Интерполяция

Выполнил Галиуллин Арслан, 1 курс факультета физики, группа 171.
1233550v@mail.ru
Желаемая оценка - 10.

Задача - построить такую функцию, чтобы её значения в узлах интерполяции совпадали с данными значениями.
Это, очевидно, можно сделать многими способами. Я сделал двумя - вручную многочленом Лагранжа и при помощи встроенной в библиотеку scipy функции interp1d.

1. Многочлен Лагранжа

$$L_N(x) = \sum_{k=0}^{N} c_k(x) * u_k, c_k(x) = \prod_{\substack{i=0 \\ i\neq}}^{N} \frac{x - x_i}{x_k - x_i}$$

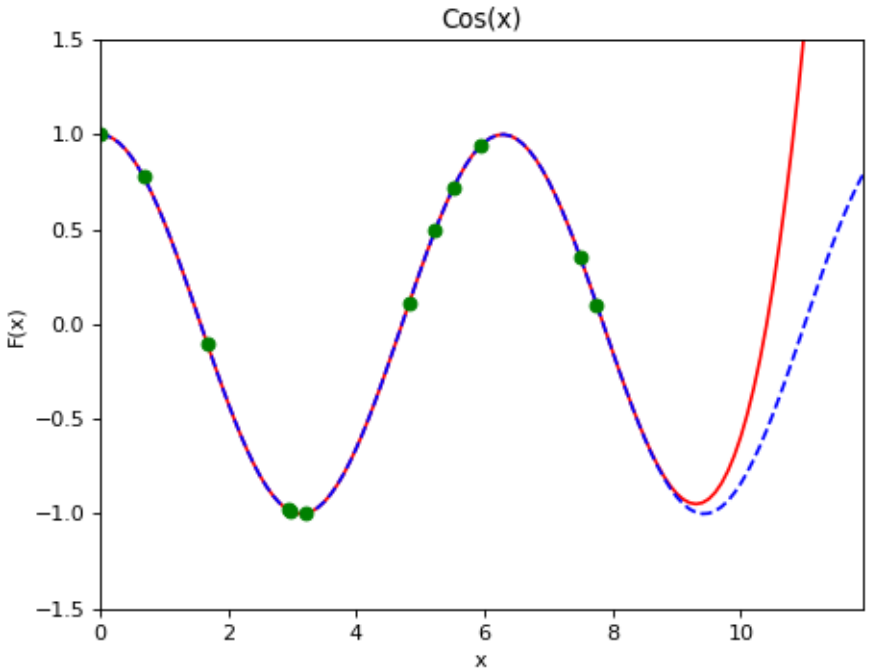
u_k - значение в узле, x_j - узел.

```
from math import *
import random
import matplotlib as mpl
import matplotlib.pyplot as plt


xs = []
cos_vals = []

x = 0.0

def f(x):
    return cos(x)

for i in range (12):
    x = i*(1-0.5*random.random())
    cos_vals += [f(x)]
    xs += [x]


x_list = xs
```

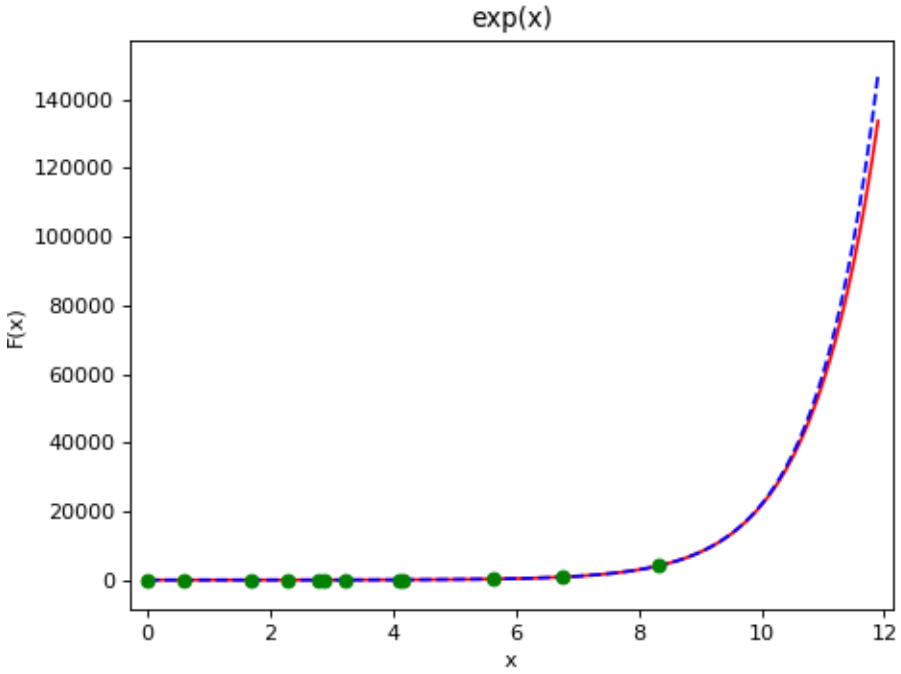
```
y_list = cos_vals
N = len(x_list)

def c_k(x, N, k, x_list):
    c = 1.0
    for i in range (N):
        if i != k:
            c *= (x - float(x_list[i]))/(float(x_list[k]) - float(x_list[
               i]))
    return c

def L(x, N, x_list, y_list):
    l = 0
    for i in range (N):
        l += float(c_k(x, N, i, x_list))*float(y_list[i])
    return l

xs_interpol = []
cos_vals_interpol = []

x = 0.0

for i in range (120):
    x = i/10
    cos_vals_interpol += [L(x, N, x_list, y_list)]
    xs_interpol += [x]


xs_real = []
cos_vals_real = []

for i in range (120):
    x = i/10
    cos_vals_real += [f(x)]
    xs_real += [x]


dpi = 80
fig = plt.figure(dpi=dpi, figsize=(512 / dpi, 384 / dpi))
mpl.rcParams.update({'font.size': 10})

plt.axis([0, 12, -1.5, 1.5])

plt.title('Cos(x)')
```

```
plt.xlabel('x')
plt.ylabel('F(x)')

plt.xlim(0, max(xs_interpol))

plt.plot(xs_interpol, cos_vals_interpol, color='red', linestyle='solid',
        label='cos(x)')
plt.plot(xs_real, cos_vals_real, color='blue', linestyle='dashed',
        label='cos(x)')
plt.plot(xs, cos_vals, 'go')
plt.show()


error = 0
for i in range (ceil(max(xs)*10)):
    error += (cos_vals_interpol[i]-cos_vals_real[i])**2
error = sqrt(error/ceil(max(xs)*10))
print(error)
```

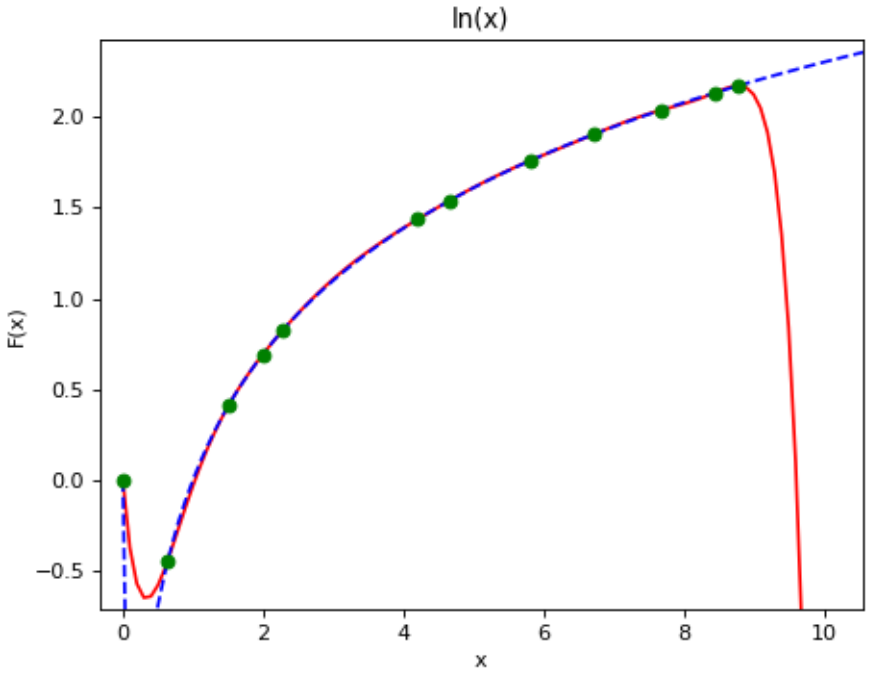
Вот пример нескольких функций:
Зелёные точки - данные значения, синяя линия - точная функция (из которой случайным образом брались значения в узлах), красная линия - полином Лагранжа.



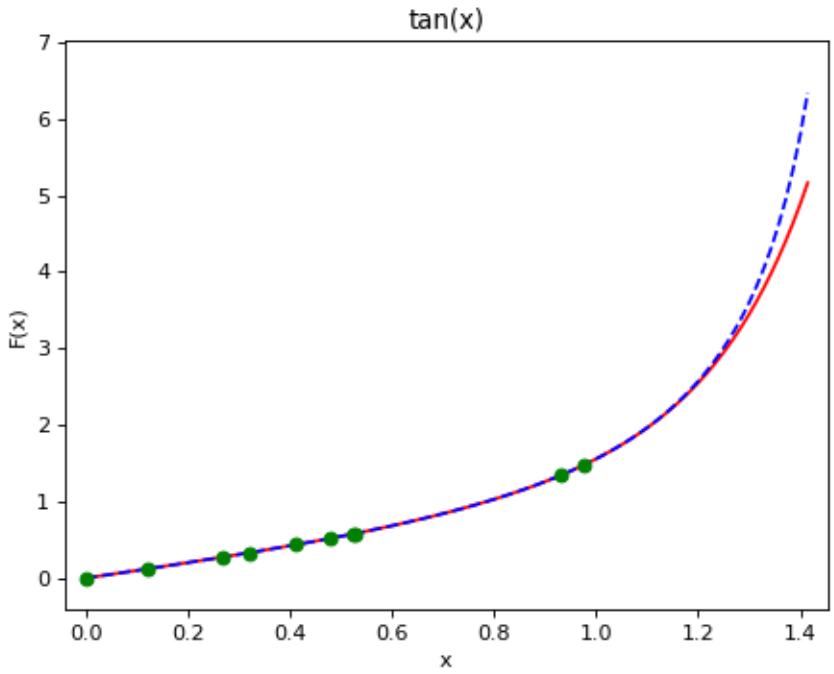
$\sigma = 3.6 \cdot 10^{-5}$ - среднеквадратичное отклонение.


exp(x)
140000
120000
100000
80000
60000
40000
20000
0
F(x)
0
2
4
6
8
10
12
x

$\sigma = 0.12$


ln(x)
2.0
1.5
1.0
0.5
0.0
−0.5
F(x)
0
2
4
6
8
10
x

$\sigma = 0.24$



$\sigma = 1.3 \cdot 10^{-6}$

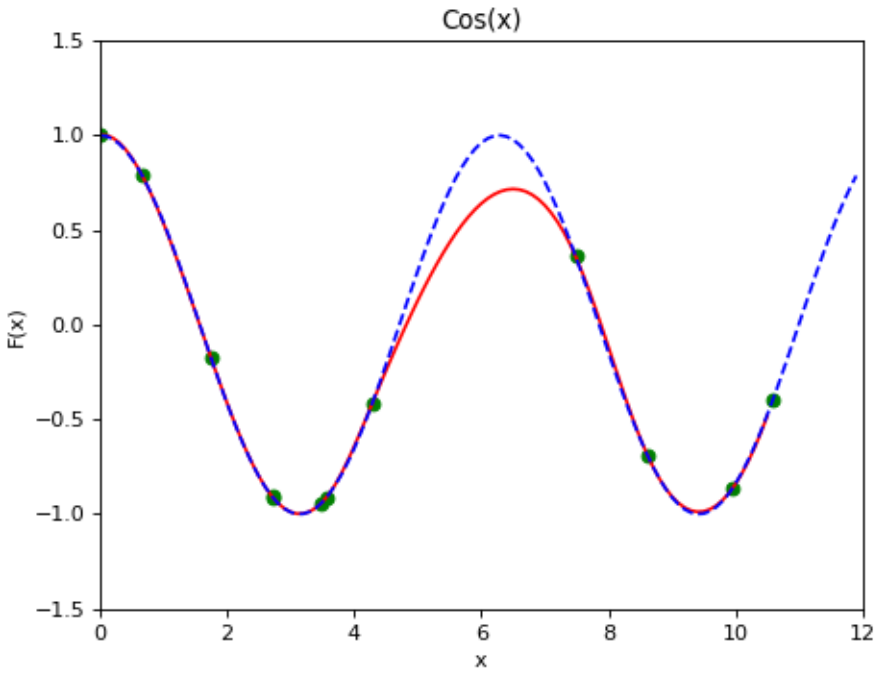
2. Встроенная функция

```
from scipy.interpolate import interp1d
from numpy import *
from math import *
import random
import matplotlib as mpl
import matplotlib.pyplot as plt


def f(x):
    return sin(x)

xs = []
cos_vals = []
x = 0.0

for i in range (12):
    x = i*(1-0.5*random.random())
```

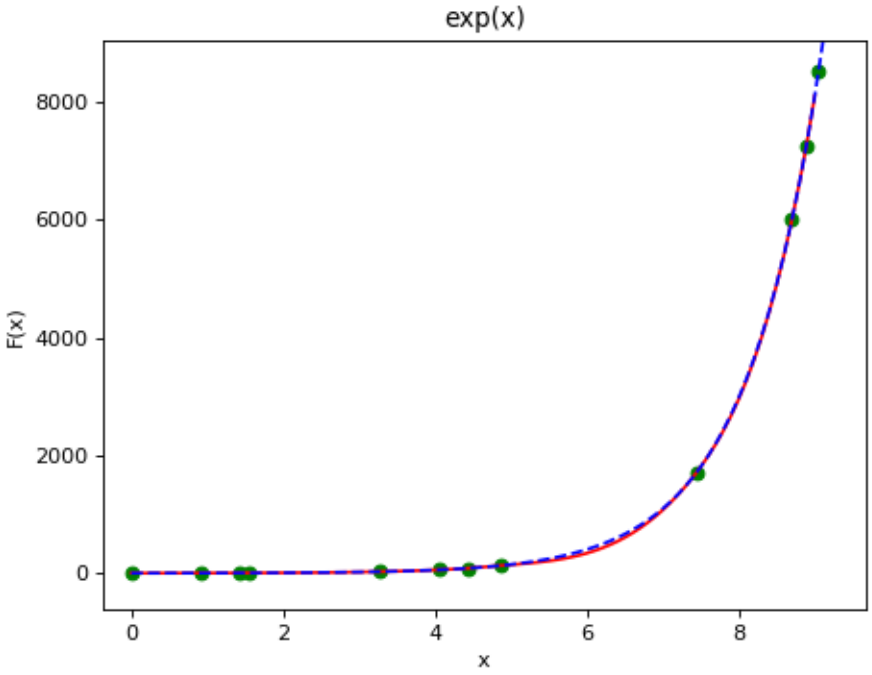
```
    cos_vals += [f(x)]
    xs += [x]

xs_real = []
cos_vals_real = []

for i in range (120):
    x = i/10
    cos_vals_real += [f(x)]
    xs_real += [x]



dpi = 80
fig = plt.figure(dpi=dpi, figsize=(512 / dpi, 384 / dpi))
mpl.rcParams.update({'font.size': 10})

plt.axis([0, 12, -1.5, 1.5])

plt.title('Cos(x)')
plt.xlabel('x')
plt.ylabel('F(x)')




g = interp1d(xs, cos_vals, kind = 3)

xs_interpol = []
cos_vals_interpol = []
x = 0.0

for i in range (120):
    try:
        x = i/10
        cos_vals_interpol += [g(x)]
        xs_interpol += [x]
    except:
        print('')

plt.plot(xs, cos_vals, 'go')
plt.plot(xs_interpol, cos_vals_interpol, color='red', linestyle='solid',
         label='cos(x)')
plt.plot(xs_real, cos_vals_real, color='blue', linestyle='dashed',
```

```
        label='cos(x)')

plt.show()
error = 0
for i in range(ceil(max(xs) * 10)):
    error += (cos_vals_interpol[i] - cos_vals_real[i]) ** 2
error = sqrt(error / ceil(max(xs) * 10))
print(error)
```

Тут точки соединяются многочленами третьей степени.
Функции:



$\sigma = 0.12$
Виден недостаток этого метода: он плохо находит экстремумы.



$\sigma = 23$ - и погрешность больше.

3. Итог

Без какой-нибудь практической задачи кажется, что интерполяция многочленом Лагранжа лучше встроенной функции.