УДК 517.518.85
DOI 10.17513/snt.39972

ИССЛЕДОВАНИЕ МЕТОДА ИНТЕРПОЛИРОВАНИЯ СПЛАЙНАМИ И ЕГО РЕАЛИЗАЦИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

[1]Курасов Д.А., [1]Волоскова М.М., [1]Сабирова Р.Д., [1]Хомутова Е.И., [2]Кутузов А.С.

[1]ФГАОУ ВО «Тюменский государственный университет», Тюмень, e-mail: naukka@mail.ru;
[2]ФГБОУ ВО «Челябинский государственный университет», Челябинск, e-mail: thething84@mail.ru

Аннотация. В статье рассматривается метод интерполирования сплайнами. Для данного метода приводятся в обобщенном виде основные формулы и расчетные зависимости, указываются необходимые условия существования сплайнов. В качестве примера построения кубического сплайна рассматривается тригонометрическая функция. Система уравнений, соответствующая кубическому сплайну, решается методом обратной матрицы в числовом редакторе Excel, что позволяет с минимальными трудозатратами найти искомые коэффициенты. Также рассмотрена реализация указанного метода на языке программирования Python для данной тригонометрической функции. Произведен сравнительный анализ полученных результатов. Визуализация расчетов представлена в виде графиков функции для sin(x), интерполяционного многочлена и графика функций, наложенных друг на друга, с целью анализа расхождения точного и приближенного значения функций. Исследование, выполненное в статье, показывает, что различные методы расчета приближенных значений для функций дают разные результаты и в любом методе имеются свои погрешности и расхождения. Но так или иначе реализация метода интерполирования сплайнами с использованием элементов Индустрии 4.0 с помощью языка программирования Python дает более точные данные, которые подтверждаются наименьшим расхождением графиков.

Ключевые слова: интерполирование, сплайны, интерполяционные многочлены, численные методы, приближения сплайнами

SPLINE INTERPOLATION STUDY AND IMPLEMENTATION IN PYTHON PROGRAMMING LANGUAGE

[1]Kurasov D.A., [1]Voloskova M.M., [1]Sabirova R.D., [1]Khomutova E.I., [2]Kutuzov A.S.

[1]Tyumen State University, Tyumen, e-mail: naukka@mail.ru;
[2]Chelyabinsk State University, Chelyabinsk, e-mail: thething84@mail.ru

Annotation. The article discusses the method of interpolation by splines. For this method, the basic formulas and calculated dependencies are summarized, and the necessary conditions for the existence of splines are indicated. The trigonometric function is considered as an example of constructing a cubic spline. The system of equations corresponding to the cubic spline is solved by the inverse matrix method in the Excel numerical editor, which allows you to find the desired coefficients with minimal effort. The implementation of this method in the Python programming language for this trigonometric function is also considered. A comparative analysis of the obtained results was carried out. The visualization of the calculations is presented in the form of graphs of the function for sin(x), the interpolation polynomial and the graph of functions superimposed on each other, in order to analyze the discrepancy between the exact and approximate values of the functions. The research carried out in the article shows that different methods of calculating approximate values for functions give different results and each method has its own errors and discrepancies. But one way or another, the implementation of the spline interpolation method using Industry 4.0 elements using the Python programming language provides more accurate data, which is confirmed by the smallest discrepancy in the graphs.

Keywords: interpolation, splines, interpolation polynomials, numerical methods, approximations by spline.s

Интерполяционные формулы Лагранжа, Ньютона, Стирлинга и иных при использовании большого числа узлов интерполяции на всем отрезке [a; b] часто приводят к плохому приближению из-за накопления погрешностей в процессе вычислений. Кроме того, из-за расходимости процесса интерполяции увеличение числа узлов не обязательно приводит к повышению точности. Для снижения погрешностей используется кусочно-полиномиальная интерполяция. В этом случае весь отрезок [a; b] разбивается на частичные отрезки, и на каждом из них функцию $f(x)$ заменяют приближенно полиномом невысокой степени.

Один из способов интерполирования на всем отрезке [a; b] является интерполирование сплайнами [1]. Сплайн представляет собой кусочно-полиномиальную функцию, определенную на отрезке [a; b] и имеющую на этом отрезке некоторое количество непрерывных производных. Преимущества интерполяции сплайнами по сравнению с обычными методами интерполяции – сходимость и устойчивость вычислительного процесса.

В качестве примера рассмотрим один из наиболее распространенных случаев – интерполирование функции кубическим сплайном [2].

Пусть на отрезке [a; b] задана непрерывная функция $f(x)$.

Введем разбиение отрезка:

$$a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b \quad (1)$$

и обозначим $y_i = f(x_i)$, $i = 0,1,2,\ldots,n$.

Интерполяционный кубический сплайн [3], соответствующий данной функции $f(x)$ и узлам интерполяции (1), будет определяться функцией $S(x)$, удовлетворяющей следующим условиям:

1) на каждом отрезке $[x_{i-1}; x_i]$, $i = 0,1,2,\ldots,n$. функция $S(x)$ является кубическим многочленом; 2) функция $S(x)$, а также ее первая и вторая производные непрерывны на отрезке [a; b]; 3) $S(x_i) = f(x_i)$, $i = 0,1,2,\ldots,n$. соответствует условию интерполирования [4].

Целью статьи является сравнительный анализ результатов приближенного вычисления с точными данными при использовании операции интерполирования методом кубического сплайна. Сравнительный анализ погрешностей и расхождений производится на основе визуализации расчетных зависимостей в виде графиков найденных интерполяционных полиномов.

Материал и методы исследования

Для примера нахождения и построения кубического сплайна [5] рассмотрим функцию $f(x) = sinx$. Используем два метода получения приближенных значений для указанной функции. В первом случае реализуем построение кубического сплайна вручную, а полученную систему уравнений решим методом обратной матрицы в числовом редакторе Excel. Во втором случае используем реализацию построения кубического сплайна с помощью языка программирования Python.

Первый метод

Определим для указанной функции три узла интерполирования. На каждом узле составим кубический сплайн, используя таблицу значений функции $f(x) = sinx$, $x \in [0; \pi / 2]$.

Значения функции $f(x) = sinx$, $x \in [0; \pi / 2]$

x	0	$\pi / 6$	$\pi / 3$	$\pi / 2$
$sin\ x$	0	1/2	$\sqrt{3} / 2$	1

Уравнения сплайнов для трех узлов интерполирования с учетом исходных данных имеют вид:

$$\begin{aligned} S_1 &= a_1 + b_1(x-0) + c_1(x-0)^2 + d_1(x-0)^3, \\ S_2 &= a_2 + b_2\left(x-\frac{\pi}{6}\right) + c_2\left(x-\frac{\pi}{6}\right)^2 + d_2\left(x-\frac{\pi}{6}\right)^3, \\ S_3 &= a_3 + b_3\left(x-\frac{\pi}{3}\right) + c_3\left(x-\frac{\pi}{3}\right)^2 + d_3\left(x-\frac{\pi}{3}\right)^3. \end{aligned} \quad (2)$$

Коэффициенты a_i, b_i, c_i, d_i с учетом трех условий прохождения через узловые точки и с учетом преобразований имеют вид:

$$\begin{aligned} 0 &= a_1, \\ 1 &= 2a_1 + \frac{\pi}{3}b_1 + \frac{\pi^2}{18}c_1 + \frac{\pi^3}{108}d_1, \\ 1/2 &= a_2, \\ \sqrt{3} &= 2a_2 + \frac{\pi}{3}b_2 + \frac{\pi^2}{18}c_2 + \frac{\pi^3}{108}d_2, \\ \sqrt{3} &= 2a_3, \\ 1 &= a_3 + \frac{\pi}{6}b_3 + \frac{\pi^2}{36}c_3 + \frac{\pi^3}{216}d_3. \end{aligned} \quad (3)$$

Также необходимо выполнение следующих условий.

1. В стыках между сплайнами должна обеспечиваться гладкость. В узлах не должно быть изломов. Отсюда следует, что в узлах интерполирования должны быть одинаковыми первые производные соседних сплайнов, соответствующих системе уравнений:

$$\begin{aligned} S_1' &= b_1 + 2c_1(x-x_0) + 3d_1(x-x_0)^2, \\ S_2' &= b_2 + 2c_2(x-x_1) + 3d_2(x-x_1)^2, \\ S_3' &= b_3 + 2c_3(x-x_2) + 3d_3(x-x_2)^2, \end{aligned} \quad (4)$$

Сплайны S_1 и S_2 стыкуются в точке $x_1 = \pi / 6$. После подстановки в систему уравнений (4) значения точки x_1, а также данных таблицы равенство $S_1'(x_1) = S_2'(x_1)$ принимает вид:

$$b_1 + 2c_1\frac{\pi}{6} + 3d_1\frac{\pi}{36}^2 = b_2. \quad (5)$$

Аналогично со сплайнами S_2 и S_3, которые стыкуются в точке $x_2 = \pi / 3$.

a1	0	a2	0,5	a3	0,866025404
b1	0,985517759	b2	0,893753458	b3	0,401428466
c1	0	c2	-0,175256904	c3	-0,76501457
d1	-0,11157201	d2	-0,375451391	d3	0,487023401

Рис. 1. Коэффициенты a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2 и a_3, b_3, c_3, d_3 в числовом редакторе Excel

Равенство $S_2'(x_2) = S_3'(x_2)$ принимает вид:

$$b_2 + 2c_2\frac{\pi}{6} + 3d_2\frac{\pi^2}{36} = b_3. \quad (6)$$

2. В узлах, где стыкуются сплайны, должна быть одинаковой кривизна соседних сплайнов. Это означает равенство вторых производных, соответствующих системе уравнений:

$$\begin{aligned} S_1'' &= 2c_1 + 6d_1(x - x_0), \\ S_2'' &= 2c_2 + 6d_2(x - x_1), \\ S_3'' &= 2c_3 + 6d_3(x - x_2). \end{aligned} \quad (7)$$

Сплайны S_1 и S_2 стыкуются в точке $x_1 = \pi/6$, а сплайны S_2 и S_3 – в точке $x_2 = \pi / 3$. После подстановки в систему уравнений (7) значения точки x_1, а также данных таблицы получаем:

$$\begin{aligned} 2c_1 + 6d_1\frac{\pi}{6} &= 2c_2, \\ 2c_2 + 6d_2\frac{\pi}{6} &= 2c_3. \end{aligned} \quad (8)$$

3. Необходимо задать поведение сплайнов на левой и правой границах, то есть в точках x_0 и x_3. Для этого необходимо задать нулевую кривизну (нулевые значения вторых производных). В точке $x_0 = 0$: $S_1''(x_0) = 0$, а в точке $x_2 = \pi / 2$: $S_3''(x_3) = 0$. После подстановки в систему уравнений (3) с учетом преобразований получаем:

$$\begin{aligned} 2c_1 &= 0, \\ 2c_3 + 6d_3\frac{\pi}{6} &= 0. \end{aligned} \quad (9)$$

В результате получили систему уравнений (3), (5), (6), (8), (9) с 12 неизвестными.

Такую систему удобно решить в числовом редакторе Excel методом обратной матрицы [6]. Найденные искомые коэффициенты представлены на рисунке 1.

Второй метод.

Используем для построения кубического сплайна высокоуровневый язык программирования Python [7]. Программный код должен быть написан таким образом, чтобы можно было выбрать любую функцию для построения кубического сплайна. Для нашего случая будем рассматривать также функцию $f(x) = sinx$ на тех же узлах интерполирования. Программный код представлен на рисунках 2, 3. В структуру кода добавлены комментарии, поясняющие алгоритм построения кубического сплайна.

Результаты исследования и их обсуждение

После запуска программы в интегрированной среде IDLE пользователь должен ответить на вопрос, что именно он хочет сделать. При этом пользователю доступны варианты с вводом любого набора точек функции и вводом любой функции. В примере на рисунке 4 выбрана функция $f(x) = sinx$. Далее вводятся значения аргументов по порядку, которые будут соответствовать каждому из трех узлов интерполирования. Следующим шагом программа строит три интерполяционных многочлена третьей степени и подставит в них посчитанные коэффициенты.

Сравнительный анализ коэффициентов, полученных в результате реализации программного кода на языке Python, и коэффициентов, полученных в результате решения системы уравнений методом обратной матрицы в числовом редакторе Excel, показывает наличие расхождений при вычислениях.

При расчетах методом обратной матрицы в числовом редакторе Excel были получены следующие значения коэффициентов:

$$a_1 = 0,\ b_1 = 0,985517759,$$

$$c_1 = 0,\ d_1 = -0,11157201$$

$$a_2 = 0,5,\ b_2 = 0,893753458,$$

$$c_2 = -0,175256904,\ d_2 = -0,375451391,$$

$$a_3 = 0,866025404,\ b_3 = 0,401428466,$$

$$c_3 = -0,76501457,\ d_3 = 0,487023401.$$

```
from math import *
import numpy
import matplotlib.pyplot as plt
def reshen(x,y): #создадим функцию, которая будет находить коэффициенты сплайнов
    raz=(len(x)-1)*3-1 #количество переменных, без "a" и "c0", значение которых можно найти сразу
    k1=[[0.0 for i in range(raz)] for b in range(raz)] #левая матрица коэффициентов
    k2=[0.0 for i in range(raz)] #правая матрица коэффициентов
    spl=len(x)-1 #кол-во сплайнов
    #начало получения матриц
    #из условия прохождения сплайнами заданных точек
    #коэффициенты перед b0, d0
    k1[0][0]=x[1]-x[0]
    k1[0][1]=(x[1]-x[0])**3
    #соответствующая правая часть уравнения
    k2[0]=y[1]-y[0]
    for i in range(1,spl): #данный цикл находит коэффициенты перед bi, ci, di и соответствующую правую часть уравнения
        t=x[i+1]-x[i]
        tri=3*i
        k1[i][tri-1]=t
        k1[i][tri]=t**2
        k1[i][tri+1]=t**3
        k2[i]=y[i+1]-y[i]
    #из условий равенства углов и выпуклостей
    #коэффициенты перед b0, d0, b1
    k1[spl][0]=1.0
    k1[spl][1]=3*((x[1]-x[0])**2)
    k1[spl][2]=-1.0
    for i in range(1,spl-1):
        n=spl+i
        t=x[i+1]-x[i]
        tri=3*i
        k1[n][tri-1]=1.0 #данная часть цикла находит коэффициенты перед bi, ci, di, b(i+1)
        k1[n][tri]=2*t
        k1[n][tri+1]=3*(t**2)
        k1[n][tri+2]=-1.0
        n=2*spl-1+i # данная часть цикла находит коэффициенты перед ci, di, c(i+1)
        k1[n][tri]=2.0
        k1[n][tri+1]=6*t
        k1[n][tri+3]=-2.0
    n=2*spl-1
    #коэффициенты перед d0, c1
    k1[n][1]=6*(x[1]-x[0])
    k1[n][3]=-2.0
    #из условия поведения в конечной точке
    k1[-1][-2]=2.0
    k1[-1][-1]=6*(x[-1]-x[-2])
    #конец получения матриц
    return numpy.linalg.solve(k1,k2) #данная функция решает систему линейных уравнений
def vivod(x,y,resh,text=""): #создадим функцию, которая будет выводить сплайны и вычислять приближённое значение с помощью них
    print(f"S0={y[0]}+{resh[0]}(x-{x[0]})+{resh[1]}(x-{x[0]})^3")
    for i in range(1,len(x)-1):
        tri=3*i
        print(f"S{i}={y[i]}+{resh[tri-1]}(x-{x[i]})+{resh[tri]}(x-{x[i]})^2+{resh[tri+1]}(x-{x[i]})^3")
    print (f"Если вы хотите вычислить приближённое значение функции с помощью сплайнов, введите \"1\".
Если вы хотите построить график{text}, введите \"2\".
Если хотите закончить работу программы, введите что-то другое.")
    inp=input()
    if inp=="1":
        while True:
            arg=eval(input("Введите значение аргумента функции: "))
            if x[0]<=arg and x[-1]>=arg:
                for i in range(len(x)):
                    tri=3*i
                    if x[i]<=arg and arg<=x[i+1]:
                        if i==0:
                            print(f"Результат: {y[0]+resh[0]*(arg-x[0])+resh[1]*(arg-x[0])**3}")
                        else:
                            print(f"Результат: {y[i]+resh[tri-1]*(arg-x[i])+resh[tri]*(arg-x[i])**2+resh[tri+1]*(arg-x[i])**3}")
                        break
            else:
                print("Значение аргумента выходит за область, в которой определены сплайны.")
            print(f"Если хотите ввести другой аргумент, введите \"1\".
Если хотите построить график{text}, введите \"2\".
Если хотите закончить работу программы, введите что-то другое.")
            inp=input()
            if inp=="1":
                pass
            elif inp=="2":
                return True
                break
            else:
                return False
                break
        return False
    elif inp=="2":
        return True
    else:
        return False
```

Рис. 2. Листинг реализации программного кода на языке программирования Python

```
def grspl(x,y,resh): #создадим функцию, которая будет возвращать точки для построения графиков сплайнов
    grX=[]
    grY=[]
    for b in range(1001):
        arg=x[0]+((x[-1]-x[0])/1000)*b
        for i in range(len(x)-1):
            tri=3*i
            if x[i]<=arg and arg<=x[i+1]:
                if i==0:
                    grX.append(arg)
                    grY.append(y[0]+resh[0]*(arg-x[0])+resh[1]*(arg-x[0])**3)
                else:
                    grX.append(arg)
                    grY.append(y[i]+resh[tri-1]*(arg-x[i])+resh[tri]*(arg-x[i])**2+resh[tri+1]*(arg-x[i])**3)
    return [grX,grY]
inp=input("Здравствуйте, Пользователь!
Если хотите ввести координаты точек, введите \"1\".
Если хотите ввести функцию и её аргументы, введите \"2\".
Если хотите закончить работу программы введите что-то другое.")
if inp=="1":
    x=list(map(eval,
    input("Введите координату x каждой точки по порядку через пробел:\n").split()))
    y=list(map(eval,
    input("Введите соответствующую координату y каждой точки через пробел:\n").split()))
    if len(x)<3:
        print("Количество x должно быть равно количеству y.")
    elif len(x)==len(y):
        resh=reshen(x,y)
        if vivod(x,y,resh):
            gr=grspl(x,y,resh)
            plt.subplot()
            plt.title("Сплайны")
            plt.plot(gr[0],gr[1],'-',x,y,'go')
            plt.grid(True)
            plt.show()
    else:
        print("Количество x должно быть равно количеству y.")
elif inp=="2":
    funcinp=input("Доступны все функции модуля math.
Примеры ввода: sin(X)*cos(X)+X; log(X,3); 2*exp(X)+X**pi.
В качестве аргумента необходимо указать "X".
Введите функцию: ")
    x=list(map(eval,input("Введите значения аргументов по порядку через пробел:\n").split()))
    y=[]
    gr1=[[],[]]
    for i in range(1001): #получаем точки для построения графика функции
        arg=x[0]+((x[-1]-x[0])/1000)*i
        func=funcinp
        while True:
            try:
                func=func[:func.index("X")]+str(arg)+func[func.index("X")+1:] #заменяем все X на нужное значение
            except ValueError:
                break
        gr1[0].append(arg)
        gr1[1].append(eval(func))
    for i in range(len(x)): #получаем координату y для точек, по которым будем строить сплайны
        func=funcinp
        while True:
            try:
                func=func[:func.index("X")]+str(x[i])+func[func.index("X")+1:]
            except ValueError:
                break
        y.append(eval(func))
    resh=reshen(x,y)
    if vivod(x,y,resh,"и"):
        gr0=grspl(x,y,resh)
        plt.subplot(131)
        plt.plot(gr0[0],gr0[1],'-',x,y,'go')
        plt.title("Сплайны")
        plt.grid(True)
        plt.subplot(132)
        plt.plot(gr1[0],gr1[1],'-r',x,y,'go')
        plt.title(funcinp)
        plt.grid(True)
        plt.subplot(133)
        plt.plot(gr0[0],gr0[1],'-',label="Сплайны")
        plt.plot(gr1[0],gr1[1],'-r',label=funcinp)
        plt.plot(x,y,'go')
        plt.title("Вместе")
        plt.legend()
        plt.grid(True)
        plt.show()
```

Рис. 3. Листинг реализации программного кода на языке программирования Python

Рис. 4. Демонстрация работы программы с построенными многочленами третьей степени

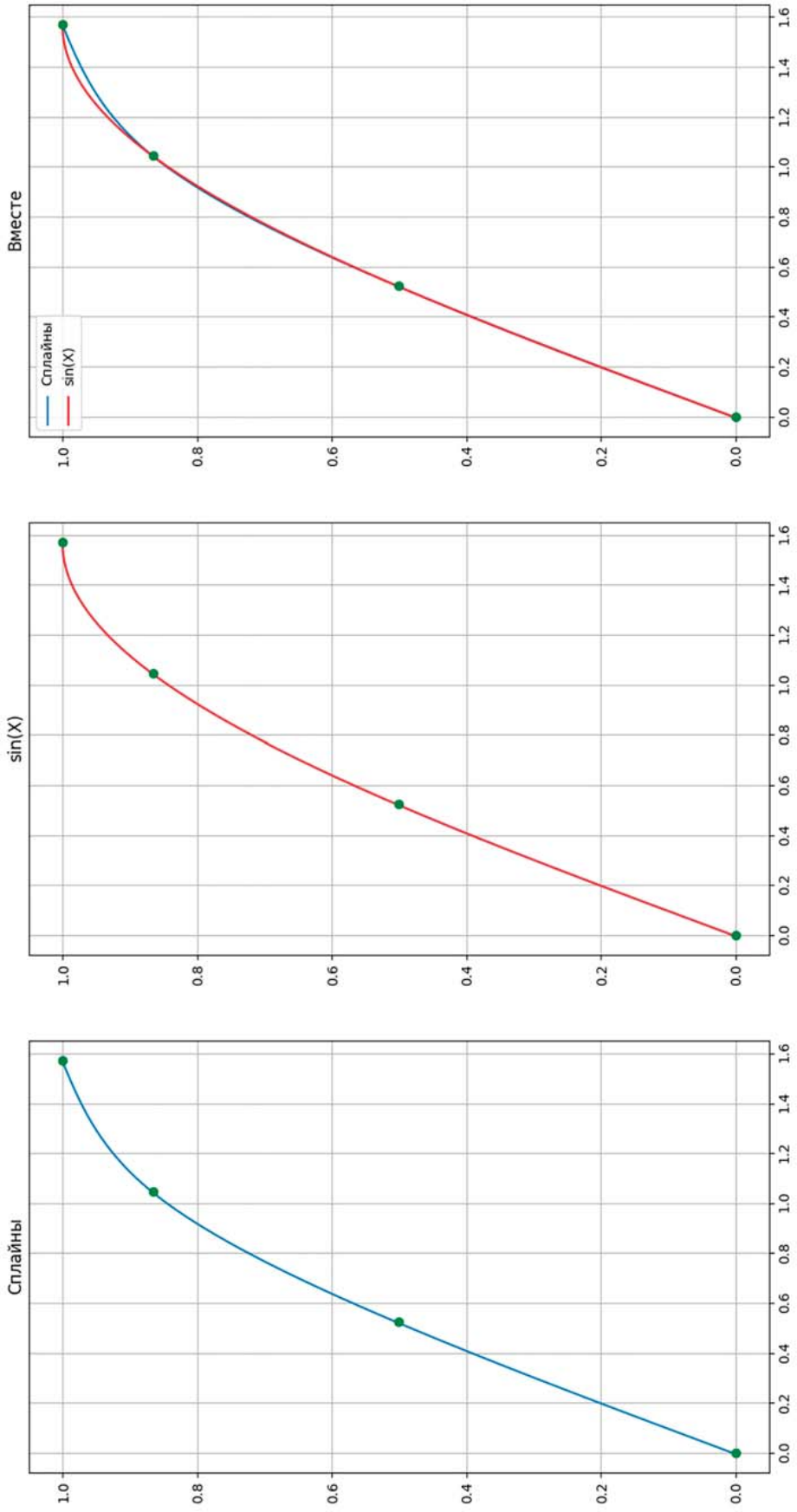

Рис. 5. Графики интерполяционных многочленов, функции f(x) = sinx и их наложения друг на друга

При вычислении коэффициентов с помощью реализации метода интерполирования на языке программирования Python получились следующие значения:

$$a_1 = 0,\ b_1 = 0,9936167336495609,$$

$$c_1 = 0,\ d_1 = -0,1411135286619067,$$

$$a_2 = 0,49999999999999994,$$

$$b_2 = 0,877555508354994,$$

$$c_2 = -0,2216606124831894,$$

$$d_2 = -0,22774379814282397,$$

$$a_3 = 0,8660254037844386,$$

$$b_3 = 0,4581212917265916,$$

$$c_3 = -0,5793997340562557,$$

$$d_3 = 0,3688573268047306.$$

По полученным данным можно построить график найденных интерполяционных полиномов в виде кубического сплайна и график функции. На рисунке 5 показаны график функции $f(x) = sinx$, график интерполяционных многочленов и график наложения функций друг на друга.

На рисунке 3 видно, что график точного значения функции и значения, рассчитанного приближенным методом с помощью интерполирования кубическими сплайнами, расходятся.

Вывод

В результате работы, выполненной в статье, можно сделать вывод о том, что различные методы расчета приближенных значений для функций дают разные результаты и в любом методе имеются свои погрешности и расхождения. Но так или иначе реализация метода интерполирования сплайнами с использованием элементов Индустрии 4.0 [8, 9] с помощью языка программирования Python дает более точные данные, которые подтверждаются наименьшим расхождением графиков. Также к плюсам второго метода исследования можно отнести меньшую трудозатратность за счет полностью автоматических расчетов программой.

Список литературы

1. Лиманова Н.И., Тренина М.А. Об использовании сплайн-функций в перколяционных исследованиях // Информационные системы и технологии: управление и безопасность. 2012. № 1. С. 192-199.

2. Sun M., Lan L., Zhu C., Lei F. Cubic spline interpolation with optimal end conditions // Journal of Computational and Applied Mathematics. 2023. Vol. 425. P. 115039.

3. Ahmad N., Khan F.D. Study of Cubic B Spline Interpolation // Journal of Ultra Scientist of Physical Sciences. 2019. Vol. 31(2). P. 4-10.

4. Муратов Е.Р. Алгоритмы предварительной обработки изображений в системах комбинированного видения летательных аппаратов: дис. … канд. техн. наук. Рязань, 2013. 177 с.

5. Петрова Е.В. Вариант реализации сплайн-интерполяции // Системы компьютерной математики и их приложения. 2016. № 17. С. 19-21.

6. Лавренов А.Н. Метод нахождения обратной матрицы порядка N // Журнал вычислительной математики и математической физики. 2012. Т. 52, № 6. С. 979.

7. Хахаев И.А. Практикум по алгоритмизации и программированию на Python. М.: ИНТУИТ, 2016. 179 с.

8. Kurasov D.A. Digital technologies Industry 4.0// CEUR Workshop Proceedings. 2021. Vol. 2843.

9. Kurasov D.A. Computer-aided manufacturing: Industry 4.0 // IOP Conference Series: Materials Science and Engineering. 2021. Vol. 1047. P. 012153.