1. Создаем таблицу значений. Параметры a и b это начало и конец отрезка, n – количество точек. Массив из n точек отрезка ab создается функцией linspace(). Затем по точкам x рассчитываем значения функции

Например для a=0, b=2, n=5

Таблица значений:

x: 0.0000 0.5000 1.0000 1.5000 2.0000

y: 0.0000 0.2874 1.1359 1.2203 -1.3790

2.

Листинг файла generate\_table.m:

function [x, y] = generate\_table(a, b, n)

x = linspace(a,b,n);

y = sin(x.^2).\*exp(0.3\*x);

end

3. Выбираем базисные функции

4. Реализуем метод наименьших квадратов для выбранных базисных функций.

Аппроксимирующая функция задается в виде

Запишем критерий среднеквадратичного приближения для метода наименьших квадратов:

Подкоренное выражение представляет собой квадратичную функцию относительно коэффициентов аппроксимирующего многочлена. Она непрерывна и дифференцируема по *c0, ..., cn*. Очевидно, что ее минимум находится в точке, где все частные производные равны нулю. Приравнивая к нулю частные производные, получим систему линейных алгебраических уравнений относительно неизвестных (искомых) коэффициентов многочлена наилучшего приближения.

5.

Где

Реализуем расчет коэффициентов системы линейных уравнений и решение самой системы в Matlab

Листинг файла lsquares.m

function c = lsquares(x,y,k)

% Базисные функции phi\_i(x)=x^k

A = zeros(k+1,k+1);

b = zeros(k+1,1);

for i=1:k+1

for j=1:k+1

sumA = 0;

sumB = 0;

for n=1:length(x)

sumA = sumA+x(n)^(i-1)\*x(n)^(j-1);

sumB = sumB+y(n)\*x(n)^(i-1);

end

A(i,j) = sumA;

b(i,1) = sumB;

end

end

% Решаем систему линейных уравнений Ac=b

c = A\b;

end

5. Расчет аппроцсимирующих функций произведем в скрипте polinomial\_fit.m

Листинг файла polinomial\_fit.m

clc

clear all

close all

a = 0;

b = 2;

n = 5;

% Создаем таблицу значений

[x\_table, y\_table] = generate\_table(a, b, n);

fprintf('Таблица значений:\n')

fprintf('x:')

fprintf(' %.4f', x\_table)

fprintf('\n')

fprintf('y:')

fprintf(' %.4f', y\_table)

fprintf('\n')

% Применяем МНК

c1 = lsquares(x\_table,y\_table,1);

c2 = lsquares(x\_table,y\_table,2);

c3 = lsquares(x\_table,y\_table,3);

% Выведем аппроксимирующие функции

fprintf('Аппроксимирующие функции:\n')

fprintf('Phi1(x) = %.4f + %.4f\*x\n',c1)

fprintf('Phi2(x) = %.4f + %.4f\*x + %.4f\*x^2\n',c2)

fprintf('Phi3(x) = %.4f + %.4f\*x + %.4f\*x^2 + %.4f\*x^3\n',c3)

% Значения точной функции

[x, y] = generate\_table(a, b, 100);

% Значения аппроксимирующих функций

Phi1 = c1(1)+c1(2).\*x;

Phi2 = c2(1)+c2(2).\*x+c2(3).\*x.^2;

Phi3 = c3(1)+c3(2).\*x+c3(3).\*x.^2+c3(4).\*x.^3;

% Строим графики

plot(x\_table,y\_table,'o')

hold on

plot(x,y)

plot(x,Phi1)

plot(x,Phi2)

plot(x,Phi3)

legend('table','real','Phi1','Phi2','Phi3')

grid on

Результат работы скрипта:

*Таблица значений:*

*x: 0.0000 0.5000 1.0000 1.5000 2.0000*

*y: 0.0000 0.2874 1.1359 1.2203 -1.3790*

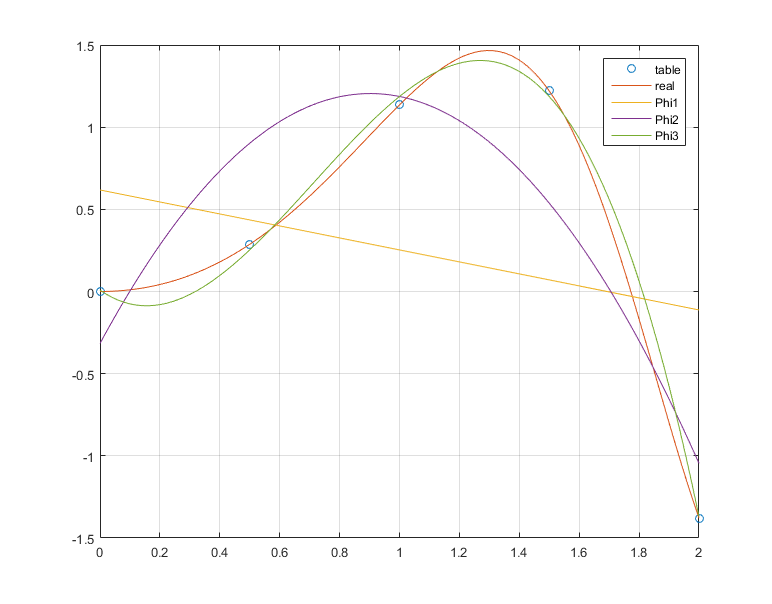
*Аппроксимирующие функции:*

*Phi1(x) = 0.6179 + -0.3650\*x*

*Phi2(x) = -0.3160 + 3.3706\*x + -1.8678\*x^2*

*Phi3(x) = 0.0085 + -1.2800\*x + 4.6214\*x^2 + -2.1631\*x^3*

6. Графики точной и приближенных функций



7. Из графика видно, что исходная функция лучше всего приближается полиномом третьей степени