Московский авиационный институт

(Национальный исследовательский университет)

Институт №8 «Компьютерные науки и прикладная математика

Кафедра вычислительной математики и программирования

Курсовая работа

по курсу «Численные методы»

«Интерполяция экспоненциальными сплайнами»

Выполнила: Богачев П.С.

Группа: М8О-402Б-20

Преподаватель:Пивоваров Д.Е.

Дата:

Оценка:

Москва, 2023

Постановка задачи

Получить интерполяционную формулу $f(x)$ по входным данным в виде вектора $[x, y]$. Входные данные задаются в оконном приложении. В качестве интерполяционной функции использовать экспоненту.

Теоретические сведения

На основе экспоненты представляются функции синус и косинус – гиперболические и тригонометрические. Такого рода обобщения называются экспоненциальными L-сплайнами и предлагаются для самого широкого использования. Однако проработанные теоретические основы оказываются весьма далекими от практического внедрения. Общие формулы, рассчитанные на все случаи жизни, нередко оказываются настолько сложными, что их не применяют ни в одном конкретном случае. Трудоемкость же вычислений часто оказывается существенно меньшей при применении рекуррентных процедур. Поэтому для практической деятельности по части формирования градуировочных зависимостей сенсорных устройств возникла необходимость выполнить интерполяцию экспонентами, хоть и не по обобщенной методике, но в значительной мере универсальной, ограничиваемой лишь условием монотонности возрастания (или убывания) требуемой функции.

На вход программы подается массив точек (x, y). Для каждых трех точек, начиная с $x_i, i = 1, \dots, n-2$, где n – количество точек, должна быть построена экспоненциальная функция в виде $y_i = C_i + B_i * \exp(A_i * x),\ i = 1, \dots, n-2$. Параметры A, B и C вычисляются следующим образом:

1. В случае, когда $\frac{y_{i+1}-y_i}{x_{i+1}-x_i} = \frac{y_{i+2}-y_i}{x_{i+2}-x_i}$, т. е. когда точки лежат на одной прямой, построить экспоненту не получится. Можно воспользоваться линейной интерполяцией.
2. Если $x_{i+2} - x_{i+1} = x_{i+1} - x_i$, можно воспользоваться формулами:

$$C_i = \frac{y_{i+1}^2 - y_i y_{i+2}}{2y_{i+1} - y_i - y_{i+2}},$$

$$A_i = \frac{\ln(\frac{y_{i+2}-y_{i+1}}{y_{i+1}-y_i})}{x_{i+2}-x_{i+1}},$$

Параметр B можно вычислить по одной из трех формул:

$$B_i = (y_i - C_i)\exp(-A_i x_i)$$
$$= (y_i - \frac{y_{i+1}^2 - y_i y_{i+2}}{2y_{i+1} - y_i - y_{i+2}})\exp\left(-\frac{\ln(\frac{y_{i+2}-y_{i+1}}{y_{i+1}-y_i})}{x_{i+2}-x_{i+1}}x_i\right)$$

$$B_i = (y_{i+1} - C_i)\exp(-A_i x_{i+1})$$
$$= (y_{i+1}$$
$$-\frac{y_{i+1}^2 - y_i y_{i+2}}{2y_{i+1} - y_i - y_{i+2}},)\exp\left(-\frac{\ln(\frac{y_{i+2}-y_{i+1}}{y_{i+1}-y_i})}{x_{i+2}-x_{i+1}}x_{i+1}\right)$$

$$B_i = (y_{i+2} - C_i)\exp(-A_i x_{i+2})$$
$$= (y_{i+2}$$
$$-\frac{y_{i+1}^2 - y_i y_{i+2}}{2y_{i+1} - y_i - y_{i+2}},)\exp\left(-\frac{\ln(\frac{y_{i+2}-y_{i+1}}{y_{i+1}-y_i})}{x_{i+2}-x_{i+1}}x_{i+2}\right).$$

3. В остальных случаях (при условии возрастания) параметры вычисляются следующим образом:

Система уравнений:

$$y_i = C_i + B_i\exp(A_i x_i)$$

$$y_{i+1} = C_i + B_i\exp(A_i x_{i+1})$$

$$y_{i+2} = C_i + B_i\exp(A_i x_{i+2})$$

Исключим параметр C:

$$y_i - y_{i+1} = B_i(\exp(A_i x_i) - \exp(A_i x_{i+1}))$$
$$y_{i+2} - y_{i+1} = B_i(\exp(A_i x_{i+2}) - \exp(A_i x_{i+1}))$$

Исключим параметр B:

$$(y_{i+2} - y_{i+1})\left(1 - \exp\left(-A_i(x_{i+1} - x_i)\right)\right) = (y_{i+1} - y_i)\left(\exp\left(A_i(x_{i+2} - x_{i+1})\right) - 1\right)$$

Полученное уравнение относительно A удобно записать в виде функции, приравниваемой к нулю (т. к. R(0) = 0):

$$R(A_i) = (y_{i+2} - y_{i+1})\left(\exp\left(-A_i(x_{i+1} - x_i)\right) - 1\right) + (y_{i+1} - y_i)\left(\exp\left(A_i(x_{i+2} - x_{i+1})\right) - 1\right) = 0$$

Найдем A_{min} через производную:

$$A_{imin} = \frac{\ln\left(\frac{(y_{i+3} - y_{i+2})(x_{i+1} - x_i)}{(y_{i+1} - y_i)(x_{i+2} - x_{i+1})}\right)}{x_{i+2} - x_i}$$

Удвоенное расстояние до точки минимума, являющееся приближенным решением уравнения, возьмем в качестве начального значения для итерационного процесса по методу касательных:

$$A_{i0} = 2A_{imin}$$

$$R(A_{i0}) + R'(A_{i0})\Delta A_i = 0$$

$$\Delta A_i = \frac{-R(A_{i0})}{R'(A_{i0})}$$

$$A_i \leftarrow A_{i0} + \Delta A_i$$

Таким образом находим коэффициент A, затем подстановкой в уравнения выше коэффициенты B и C.

Интервалы от x_k до $x_{k+1}, k \in [2; n-1]$ являются общими для функций F_k и F_{k+1}. Поэтому необходимо произвести склеивание функций на данных интервалах. Это можно сделать двумя способами:

1. Усреднение:

$$H(x) = \frac{F_k(x) + F_{k+1}(x)}{2}$$

В этом случае непрерывность производной не соблюдается.

2. Склеивающая функция вида:

$$G(x) = \frac{(x_{k+1} - x)F_k(x) + (x - x_k)F_{k+1}(x)}{x_{k+1} - x_k}$$

В данном случае соблюдается непрерывность производной.

Программа

```
import tkinter as tk
import numpy as np
import matplotlib.pyplot as plt
import scipy.interpolate as spi


def print_entered_value():
    global value
    global value1
    value = entry.get().split()
    value = np.array(value, dtype=float)
    value1 = entry1.get().split()
    value1 = np.array(value1, dtype=float)

    window.quit()


def ff(x):
    return 16 * x * x * x/ (x * x + 1) ** 3 - 12 * x / (x ** 2 + 1) ** 2


def F(A, B, C, x):
    return C + B * np.exp(A * x)


def check(x, y):
    swapped = True
    while swapped:
        swapped = False
        for i in range(len(x) - 1):
            if x[i] > x[i + 1]:
                x[i], x[i + 1] = x[i + 1], x[i]
                y[i], y[i + 1] = y[i + 1], y[i]
                swapped = True
    for i in range((len(x) - 1)):
        if (y[i + 1] < y[i]):
            return 1, x, y

    return 0, x, y


def exp_interpolation(x, y):
    A = np.zeros(len(x) - 2)
    B = np.zeros(len(x) - 2)
    C = np.zeros(len(x) - 2)
    res = np.zeros(len(x) - 2)

    d = 0.00001
```

```
    for i in range(len(A)):
        dif = (y[i + 2] - y[i + 1]) * (x[i + 1] - x[i]) / ((y[i + 1] - y[i])
* (x[i + 2] - x[i + 1]))

        if check(x, y)[0] == 1:
            A[i], B[i], C[i], res = 0, 0, 0, 0
            break
        else:
            x, y = check(x, y)[1:]

        if abs(dif - 1) < d:
            A[i] = (y[i + 2] - y[i]) / (x[i + 2] - x[i])
            B[i] = y[i] - A[i] * x[i]
            print('При данном наборе точек применима только линейная
интерполяция')
            C[i] = 0
            res[i] = 1

        if x[i + 2] + x[i] == x[i + 1] + x[i + 1]:
            z = (2 * y[i + 1] - y[i] - y[i + 2])
            r = (y[i + 2] - y[i + 1]) / (y[i + 1] - y[i])
            A[i] = np.log(r) / (x[i + 2] - x[i + 1])
            C[i] = (y[i + 1] * y[i + 1] - y[i] * y[i + 2]) / z
            B[i] = (y[i] - C[i]) * np.exp(-A[i] * x[i])
            res[i] = 2

        else:

            Amin = np.log(dif) / (x[i + 2] - x[i])
            A0 = 2 * Amin
            while (1):
                u = np.exp(A0 * (x[i + 2] - x[i + 1]))
                v = np.exp(-A0 * (x[i + 1] - x[i]))
                F = (y[i + 1] - y[i]) * (u - 1) + (y[i + 2] - y[i + 1]) * (v
- 1)
                FF = (y[i + 1] - y[i]) * (x[i + 2] - x[i + 1]) * u - (y[i +
2] - y[i + 1]) * (x[i + 1] - x[i]) * v
                dA = -F / FF
                A0 += dA
                if (abs(dA / Amin) < 0.000000001):
                    break

            A[i] = A0
            B[i] = (y[i] - y[i + 1]) / (np.exp(A0 * x[i]) - np.exp(A0 * x[i +
1]))
            C[i] = y[i] - B[i] * np.exp(A0 * x[i])
            res[i] = 2

    return A, B, C, res


def plot_graphics(A, B, C, x, y):
    max = 0
    n = len(A)

    x1 = np.arange(x[0], x[1] + 0.01, 0.01)
    y1 = C[0] + B[0] * np.exp(A[0] * x1)
    for i in range (len(x1)):
        if abs(y1[i]-ff(x1)[i])> max:
            max = abs(y1[i]-ff(x1)[i])
    yy = ff(x1)
    plt.title("Интерполяция экспонентой с использованием G(x)")
    plt.plot(x1, y1, color="red")
    plt.plot(x1, yy, color="black")
```

```
    plt.scatter(x, y, c='b')

    if (n > 1):
        for i in range(1, n):
            x1 = np.arange(x[i], x[i + 1] + 0.01, 0.01)
            y1 = ((x[i + 1] - x1) * (C[i - 1] + B[i - 1] * np.exp(A[i - 1] *
x1)) + (x1 - x[i]) * (
                    C[i] + B[i] * np.exp(A[i] * x1))) / (x[i + 1] - x[i])
            yy = ff(x1)
            for i in range(len(x1)):
                if abs(y1[i] - ff(x1)[i]) > max:
                    max = abs(y1[i] - ff(x1)[i])
            plt.plot(x1, y1, color="red")
            plt.plot(x1, yy, color="black")
    x1 = np.arange(x[-2], x[-1] + 0.01, 0.01)
    y1 = C[n - 1] + B[n - 1] * np.exp(A[n - 1] * x1)
    yy = ff(x1)
    plt.plot(x1, y1, color="red",label="Интерполяционная функция")
    plt.plot(x1, yy, color="black", label="Искомая функция  ")
    plt.legend()
    plt.grid()
    plt.show()
    plt.title("Интерполяция экспонентой с использованием H(x)")
    x1 = np.arange(x[0], x[1] + 0.01, 0.01)
    y1 = C[0] + B[0] * np.exp(A[0] * x1)
    yy = ff(x1)
    for i in range (len(x1)):
        if abs(y1[i]-ff(x1)[i])> max:
            max = abs(y1[i]-ff(x1)[i])

    plt.plot(x1, y1, color="green")
    plt.plot(x1, yy, color="black")
    plt.scatter(x, y, c='b')

    if (n > 1):
        for i in range(1, n):
            x1 = np.arange(x[i], x[i + 1] + 0.01, 0.01)
            y2 = (C[i] + B[i] * np.exp(A[i] * x1) + C[i - 1] + B[i - 1] *
np.exp(A[i - 1] * x1)) / 2
            yy = ff(x1)
            plt.plot(x1, y2, color="green")
            plt.plot(x1, yy, color="black")

    x1 = np.arange(x[-2], x[-1] + 0.01, 0.01)
    y1 = C[n - 1] + B[n - 1] * np.exp(A[n - 1] * x1)
    yy = ff(x1)
    plt.plot(x1, y1, color="green", label="Интерполяционная функция")
    plt.plot(x1, yy, color="black", label="Искомая функция  ")
    plt.grid()
    plt.show()
    return max


def lagranz(x, y, t):
    z = 0
    for j in range(len(y)):
        p1 = 1;
        p2 = 1
        for i in range(len(x)):
            if i == j:
                p1 = p1 * 1;
                p2 = p2 * 1
            else:
                p1 = p1 * (t - x[i])
```

```
                p2 = p2 * (x[j] - x[i])
        z = z + y[j] * p1 / p2
    return z

def _poly_newton_coefficient(x, y):
    """
    x: list or np array contanining x data points
    y: list or np array contanining y data points
    """

    m = len(x)

    x = np.copy(x)
    a = np.copy(y)
    for k in range(1, m):
        a[k:m] = (a[k:m] - a[k - 1])/(x[k:m] - x[k - 1])

    return a

def newton_polynomial(x_data, y_data, x):
    """
    x_data: data points at x
    y_data: data points at y
    x: evaluation point(s)
    """
    a = _poly_newton_coefficient(x_data, y_data)
    n = len(x_data) - 1  # Degree of polynomial
    p = a[n]

    for k in range(1, n + 1):
        p = a[n - k] + (x - x_data[n - k])*p

    return p




window = tk.Tk()
window.geometry('500x300+200+100')
window.title("Ввод данных")
label = tk.Label(window, text="Введите координаты точек по x и y без
запятых")
label.pack()

entry = tk.Entry(window)
entry.pack()
entry1 = tk.Entry(window)
entry1.pack()

button = tk.Button(window, text="Построить график",
command=print_entered_value)
button.pack()

window.mainloop()

x = value
y = value1

res = check(x, y)[0]
x, y = check(x, y)[1:]
res
if res == 1:

    window1 = tk.Tk()
```

```
    window1.geometry('300x200+200+100')
    window1.title("Ошибка")
    label1 = tk.Label(window1, text="Нарушен порядок возрастания")
    label1.pack(expand=True, ipadx=10, ipady=10)
    window1.mainloop()
else:
    A, B, C, res = exp_interpolation(x, y)
    max = plot_graphics(A, B, C, x, y)
    max_l = 0
    max_n = 0
    max_lin = 0
    max_q = 0
    max_c = 0
    xx = np.arange(x[0],x[-1]+0.01,0.01)
    ynew = [lagranz(x, y, i) for i in xx]
    y_n = newton_polynomial(x,y,xx)
    yy = ff(xx)
    f1 = spi.interp1d(x, y, kind='linear',bounds_error = False,fill_value=
y[-1])
    f2 = spi.interp1d(x, y, kind='quadratic', bounds_error=False,
fill_value=y[-1])
    f3 = spi.interp1d(x, y, kind='cubic', bounds_error=False, fill_value=y[-
1])
    for i in range(len(xx)):
        if abs(ynew[i] - yy[i]) > max_l:
            max_l = abs(ynew[i] - yy[i])
        if abs(y_n[i] - yy[i]) > max_n:
            max_n = abs(y_n[i] - yy[i])
        if abs(f1(xx)[i] - yy[i]) > max_lin:
            max_lin = abs(f1(xx)[i] - yy[i])
        if abs(f2(xx)[i] - yy[i]) > max_q:
            max_q = abs(f2(xx)[i] - yy[i])
        if abs(f3(xx)[i] - yy[i]) > max_c:
            max_c = abs(f3(xx)[i] - yy[i])
    print('Ошибки интерполяции:')
    print('Экспоненциальная функция:', max)
    print('Полином Лагранжа: ',max_l)
    print('Полином Ньютона: ',max_n)
    print('Линейная интерполяция: ',max_lin)
    print('Квадратичная интерполяция: ', max_q)
    print('Кубическая интерполяция: ', max_c)
    plt.title("Интерполяционный многочлен Лагранжа")
    plt.plot(xx, ynew,label="многочлен Лагранжа  ")
    plt.plot(xx, yy, color="black", label="Искомая функция  ")
    plt.scatter(x, y, c='b')
    plt.legend()
    plt.grid()
    plt.show()

    plt.title("Интерполяционный многочлен Ньютона")
    plt.plot(xx, y_n,label="многочлен Ньютона  ")
    plt.plot(xx, yy, color="black", label="Искомая функция  ")
    plt.scatter(x, y, c='b')
    plt.legend()
    plt.grid()
    plt.show()
    plt.title("Линейная интерполяция")
    plt.plot(xx, f1(xx),label ='Линейная интерполяция')
    plt.plot(xx, yy, color="black", label="Искомая функция  ")
    plt.scatter(x, y, c='b')
    plt.legend()
    plt.grid()
    plt.show()
    plt.title("Квадратичная интерполяция")
```

```
    plt.plot(xx, f2(xx),label ='Квадратичная интерполяция')
    plt.plot(xx, yy, color="black", label="Искомая функция  ")
    plt.scatter(x, y, c='b')
    plt.legend()
    plt.grid()
    plt.show()
    plt.title("Кубическая интерполяция")
    plt.plot(xx, f3(xx),label ='Кубическая интерполяция')
    plt.plot(xx, yy, color="black", label="Искомая функция  ")
    plt.scatter(x, y, c='b')
    plt.legend()
    plt.grid()
    plt.show()
```

Вывод программы

В качестве анализа метода возьмем возрастающую функцию, например, $y = \frac{16x^3}{(x^2+1)^3} - \frac{12x}{(x^2+1)^2}$ на промежутке $[0{,}45; 1.7]$

0.45	0.52	0.54	0.6	0.72	0.79	0.9	1.06	1.1	1.23	1.3	1.4	1.5	1.6	1.7
-2.896	-2.769	-2.715	-2.519	-2.042	-1.753	-1.333	-0.831	-0.733	-0.461	-0.355	-0.225	-0.131	-0.062	-0.012

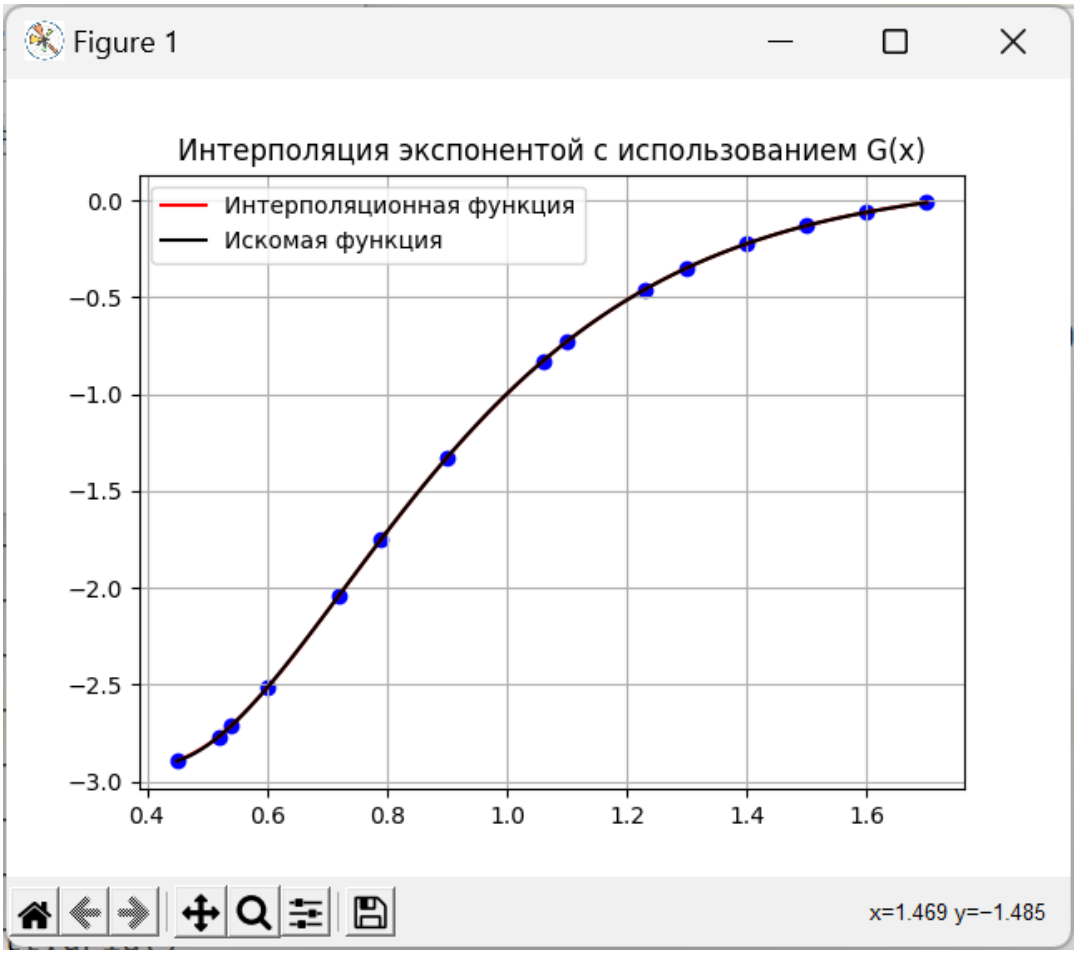
Figure 1
Интерполяция экспонентой с использованием G(x)
Интерполяционная функция
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6
x=1.469 y=−1.485

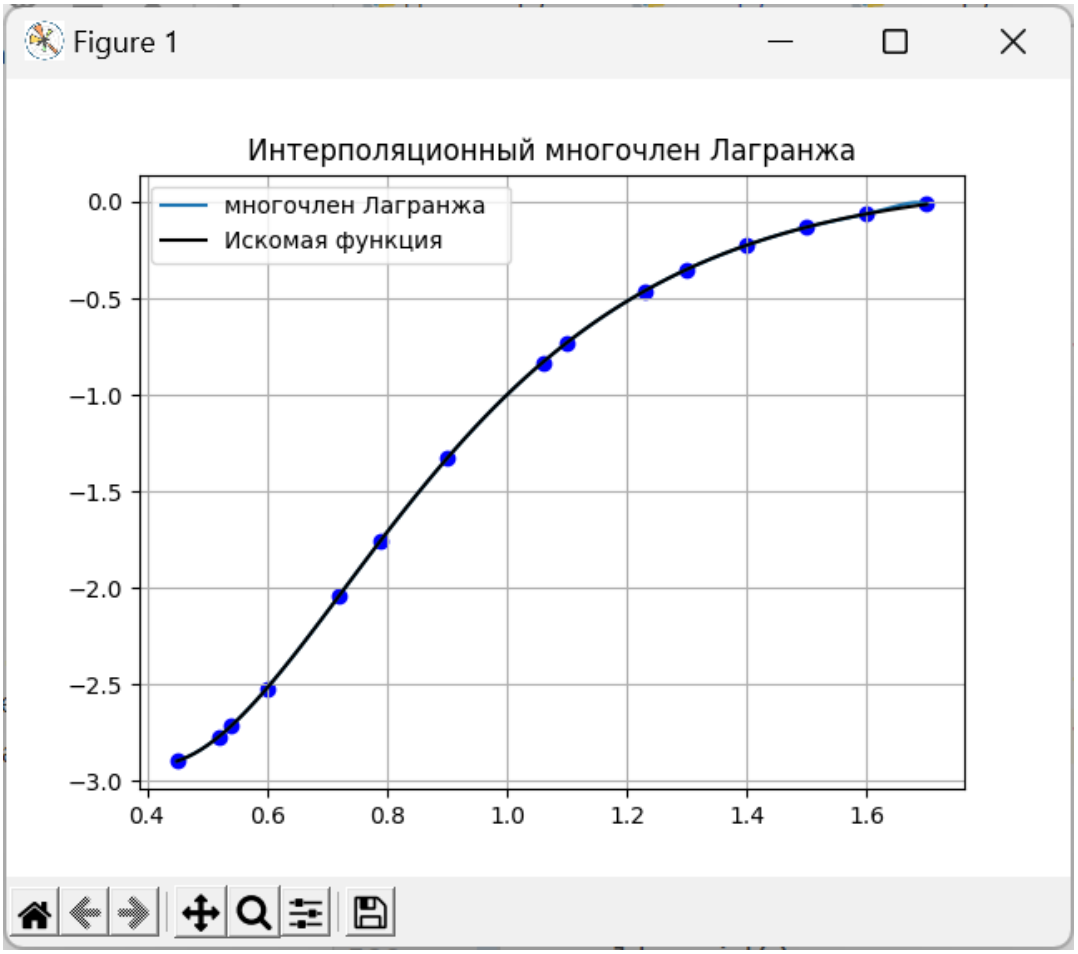
Figure 1
Интерполяционный многочлен Лагранжа
многочлен Лагранжа
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6

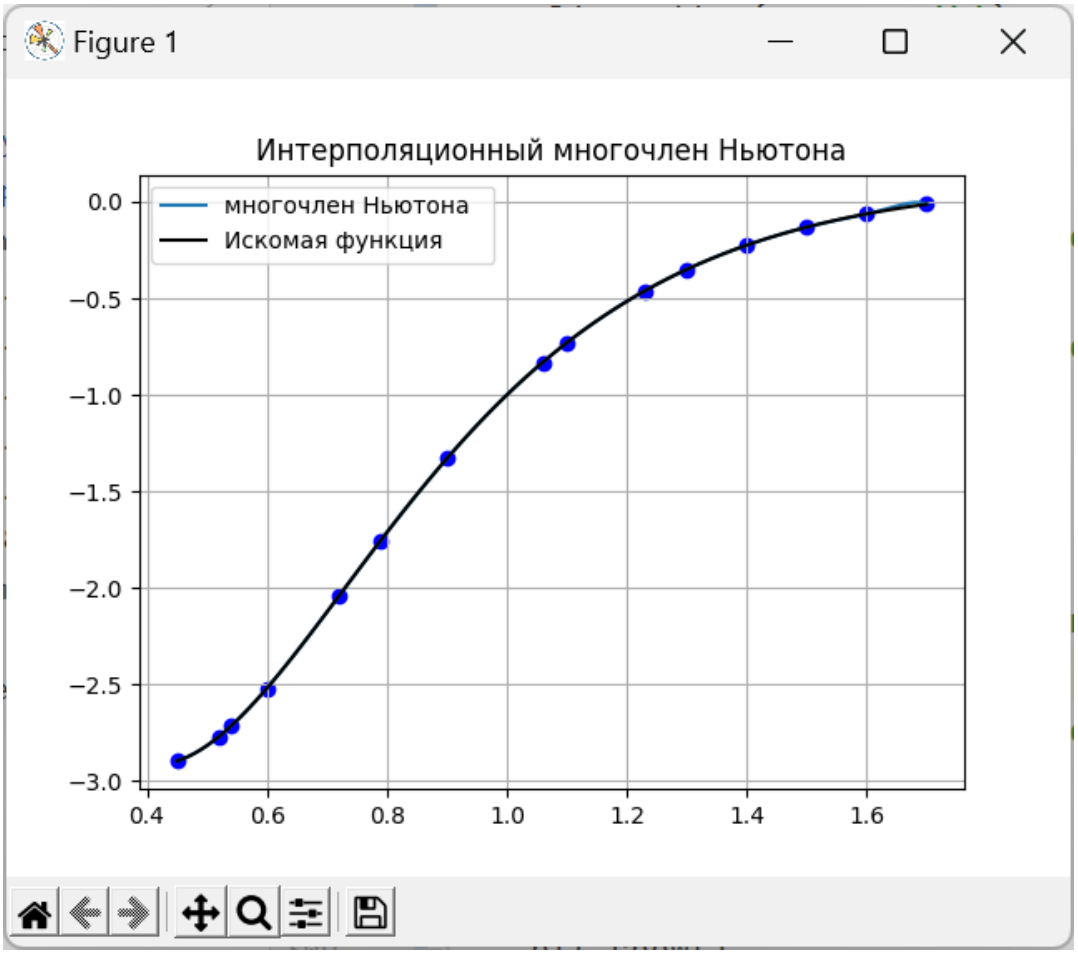
Figure 1
Интерполяционный многочлен Ньютона
многочлен Ньютона
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6

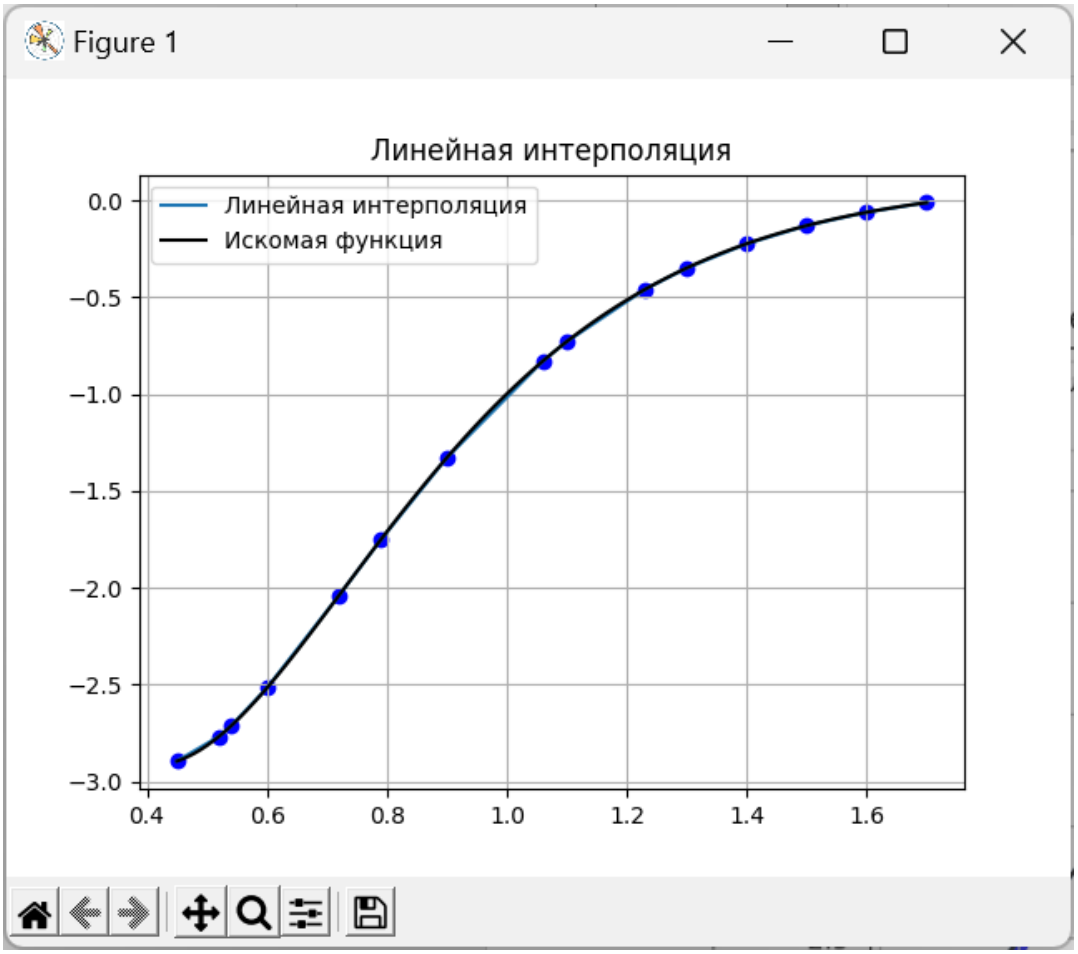
Figure 1
Линейная интерполяция
Линейная интерполяция
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6

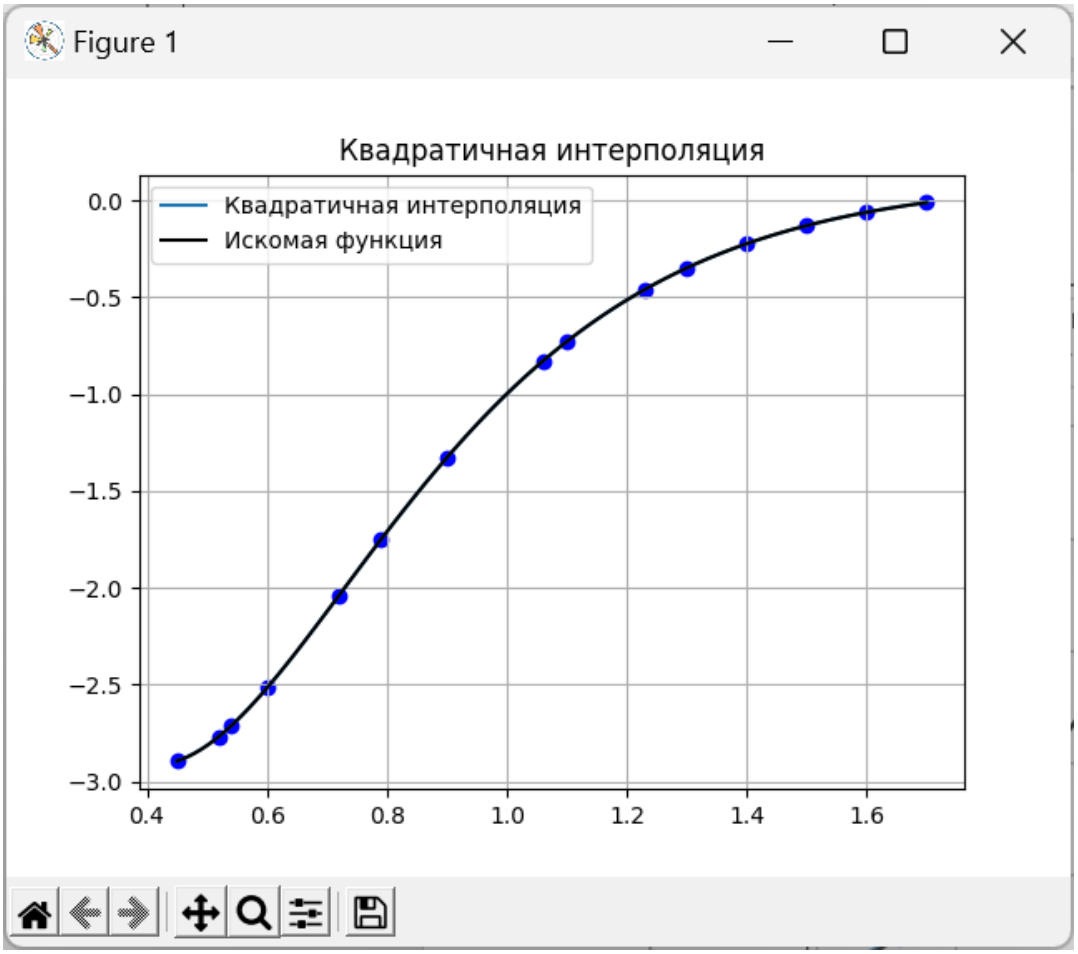
Figure 1
Квадратичная интерполяция
Квадратичная интерполяция
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6

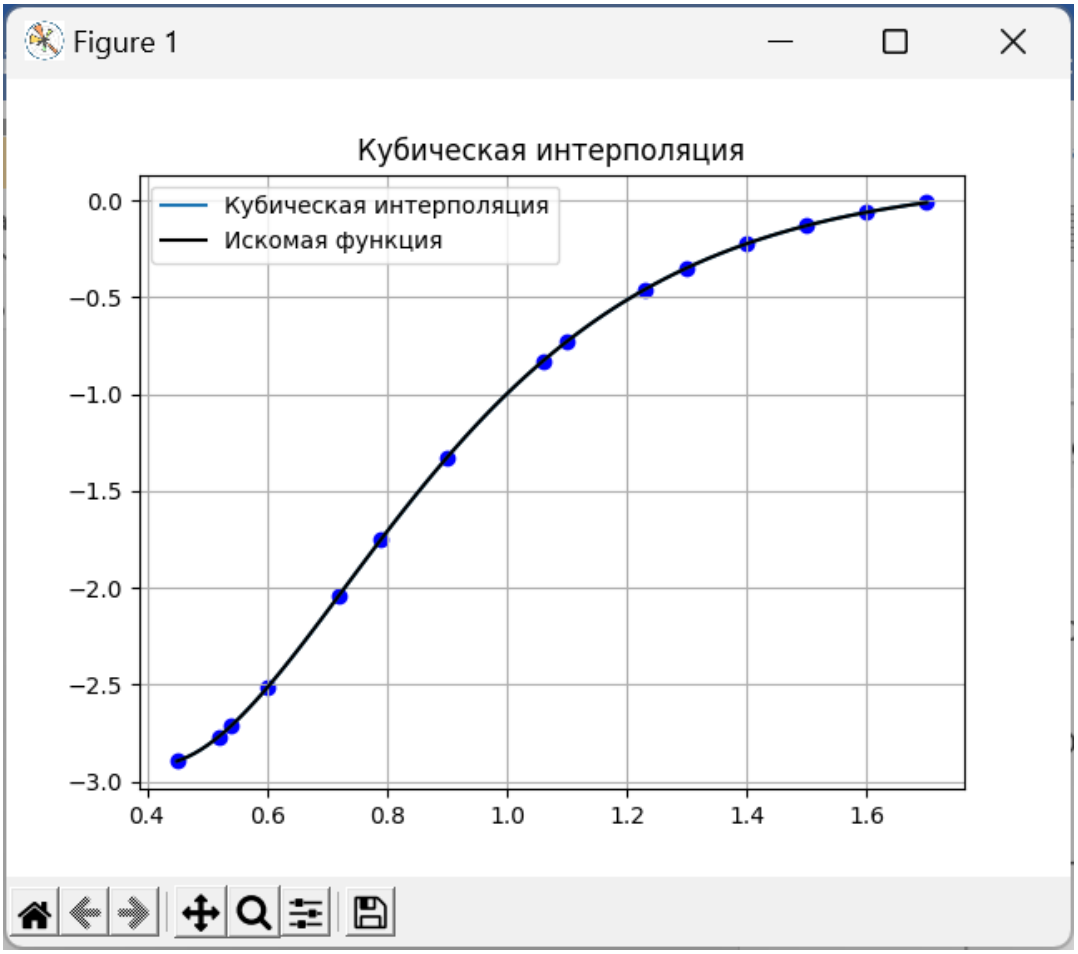
Figure 1
Кубическая интерполяция
Кубическая интерполяция
Искомая функция
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.4
0.6
0.8
1.0
1.2
1.4
1.6

```
Ошибки интерполяции:
Экспоненциальная функция: 0.003488741639925852
Полином Лагранжа:  0.01844736575541019
Полином Ньютона:  0.01844736575547734
Линейная интерполяция:  0.019300617023926936
Квадратичная интерполяция:  0.0014335126573756796
Кубическая интерполяция:  0.0007072856865726444
```

Прикладные задачи

В настоящее время имеется значительное число работ, посвященных численным аспектам интерполяции сплайнами функций одного переменного, связанных с локальным наследованием сплайном свойств монотонности и выпуклости исходных данных.

Сплайн можно использовать для уменьшения веса хранимых данных. Легче хранить коэффициенты интерполяции, чем всю таблицу точек. Например, если нужно хранить форму крыла, которая имеет вид, похожий на функцию экспоненты.

Формулу просто интегрировать, находить производную. К ней можно применять математические методы для исследования. Это следует из свойств функции экспоненты.

Также важной областью применения алгоритма является робототехника, прежде всего – сенсорные устройства. Обработка сигналов от датчиков обычно происходит с применением градуировочных функций, определяющих зависимость электрического сигнала от измеряемых физических величин внешнего пространства. Известно, что для обработки экспериментальных данных выбор метода интерполяции целесообразно осуществлять с учетом условий, характерных для данного анализируемого эксперимента: характер изменения данных, количество точек, точность измерения. Для градуировочных функций типично, что природа нам предоставляет монотонно возрастающие зависимости. Однако возрастание может быть неравномерным

и переходящим в асимптотическое приближение к некоторому предельному уровню. Если на таких участках получены редкие исходные данные, то обычные способы построения интерполяционной кривой с помощью полиномов дадут неверную картину с выпуклостями. В связи с этим заслуживают внимание экспоненты.

Вывод

Как показал анализ метода, интерполяция экспонентой дает неплохие показатели. Судя по ошибке интерполяции, экспонента может сравниться с квадратичной интерполяцией. Лучше всего из вышеприведенных алгоритмов справилась кубическая интерполяция. Линейная интерполяция, интерполяция многочленом Лагранжа и Ньютона дают большую ошибку.

Литература

1. **Юревич Е.И.** Сенсорные системы в робототехнике: учеб. пособие. СПб.: Изд-во Политехн. ун-та, 2013. 100 с.
2. **Дьяконов В.П.** Mathcad 2001: спец. справочник. СПб.: Питер, 2002. 832 с.
3. Численные методы решения задач. обработка экспериментальных данных средствами MathCAD [Электронный ресурс] / URL: http:// tib.znaimo.com.ua/docs/1100/index-26510-1.html
4. **Утешев А.Ю.** Интерполяция [Электронный ресурс] / URL: http://pmpu.ru/vf4/interpolation
5. **Винниченко Л.Ф.** Экспоненциальные гистосплайны: предпосылки введения, конф. европейская наука XXI века, 2009 [Электронный ресурс] / URL: http://www.rusnauka.com/14_ ENXXI_2009/Matemathics/46070.doc.htm
6. **Стрелкова Е.В.** Аппроксимация локальными L-сплайнами. дис. … канд. физ.-мат. наук. екатеринбург: ИММ Уро РАН, 2009.
7. **Шевалдина Е.В.** Аппроксимация локальными экспоненциальными сплайнами с произвольными узлами // Сиб. журн. вычисл. матем. 2006. № 9:4. С. 391–402.
8. **Калиткин Н.Н.** Численные методы: учеб. пособие. 2-е изд., испр. СПб.: бхВ-Петербург, 2013. 592 с.