Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Вычислительная математика

Лабораторная работа №3

Аппроксимация функций методом наименьших квадатов

Выполнил: **Шкаруба Н.Е.**

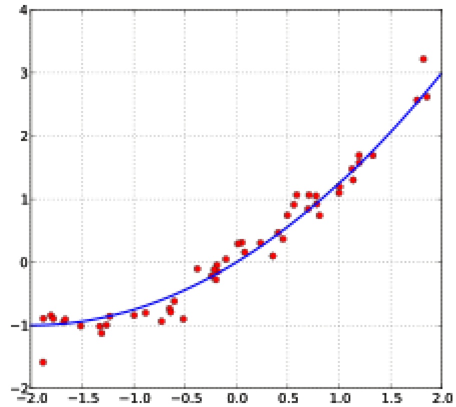
Проверил: **Петрова М.**

группа: **P3218**

год: **2015**

Метод наименьших квадратов

1. Описание метода:



Мера отклонения многочлена от заданной функции f(x) на множестве точек (x_i, y_i) (i = 0, 1, …, n) при среднеквадратичном приближении является величина S, равная сумме квадратов разностей между значениями многочлена и функции в данных точках:

$$S = \sum_{i=0}^{n} [\varphi(x_i) - y_i]^2$$

Если мы решим, что будем аппроксимировать полиномами, то запишем сумму квадратов отклонений для всех точек x_0, x_1, …, x_n :

$$S = \sum_{i=0}^{n} \varepsilon_i^2 = \sum_{i=0}^{n} [\varphi(x_i, a_0, a_1, \ldots, a_m) - y_i]^2.$$

Суть метода наименьших квадратов в том, что параметры a_0, a_1, …, a_n, будем находить из условия минимума функции S :

$$S = S(a_0,\ a_1,\ \ldots,\ a_m)$$

Однако, важно(!), чтобы отклонения e_i подчиняются нормальному закону распределения (В самом грубом случае: необходимость, чтобы функция отклонений была плавной, не резкой), тогда полученный рассматриваемым методом значения параметров наиболее вероятны.

Поскольку параметры a_0, a_1, …, a_n выступают в роли независимых переменных функции S, то её минимум найдём, приравнивая к нулю частные производные по этим переменным (Из курса линейной алгебры):

$$\frac{\partial S}{\partial a_0} = 0, \quad \frac{\partial S}{\partial a_1} = 0, \ldots, \frac{\partial S}{\partial a_m} = 0. \qquad (*)$$

Полученные отношения – система уравнений для определения a_0, a_1, …, a_n .

Самый частый случай, аппроксимируемый по МНК это многочлен вида:

$$\varphi(x) = a_0 + a_1 x + \ldots + a_m x^m$$

Для составления системы уравнений коэффициентов найдём частные производные функции S:

$$\frac{\partial S}{\partial a_0} = 2\sum_{i=0}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i),$$

$$\frac{\partial S}{\partial a_1} = 2\sum_{i=0}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i)\, x_i,$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\frac{\partial S}{\partial a_m} = 2\sum_{i=0}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i)\, x_i^m.$$

Приравнивая эти выражения к нулю, в соответствии с (*) и собирая коэффициенты при неизвестных $a_0, a_1, \ldots, a_n$, получаем следующую систему линейных(!) уравнений, которую мы умеем решать :

$$
\begin{aligned}
&(n+1)a_0 + a_1 \sum_{i=0}^{n} x_i + a_2 \sum_{i=0}^{n} x_i^2 + \ldots + a_m \sum_{i=0}^{n} x_i^m = \sum_{i=0}^{n} y_i,\\
&a_0 \sum_{i=0}^{n} x_i + a_1 \sum_{i=0}^{n} x_i^2 + a_2 \sum_{i=0}^{n} x_i^3 + \ldots + a_m \sum_{i=0}^{n} x_i^{m+1} = \sum_{i=0}^{n} x_i y_i,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&a_0 \sum_{i=0}^{n} x_i^m + a_1 \sum_{i=0}^{n} x_i^{m+1} + a_2 \sum_{i=0}^{n} x_i^{m+2} + \ldots + a_m \sum_{i=0}^{n} x_i^{2m} = \sum_{i=0}^{n} x_i^m y_i
\end{aligned}
$$

Эту систему можно записать в более компактном виде

$$
\begin{gathered}
b_{00}a_0 + b_{01}a_1 + \ldots + b_{0m}a_m = c_0,\\
b_{10}a_0 + b_{11}a_1 + \ldots + b_{1m}a_m = c_1,\\
\cdots\cdots\cdots\cdots\\
b_{m0}a_0 + b_{m1}a_1 + \ldots + b_{mm}a_m = c_m;\\
b_{kl} = \sum_{i=0}^{n} x_i^{k+l} \qquad c_k = \sum_{i=0}^{n} x_i^k y_i, \qquad k, l = 0, 1, \ldots, m.
\end{gathered}
$$

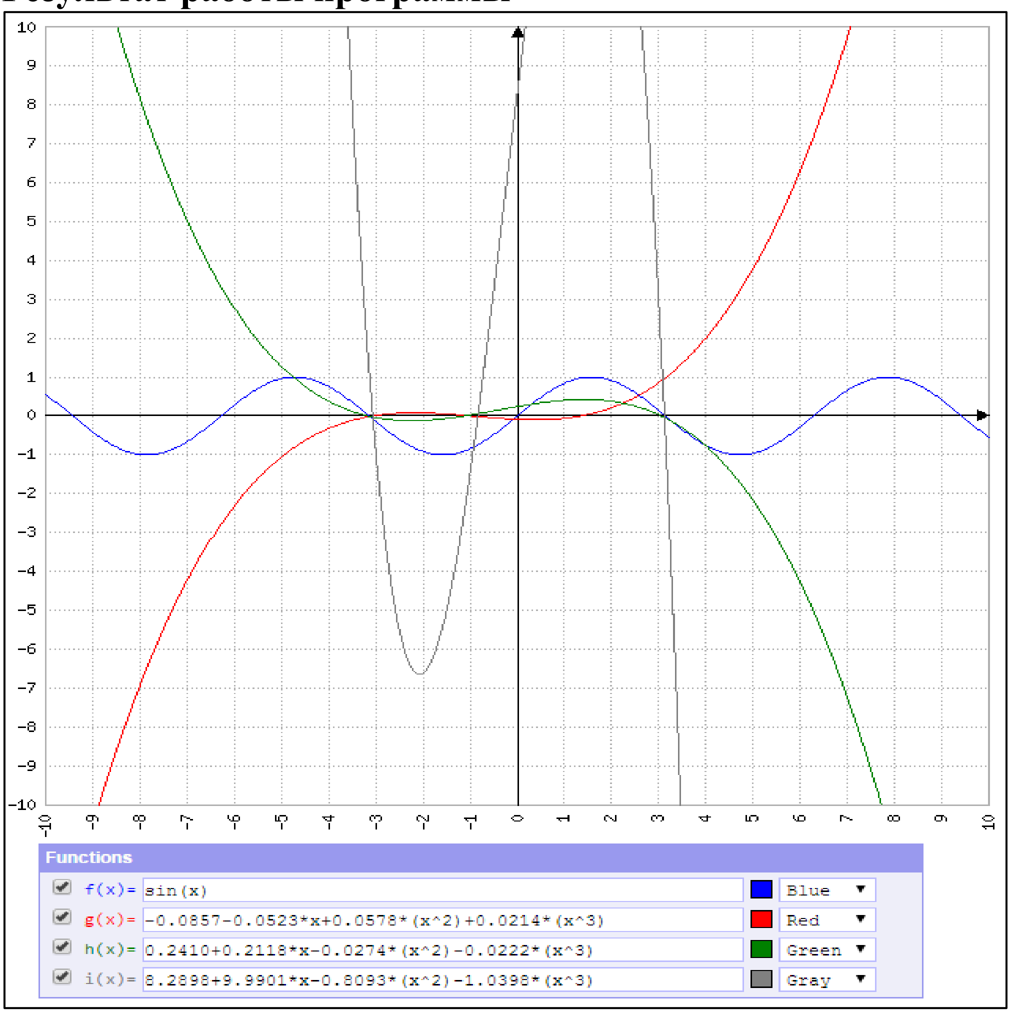
2. Исходный код

```
function<float(float)> OrdinaryLeastSquares(vector<Point> points, size_t polynomRang=5) {
	// Matrix system is bA = c
	// polynomRang = points.size() - 1
	vector<vector<float>> b(polynomRang, vector<float>(polynomRang));
	vector<float> c(polynomRang);
	vector<float> coefficients(polynomRang);

	// compute b
	for (size_t k = 0; k < polynomRang; k++)
		for (size_t l = 0; l < polynomRang; l++)
			for (size_t i = 0; i < points.size(); i++)
				b[k][l] += powf(points[i].x, k+l);

	// compute c
	for (size_t k = 0; k < polynomRang; k++)
		for (size_t i = 0; i < points.size(); i++)
			c[k] = powf(points[i].x, k) + points[i].y;

	// compute coefficients
	coefficients = GaussSeidel(b, c, 0.0001);

	// appriximated function. Don't be ashamed of lambda's
	return [coefficients](float x) -> float {
		return computePolynom(x, coefficients);
	};
}
```

3. Пример работы программы

```
What method do you want to run?
1. Gauss-Seidel method
2. Reqtangle method(integrals)
3. Ordinary Least Squares(approximation)
4.
5.
0 - exit

Command: 3
Ordinary Least Squares method
Input points:
x: 0
y: 0
'q' - exit; other key - continue. command: s

x: 1
y: 1
'q' - exit; other key - continue. command: s

x: 2
y: 4
'q' - exit; other key - continue. command: s

x: 3
y: 9
'q' - exit; other key - continue. command: q

Input expected polynom rank or 'q' if you don't know it: 4

input 'q' to exit, or if you want to calculate y, inout x:
y: 39.8701What method do you want to run?
```

4. Результат работы программы



5. Вывод: ez