БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Отчет

Методы численного анализа

Лабораторная работа 3

Выполнила

Юрковская Екатерина Артуровна

Студентка 2 курса 3 группы

Минск 2020

1) Постановка задачи.

Постановка задачи. Для заданной функции $f : [a, b] \to \mathbb{R}$ (берется из предыдущей лабораторной работы):

- Произвести интерполяцию кубическими сплайнами на отрезке $[-2, 2]$ по равноотстоящим узлам с естесственными граничными условиями.
- Построить графики получившихся приближений для сеток с количеством узлов, равным $N_i = 10i, \quad i = 1, 2, \dots, 10$. На графике должны быть изображены построенное приближение и исходная функция.
- Для каждого построения экспериментально определить максимум-норму погрешности: взять сетку из 1000 равноотстоящих узлов и определить максимум величины $|f(x_i) - S(x_i)|, \quad i = 1, \dots, 1000$. При каждом вычислении нормы замерять затраченное время с точностью до милисекунд.
- Используя программу из лабораторной работы №2 получить аналогичные данные для интерполяционного многочлена из предыдущей лабораторной работы (используются чебышовские узлы). Результат представить в виде таблицы:

Примечание: $f(x) = (\cos x)^2 - x$

2) Теоретические сведения

На каждом отрезке $[x_{i-1}, x_i]$, $i = \overline{1, N}$ функция $S(x)$ есть полином третьей степени $S_i(x)$, коэффициенты которого надо определить. Запишем для удобства $S_i(x)$ в виде:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Коэффициенты будем искать по следующим формулам:

$$a_i = f(x_i);$$

$$d_i = \frac{c_i - c_{i-1}}{3 \cdot h_i};$$

$$b_i = \frac{a_i - a_{i-1}}{h_i} + \frac{2 \cdot c_i + c_{i-1}}{3} \cdot h_i;$$

$$c_{i-1} \cdot h_i + 2 \cdot c_i \cdot (h_i + h_{i+1}) + c_{i+1} \cdot h_{i+1} = 3 \cdot \left(\frac{a_{i+1} - a_i}{h_{i+1}} - \frac{a_i - a_{i-1}}{h_i} \right),$$

причем $c_N = S''(x_N) = 0$ и $c_1 - 3 \cdot d_1 \cdot h_1 = S''(x_0) = 0$.

Если учесть, что $c_0 = c_N = 0$, то вычисление c можно провести с помощью метода прогонки для трёхдиагональной матрицы.

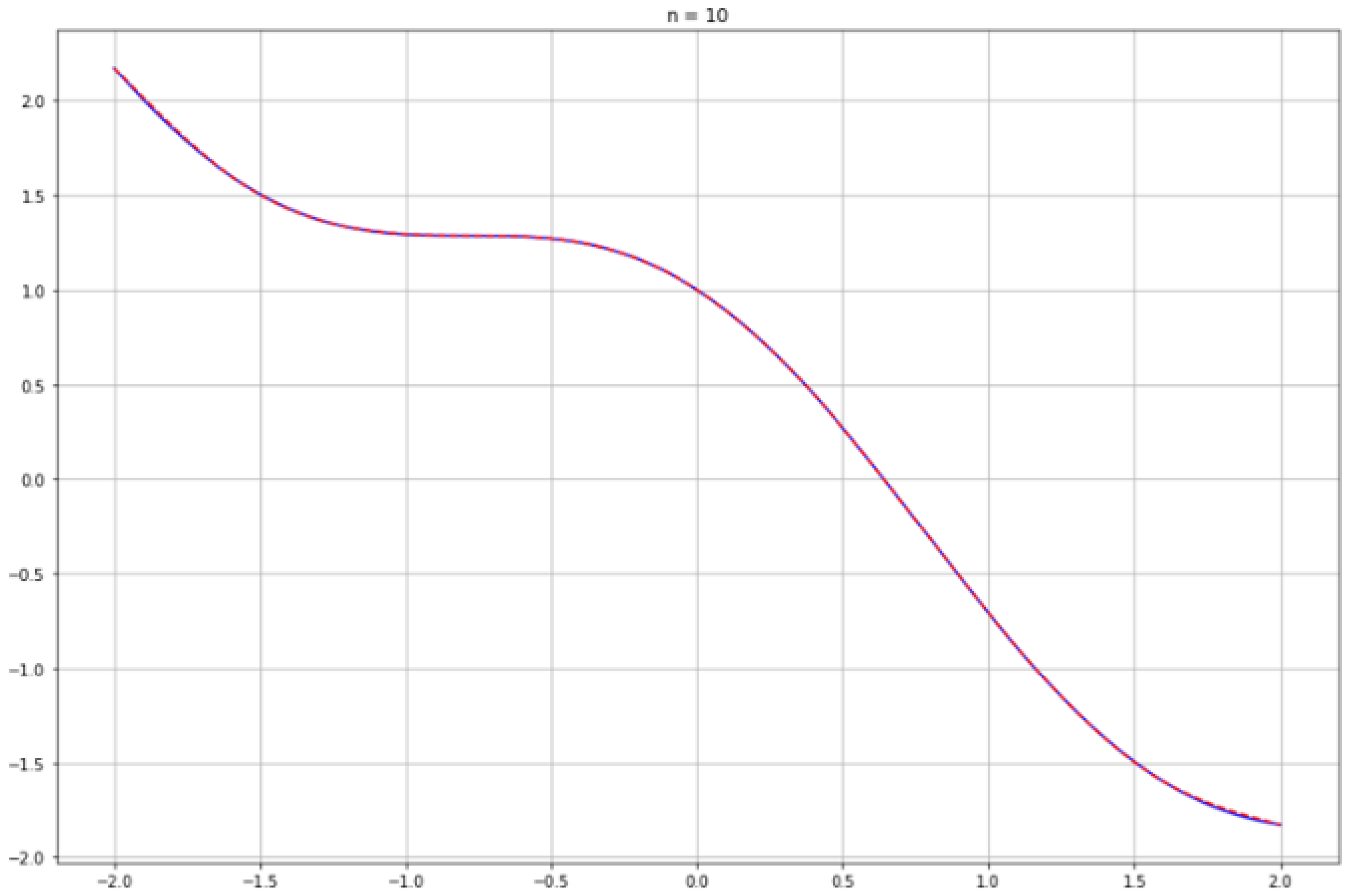
3) Исходный код программы.

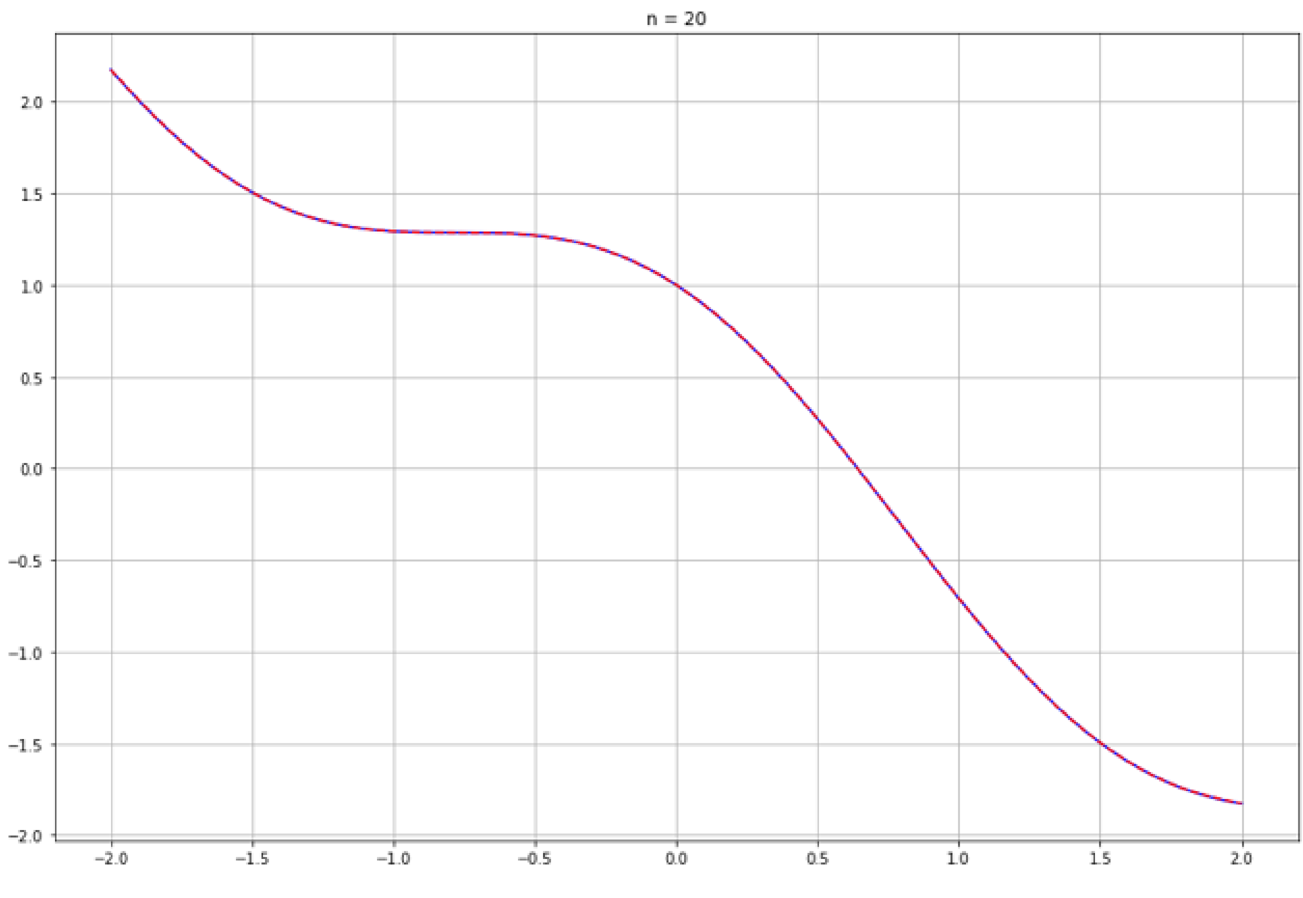
```
import numpy as np
import matplotlib.pyplot as plt
```

```
import math
from pylab import *
from time import time

#исходная функция
func_x = np.linspace(-2.0, 2.0, 1000)
func_y = [ math.cos(x)* math.cos(x) - x for x in func_x]

def progonka(c,values,n):
    #матрица неизвестных коэффициентов
    a = [0.0 for x in range(0, n - 1)]
    b = [0.0 for x in range(0, n - 1)]
    #шаг h[i]==h[i+1], тк узлы раноотстоящие
    h = 4.0/ (n-1)

    #прямой ход метода прогонки
    for i in range(1, n - 1):
        koef_a = h
        koef_b = h
        koef_c = 2.0 * 2* h

        F = 6.0/ h*((values[i + 1] - values[i]) - (values[i] - values[i - 1]))
        z = (koef_a * a[i - 1] + koef_c)
        a[i] = -koef_b / z
        b[i] = (F - koef_a * b[i - 1]) / z

    #обратный ход мп
    for i in range(n - 2, 0, -1):
        c[i] = a[i] * c[i+1] + b[i]


def makePoly(P,nodes,values,n):

    x_i=nodes
    a=values
    b=[0.0 for x in range(n)]
    c=[0.0 for x in range(n)]
    d=[0.0 for x in range(n)]


    c[0]=c[n-1]=0
    progonka(c,values,n)
    h=4.0/(n-1)
    for i in range(n - 1, 0, -1):
        d[i] = ((c[i] - c[i-1]) / h)
        b[i]=(values[i] - values[i - 1]) / h + c[i] * h / 2 - d[i] * h ** 2 / 6

    for i in range (n):
        c[i]=c[i]/2.0
        d[i]=d[i]/6.0

    for i in range (n):
        P[i][0]=a[i]
        P[i][1]=b[i]
        P[i][2]=c[i]
        P[i][3]=d[i]


def calc_f(x, x_i, a,b,c,d):
    return a + b * (x- x_i) + c * (x- x_i)**2+ d* (x - x_i)**3

def calc_spline(x,nodes,P,n):

    for i in range (1,n):
        if  x<=nodes[i] and x>=nodes[i-1]:
            return calc_f(x,nodes[i],P[i][0],P[i][1],P[i][2],P[i][3])
```

```
def draw_cubic_spline(x, y, n):
    fig = plt.figure(figsize=(15,10))
    ax = plt.subplot(111)
    plt.grid(True)
    plt.title('n = {}'.format(n))
    ax.plot(func_x, func_y, 'b-', label='function')
    ax.plot(x, y, 'r--', label='cubic spline')
    plt.show()

def find_max(y):
    max_n= 0
    for i in range(1000):
        max_n = max(max_n, abs(func_y[i] - y[i]))
    print(max_n)

n=10
# получаем узлы
nodes = np.linspace(-2.0, 2.0, n+1)
values = [ math.cos(x)* math.cos(x) - x for x in nodes]


#матрица с коэффициентами полинома
P=[[0.0 for x in range(4)] for x in range(n+1)]

time1=time()
makePoly(P,nodes,values,n+1)
time2=time()


x=func_x
y=[calc_spline(x,nodes,P,n+1) for x in x]


draw_cubic_spline(x, y,n)

time3=time()
find_max(y)
time4=time()

print(time4-time3+time2-time1)
```

4) Графики.


n = 10
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


n = 20
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Примечание: для n= 20,30, 40, … , 100 графики функций выглядят одинаково

5) Таблица

N	Норма (сплайн)	Норма (чебышевские узлы)	Время (сплайн)	Время (чеб.узлы)
10	0.011348720561880077	0.00020785128659173147	0.0004994869232177734	0.061063289642333984
20	0.002626773812326899	0.00020785128659173147	0.001020193099975586	0.061063289642333984
30	0.0011517777559326703	0.00020785128659173147	0.0004999637603759766	0.061063289642333984
40	0.0006440220318140177	0.00020785128659173147	0.0010018348693847656	0.061063289642333984
50	0.0004113359702437869	0.00020785128659173147	0.002001047134399414	0.061063289642333984
60	0.000285292952087568	0.00020785128659173147	0.002001523971557617	0.061063289642333984
70	0.00020889434163451348	0.00020785128659173147	0.0020029544830322266	0.061063289642333984
80	0.0001603301796420098	0.00020785128659173147	0.0010001659393310547	0.061063289642333984
90	0.00012662906978966326	0.00020785128659173147	0.0015001296997070312	0.061063289642333984
100	0.00010256076537640268	0.00020785128659173147	0.002001047134399414	0.061063289642333984