# Аппроксимация функций методом наилучшего среднеквадратичного приближения

## Описание метода

Задана функция, определённая на R. Будем строить ее наилучшее среднеквадратическое приближение с помощью полиномов Эрмита.

Построенная функция будет линейной комбинацией этих полиномов. Коэффициенты будут находиться из следующих условий.

Из условия минимизации

$$\min_{a_k} \left\| f(x) - \sum_{k=0}^{n} a_k \varphi_k(x) \right\|_{\mathcal{H}}^2$$

следует, что коэффициенты буду удовлетворять следующему соотношению:

$$\sum_{k=0}^{n} a_k \int_a^b \rho(x)\varphi_k(x)\varphi_i(x)dx = \int_a^b \rho(x) f(x)\varphi_i(x)dx$$

Однако полиномы Эрмита создают ортогональную систему на интервале (-∞, + ∞) с весовой функцией $\rho(x) = e^{x^{-2}}$, следовательно выражение выше упрощается до

$$a_i = \int_a^b \rho(x) f(x)\varphi_i(x)dx,\ i = \overline{0,n}$$

Таким образом, зная функцию веса, исходную функцию и соответствующие каждому I полиномы, можно посчитать все коэффициенты.

Также важно заметить, что многочлены нужно еще и нормировать, так как в общем случае получается

$$\int_{-\infty}^{\infty} H_n(x) H_m(x)\, e^{-x^2}\, dx = 2^n n! \sqrt{\pi}\, \delta_{nm}$$

Поэтому разделю каждый многочлен на его норму (корень из того, что выше).

Тогда все коэффициенты можно посчитать и построить приближение.

Программная реализация

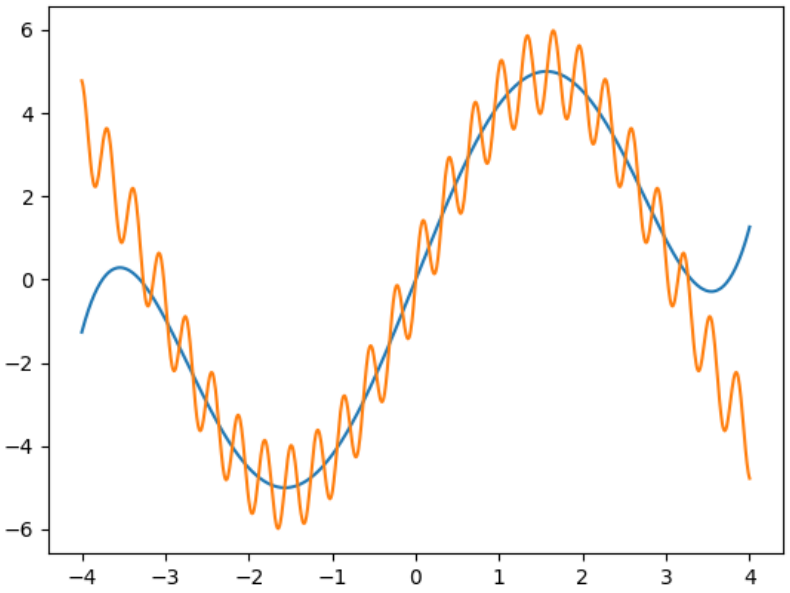
```python
from math import sqrt, pi, exp, factorial, sin
import scipy.integrate as integrate
import numpy as np
import matplotlib.pyplot as plt


fi = [(lambda x: 1),
      (lambda x: 2 * x),
      (lambda x: 4*x ** 2 - 2),
      (lambda x: 8*x ** 3 - 12 * x),
      (lambda x: 16*x ** 4 - 48 * x ** 2 + 12),
      (lambda x: 32*x ** 5 - 160 * x ** 3 + 120 * x),
      (lambda x: 64*x ** 6 - 480 * x ** 4 + 720 * x ** 2 - 120)]

r = lambda x: exp(-x ** 2)
norm = lambda n: 1/(2**n * factorial(n)*sqrt(pi))

def approximation(f, n):
    global fi, r
    n = min(7, n)
    res = list()
    for i in range(n):
        cur_int = integrate.quad(lambda x: r(x)*fi[i](x)*f(x)*norm(i), -100, 100)
        res.append(cur_int[0])
    return res

def get_value(a, x, n):
    res = 0
    n = min(7, n)
    global fi
    for i in range(n):
        res += a[i]*fi[i](x)
    return res
```

## Проверка правильности решения

Возьму функцию $f(x) = 5 * \sin(x) + \sin(20x)$

Приближение будет строиться на промежутке от -4 до 4. Используется 7 полиномов

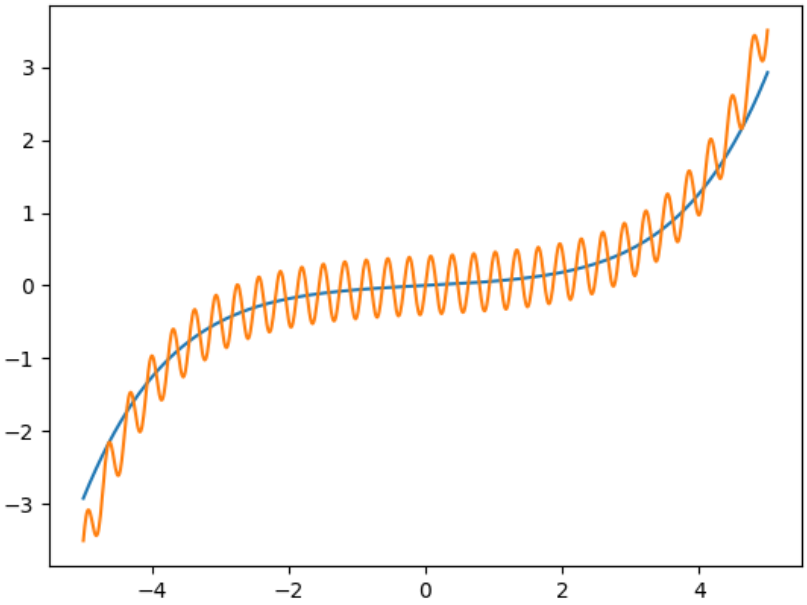
Результат:




Возьму другую функцию

$f(x) = sh(x)/20 + 0.4\sin(20x)$

Результат:




Из графиков видно, что программа работает правильно, заданные функции хорошо аппроксимируются полиномами. В целом можно сказать, что результат довольно приемлемый.