Интерполяция функции полиномом Лагранжа и кубическими сплайнами

Лебедев Андрей, группа 210

Ноябрь 2022

1 Постановка задачи

1.1 Общий вид задачи

Функция $f(x)$ задана таблично на отрезке $[0, a]$ в точках $x_i : x_i = i * h, i = 0, 1, ..., N, h = a/n$

1. Построить интерполяционный многочлен по точкам x_i
2. Интерполировать функцию кубическим сплайном.
3. Результаты сравнить. Оценить разность.

1.2 Параметры интерполируемой функции

В рамках данной работы реализуется интерполяция при $a = 2, N = 20$, значения в точках:

i	0	1	2	3	4	5	6
x	0	0.1	0.2	0.3	0.4	0.5	0.6
f(x)	0	0.529847	1.027775	1.346477	1.356512	0.986714	0.257137
i	7	8	9	10	11	12	13
x	0.7	0.8	0.9	1	1.1	1.2	1.3
f(x)	0.706391	1.684295	2.404336	2.606626	2.11956	0.927692	0.789339
i	14	15	16	17	18	19	20
x	1.4	1.5	1.6	1.7	1.8	1.9	2
f(x)	2.664212	4.203824	4.900323	4.370876	2.493172	0.502452	4.019803

2 Интерполяционный многочлен Лагранжа

2.1 Математическая модель

Для нахождения интерполяционного многочлена воспользуемся методом Лагранжа.

$$L(X) = \sum_{i=0}^{n} x_i * l(x)_i \quad (1)$$

Где $L(X)$ - искомый многочлен, y_i - ординаты заданных точек $(i = \overline{0, N})$, l_i - базисный полином, вычисляемый по следующей формуле:

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad (2)$$

Заметим, что $\forall i$ многочлен l_i имеет степень n и $l_j(x_i) = \delta_{ij}$. Значит искомый многочлен Лагранжа представляет собой линейную комбинацию базисных полиномов и имеет степень не больше n.

2.2 Математическая модель

Поиск коэффициентов и построение интерполяционного многочлена Лагранжа реализованы на языке Python3. Реализация состоит из 2-х основных функций.

1. Функция, которая по значениям абсцисс точек и индексу i возвращает кортеж коэффициентов $(a_n, a_{n-1}, ..., a_0)$ базисного многочлена $l_i(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$

   Реализация функции на языке Python3

```
def lagrJCoef(j):
  k = 1
  for i in range(N+1):
      if i != j:
          k *= (x_array[j]-x_array[i])
  t = (1/k,)
  temp = deepcopy(x_array)
  del temp[j]
  for i in range(N):
      isum = 0
      for p in combinations(temp, (i+1)):
          isum += np.prod(p)
      if i%2 == 0:
          isum *= -1
      t = *t, isum/k
  return t
```

   Функция combinations возвращает массив сочетаний без повторений из массива temp (абсцисс точек) длины $i + 1$ (функция исключительно комбинаторная). Использование такого алгоритма обусловлено тем, что $a_k = 1$, если $k = n$, и $a_k =$ сумма всевозможных произведений, состоящих из $n - k$ корней x_j функции (2) для всех остальных k.
   Код реализующий собственную вариацию функции combinations:

```
def combinations(iterable, r):
   pool = tuple(iterable)
   n = len(pool)
   if r > n:
       return
   indices = list(range(r))
   yield tuple(pool[i] for i in indices)
   while True:
       for i in reversed(range(r)):
           if indices[i] != i + n - r:
               break
       else:
           return
```

```
        indices[i] += 1
        for j in range(i+1, r):
            indices[j] = indices[j-1] + 1
        yield tuple(pool[i] for i in indices)
```

2. Функция, которая по значениям точек возвращает кортеж коэффициентов интерполяционного многочлена Лагранжа, используя предыдущую функцию, согласно формуле (1)

   Реализация функции на языке Python3

```
def LagrCoef():
  answer = [0 for i in range(N+1)]
  for j in tqdm(range(N+1)):
      a = list(lagrJCoef(j))
      for i in range(N+1):
          answer[i] += a[i]*y_array[j]
  return tuple(answer)
```

3 Интерполяция кубическими сплайнами

3.1 Математическая модель

Необходимо на отрезке $[0, a]$ по $n+1$ точке с координатами (x_i, y_i), где $i = \overline{0, N}$ построить кусочную функцию $S(x)$, удовлетворяющую следующим условиям:

- На каждом отрезке $[x_{i-1}, x_i], i = \overline{1, n}, S(x)$ представляет собой многочлен степени не выше 3
- $S(x) \in C^2_{[0,a]}$
- $S(x_i) = y_i$
- $S^{(2)}(0) = S^2(a) = 0$ – условие, необходимое для однозначности задания функции $S(x)$, задающее "естественный сплайн"
- $S^3(0) = S^3(a) = 0$ – граничные условия непрерывности второй производной

Заметим, что обозначенные выше условия однозначно задают функцию $S(x)$

$$S(x) = \begin{cases} a_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3, & x \in [x_0; x_1] \\ \dots \\ a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, & x \in [x_{i-1}; x_i] \\ \dots \\ a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3, & x \in [x_{n-1}; x_n] \end{cases}$$

Тогда естественным образом:

$$S_i(x_i) = a_i \qquad S_i^{'}(x_i) = b_i \qquad S_i^2(x_i) = 2c_i \qquad S_i^3(x_i) = 6d_i, \qquad i = \overline{1, n}$$

Добавим к этим уравнениям уравнения, задающие непрерывность производных:

$$S_i(x_{i-1}) = S_{i-1}(x_{i-1}) \qquad S_i^{'}(x_{i-1}) = S_{i-1}^{'}(x_{i-1}) \qquad S_i^{(2)}(x_{i-1}) = S_{i-1}^{(2)}(x_{i-1}), \qquad i = \overline{1, n}$$

И уравнение $S(x_i) = y_i$ тогда

1. $a_i = y_i$
2. $b_i = \frac{a_i - a_{i-1}}{x_i - x_{i-1}} + \frac{2c_i + c_{i-1}}{3} * (x_i - x_{i-1})$
3. $c_{i-1} * (x_i - x_{i-1}) + 2c_i * (x_{i+1} - x_{i-1}) + c_{i+1} * (x_{i+1} - x_i) = 3 * (\frac{a_{i+1} - a_i}{x_{i+1} - x_i} - \frac{a_i - a_{i-1}}{x_i - x_{i-1}})$,
   Причем $c_0 = {}_n = 0$
4. $d_i = \frac{c_i - c_{i-1}}{3(x_i - x_{i-1}}$

   $i = \overline{1, n}$

3.2 Реализация

Реализация функции на языке Python3

```
def CubicSplineCoefs():
  a = [y_array[i] for i in range(N+1)]
  k = [0, 0] + [x_array[i+2] - x_array[i+1] for i in range(N-2)]
  p = [0] + [2*(x_array[i+2] - x_array[i]) for i in range(N-1)]
  t = [0] + [(x_array[i+2] - x_array[i+1]) for i in range(N-2)]
  q = [0] + [3*((a[i+2] - a[i+1])/(x_array[i+2] - x_array[i+1])
              - (a[i+1] - a[i])/(x_array[i+1] - x_array[i])) for i in range(N-1)]
  y = [0, p[1]]
  alph = [0, -t[1]/y[1]]
  beth = [0, q[1]/y[1]]
  for i in range(N-3):
      y = y + [p[i+2]+k[i+2]*alph[-1]]
      alph = alph + [-t[i+2]/y[-1]]
      beth = beth + [(q[i+2]-k[i+2]*beth[-1])/y[-1]]
  y = y + [p[N-1]+k[N-1]*alph[-1]]
  beth = beth + [(q[N-1]-k[N-1]*beth[-1])/y[-1]]
  c = [beth[-1], 0]

  for i in range(N-1):
      temp = c[0]
      c = [alph[-1-i]*temp + beth[-2-i]] + c
  b = [0] + [(a[i+1]-a[i])/(x_array[i+1]-x_array[i]) + \
  (2*c[i+1]+c[i])*(x_array[i+1]-x_array[i])/3 for i in range(N)]
  d = [0] + [((c[i+1]-c[i])/(x_array[i+1]-x_array[i]))/3 for i in range(N)]
  coefs = [(d[i+1], c[i+1], b[i+1], a[i+1]) for i in range(N)]
  return tuple(coefs)
```

Большую часть программы занимает вычисление c_i. В выше представленном коде k, p и t – списки коэффициентов при c_{i-1}, c_i, c_{i+1} соответственно.

$$k_i = (x_i - x_{i-1})$$

$$p_i = 2 * (x_{i+1} - x_{i-1})$$

$$t_i = (x_{i+1} - x_i)$$

А q – список правых частей формулы рекурсивного вычисления $_i$.

$$q_i = \frac{c_i - c_{i-1}}{3(x_i - x_{i-1}}$$

Тогда система, позволяющая найти $c_1 - c_{n-1}$ выглядит следующим образом:

$$\begin{cases} p_1 * c_1 + t_1 * c_2 = q_1 \\ k_2 * c_1 + p_2 * c_2 + t_2 * c_3 = q_2 \\ \dots \\ k_{n-2} * c_{n-3} + p_{n-2} * c_{n-2} + t_{n-2} * c_{n-1} = q_{n-2} \\ k_{n-1} * c_{n-2} + p_{n-1} * c_{n-1} = q_{n-1} \end{cases}$$

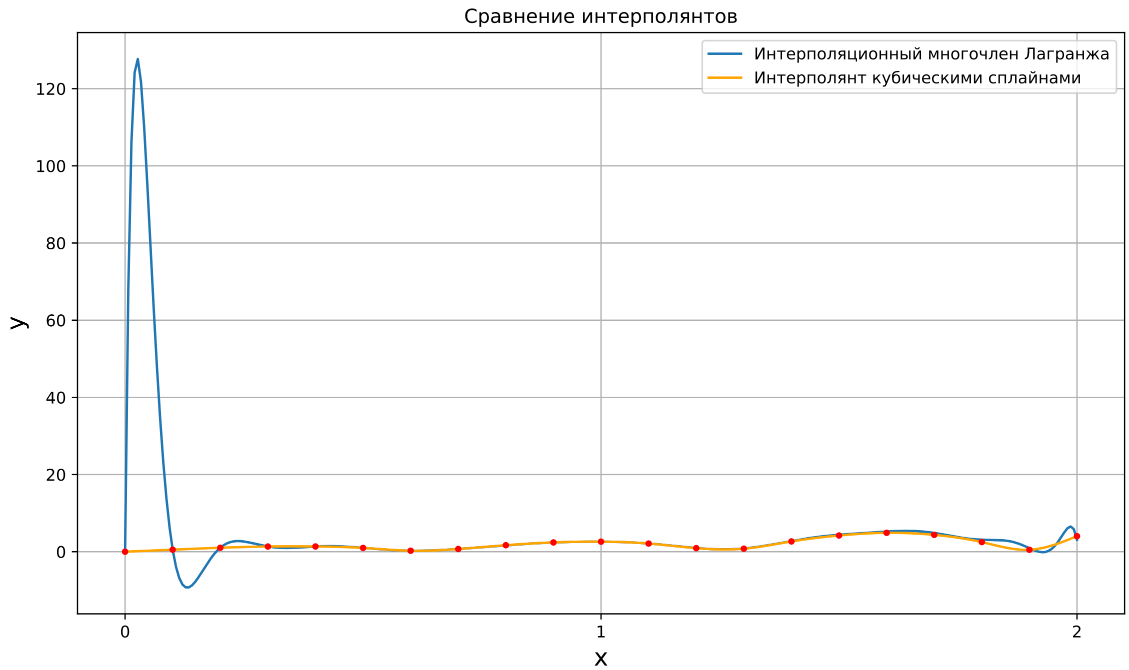
В матричном виде эта система имеет следующий вид:

$$\begin{pmatrix} p_1 & t_1 & \dots & 0 & 0 \\ k_2 & p_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & p_{n-2} & t_{n-2} \\ 0 & 0 & \dots & k_{n-2} & p_{n-1} \end{pmatrix} * \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-2} \\ q_{n-1} \end{pmatrix}$$

Методом прогонки через вспомогательные списки alph и beth находим c_i. Остальные коэффициенты находим по соответствующим формулам.

4 Анализ полученных данных

В результате работы программы были построены следующие графики:



Где красным выделены точки представляющие входные данные, синим – график интерполяционного многочлена Лагранжа, а оранжевым – функция интерполяции кубическими сплайнами
Уравнение многочлена Лагранжа:
$y = -326952.877*x^{20}+6715312.016*x^{19}-64159283.994*x^{18}+378658980.093*x^{17}-1546091043.072*x^{16}+4635189785.716*x^{15}-10568527407.189*x^{14}+18728774288.173*x^{13}-26138579392.001*x^{12}+28934152687.452*x^{11}-25458890560.216*x^{10}+17765359201.918*x^{9}-9764203659.327*x^{8}+4176149188.207*x^{7}-1364092110.885*x^{6}+330840556.278*x^{5}-57083269.296*x^{4}+6539690.724*x^{3}-438781.174*x^{2}+12772.070*x$

Система уравнений, задающая интерполянт кубическими сплайнами:

$$\begin{cases} -0.542*x^3-0.163*x^2+5.288*x+0.53, & x\in[0.0;0.1) \\ -29.21*x^3-8.926*x^2+4.379*x+1.028, & x\in[0.1;0.2) \\ -29.923*x^3-17.903*x^2+1.696*x+1.346, & x\in[0.2;0.3) \\ 19.463*x^3-12.064*x^2-1.301*x+1.357, & x\in[0.3;0.4) \\ -119.095*x^3-47.792*x^2-7.286*x+0.987, & x\in[0.4;0.5) \\ 476.972*x^3+95.299*x^2-2.536*x+0.257, & x\in[0.5;0.6) \\ -250.183*x^3+20.244*x^2+9.019*x+0.706, & x\in[0.6;0.7) \\ -126.42*x^3-17.682*x^2+9.275*x+1.684, & x\in[0.7;0.8) \\ -30.648*x^3-26.876*x^2+4.819*x+2.404, & x\in[0.8;0.9) \\ -10.875*x^3-30.139*x^2-0.882*x+2.607, & x\in[0.9;1.0) \\ -97.455*x^3-59.375*x^2-9.834*x+2.12, & x\in[1.0;1.1) \\ 385.25*x^3+56.2*x^2-10.151*x+0.928, & x\in[1.1;1.2) \\ 314.772*x^3+150.631*x^2+10.532*x+0.789, & x\in[1.2;1.3) \\ -684.627*x^3-54.757*x^2+20.119*x+2.664, & x\in[1.3;1.4) \\ 75.248*x^3-32.182*x^2+11.425*x+4.204, & x\in[1.4;1.5) \\ -124.218*x^3-69.448*x^2+1.262*x+4.9, & x\in[1.5;1.6) \\ 38.79*x^3-57.811*x^2-11.463*x+4.371, & x\in[1.6;1.7) \\ -153.253*x^3-103.787*x^2-27.623*x+2.493, & x\in[1.7;1.8) \\ 1809.462*x^3+439.052*x^2+5.903*x+0.502, & x\in[1.8;1.9) \\ -1463.507*x^3+49.809*x+4.02, & x\in[1.9;2.0) \end{cases}$$

Как видно из графиков метод интерполяции кубическими сплайнами создает “менее амплитудную функцию”, однако делать вывод о том, какой метод более точный не просто не корректно, но и невозможно в виду отсутствия информации о первоначальной функции, по частным значениям которой строятся интерполянты. Заметим, что оба интерполянта для данных точек единственны:

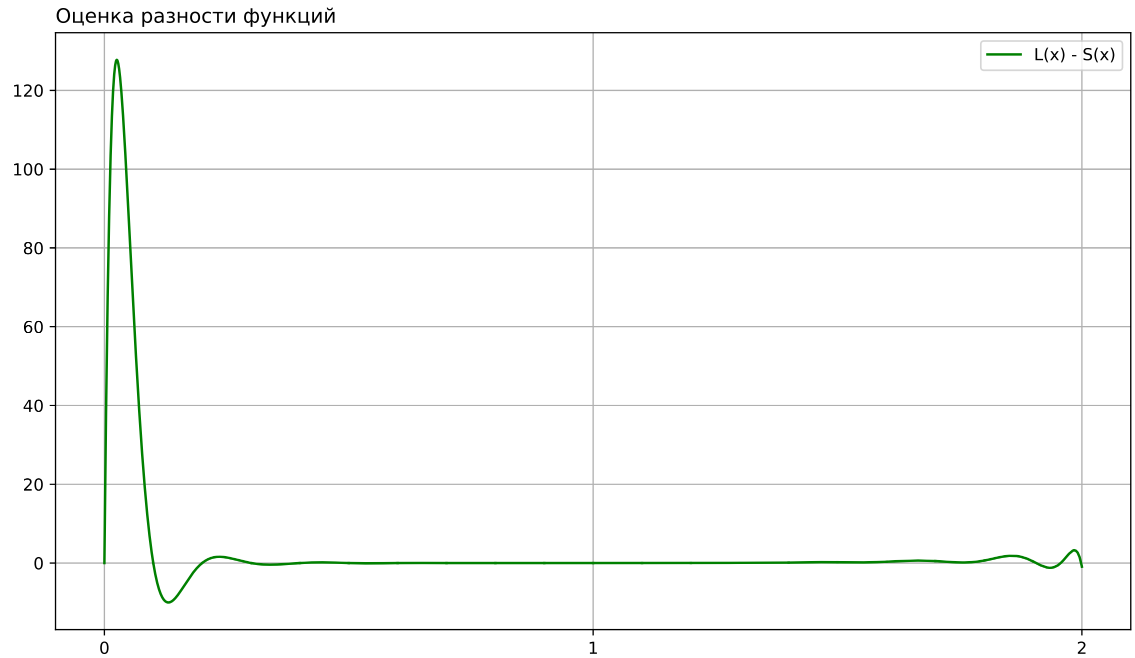
Теорема. Существует единственный многочлен степени не превосходящей n, принимающий заданные значения в $n+1$ заданной точке.

Доказательство. Предположим, что существуют два различных многочлена $P(x)$ и $Q(x)$ степени не более n, для которых верно, что для $n+1$ пар чисел $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, где все $x_j x_j$ различны, $P(x_j) = Q(x_j) = y_j$. Рассмотрим многочлен $L(x) = P(x) - Q(x)$. Подставляя в него x_j $(0 \leq j \leq n)$, получаем, что $L(x_j) = P(x_j) - Q(x_j) = y_j - y_j = 0$. Таким образом, многочлен $L(x)$ имеет $n+1$ корней и все они различны. Следовательно $L(x) \equiv 0$, так как ненулевой многочлен степени не превосходящей n имеет не более n корней. Следовательно, $P(x) = Q(x)$.

Теорема. Для любой функции f и любого разбиения отрезка $[a, b]$на части $[x_{i-1}, x_i]$существует ровно один естественный сплайн $S_i(x)$, удовлетворяющий условиям пункта 3.1.

Эта теорема является следствием более общей теоремы Шёнберга-Уитни об условиях существования интерполяционного сплайна.

Чтобы визуализировать разность функций в каждой точке отрезка построим вспомогательный график, представляющий собой f(x) = L(x) - S(x)



Для анализа различия решений, воспользуемся 2-мя метриками:

$$\parallel f(x) \parallel_{C[0,2]} = \sup_{x \in [0,2]} |L(x) - S(x)| = 127.761$$

$$\parallel f(x) \parallel_{1} = \frac{1}{2} \int_0^2 |L(x) - S(x)| \, \mathrm{d}x = 1.861$$