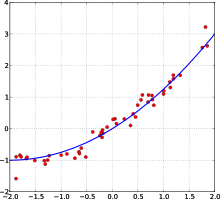
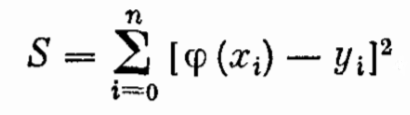
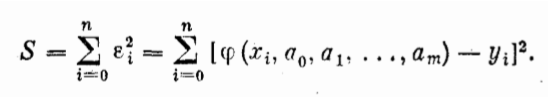
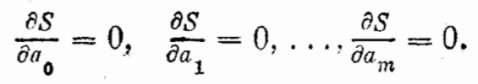
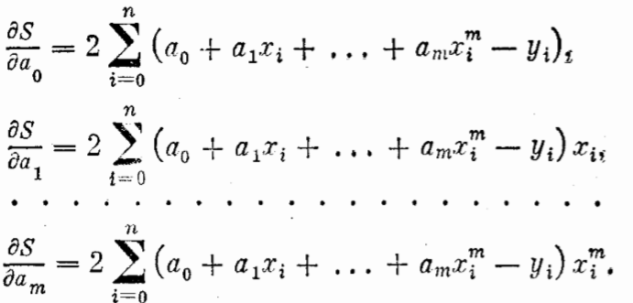
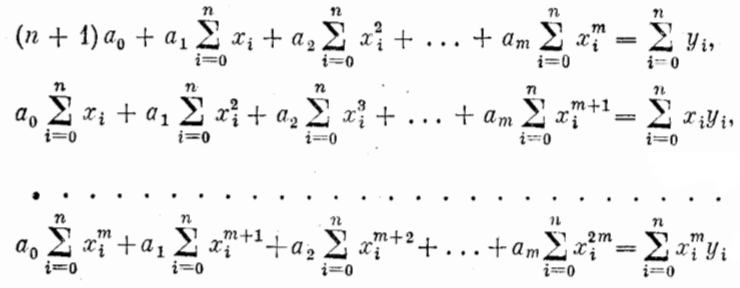
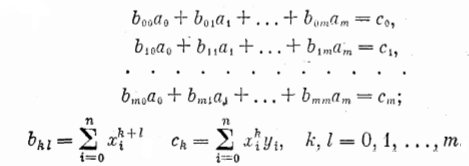
Санкт-Петербургский национальный исследовательский университет  
 информационных технологий, механики и оптики  
Кафедра информатики и прикладной математики

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 (xi, yi) (i = 0, 1, …, n) при среднеквадратичном приближении является величина S, равная сумме квадратов разностей между значениями многочлена и функции в данных точках:  
     
     
     
     
     
     
     
     
   Если мы решим, что будем аппроксимировать полиномами, то запишем сумму квадратов отклонений для всех точек x0,   
   x1,…, xn :   
     
     
     
     
     
     
     
     
     
   Суть метода наименьших квадратов в том, что параметры a0, a1,…, an, будем находить из условия минимума функции S :   
     
     
     
     
   Однако, важно(!), чтобы отклонения ei 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 a0, a1,…, an выступают в роли независимых переменных функции S, то её минимум найдём, приравнивая к нулю частные проихводные по этим переменным (Из курса линейной алгебры):  
     
     
     
     
     
   (\*)  
   Полученные отношения – система уравнений для определения a0, a1,…, an .  
     
   Самый частый случай, аппроксимируемый по МНК это многочлен вида:   
     
     
     
     
   Для составления системы урванений коэффициентов найдём частные производные функции S:  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   Приравнивая эти выражения к нулю, в соответствии с (\*) и собирая коэффициенты при неизвестных a0, a1,…, an, получаем следующую систему линейных(!) уравнений, которую мы умеем решать :  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   Эту систему можно записать в более компактном виде  
   
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);

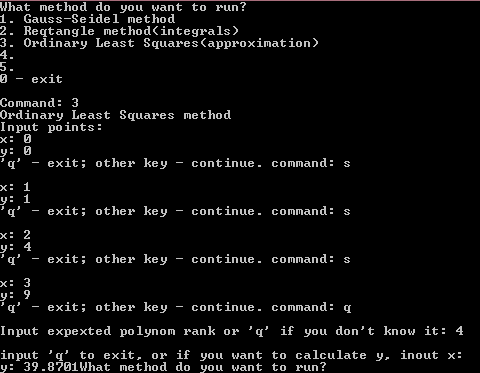
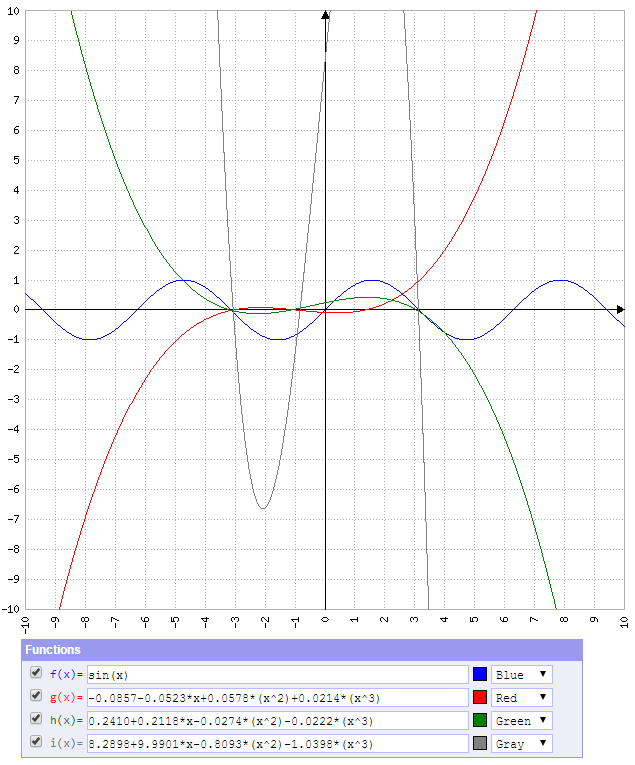
// appriximated function. Don't be ashamed of lambda's

return [coefficients](float x) -> float {

return computePolynom(x, coefficients);

};

}

1. **Пример работы программы**
2. **Результат работы программы  
   **
3. **Вывод**: ez