ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина:

«Вычислительная математика»

Лабораторная работа №1

«Аппроксимация и интерполяция»

Выполнил:

Студент группы P3231

Колмаков Дмитрий Владимирович

Преподаватель:

Перл О.В.

Санкт-Петербург

2024г.

Оглавление

Задание

Дан набор точек, по которым необходимо построить аппроксимацию по методу сигмоид. Необходимо найти значение наибольшего отклонения среди заданных точек относительно полученной аппроксимации.

Формат входных данных:

x1 x2 x3 ...

y1 y2 y3 ...

где x1...xn - список значений аргумента для узлов интерполяции, y1...yn - список значений функции для соответствующего значения аргумента для узлов интерполяции. В тестах также вначале задаётся количество задаваемых точек, однако, в функцию этот параметр не передается.

Формат выходных значений: вещественное число, являющееся значением наибольшего отклонения исходных данных от полученной аппроксимации.

Теория

Аппроксимация и интерполяция - это два метода приближения функций или данных. Аппроксимация - это процесс замены сложной функции более простой для вычисления и анализа функцией, достаточно близкой к исходной. Интерполяцией же называют разновидность аппроксимации, при которой построенной функции обязательно проходит точно через заданные точки.

Среди методов аппроксимации можно выделить:

- Метод наименьших квадратов;
- Метод наименьших модулей;
- Метод сигмоид.

Среди методов интерполяции:

- Полином Лагранжа;
- Полином Ньютона;
- Кубические сплайны.

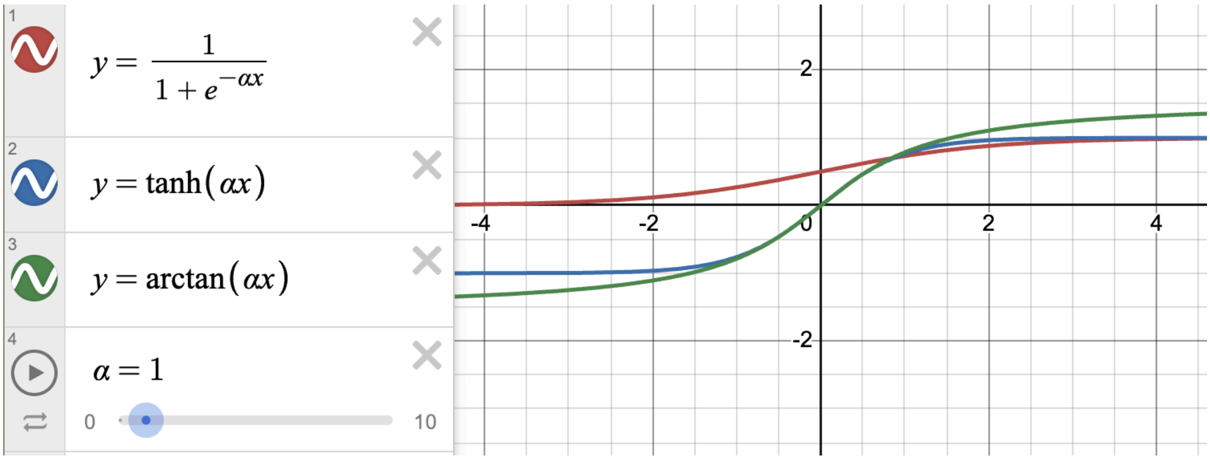
Каждый из методов имеет свои плюсы и минусы. Так, среди методов аппроксимации Метод наименьших квадратов позволяет найти модель, которая “наилучшим образом" соответствует данным, а Метод наименьших модулей дает меньшую чувствительность к выбросам, но при этом более трудоемкий.

Рассмотрим Метод сигмоид. В 1989 году Джордж Цыбенко доказал Универсальную теорему аппроксимации. Она гласит, что если дана любая непрерывная функция f, то ее можно аппроксимировать с любой заданной точностью с помощью любой непрерывной сигмоидной функции. В семейство функций класса сигмоид входят такие функции, как арктангенс, гиперболический тангенс и другие функции подобного вида:

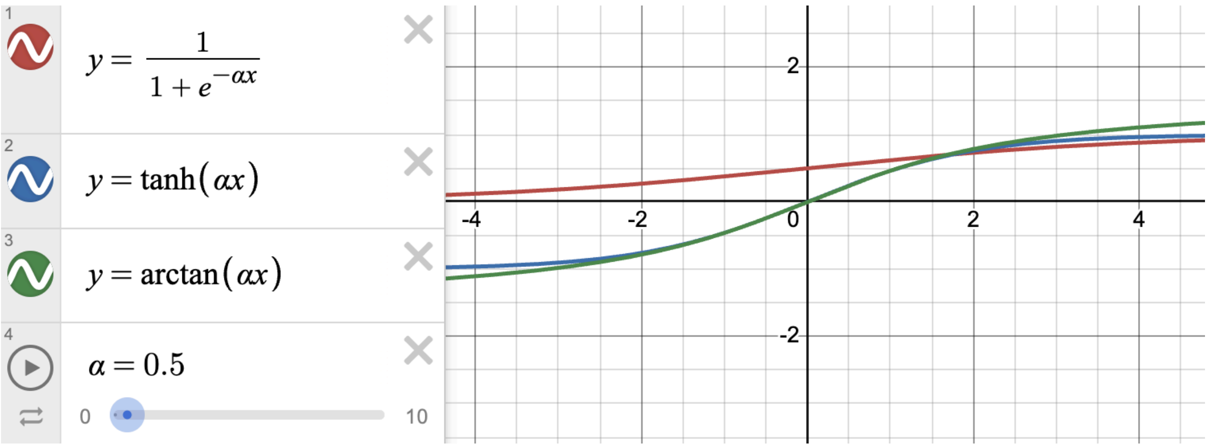
- $\sigma(x) = \frac{1}{1+e^{-\alpha x}},\ \alpha > 0;$
- $\sigma(x) = tanh(\alpha x) = \frac{e^{\alpha x}-e^{-\alpha x}}{e^{\alpha x}+e^{-\alpha x}},\ \alpha > 0;$
- $\sigma(x) = arctg(\alpha x), \alpha > 0$

и другие, причем:

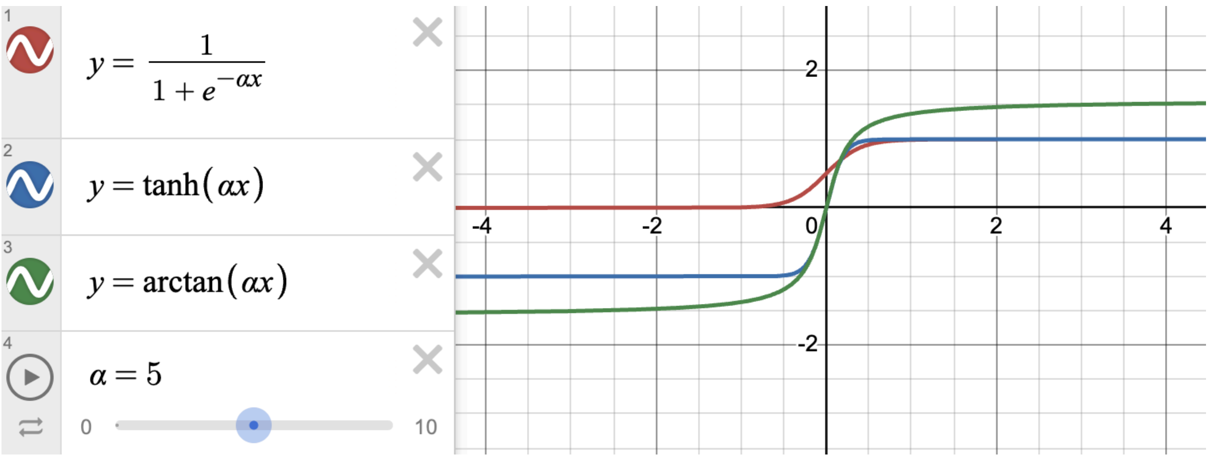
$$\begin{cases} \lim_{x\to-\infty} \sigma(x) = a \\ \lim_{x\to+\infty} \sigma(x) = b \end{cases}$$



При этом, при уменьшении параметра α сигмоиды становятся более гладкими:



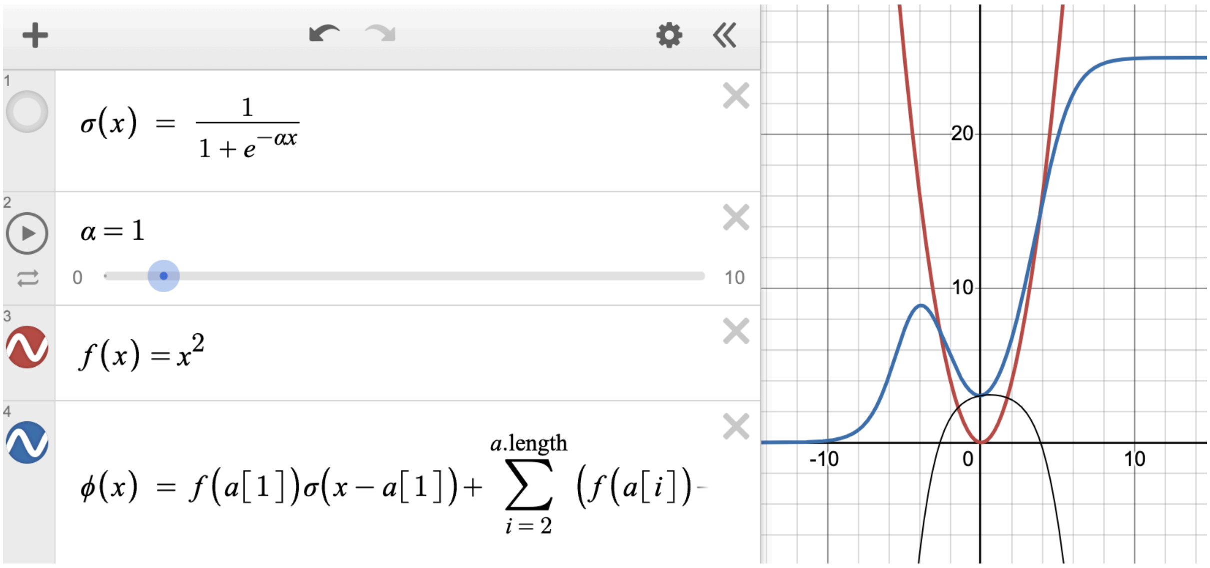
А при увеличении - наоборот:

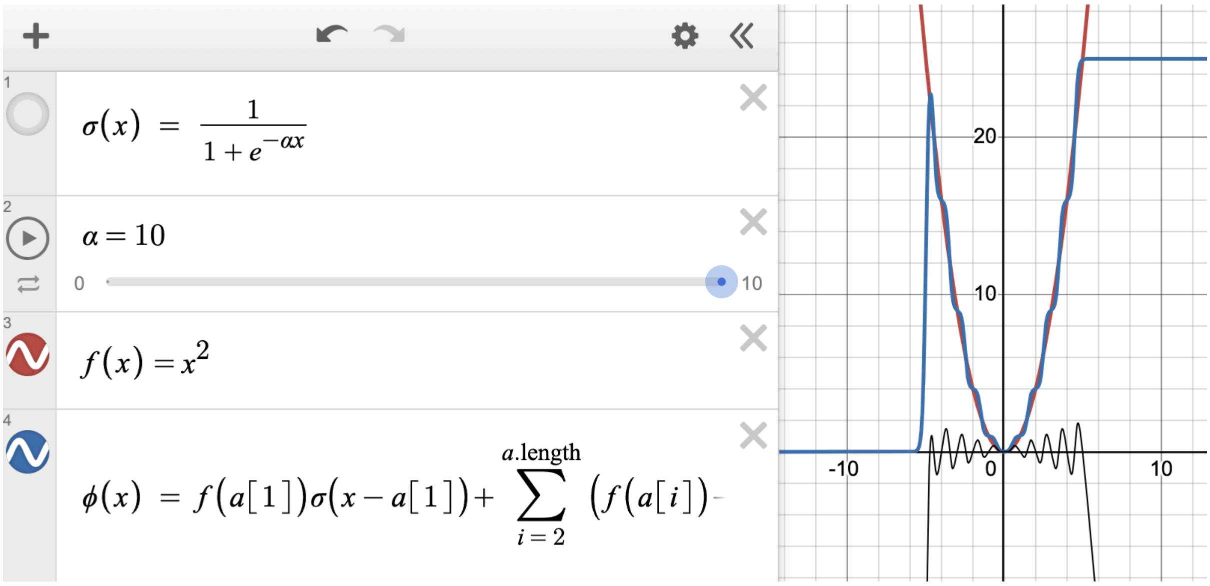


Тогда приближение будет иметь вид:

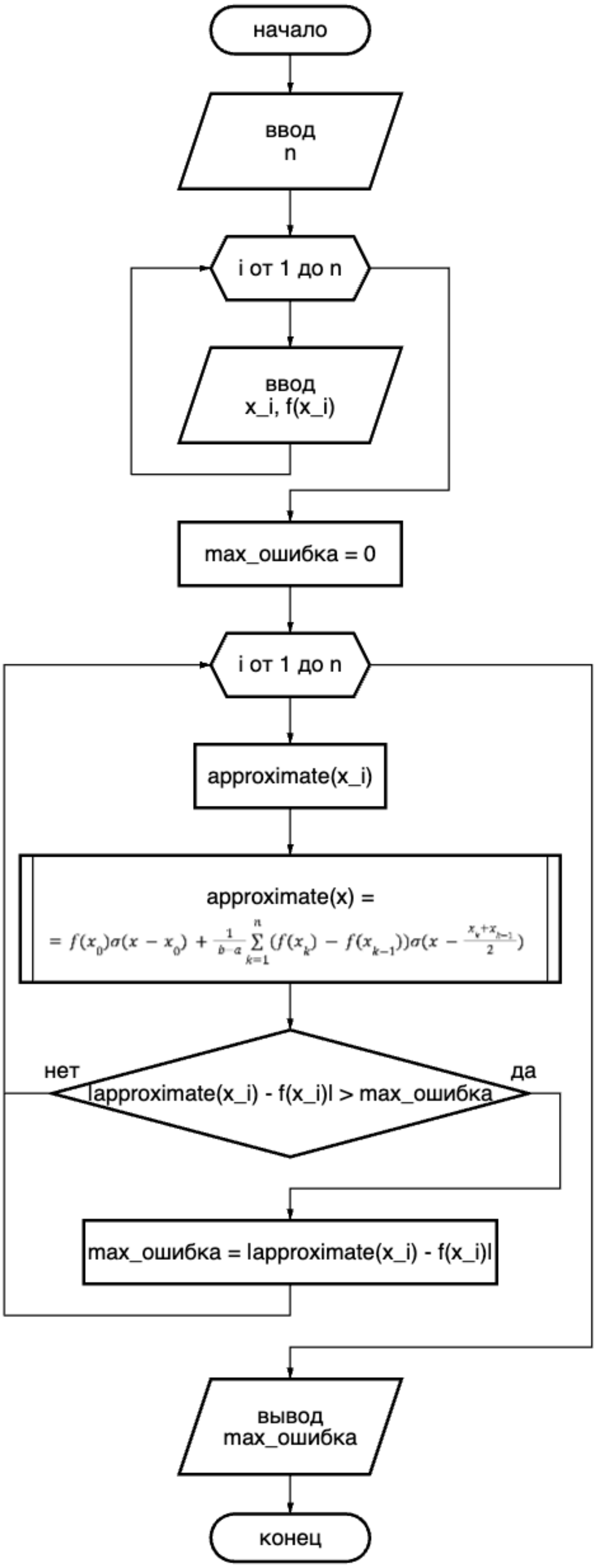
$$f(x) \approx \sum_{k=1}^{n} \alpha_k \sigma(< \overline{w_k}, \overline{x} > + \theta_k) = f(x_0)\sigma(x - x_0) + \frac{1}{b-a}\sum_{k=1}^{n} (f(x_k) - f(x_{k-1}))\sigma(x - \frac{x_k + x_{k-1}}{2})$$

Таким образом, меняя параметр α у сигмоиды, можно добиться большей гладкости, но меньшей точности, или, наоборот, меньшей гладкости, но большей точности (пример для $f(x) = x^2$, красным - исходная функция, синим - аппроксимация, черным - график погрешности):





Блок-схема



начало
ввод
n
i от 1 до n
ввод
x_i, f(x_i)
max_ошибка = 0
i от 1 до n
approximate(x_i)
approximate(x) =
$= f(x_0)\sigma(x - x_0) + \frac{1}{b-a}\sum_{k=1}^{n}(f(x_k) - f(x_{k-1}))\sigma(x - \frac{x_k+x_{k-1}}{2})$
нет
да
|approximate(x_i) - f(x_i)| > max_ошибка
max_ошибка = |approximate(x_i) - f(x_i)|
вывод
max_ошибка
конец

Реализация программы

```
import math


def approximate_sigmoid(x_axis, y_axis):
   if len(x_axis) < 3 or len(y_axis) < 3 or len(x_axis) != len(y_axis):
       return 0

   def approximate(x):
       result_sum = y_axis[0] * sigmoid(x - x_axis[0])
       for i in range(1, len(x_axis)):
              result_sum += (y_axis[i] - y_axis[i - 1]) * sigmoid(x - (x_axis[i] + x_axis[i - 1]) / 2)
       return result_sum

   errors = []
   for (x, y) in zip(x_axis, y_axis):
       errors.append(abs(approximate(x) - y))
   return max(errors)


def sigmoid(x):
   k = 1
   if x < 0:
       return 1 - 1 / (1 + math.exp(k * x))
   return 1 / (1 + math.exp(-k * x))


if __name__ == '__main__':
   axis_count = int(input().strip())

   x_axis = list(map(float, input().rstrip().split()))

   y_axis = list(map(float, input().rstrip().split()))

   result = approximate_sigmoid(x_axis, y_axis)

   print(str(result) + '\n')
```

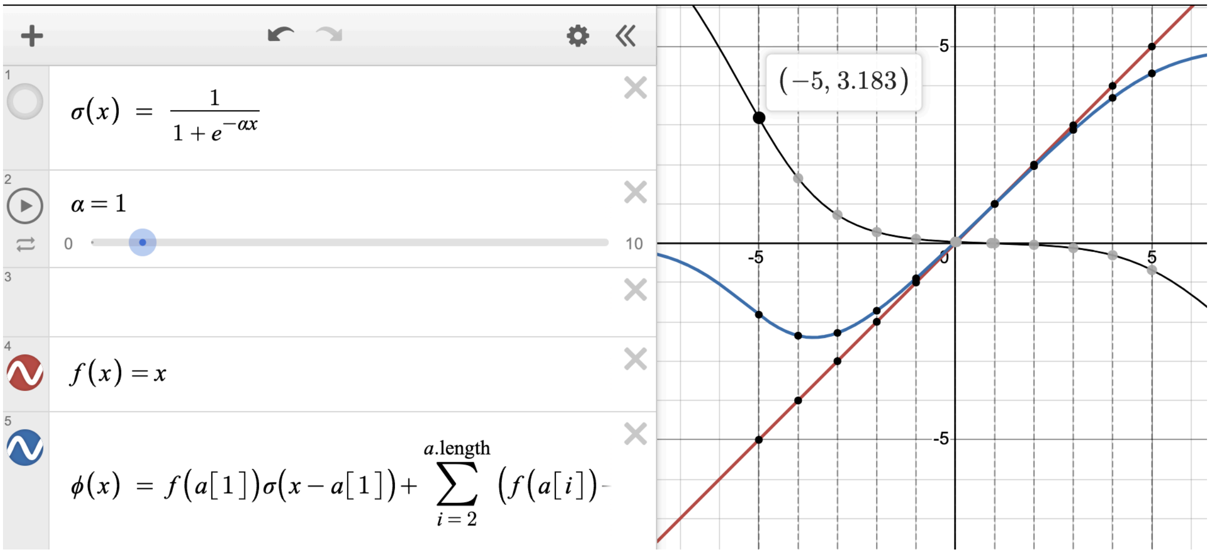
Тесты

Тест 1

Ошибочный, малое количество данных
Входные данные:
1
0
0
Выходные данные:
0

Тест 2

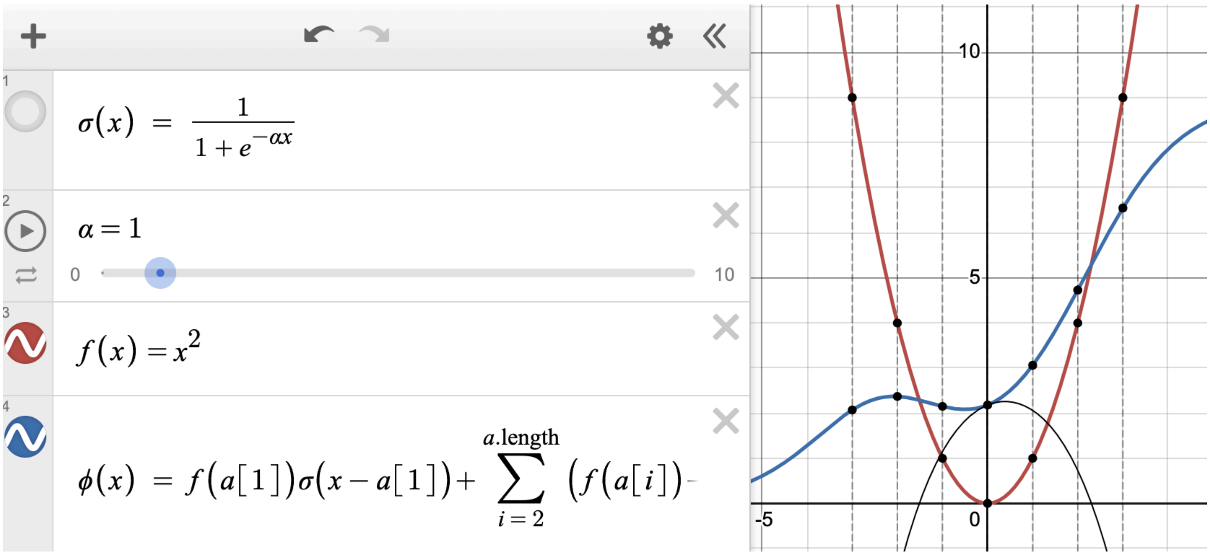
Аппроксимация y = x



Входные данные:
11
-5 -4 -3 -2 -1 0 1 2 3 4 5
-5 -4 -3 -2 -1 0 1 2 3 4 5
Выходные данные:
3.182525917822817

Тест 3

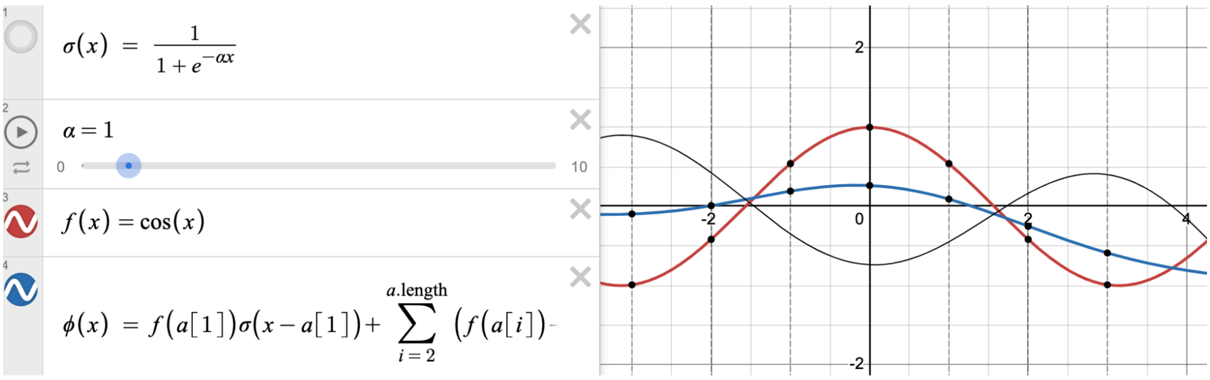
Аппроксимация y = x^2



Входные данные:
7
-3 -2 -1 0 1 2 3
9 4 1 0 1 4 9
Выходные данные:
6.9282143482084235

Тест 4

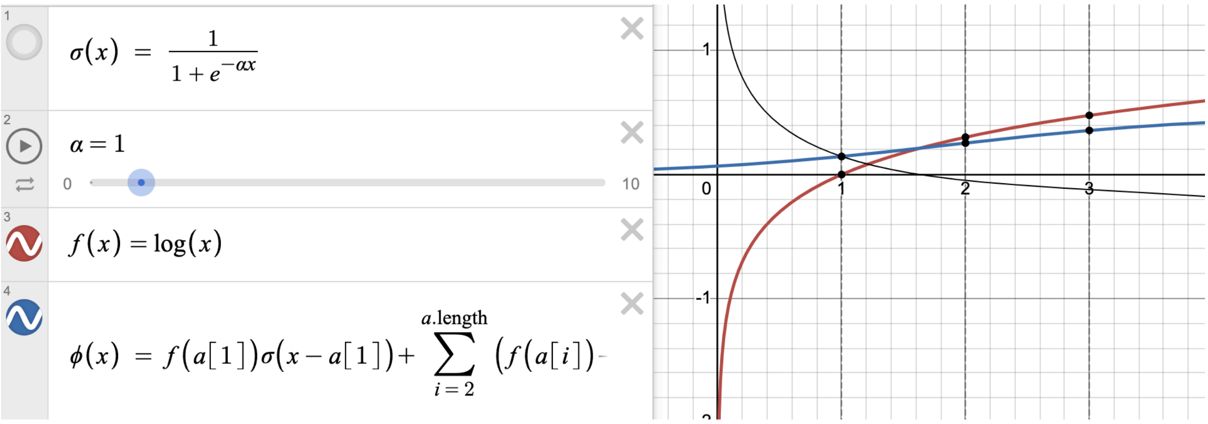
Аппроксимация y = cos(x)



Входные данные:
7
-3 -2 -1 0 1 2 3
-0.99 -0.416 0.54 0.659 0.54 -0.416 -0.99
Выходные данные:
0.8788063364083573

Тест 5

Аппроксимация y = log10(x)



Входные данные:

3

1 2 3

0 0.301 0.477

Выходные данные:

0.14574663349816047

Вывод

В ходе проделанной работы я разобрался в понятиях аппроксимации и интерполяции, научился работать с различными методами приближений. Подробно изучил метод сигмоид. Работая над ним, выяснил, что:

- разные сигмоиды дают разный результат, например, логистическую функцию лучше использовать для аппроксимации более гладких функций;
- параметр α необходимо изменять для более точной подгонки функции под данные. Чем больше этот параметр, тем точнее приближение. Чем он меньше, тем более плавной будет функция;
- метод сигмоид отлично подходит для аппроксимации циклических функций, таких как sin или cos;
- использовать этот метод в вычислительной технике не всегда удобно. Часто само значение сигмоиды стараются аппроксимировать, так как считать ее достаточно сложно.

Также мне удалось написать численный метод, позволяющий находить приближение по заданным данным и считать максимальное отклонение аппроксимации от данных.