МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа № 2

Выполнила: Яроцкас Анастасия, 2 курс, 4 группа
Преподаватель: Полещук Максим Александрович

Минск
2018

Постановка задачи:

Задание 1. Построить квадратичную функцию $P_2(x) = c_2x^2 + c_1x + c_0$, которая дает для $f(x)$ наилучшее приближение по методу наименьших квадратов

решением системы линейных уравнений $X^T Xc = X^T y$ с нормальной матрицей $X^T X = \left(\sum_{i=0}^{n} x_i^{k+l-2} \right)$ (normal equations). Функцию $f(x)$ взять из задания лабораторной работы 1. Задать значения y_i функции $f(x)$, определенной на интервале $[a,b]$, в узлах $x_i = a + \frac{b-a}{n} i, i = \overline{0,n}, n = 30$. Код решения системы линейных уравнений методом Гаусса с выбором главного элемента по столбцу (алгоритм GEPP) взять из лабораторной работы 1 курса «Вычислительные методы алгебры». Для вычислений использовать тип float.

В отчёте представить значения коэффициентов c_0, c_1, c_2, значение $\hat{\sigma}$ несмещенной оценки дисперсии случайных ошибок для выборок малого объема, $\hat{\sigma} = \sqrt{\frac{RSS}{n+1-k}} = \sqrt{\frac{1}{n+1-k} \sum_{i=0}^{n} (y_i - P_2(x_i))^2}$, где k — число степеней свободы, k=3. Построить на одном рисунке графики функций $f(x)$ и $P_2(x)$ на интервале $[a,b]$ для визуализации результата работы программы.

Задание 2. Для условия из задания 1 применить алгоритм QR-разложения матрицы X методом Хаусхолдера: решить систему линейных уравнений $Rc = Q^T y$. Код алгоритма QRразложения методом Хаусхолдера взять из лабораторной работы 2 курса «Вычислительные методы алгебры». Для вычислений использовать тип float.

В отчёте представить значения величин, указанных в задании 1. Сравнить значение величины среднеквадратичного уклонения со значением, полученным в задании 1. В этом задании графики функций строить не требуется.

Входные данные:

$2(\pi + 3x)^{1/2}, [1;10]$

Листинг1:

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <fstream>
#include <math.h>
#include <iomanip>

int n = 30, a = 1, b = 10;
float** table, **secondTable;
float* y, *tableY, *c, **tableX, *p;
```

```
void create() {
    table = new float*[n + 1];

    for (int i = 0; i < n+1; i++){
        table[i] = new float[3];
    }

    secondTable = new float*[3];
    for (int i = 0; i < 3; i++){
        secondTable[i] = new float[n+1];
    }

    y = new float[n + 1];

    tableX = new float*[3];

    for (int i = 0; i < 3; i++){
        tableX[i] = new float[3];
    }

    tableY = new float[3];
    c = new float[3];

    p = new float[n + 1];
}

void clear(){
    for (int i = 0; i < n+1; i++){
        delete table[i];
    }

    delete table;

    for (int i = 0; i < 3; i++){
        delete secondTable[i];
    }

    delete secondTable;

    delete y;

    for (int i = 0; i < 3; i++){
        delete tableX[i];
    }

    delete tableX;

    delete tableY;
    delete c;
    delete p;
}

float findSigma(){
    float sigma = 0;
```

```
    for (int i = 0; i < n + 1; i++){
        sigma += (y[i] - p[i])*(y[i] - p[i]);
    }

    sigma /= n - 2;

    return (float)sqrt(sigma);
}

void firstTask(){
    create();
    for (int i = 0; i < n + 1; i++){
        table[i][0] = secondTable[0][i] = 1;
        table[i][1] = secondTable[1][i] = a + (float)i*(b - a) / (n +
1);
        table[i][2] = secondTable[2][i] = (float)pow(a + (float)i*(b -
a) / (n + 1), 2);
    }

    for (int i = 0; i < n + 1; i++){
        y[i] = (float)(2 * pow((M_PI + 3 * table[i][1]), 0.5));
    }

    for (int i = 0; i < 3; i++){
        for (int l = 0; l < 3; l++){
            tableX[i][l] = 0;
            for (int j = 0; j < n + 1; j++){
                tableX[i][l] += secondTable[i][j] * table[j][l];
            }
        }
    }

    for (int i = 0; i < 3; i++) {
        tableY[i] = 0;

        for (int j = 0; j < n + 1; j++){
            tableY[i] += secondTable[i][j] * y[j];
        }

    }

    for (int index = 0; index < 3; index++) {
        int max = index;
        for (int i = index + 1; i < 3; i++){
            if (abs(tableX[i][index]) > abs(tableX[max][index])){
                max = i;
            }
        }

        float*temp = tableX[index];
        tableX[index] = tableX[max];
        tableX[max] = temp;

        float t = tableY[index];
        tableY[index] = tableY[max];
```

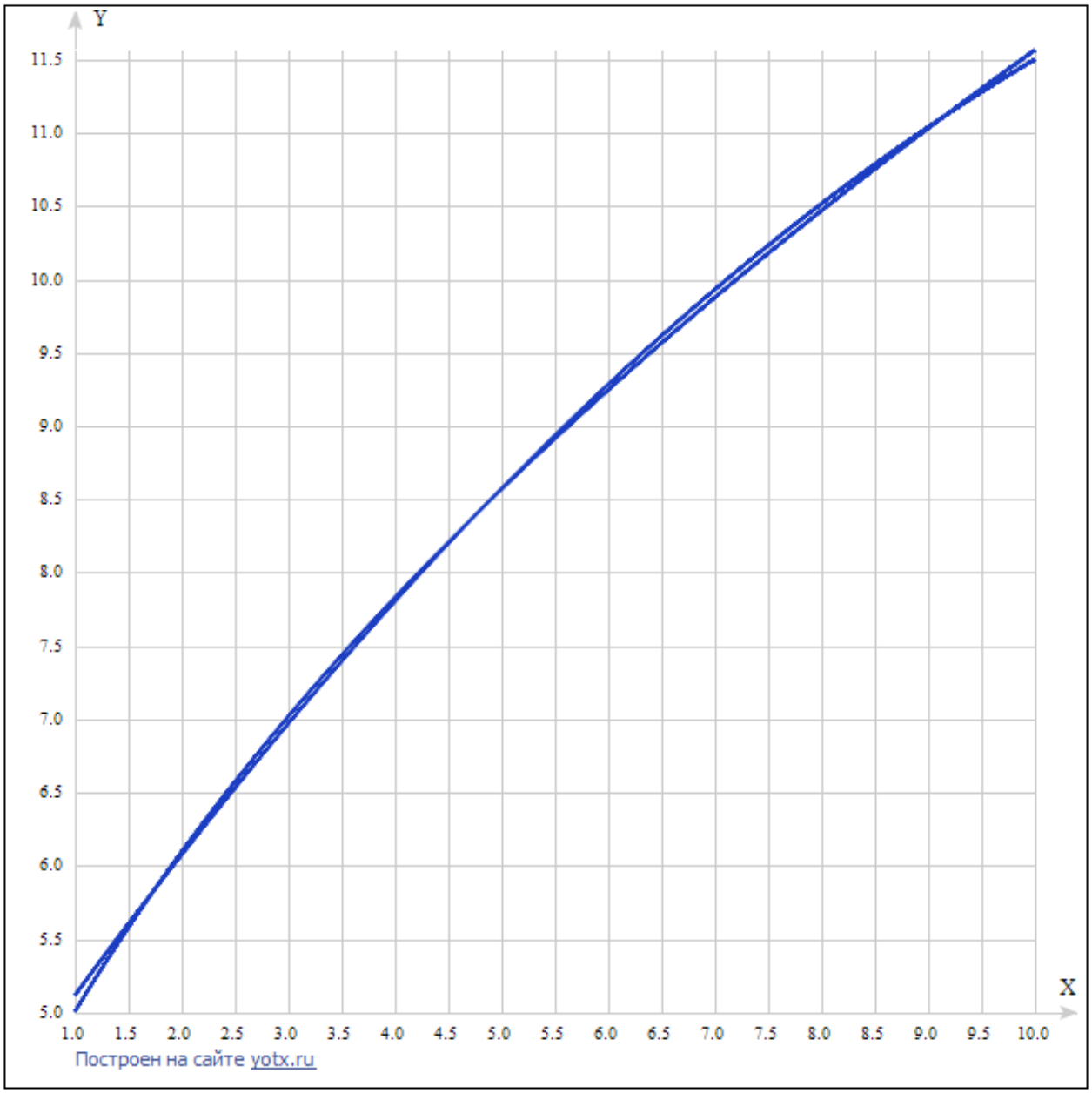
```
        tableY[max] = t;

        for (int i = index + 1; i < 3; i++) {
            float shadowTemp = tableX[i][index] / tableX[index][index];
            tableY[i] -= shadowTemp * tableY[index];

            for (int j = index; j < 3; j++){
                tableX[i][j] -= shadowTemp * tableX[index][j];
            }
        }

    }

    for (int i = 2; i >= 0; i--){
        float summa = 0;

        for (int j = i + 1; j < 3; j++){
            summa += tableX[i][j] * c[j];
        }

        c[i] = (tableY[i] - summa) / tableX[i][i];
    }

    std::cout << "Vector c: " << std::endl;

    for (int i = 0; i < 3; i++) {
        std::cout << std::setprecision(3) << c[i] << " ";
    }
    std::cout << std::endl;

    for (int i = 0; i < n + 1; i++){
        p[i] = c[2] * table[i][1] * table[i][1] + c[1] * table[i][1] +
c[0];
    }

    std::cout << "Sigma = " << std::setprecision(3) << findSigma();

    clear();
}

int main(){
    firstTask();
}
```

Вывод:

Vector c:

4.05 1.05 -0.0313

Sigma = 0.0414



Листинг2:

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <fstream>
#include <math.h>
#include <iomanip>

using namespace std;

int n = 30, a = 1, b = 10;
float** table, **secondTable, **W, *y, **w, **w_t, **E, **Q, **R,
    **Qt, *d, *c;
float** temp1, **temp2, **temp, *p;

void create(){
    table = new float*[n + 1];
    W = new float*[n + 1];
```

```
for (int i = 0; i < n + 1; i++) {
    table[i] = new float[3];
    W[i] = new float[3];
}

secondTable = new float*[n+1];

for (int i = 0; i < n+1; i++){
    secondTable[i] = new float[3];
}

y = new float[n + 1];
w = new float*[n + 1];

for (int i = 0; i < n + 1; i++){
    w[i] = new float[1];
}

w_t = new float*[1];

for (int i = 0; i < 1; i++){
    w_t[i] = new float[n + 1];
}

E = new float*[n + 1];

for (int i = 0; i < n + 1; i++){
    E[i] = new float[n + 1];
}

temp = new float*[n + 1];
temp1 = new float*[n + 1];
temp2 = new float*[n + 1];

for (int i = 0; i < n + 1; i++) {

    temp1[i] = new float[n + 1];
    temp2[i] = new float[n + 1];
    temp[i] = new float[n + 1];

    for(int j = 0; j < n + 1; j++){

        temp1[i][j] = 0;
        temp2[i][j] = 0;
        temp[i][j] = 0;

    }
}

Q = new float*[n + 1];

for (int i = 0; i < n + 1; i++)
    Q[i] = new float[3];

R = new float*[3];
```

```
    for (int i = 0; i < 3; i++) {
        R[i] = new float[3];
        for (int j = 0; j < 3; j++)
            R[i][j] = 0;
    }

    d = new float[3];
    c = new float[3];

    Qt = new float*[3];

    for (int i = 0; i < 3; i++)

        Qt[i] = new float[n+1];

    p = new float[n + 1];
}

void clear(){
    for (int i = 0; i < n + 1; i++) {
        delete table[i];
        delete W[i];
    }

    delete table;
    delete W;

    for (int i = 0; i < n+1; i++){
        delete secondTable[i];
    }

    delete secondTable;

    delete y;

    for (int i = 0; i < n + 1; i++){
        delete w[i];
    }

    delete w;

    for (int i = 0; i < 1; i++){
        delete w_t[i];
    }
    delete w_t;

    for (int i = 0; i < n + 1; i++){
        delete E[i];
    }

    delete E;

    for (int i = 0; i < n + 1; i++) {
        delete temp1[i];
        delete temp2[i];
        delete temp[i];
```

```
    }

    delete temp;
    delete temp1;
    delete temp2;

    for (int i = 0; i < n + 1; i++)
        delete Q[i];

    delete Q;

    for (int i = 0; i < 3; i++)
        delete R[i];

    delete R;

    delete d;
    delete c;

    delete Qt;

    for (int i = 0; i < 3; i++)
        delete Qt[i];

    delete p;
}

float getSigma(){
    float sigma = 0;

    for (int i = 0; i < n + 1; i++)

        sigma += (y[i] - p[i])*(y[i] - p[i]);

    sigma /= (n - 2);

    return (float)sqrt(sigma);
}

int main() {
    create();

    float s = 0, shadow = 0;


    for (int i = 0; i < n + 1; i++) {

        table[i][0]= 1;

        secondTable[i][0] = 1;

        table[i][1] =(float)a + (float)i*(b - a) / (n + 1);

        secondTable[i][1] = (float)a + (float)i*(b - a) / (n + 1);
```

```
        table[i][2] = (float)pow((float)a + (float)i*(b - a) / (n + 1),
2);

        secondTable[i][2] = (float)pow((float)a + (float)i*(b - a) / (n
+ 1), 2);

    }

    for (int i = 0; i < n + 1; i++){
        y[i] = (float)(2 * pow((M_PI + 3 * table[i][1]), 0.5));
    }

    for (int j = 0; j < 3; j++) {

        s = 0;

        for (int i = j; i < n + 1; i++)
            s += table[i][j] * table[i][j];

        s = (float)sqrt(s);

        if (table[j][j] > 0) d[j] = (-1)*s;

        else d[j] = s;

        shadow = sqrt(s*(s + abs(table[j][j])));

        table[j][j] -= d[j];

        for (int k = j; k < n + 1; k++)
            table[k][j] /= shadow;

        for (int i = j + 1; i < 3; i++) {

            s = 0;

            for (int k = j; k < n + 1; k++)
                s += table[k][j] * table[k][i];

            for (int k = j; k < n + 1; k++)
                table[k][i] -= table[k][j] * s;

        }

    }

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {

            R[i][i] = d[i];

            if (i < j)
                R[i][j] = table[i][j];

        }
    }
```

```
for (int i = 0; i < n + 1; i++) {
    for (int j = 0; j < 3; j++) {
        if (j <= i)
            W[i][j] = table[i][j];
        else W[i][j] = 0;
    }
}

for (int i = 0; i < n + 1; i++){
    for (int j = 0; j < n + 1; j++) {
        if (i == j) {
            E[i][j] = 1;
            temp2[i][j] = 1;
        }
        else {
            E[i][j] = 0;
        }
    }
}

for (int k = 0; k < 3; k++){
    for (int i = 0; i < n + 1; i++) {
        w[i][0] = W[i][k];
        w_t[0][i] = W[i][k];
    }

    for (int i = 0; i < n + 1; i++) {
        for (int l = 0; l < n + 1; l++) {
            s = 0;

            for (int j = 0; j < 1; j++)
                s += w[i][j] * w_t[j][l];

            temp[i][l] = s;
        }
    }

    for (int i = 0; i < n + 1; i++)
        for (int j = 0; j < n + 1; j++)
            temp[i][j] = E[i][j] - temp[i][j];

    for (int i = 0; i < n + 1; i++){
        for (int l = 0; l < n + 1; l++){
            s = 0;

            for (int j = 0; j < n + 1; j++)
                s += temp2[i][j] * temp[j][l];

            temp1[i][l] = s;
        }
    }

    for (int i = 0; i < n + 1; i++)
        for (int j = 0; j < n + 1; j++)
            temp2[i][j] = temp1[i][j];
```

```
}

for (int i = 0; i < n + 1; i++)
    for (int j = 0; j < 3; j++)
        Q[i][j] = temp2[i][j];

for (int i = 0; i < 3; i++)
    for (int j = 0; j < n + 1; j++)
        Qt[i][j] = Q[j][i];

float* Qty = new float[3];

for (int i = 0; i < 3; i++) {
    Qty[i] = 0;

    for (int j = 0; j < n + 1; j++)
        Qty[i] += Qt[i][j] * y[j];

}

for (int index = 0; index < 3; index++){

    int max = index;

    for (int i = index + 1; i < 3; i++) {
        if (abs(R[i][index]) > abs(R[max][index]))
            max = i;
    }

    float* tmp = R[index];

    R[index] = R[max];

    R[max] = tmp;

    float tmp2 = Qty[index];

    Qty[index] = Qty[max];

    Qty[max] = tmp2;

    for (int i = index + 1; i < 3; i++) {
        float shadowTemp = R[i][index] / R[index][index];
        Qty[i] -= shadowTemp * Qty[index];
        for (int j = index; j < 3; j++)
            R[i][j] -= shadowTemp * R[index][j];
    }
}

for (int i = 2; i >= 0; i--){
    float summa = 0;
    for (int j = i + 1; j < 3; j++)
        summa += R[i][j] * c[j];
    c[i] = (Qty[i] - summa) / R[i][i];
```

```
    }

    cout<<("Vector c: ");

    for (int i = 0; i < 3; i++)
        cout<<setprecision(3)<<c[i]<< " ";

    cout << endl;

    for (int i = 0; i < n + 1; i++)
        p[i] = c[2] * secondTable[i][1] * secondTable[i][1] + c[1] *
secondTable[i][1] + c[0];

    cout<<"Sigma = " << setprecision(3) << getSigma() << std::endl;

    clear();
}
```

Вывод:
Vector c: 4.05 1.05 -0.0313
Sigma = 0.0414