Университет ИТМО МФ КТиУ, Ф ПИиКТ

**Лабораторная работа №5**

**Дисциплина «Вычислительная математика»**

**Интерполяция функции**

**Выполнил** Аскаров Эмиль Рамилевич

**Преподаватель:**

Машина Екатерина Алексеевна

г. Санкт-Петербург 2024 г.

# Вычислительная реализация задачи

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | 1.10 | 1.25 | 1.40 | 1.55 | 1.70 | 1.85 | 2.00 |
| y | 0.2234 | 1.2438 | 2.2644 | 3.2984 | 4.3222 | 5.3516 | 6.3867 |

n = 6

Таблица конечных разностей:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| xi | 1.10 | 1.25 | 1.40 | 1.55 | 1.70 | 1.85 | 2.00 |
| yi | 0.2234 | 1.2438 | 2.2644 | 3.2984 | 4.3222 | 5.3516 | 6.3867 |
| Δyi | 1.0204 | 1.0206 | 1.034 | 1.0238 | 1.0294 | 1.0351 | - |
| Δ2yi | 0.0002 | 0.0134 | -0.0102 | 0.0056 | 0.0057 | - | - |
| Δ3yi | 0.0132 | -0.0236 | 0.0158 | 0.0001 | - | - | - |
| Δ4yi | -0.0368 | 0.0394 | -0.0157 | - | - | - | - |
| Δ5yi | 0.0762 | -0.0551 | - | - | - | - | - |
| Δ6yi | -0.1313 | - | - | - | - | - | - |

X1 = 1.121

x0 <= X1 <= x1 => первая интерполяционная формула Ньютона

h = 1.25 – 1.10 = 0.15

t = (x – x0) / h = (x – 1.10) / 0.15

x = 1.121=> t = (1.121 – 1.10) / 0.15 = 0.14

=0.2234+0.142856+0+0.00049+0.00098+0.00156+0.00219 = **0.3715**

X2 = 1.482

a = x3 = 1.55

x2 < X2 < x3 => X2 < a => вторая интерполяционная формула Гаусса

h = 1.25 – 1.10 = 0.15

t = (x – a) / h = (x – 1.55) / 0.15

Таблица конечных разностей:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| i | -3 | -2 | -1 | 0 | 1 | 2 | 3 |
| xi | 1.10 | 1.25 | 1.40 | 1.55 | 1.70 | 1.85 | 2.00 |
| yi | 0.2234 | 1.2438 | 2.2644 | 3.2984 | 4.3222 | 5.3516 | 6.3867 |
| Δyi | 1.0204 | 1.0206 | 1.034 | 1.0238 | 1.0294 | 1.0351 | - |
| Δ2yi | 0.0002 | 0.0134 | -0.0102 | 0.0056 | 0.0057 | - | - |
| Δ3yi | 0.0132 | -0.0236 | 0.0158 | 0.0001 | - | - | - |
| Δ4yi | -0.0368 | 0.0394 | -0.0157 | - | - | - | - |
| Δ5yi | 0.0762 | -0.0551 | - | - | - | - | - |
| Δ6yi | -0.1313 | - | - | - | - | - | - |

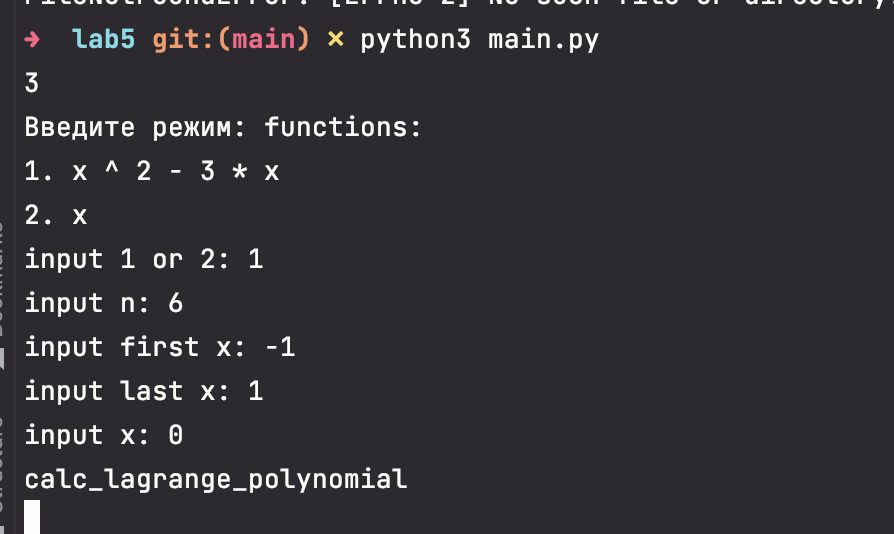
x = 1.482 => t = (1.482 – 1.55) / 0.15 = -0.453

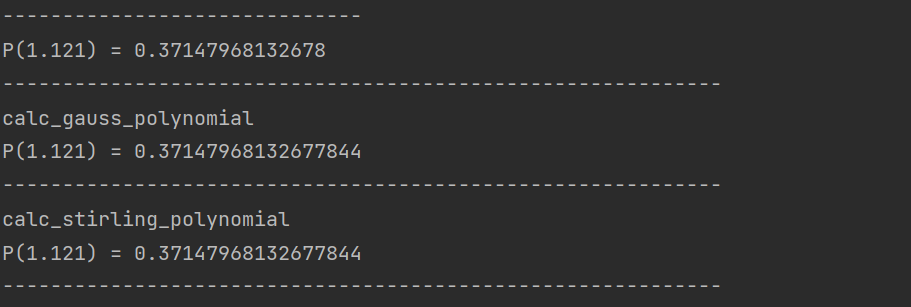
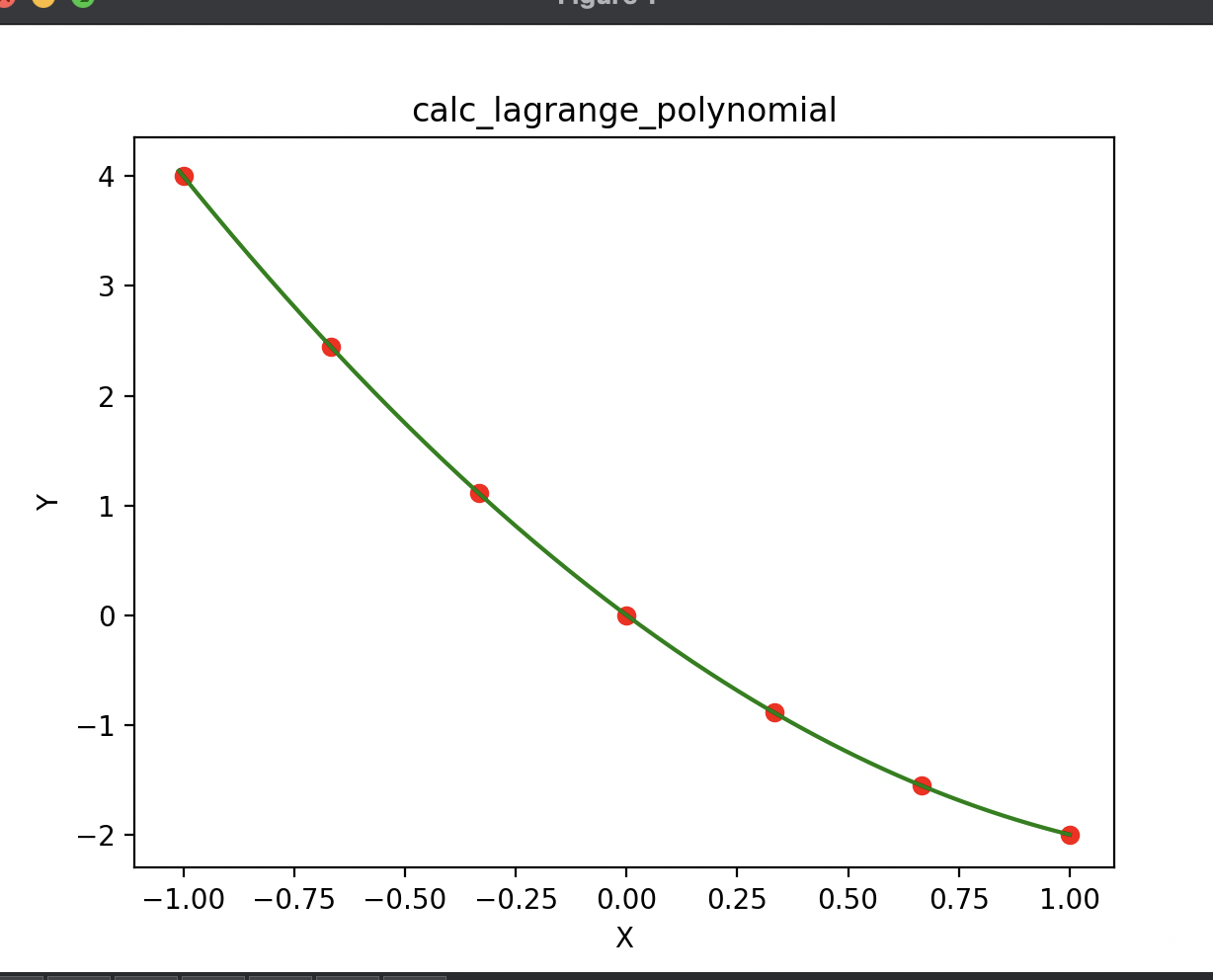
= 3.2984 - 0.4684 + 0.00126 - 0.00141 - 0.00091 - 0.00087 + 0.00063 = **2.8287**

# Программная реализация задачи

from functools import reduce  
from math import factorial  
  
from matplotlib import pyplot as plt  
  
  
def calc\_lagrange\_polynomial(*xs*, *ys*):  
 n = *len*(*xs*) - 1  
 f = lambda *x*: *sum*([*ys*[i] \*  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[j]) / (*xs*[i] - *xs*[j])  
 for j in *range*(n + 1) if i != j])  
 for i in *range*(n + 1)])  
 return f  
  
  
def calc\_newton\_divided\_difference\_polynomial(*xs*, *ys*):  
 div\_difs = []  
 div\_difs.append(*ys*[:])  
 n = *len*(*xs*) - 1  
 for k in *range*(1, n + 1):  
 new = []  
 last = div\_difs[-1][:]  
 for i in *range*(n - k + 1):  
 new.append((last[i + 1] - last[i]) / (*xs*[i + k] - *xs*[i]))  
 div\_difs.append(new[:])  
 *print*("divided differences:")  
 for row in div\_difs:  
 *print*(\**map*(lambda *a*: *round*(a, 5), row), *sep*='\t')  
 *print*('-' \* 30)  
 f = lambda *x*: *ys*[0] + *sum*([  
 div\_difs[k][0] \* reduce(lambda *a*, *b*: a \* b,  
 [x - *xs*[j] for j in *range*(k)])  
 for k in *range*(1, n + 1)])  
 return f  
  
  
def calc\_newton\_finite\_difference\_polynomial(*xs*, *ys*):  
 fin\_difs = []  
 fin\_difs.append(*ys*[:])  
 n = *len*(*xs*) - 1  
 for k in *range*(1, n + 1):  
 last = fin\_difs[-1][:]  
 fin\_difs.append(  
 [last[i + 1] - last[i] for i in *range*(n - k + 1)])  
 *print*("finite differences:")  
 for row in fin\_difs:  
 *print*(\**map*(lambda *a*: *round*(a, 5), row), *sep*='\t')  
 *print*('-' \* 30)  
 h = *xs*[1] - *xs*[0]  
 f = lambda *x*: *ys*[0] + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[0]) / h - j for j in *range*(k)])  
 \* fin\_difs[k][0] / factorial(k)  
 for k in *range*(1, n + 1)])  
 return f  
  
  
def calc\_gauss\_polynomial(*xs*, *ys*):  
 n = *len*(*xs*) - 1  
 alpha\_ind = n // 2  
 fin\_difs = []  
 fin\_difs.append(*ys*[:])  
  
 for k in *range*(1, n + 1):  
 last = fin\_difs[-1][:]  
 fin\_difs.append(  
 [last[i + 1] - last[i] for i in *range*(n - k + 1)])  
  
 h = *xs*[1] - *xs*[0]  
 dts1 = [0, -1, 1, -2, 2, -3, 3, -4, 4]  
 f1 = lambda *x*: *ys*[alpha\_ind] + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h + dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2] / factorial(k)  
 for k in *range*(1, n + 1)])  
 f2 = lambda *x*: *ys*[alpha\_ind] + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h - dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2 - (1 - *len*(fin\_difs[k]) % 2)] / factorial(k)  
 for k in *range*(1, n + 1)])  
 return lambda *x*: f1(x) if x > *xs*[alpha\_ind] else f2(x)  
  
  
def calc\_stirling\_polynomial(*xs*, *ys*):  
 n = *len*(*xs*) - 1  
 alpha\_ind = n // 2  
 fin\_difs = []  
 fin\_difs.append(*ys*[:])  
  
 for k in *range*(1, n + 1):  
 last = fin\_difs[-1][:]  
 fin\_difs.append(  
 [last[i + 1] - last[i] for i in *range*(n - k + 1)])  
  
 h = *xs*[1] - *xs*[0]  
 dts1 = [0, -1, 1, -2, 2, -3, 3, -4, 4]  
 f1 = lambda *x*: *ys*[alpha\_ind] + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h + dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2] / factorial(k)  
 for k in *range*(1, n + 1)])  
 f2 = lambda *x*: *ys*[alpha\_ind] + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h - dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2 - (1 - *len*(fin\_difs[k]) % 2)] / factorial(k)  
 for k in *range*(1, n + 1)])  
 return lambda *x*: (f1(x) + f2(x)) / 2  
  
  
def calc\_bessel\_polynomial(*xs*, *ys*):  
 n = *len*(*xs*) - 1  
 alpha\_ind = 0  
 fin\_difs = []  
 fin\_difs.append(*ys*[:])  
  
 for k in *range*(1, n + 1):  
 last = fin\_difs[-1][:]  
 fin\_difs.append(  
 [last[i + 1] - last[i] for i in *range*(n - k + 1)])  
  
 h = *xs*[1] - *xs*[0]  
  
 dts1 = [0, -1, 1, -2, 2, -3, 3, -4, 4, -5, 5]  
 f = lambda *x*: (*ys*[alpha\_ind] + *ys*[alpha\_ind + 1]) / 2 + *sum*([  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h + dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2] / factorial(2 \* k) +  
 ((x - *xs*[alpha\_ind]) / h - 1 / 2) \*  
 reduce(lambda *a*, *b*: a \* b,  
 [(x - *xs*[alpha\_ind]) / h + dts1[j] for j in *range*(k)])  
 \* fin\_difs[k][*len*(fin\_difs[k]) // 2] / factorial(2 \* k + 1)  
 for k in *range*(1, n + 1)])  
 return f  
  
  
def draw\_plot(*a*, *b*, *func*, *dx*=0.01):  
 xs, ys = [], []  
 *a* -= *dx  
 b* += *dx* x = *a* while x <= *b*:  
 xs.append(x)  
 ys.append(*func*(x))  
 x += *dx* plt.plot(xs, ys, 'g')  
  
  
def main(*xs*, *ys*, *x*):  
 methods = [calc\_lagrange\_polynomial,  
 calc\_newton\_divided\_difference\_polynomial,  
 calc\_newton\_finite\_difference\_polynomial,  
 calc\_gauss\_polynomial,  
 calc\_stirling\_polynomial,  
 calc\_bessel\_polynomial]  
 for method in methods:  
 *# для гаусса и стирлинга нечётное число узлов должно быть* if (method is calc\_gauss\_polynomial or method is calc\_stirling\_polynomial) \  
 and *len*(*xs*) % 2 == 0:  
 continue  
 *# для бесселя чётное число узлов должно быть* if method is calc\_bessel\_polynomial and *len*(*xs*) % 2 == 1:  
 continue  
 *print*(method.*\_\_name\_\_*)  
  
 P = method(*xs*, *ys*)  
  
 plt.title(method.*\_\_name\_\_*)  
  
 draw\_plot(*xs*[0], *xs*[-1], P)  
 for i in *range*(*len*(*xs*)):  
 plt.scatter(*xs*[i], *ys*[i], *c*='r')  
 plt.xlabel("X")  
 plt.ylabel("Y")  
 plt.show()  
  
 *print*(f'P({*x*}) = {P(*x*)}')  
 *print*('-' \* 60)  
  
  
def read\_number(*s*: *str*):  
 while True:  
 try:  
 return *float*(*input*(*s*))  
 except *Exception*:  
 continue  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 mode = read\_number("Введите режим: ")  
 if mode == 1:  
 xs = *list*(*map*(*float*, *input*('input xs: ').split()))  
 ys = *list*(*map*(*float*, *input*('input ys: ').split()))  
 x = *float*(*input*('input x: '))  
 elif mode == 2:  
 with *open*('input.txt') as f:  
 xs = *list*(*map*(*float*, f.readline().strip().split()))  
 ys = *list*(*map*(*float*, f.readline().strip().split()))  
 x = *float*(f.readline().strip())  
 elif mode == 3:  
 *print*('functions: ')  
 *print*('1. x ^ 2 - 3 \* x')  
 *print*('2. x ^ 5')  
 func\_number = *int*(*input*('input 1 or 2: '))  
 f = lambda x: x \*\* 2 - 3 \* x if func\_number == 1 else x \*\* 5  
 n = *int*(*input*('input n: '))  
 x0 = *float*(*input*('input first x: '))  
 xn = *float*(*input*('input last x: '))  
 h = (xn - x0) / (n - 1)  
 xs = [x0 + h \* i for i in *range*(n)]  
 ys = *list*(*map*(f, xs))  
 x = *float*(*input*('input x: '))  
 else:  
 *# xs = [0.15, 0.2, 0.33, 0.47]* xs = [0.15, 0.2, 0.25, 0.3]  
 ys = [1.25, 2.38, 3.79, 5.44]  
 x = 0.22  
 *# main(xs, ys, x)* xs = *sorted*(xs)  
 if *len*(*set*(xs)) != *len*(xs):  
 *print*('Иксы должны быть разными')  
 else:  
 main(xs, ys, x)

# Тестовые данные





calc\_lagrange\_polynomial

P(0.0) = 5.551115123125783e-17

------------------------------------------------------------

calc\_newton\_divided\_difference\_polynomial

divided differences:

4.0 2.16 0.64 -0.56 -1.44 -2.0

-4.6 -3.8 -3.0 -2.2 -1.4

1.0 1.0 1.0 1.0

-0.0 0.0 -0.0

0.0 -0.0

-0.0

------------------------------

P(0.0) = 0.0

------------------------------------------------------------

calc\_newton\_finite\_difference\_polynomial

finite differences:

4.0 2.16 0.64 -0.56 -1.44 -2.0

-1.84 -1.52 -1.2 -0.88 -0.56

0.32 0.32 0.32 0.32

-0.0 0.0 -0.0

0.0 -0.0

-0.0

------------------------------

P(0.0) = 0.0

------------------------------------------------------------

calc\_bessel\_polynomial

P(0.0) = 0.6499999999999995

------------------------------------------------------------

➜ lab5 git:(main) ✗ python3 main.py

Введите режим: 3

functions:

1. x ^ 2 - 3 \* x

2. x

input 1 or 2: 1

input n: 7

input first x: -1

input last x: 1

input x: 0

calc\_lagrange\_polynomial

P(0.0) = 0.0

------------------------------------------------------------

calc\_newton\_divided\_difference\_polynomial

divided differences:

4.0 2.44444 1.11111 0.0 -0.88889 -1.55556 -2.0

-4.66667 -4.0 -3.33333 -2.66667 -2.0 -1.33333

1.0 1.0 1.0 1.0 1.0

-0.0 -0.0 0.0 0.0

-0.0 0.0 -0.0

0.0 -0.0

-0.0

------------------------------

P(0.0) = 0.0

------------------------------------------------------------

calc\_newton\_finite\_difference\_polynomial

finite differences:

4.0 2.44444 1.11111 0.0 -0.88889 -1.55556 -2.0

-1.55556 -1.33333 -1.11111 -0.88889 -0.66667 -0.44444

0.22222 0.22222 0.22222 0.22222 0.22222

0.0 0.0 -0.0 -0.0

-0.0 -0.0 0.0

-0.0 0.0

0.0

------------------------------

P(0.0) = -8.881784197001252e-16

------------------------------------------------------------

calc\_gauss\_polynomial

P(0.0) = 0.0

------------------------------------------------------------

calc\_stirling\_polynomial

P(0.0) = 0.0

------------------------------------------------------------

# Вывод

В ходе лабораторной работы я познакомился с интерполяцией функции разными методами (линейная, квадратичная, многочлен Лагранжа, многочлен Ньютона, многочлены Гаусса, Стирлинга и Бесселя).

Линейная и квадратичная интерполяция – простые методы, но неточные.

Многочлен Лагранжа – хороший метод, но много вычислений. Малая погрешность при небольших n, с изменением числа узлов все вычисления заново.

Многочлен Ньютона с разделёнными разностями – хороший метод. Используется для неравноотстоящих узлов. При добавлении новых узлов первые члены многочлена остаются неизменными.

Многочлен Ньютона с конечными разностями – хороший метод. Используется для равноотстоящих узлов. При добавлении новых узлов первые члены многочлена остаются неизменными. Есть формулы для интерполирования вперёд и назад. Можно использовать для экстраполирования (но будут бОльшие погрешности).