Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы численного анализа»

**ОТЧЕТ**

к лабораторной работе №7

на тему:

**«Интреполяция Сплайнами»**

БГУИР 1-40 04 01

|  |
| --- |
| Выполнил студент группы 253505  БЕКАРЕВ Станислав Сергеевич |
|  |
| (дата, подпись студента) |
| Проверил доцент кафедры информатики  АНИСИМОВ Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

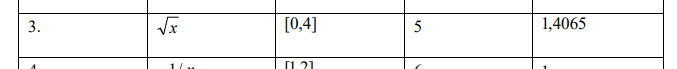
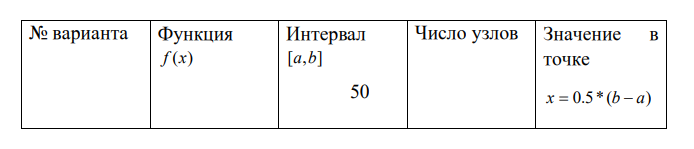
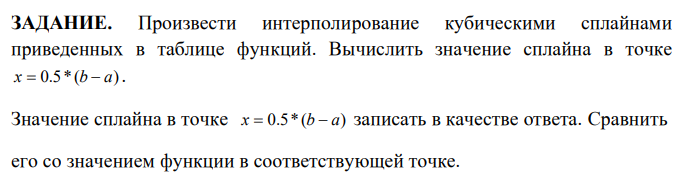
Минск 2023

**Содержание**

1. Цель работы
2. Задание
3. Программная реализация
4. Полученные результаты
5. Оценка полученных результатов
6. Вывод

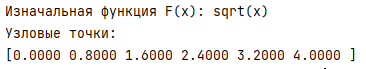
**Цель работы**

* изучить построение кубических интерполяционных сплайнов ;
* составить программу построения кубических интерполяционных сплайнов;
* выполнить тестовые примеры и проверить правильность работы программы



**Вариант 3**

*Исходные данные*



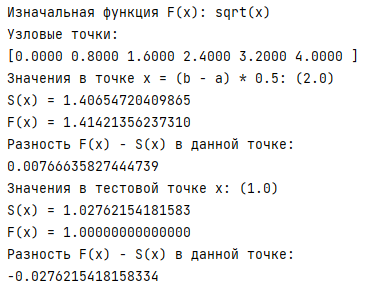
**Программная реализация**

def spline\_method(left, right, point\_amount, func):  
 point\_amount += 2  
 X = [], Y = [], A = [], B = [], C = [], D = [], S = []

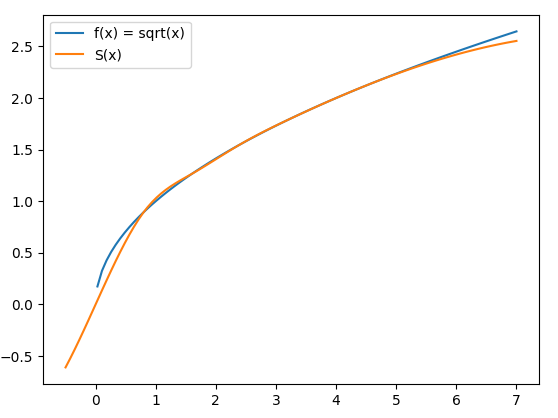
alpha = []

betta = []  
 for i in range(point\_amount):  
 X.append(left + (right - left)/(point\_amount - 1) \* i)  
 Y.append(func.subs(x, X[i]))  
  
 alpha.append(-1 \* (X[2] - X[1]) / (2 \* (X[2] - X[0])))  
 betta.append((((Y[2] - Y[1]) / (X[2] - X[1])) - ((Y[1] - Y[0]) / (X[1] - X[0]))) / (X[2] - X[0]) \* 3 / 2)  
 for i in range(1, point\_amount - 2):  
 a = (X[i + 1] - X[i]) / 3  
 b = 2 \* ((X[i + 2] - X[i]) / 3)  
 c = (X[i + 2] - X[i + 1]) / 3  
 d = (((Y[i + 2] - Y[i + 1]) / (X[i + 2] - X[i + 1])) - ((Y[i + 1] - Y[i]) / (X[i + 1] - X[i])))  
 alpha.append(-1 \* c / (a \* alpha[i - 1] + b))  
 betta.append((d - a \* betta[i - 1]) / (a \* alpha[i - 1] + b))  
 alpha.reverse()  
 betta.reverse()  
 C.append(betta[0])  
 for i in range(1, point\_amount - 1):  
 C.append(alpha[i - 1] \* C[i - 1] + betta[i - 1])  
 C.append(0.)  
 C.reverse()  
 C.append(0.)for i in range(point\_amount - 1):  
 B.append((Y[i + 1] - Y[i]) / (X[i + 1] - X[i]) - C[i] \* (X[i + 1] - X[i]) - ((C[i + 1] - C[i]) \* (X[i + 1] - X[i]) / 3))  
 D.append((C[i + 1] - C[i]) / (3 \* (X[i + 1] - X[i])))  
 A.append(Y[i])  
 for i in range(point\_amount - 1):  
 f = A[i] + B[i]\*(x - X[i]) + C[i]\*(x - X[i])\*\*2 + D[i]\*(x - X[i])\*\*3  
 S.append((f, x <= X[i+1]))  
 S = Piecewise(\*S)  
 return S, X

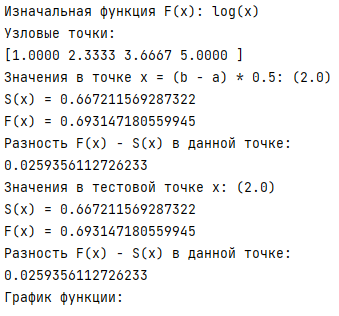
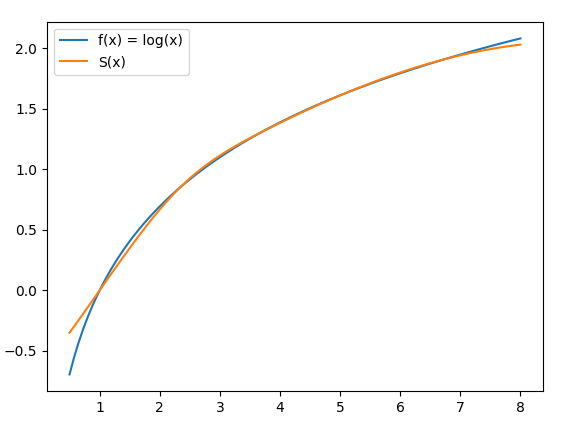
**Полученные результаты**



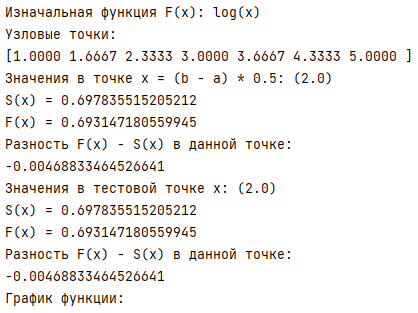
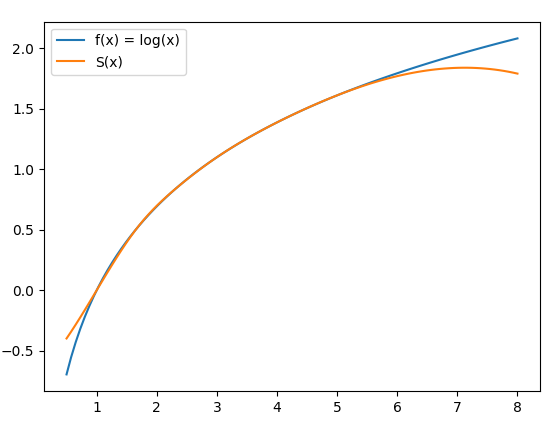
Графики функций:



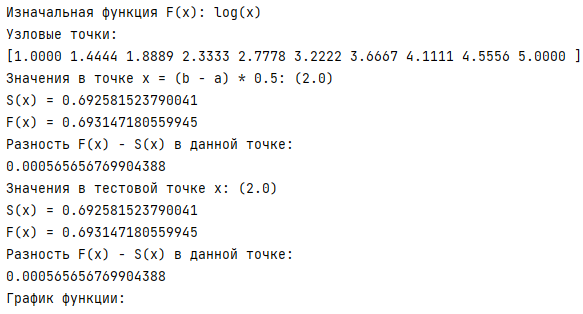
*Тестовый пример 1. Четыре узла*

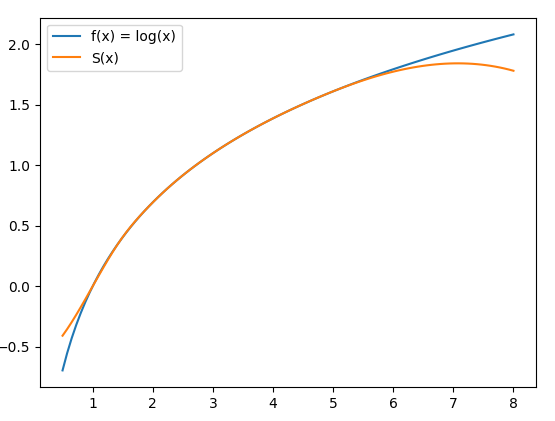
**

*Тестовый пример 2. Семь узлов*

**

*Тестовый пример 3. Десять узлов*



****

**Вывод**

В ходе выполнения лабораторной работы я изучил построения кубических интерполяционных сплайнов, написал программу их реализации на языке Python, правильность работы программы проверил на тестовых примерах.

На основании тестов можно сделать следующие выводы:

* Достоинством построения кубических интерполяционных сплайнов является то, что на каждом из отрезков (сплайнов) многочлен имеет определенную степень(в данном случае третью степень), что облегчает нахождение значения функции в каждой из точек.
* Точность интерполяции функции увеличивается с ростом количества узлов (точек Xi) отобранных для построения кубических интерполяционных сплайнов