Интерполяция и аппроксимация таблично заданной функции

Валеев Арслан Рустамович

26 Декабря 2022 г

1. Математическая постановка задачи

Известна таблично заданная функция $(x_i, f_i),\ \ i = \overline{1, n}$. Требуется построить интерполяционный многочлен и аппроксимировать данную функцию многочленом степени m. Будем называть множество $\{x_i\},\ \ i = \overline{1, n}$. узлами функции.

2. Описание алгоритмов

Далее будут описаны два алгоритма:

1. Построение интерполяционного многочлена в форме Лагранжа
2. Аппроксимация функции многочленом заданной степени

В следующей главе будет проведено подробное описание данных алгоритмов.

2.a Интерполирование многочленом в форме Лагранжа

Интерполяционный многочлен в форме Лагранжа строится по следующей формуле:

$$L_n(x) = \sum_{i=1}^{n} f_i \cdot Q_{n,i}(x)$$

В свою очередь $Q_{n,i}(x)$ имеет вид:

$$Q_{n,i}(x) = \prod_{j=1, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Можно заметить, что $Q_{n,i}(x)$ на узлах функции принимает значения:

$$Q_{n,i}(x_j) = \begin{cases} 1, j = i \\ 0, j \neq i \end{cases}$$

Значит:

$$L_n(x_i) = f_i, i = \overline{1, n}.$$

Для вычисления значения $L_n(x)$ в точке x необходимо произвести $n \cdot (n-1)$ операций. Следовательно, сложность данного алгоритма $\underline{O}(n^2)$.

Теорема 1. *Теорема о сходимости:*

Так как $x_1 < x_2 < x_3 \ldots < x_n$, то можно гарантировать, что $\deg(L_n) = n$. Значит, полученный многочлен может иметь довольно высокую амплитуду. Поэтому, используя этот метод восстановления исходной функции нужно иметь в виду, что между узлами функции интерполяционный многочлен может принимать неожиданно большие значения.

2.b Приближение функции многочленом заданной степени

Рассмотрим многочлен степени m, зависящий от коэффициентов $\{c_i\}, i = \overline{0, m}$:

$$G(x) = \sum_{i=0}^{m} c_i \cdot x^i$$

Требуется подобрать такие c_i , что:

$$\sum_{i=1}^{n} (f_i - G(x_i))^2 \to min$$

То есть, найти минимум функции:

$$L(c_i) = Loss(c_0, c_1, c_2, \dots, c_m) = \sum_{i=1}^{n} \left(f_i - \left(\sum_{k=0}^{m} c_k \cdot x_i^k \right) \right)^2$$

Из курса математического анализа мы знаем, что необходимое условие наличия экстремума у функции выглядит следующим образом:

$$\frac{\partial L}{\partial c_i} = 0 , i = \overline{0, m}$$

т. е.

$$\frac{\partial L}{\partial c_j} = -2 \cdot \sum_{i=1}^{n} \left(f_i - \left(\sum_{k=0}^{m} c_k \cdot x_i^k \right) \right) \cdot x_i^j = 0$$

Это условие можно переписать следующим образом:

$$\sum_{\mathrm{k}=0}^{\mathrm{m}} \mathrm{a}_{\mathrm{jk}} \cdot \mathrm{c}_{\mathrm{k}} = \mathrm{b}_{\mathrm{j}} \ , \ \ \mathrm{j} = \overline{0, \mathrm{m}}$$

Или в виде системы:

$$\begin{cases} \mathrm{a}_{00}\mathrm{c}_0 + \mathrm{a}_{01}\mathrm{c}_1 + \ \dots + \mathrm{a}_{0\mathrm{m}}\mathrm{c}_\mathrm{m} = \mathrm{b}_0 \\ \mathrm{a}_{10}\mathrm{c}_0 + \mathrm{a}_{11}\mathrm{c}_1 + \ \dots + \mathrm{a}_{1\mathrm{m}}\mathrm{c}_\mathrm{m} = \mathrm{b}_1 \\ \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \\ \mathrm{a}_{m0}\mathrm{c}_0 + \mathrm{a}_{m1}\mathrm{c}_1 + \ \dots + \mathrm{a}_{m\mathrm{m}}\mathrm{c}_\mathrm{m} = \mathrm{b}_m \end{cases}$$

$$A\bar{c} = b$$

где:

$$a_{jk} = \sum_{i=0}^{n} x_i^k \cdot x_i^j; \quad b_j = \sum_{i=0}^{n} f_i \cdot x_i^j \qquad j = \overline{0, m}$$

На практике данное условие для метода наименьших квадратов также является и достаточным.

Можно заметить, что матрица A имеет вид:

$$A = C^T * C \tag{1}$$

где:

$$C = \begin{pmatrix} x_1^0 & \cdots & x_1^m \\ \vdots & \ddots & \vdots \\ x_n^0 & \cdots & x_n^m \end{pmatrix}$$

Из курса линейной алгебры мы знаем, что матрица (1) обратима для любой матрицы C

Для решения системы линейных алгебраических уравнений использовался метод Гаусса.

Неравенство $[C^T * C]_{i,i} = \sum_{k=1}^{m+1} [C]_{k,i}^2 \geq [C]_{1,i}^2 = 1$ гарантирует корректность работы алгоритма без перестановки строк.

Система линейных алгебраических уравнений является плохо обусловленной, так как $\|A\| \cdot \|A^{-1}\| > 3 \cdot 10^5 \gg 1$. Следовательно, далее будет необходимо исследовать устойчивость коэффициентов при малых изменениях узлов функции x_i.

3 Программная реализация

3.a Интерполяционный многочлен в форме Лагранжа

Для интерполяции многочленом я написал две функции:

```
def Q(grid: List[float], i: int) -> Callable[[float], float]: # возвращает функцию Q_n,i
    def q(x: float) -> float:
        res = 1
        for j, node in enumerate(grid):
            if j != i:
                res *= (x - node)/(grid[i] - node)
        return res
    return q
```

```
def interpolate_poly(x_all: List[float], y: List[float]) -> Callable[[float], float]: # Возвращает L_n
    all_func_q = [Q(x_all, i) for i in range(len(x_all))]

    def inter_func(x: float) -> float:
        res = 0
        for i in range(len(x_all)):
            res += y[i] * all_func_q[i](x)
        return res
    return inter_func
```

Первая функция принимает $\{x_j\}, j = \overline{1, n}$; i и возвращает функцию $Q_{n,i}(x)$.

Вторая принимает сетку (x_i, f_i) и возвращает функцию $L_n(x)$

3.a Аппроксимация методом наименьших квадратов

Для приближения понадобилось три функции:

mse_built_matrix – строит матрицу A и вектор значений f, такие, что если x – решение уравнения $Ax = f$, то x – нужный набор коэффициентов для приближающей функции

gauss – по полученным A, f находит решение уравнения $Ax = f$

approx – принимает на вход сетку (x_i, f_i) и m – степень аппроксимирующего многочлена, и используя две вышеописанные функции возвращает функцию $G(x)$.

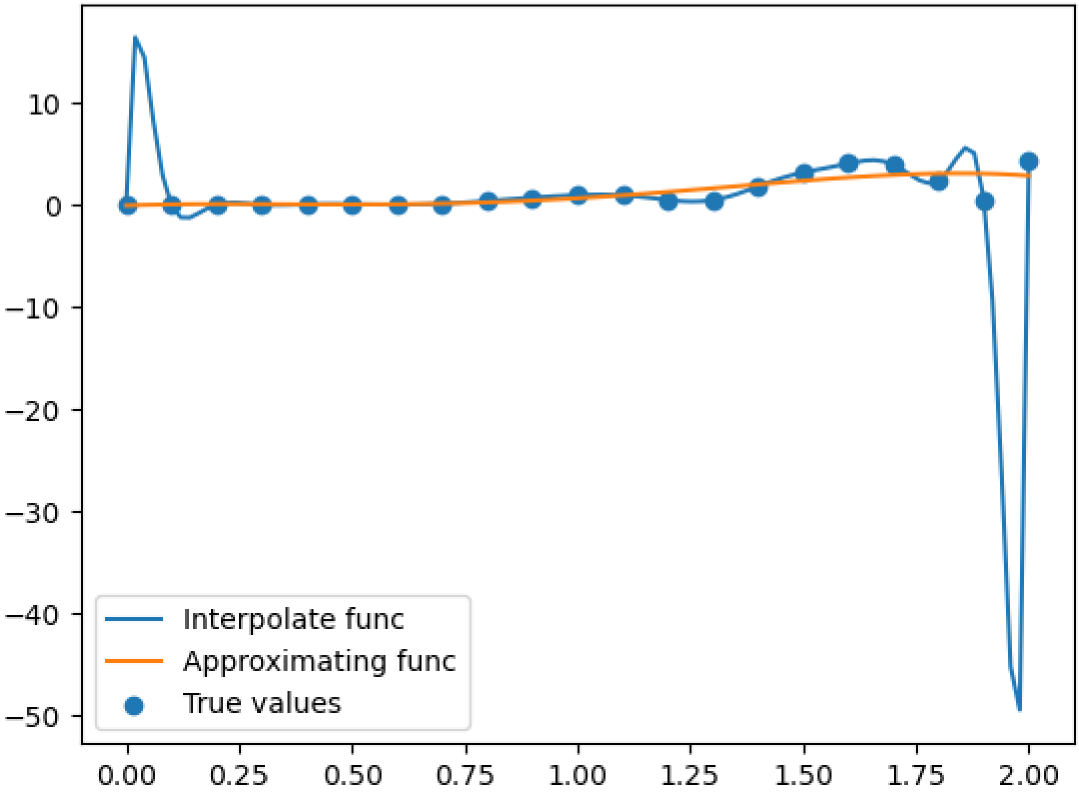
```
def mse_built_matrix(x_all: List[float], y: List[float], n: int) ->
Tuple[List[List[float]], List[float]]:
    tmp_matrix = []
    for i in range(n + 1):
        tmp_matrix.append([x ** i for x in x_all])
    A = []
    for i in range(n + 1):
        A.append([0] * (n + 1))
    for i in range(n + 1):  # Строим матрицу A для решения СЛАУ
        for j in range(n + 1):
            A[i][j] = sum([tmp_matrix[i][t] * tmp_matrix[j][t] for t in
range(len(x_all))])
    f = [sum([y[j] * tmp_matrix[i][j] for j in range(len(x_all))]) for i in
range(n + 1)]
    return A, f
```

```
def gauss(A: List[List[float]], f: List[float]) -> List[float]:
    n = len(A)
    for i in range(n - 1):
        for j in range(i + 1, n):
            f[j] -= f[i] * (A[j][i] / A[i][i])
            for k in range(i + 1, n):
                A[j][k] -= A[i][k] * (A[j][i] / A[i][i])
            A[j][i] = 0
    x = [0] * n
    for i in reversed(range(n)):
        x[i] = (f[i] - sum(A[i][j] * x[j] for j in range(i + 1, n))) /
A[i][i]
    return x
```

```
def approx(x_all: List[float], y: List[float], n: int) -> Callable[[float],
float]:
    A, f = mse_built_matrix(x_all, y, n)
    coefficients = gauss(A, f)

    def approx_func(x: float) -> float:
        return sum([coefficients[i] * (x ** i) for i in range(n + 1)])
    return approx_func
```

4 Анализ полученных результатов



Графики $(x_i, f_i), L_n(x), G(x)$

Для начала поймём, насколько полученные функции отклоняются от заданных значений. $L_n(x)$ по построению проходит через все заданные точки. А для приближающей функции вычислим среднеквадратичное отклонение:

$$\frac{\sum_{i=1}^{n}(G(x_i) - f_i)^2}{n} \approx 0.45$$

Далее проверим полученное методом Гаусса решение системы алгебраических уравнений:

$$\|A\bar{x} - f\| \approx 8 \cdot 10^{-14}$$

где $\bar{x}$ – полученное решение.

Результаты показывают, что метод наименьших квадратов и метод Гаусса работают хорошо.

Далее проверим функции на устойчивость к малым изменениям сетки.

4.a Анализ устойчивости к изменению x_i

Как было замечено ранее, решение СЛАУ $Ax = f$ может оказаться неустойчивым из-за плохой обусловленности матрицы. Требуется исследовать поведение интерполяционного и аппроксимирующего многочленов при малых изменениях сетки.

Пусть $d = (d_1, d_2, \dots, d_n) - n$ – мерный нормально распределённый случайный вектор с математическим ожиданием = 0 и дисперсией = 1. $F_{d,s}(x), L_{d,s}(x)$ – приближающий и интерполяционный многочлены таблично заданной функции $(x_i + d_i \cdot s, f_i)$ соответственно (s – некое число > 0). Тогда обозначим:

$$\Delta F_s(d) = \mathbb{E}\sqrt{\int_0^2 \left(F_{d,s}(x) - G(x)\right)^2 dx}$$

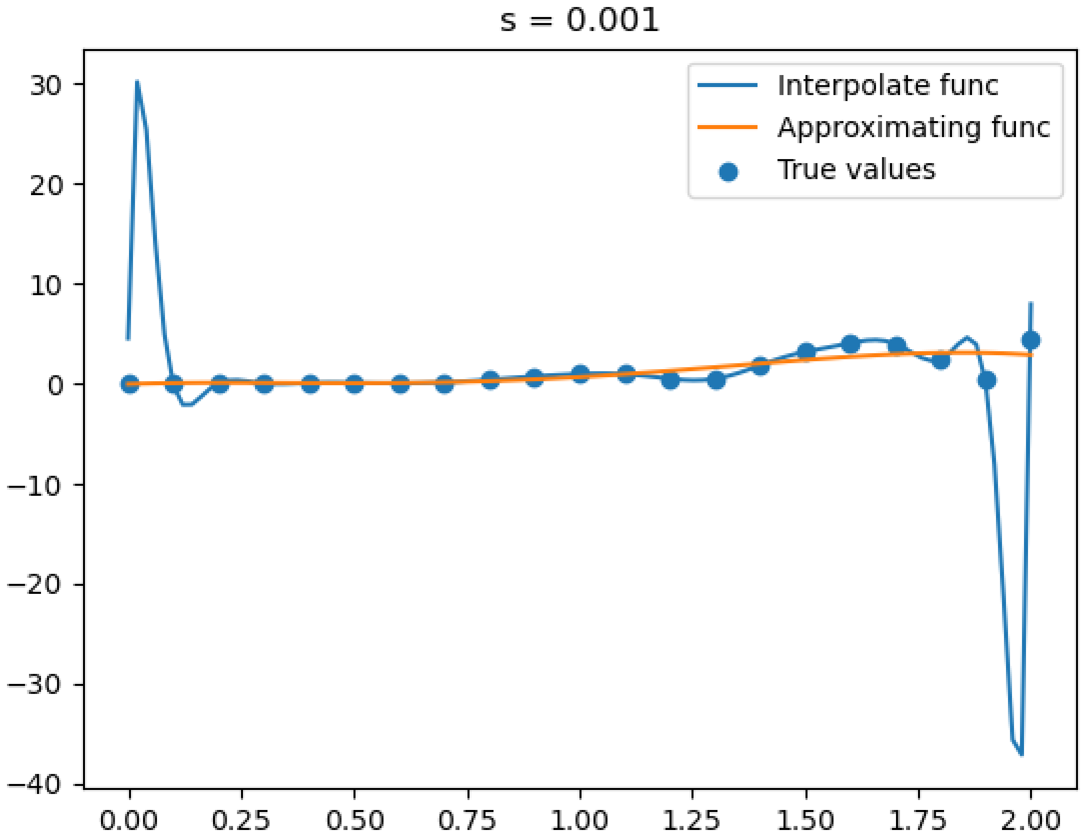
$$\Delta L_s(d) = \mathbb{E}\sqrt{\int_0^2 \left(L_{d,s}(x) - G(x)\right)^2 dx}$$

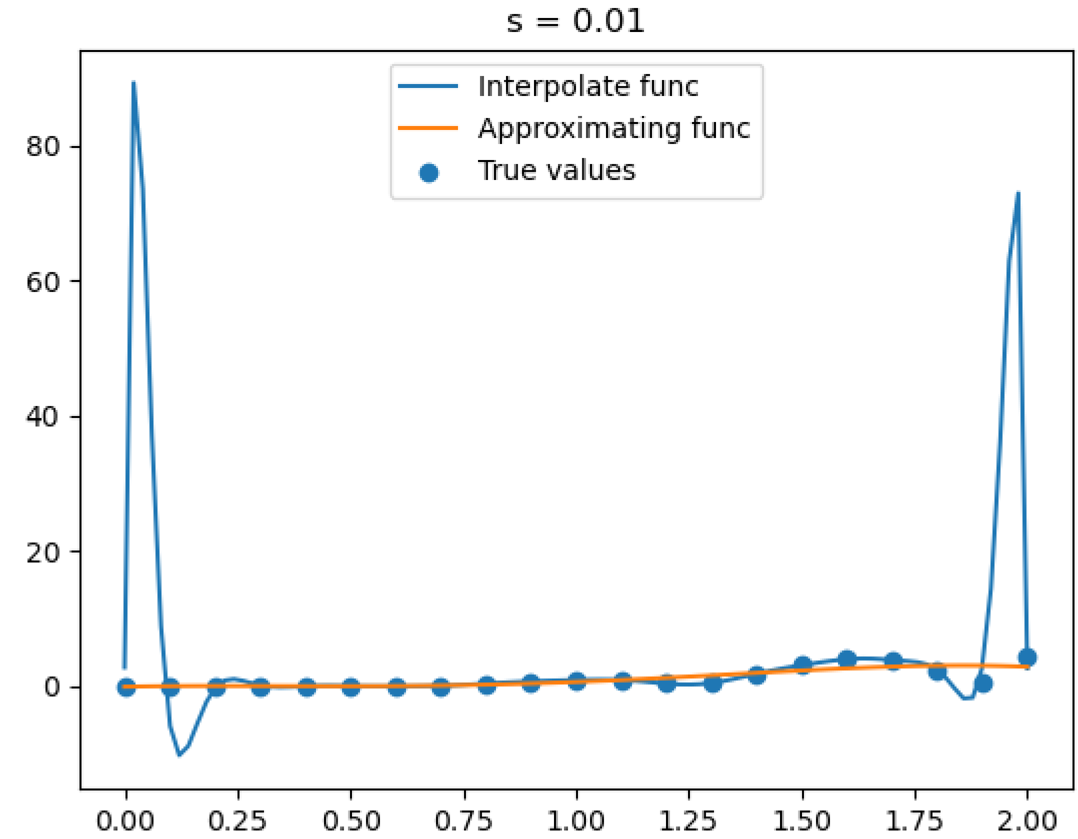
Рассмотрим несколько графиков и $\Delta F_s, \Delta L_s$ при разных значениях s:

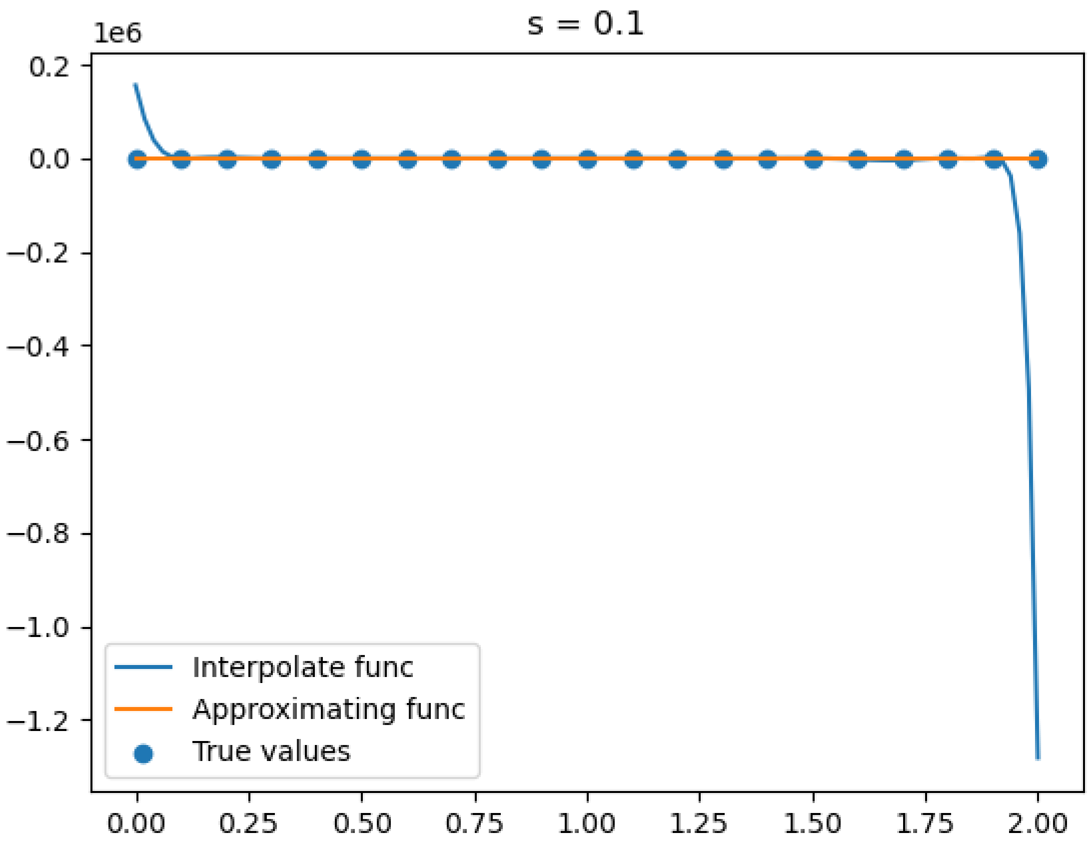
$s = 10^{-3}$: $\Delta F_s \approx 3 \cdot 10^{-3}$, $\Delta L_s \approx 4$

$s = 10^{-2}$: $\Delta F_s \approx 1 \cdot 10^{-2}$, $\Delta L_s \approx 30$

$s = 10^{-1}$: $\Delta F_s \approx 3 \cdot 10^{-1}$, $\Delta L_s \approx 1{,}2 \cdot 10^5$




s = 0.01
Interpolate func
Approximating func
True values
80
60
40
20
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


s = 0.1
1e6
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
Interpolate func
Approximating func
True values
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Из полученных результатов следует, что $\Delta L_s \gg \Delta F_s$ при $\forall$ s. Так как приближающий многочлен устойчив к изменениям в сетке, то метод Гаусса на данной матрице выдаёт приемлемый результат.

4.b Анализ устойчивости к изменению $\boldsymbol{f_i}$

Аналогично 4.a, введём $\Delta F_s, \Delta L_s$ для сетки $(x_i, f_i + d_i \cdot s)$:

$$\Delta F_s(d) = \mathbb{E}\sqrt{\int_0^2 \left(F_{d,s}(x) - G(x)\right)^2 dx}$$

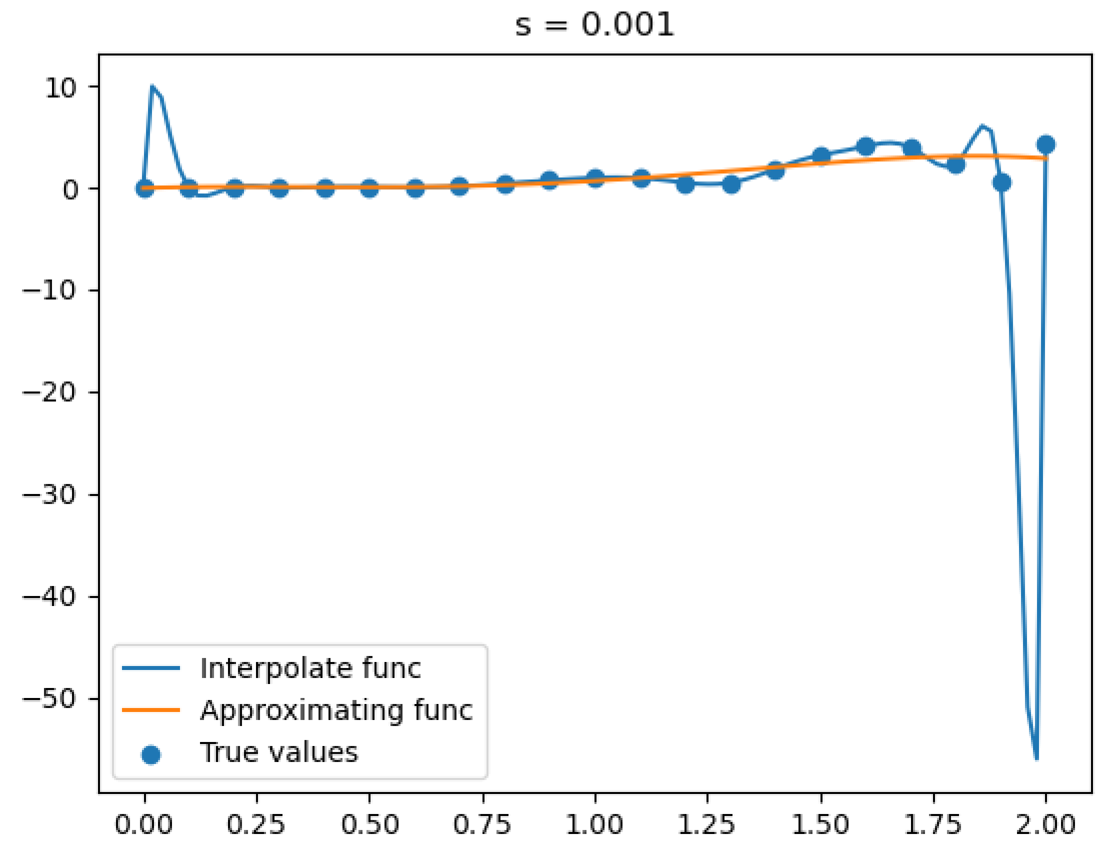
$$\Delta L_s(d) = \mathbb{E}\sqrt{\int_0^2 \left(L_{d,s}(x) - G(x)\right)^2 dx}$$

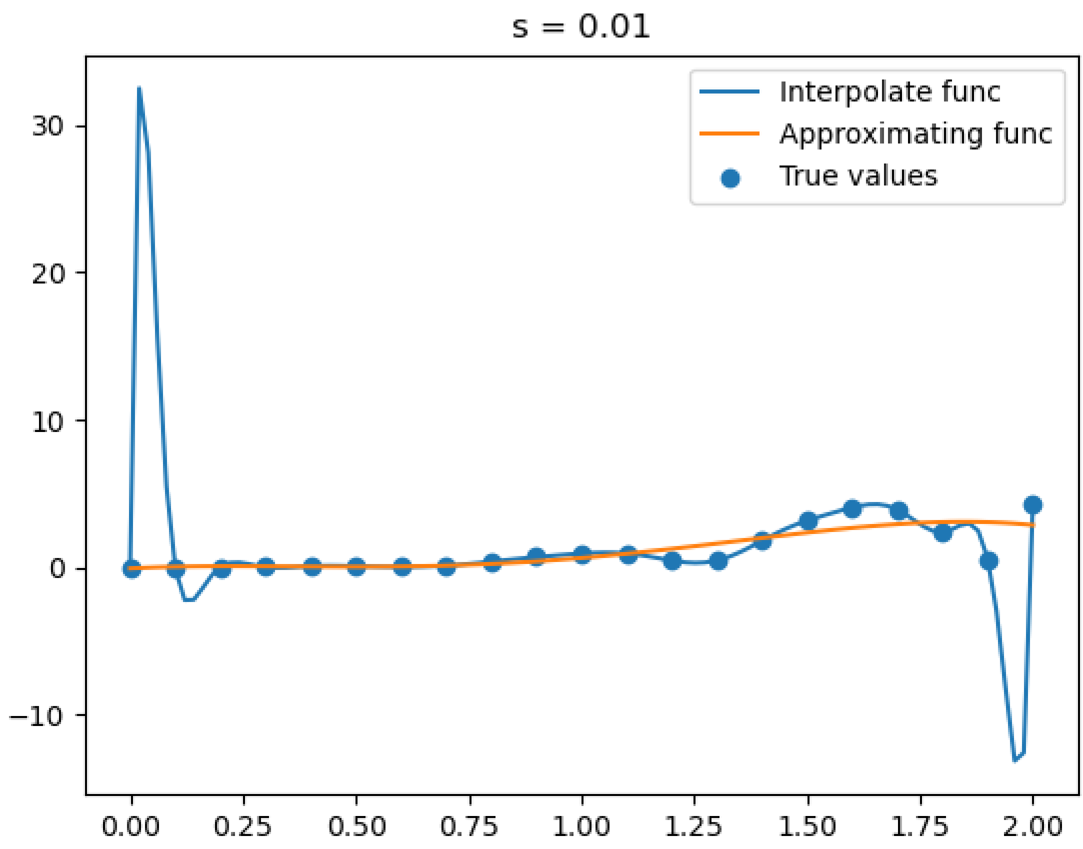
Рассмотрим несколько графиков и $\Delta F_s, \Delta L_s$ при разных значениях s:

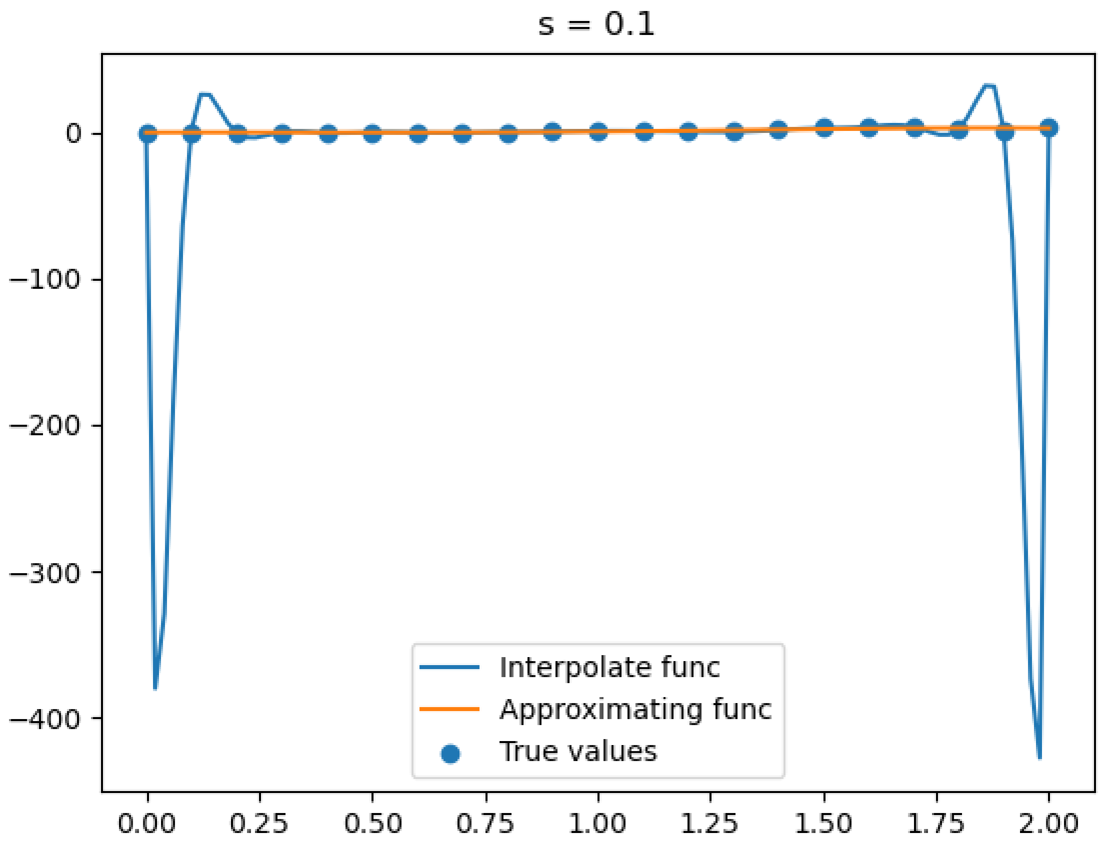
$s = 10^{-3}$: $\Delta F_s \approx 6 \cdot 10^{-4}$, $\Delta L_s \approx 2$

$s = 10^{-2}$: $\Delta F_s \approx 9 \cdot 10^{-3}$, $\Delta L_s \approx 8$

$s = 10^{-1}$: $\Delta F_s \approx 5 \cdot 10^{-2}$, $\Delta L_s \approx 1{,}2 \cdot 10^{2}$


s = 0.001
10
0
−10
−20
−30
−40
−50
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Interpolate func
Approximating func
True values


s = 0.01
30
20
10
0
−10
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Interpolate func
Approximating func
True values



Можно заметить, что оба метода куда устойчивее к шуму в f_i , чем x_i.

5 Выводы

В данной работе было исследовано поведение интерполяционного и приближающего многочлена для таблично заданной функции $(x_i, f_i),\ i = \overline{1, n}$. Было выяснено, что интерполяционный многочлен не устойчив для заданной нам конфигурации сетки (x_i, f_i). Если предположить, что при задании табличной функции были допущены погрешности, то аппроксимирующий многочлен куда лучше справится с восстановлением функции, нежели интерполяционный многочлен Лагранжа.