(обозначена зеленой линией) и доверительного интервала(область, закрашенная желтым цветом) для случая погрешности в y

7 Заключение.

1. Погрешности в абсциссах имеют большее влияние на итог интерполяции, чем та же погрешность в ординатах. Для полинома Лагранжа это особенно заметно на краях интервала.

2. Интерполяция кубическим сплайном является более точной, чем аппроксимация полиномом Лагранжа.

3. Для небольшого числа узлов логичнее использовать интерполяцию полиномом Лагранжа, так как она менее затратная по времени(хотя в данной работе это не так заметно, ведь операции с матрицами мы осуществляем через функции numpy), но в случае с большим количеством узлов стоит перейти на интерполяцию кубическими сплайнами, это спасет нас от паразитных осцилляций на краях интервала.

4. Если у наших измерений есть некая погрешность стоит задуматься об интерполяции кубическим сплайном, так как изменения в итоговых значениях будут не так ощутимо расходиться.

Список использованных источников.

1. Першин А. Ю. "Лекции по вычислительной математике". Кафедра РК6 МГТУ им. Н. Э. Баумана, 2020

2. Майкл Доусон "Программируем на Python". 2014

3. Феномен Рунге "https://ru.abcdef.wiki/wiki/Runge27s_phenomenon"

Выходные данные

Кутепов Н.С.. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2021. — 18 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, Кутепов Н.С.*

2021, осенний семестр