Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчёт по практике**

Дисциплина: Схемотехника операционных устройств

Тема: Определение статических характеристик преобразования средств измерений по экспериментальным данным.

|  |
| --- |
|  |
|  |  |

  Выполнил студент гр. 5130901/10101   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Т. Непомнящий

                                                                        (подпись)

Принял преподаватель   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ З. В. Куляшова

                                                                        (подпись)

                           “    ”                                              2023 г.

Санкт-Петербург

2023

**Оглавление**

[1. Цель работы 3](#_Toc147705954)

[2. Исходные данные 3](#_Toc147705955)

[3. Код программы 5](#_Toc147705956)

[4. Вывод программы 16](#_Toc147705957)

[5. Вывод 19](#_Toc147705958)

# Цель работы

Написать уравнение полинома, для которого известны результаты, полученные для x и y.

Задача состоит в определении количества и значений коэффициентов степенного полинома по результатам измерений входного и выходного сигналов так, чтобы полученный полином, аппроксимирующий полином (3.32), отличался от него не более, чем на погрешность, обусловленную погрешностью измерений.

# Исходные данные

Значения измеряемой величины х:

x = [

0.0712

2.7897

2.9809

3.7974

4.0810

4.7488

]

Результаты измерений (значения y, представлены в виде матрицы):

y = [

0.9804 18.5756 21.5603 35.8649 41.9397 57.8564

1.4286 18.5735 21.1411 36.0852 42.0414 57.1727

1.3475 19.1585 21.3784 35.8312 42.3798 58.2495

1.7985 18.8075 22.2480 36.0961 42.5143 58.3126

0.6506 18.6963 21.7131 36.4700 42.1848 57.7973

1.0413 18.2677 21.6050 35.9774 41.8115 58.3447

]

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Рис. 1 – Представление исходных данных на графике

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 2 – Задание исходных данных

**Изображение выглядит как снимок экрана, линия, текст, число

Автоматически созданное описание**

Рис. 3 – Средние значения (фиолетовые точки на графике)

# Код программы

Для ускорения подсчётов, чтобы избежать дополнительных ошибок при вводе данных и сделать решение более гибким, создадим программу средствами Matlab для нахождения. В программе реализован поиск полинома, а также обработка исходных данных и таблиц (например, по заданным n и k, которые есть в шапке таблицы, код сможет определить какое значение имеет коэффициент G). Для удобства в коде указаны комментарии

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 1. Вычисление мат. ожиданий

disp**(**'1. Вычисление мат. ожиданий'**);**

disp**(**' '**);**

% Создание множества занчений y

y **=** **[**

0.9804 18.5756 21.5603 35.8649 41.9397 57.8564**;**

1.4286 18.5735 21.1411 36.0852 42.0414 57.1727**;**

1.3475 19.1585 21.3784 35.8312 42.3798 58.2495**;**

1.7985 18.8075 22.2480 36.0961 42.5143 58.3126**;**

0.6506 18.6963 21.7131 36.4700 42.1848 57.7973**;**

1.0413 18.2677 21.6050 35.9774 41.8115 58.3447

**];**

% Создание множества занчений x

x **=** **[**

0.0712

2.7897

2.9809

3.7974

4.0810

4.7488

**];**

% число экспериментов

tests\_amount **=** 6**;** % k

parallel\_tests **=** 6**;** % n

% Инициализация пустого вектора для хранннения решений

y\_average **=** zeros**(**tests\_amount**,**1**);**

% Подсчёт s\_n

**for** i **=** 1**:**tests\_amount

% s\_n - сумма от 1 до n, обнуляем сумму после нахождения s\_n\_i

s\_n **=** 0**;**

% Перебор всех комбинаций элементов из двух векторов

**for** j **=** 1**:**parallel\_tests

% Укажем элемент матрицы, который рассматриваем в данный момент

% внутри цикла

element\_matrix **=** y**(**i**,**j**);**

% Решение уравнения с использованием текущего элемента

s\_n **=** s\_n **+** element\_matrix**;**

**end**

% Добавление значение s\_i

y\_average**(**i**)** **=** s\_n**/**parallel\_tests**;**

**end**

disp**(**'Вычислим мат. ожидания y:'**);**

**for** k **=** 1**:**tests\_amount

% Вывод множества решений с 5 знаками после запятой

fprintf**(**'y\_average\_%d = %.8f\n'**,** k**,** y\_average**(**k**));**

**end**

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 2. Вычисление дисперсии

disp**(**'2. Вычисление дисперсии'**);**

% Инициализация пустого вектора для хранения решений

s\_i **=** zeros**(**1**,**tests\_amount**);**

% Инициализация пустого вектора для хранения решений

s\_i\_2 **=** zeros**(**1**,**tests\_amount**);**

% Подсчёт s\_i^2

**for** i **=** 1**:(**length**(**y\_average**))**

% s\_n - сумма от 1 до n, обнуляем сумму после нахождения s\_i^2

s\_n **=** 0**;**

s\_i\_res **=** 0**;**

% Перебор всех комбинаций элементов из двух векторов

**for** j **=** 1**:**parallel\_tests

% Укажем элемент матрицы и вектора, который рассматриваем в данный

% момент внутри цикла

element\_matrix **=** y**(**i**,**j**);**

element\_vector **=** y\_average**(**j**);**

% Решение уравнения с использованием элементов, указанных выше

s\_n **=** s\_n **+** **((**element\_matrix**-**element\_vector**)^**2**);**

s\_i\_res **=** s\_i\_res **+** **(**element\_matrix**-**element\_vector**);**

**end**

% Добавление значение s\_i

s\_i**(**i**)** **=** s\_i\_res**/(**parallel\_tests**-**1**);**

% Добавление значение s\_i^2

s\_i\_2**(**i**)** **=** s\_n**/(**parallel\_tests**-**1**);**

**end**

fprintf**(**'\nМножество решений s\_i^2:\n'**);**

**for** k **=** 1**:**tests\_amount

% Вывод множества решений

disp**([**'s\_'**,** num2str**(**k**),**' = '**,** sprintf**(**'%.8f'**,** s\_i\_2**(**k**))]);**

**end**

disp**(**' '**);**

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 3. Проверка выполнения гипотезы Кохрена

disp**(**'3. Проверка выполнения гипотезы Кохрена'**);**

% Нахождение максимального значения среди s\_i^2

max\_s\_i\_2 **=** max**(**s\_i\_2**);**

fprintf**(**'\nmax s\_i^2 = %f\n'**,** max\_s\_i\_2**);**

% Обнуляем сумму

sum\_s\_i **=** 0**;**

fprintf**(**'\nКритерий Кохрена:\n'**);**

**for** g **=** 1**:**tests\_amount

sum\_s\_i **=** sum\_s\_i **+** s\_i\_2**(**g**);**

**end**

G **=** max\_s\_i\_2 **/** sum\_s\_i**;**

disp**([**'G = '**,** sprintf**(**'%.8f'**,** G**)]);**

disp**(**' '**);**

% Зададим критические значения критерия Кохрена в виде матрицы:

% \*берём за данное, что alpha = 0.05

Cochran **=** **[**

% n\k

1 2 3 4 5 6 7 **;**

6 0.8772 0.7071 0.5895 0.5063 0.4447 0.3974 **;**

8 0.8332 0.6530 0.5365 0.4564 0.3980 0.3535 **;**

10 0.8010 0.6167 0.5017 0.4241 0.3682 0.3259 **;**

12 0.7910 0.5020 0.4780 0.4020 0.3460 0.3050

**];**

alpha **=** 0.05**;**

Cochran\_n **=** parallel\_tests**;**

Cochran\_k **=** tests\_amount**;**

disp**([**'alpha = '**,** sprintf**(**'%.2f'**,** alpha**)]);**

fprintf**(**'n = %d\nk = %d\n'**,** Cochran\_n**,** Cochran\_k**);**

% Поиск крит. значения согласно таблице выше

G\_n **=** **-**1**;**

**for** n **=** 1**:**5

**for** k **=** 1**:**7

**if** Cochran**(**n**,**1**)** **==** Cochran\_n % ищем в 1 столбце

G\_n **=** n**;**

% fprintf('\nn = %d\n', G\_n);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** G\_n **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 5 **&&** k **==** 7

disp**(**"Error: cell wasn't find"**);**

**end**

**end**

G\_k **=** **-**1**;**

**for** n **=** 1**:**5

**for** k **=** 1**:**7

**if** Cochran**(**1**,**k**)** **==** Cochran\_n % ищем в 1 строке

G\_k **=** k**;**

% fprintf('\nk = %d\n', G\_k);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** G\_k **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 5 **&&** k **==** 7

disp**(**"Error: cell wasn't find"**);**

**end**

**end**

G\_critical **=** Cochran**(**G\_n**,**G\_k**);** % выделяем нужный элемент

fprintf**(**'\nКрит. значение критерия Кохрена:\n'**);**

disp**([**'G\_critical = '**,** sprintf**(**'%.4f'**,** G\_critical**)]);**

disp**(**' '**);**

disp**(**'Вектор ср. значений Y:'**);**

Y **=** y\_average**;**

disp**(**Y**);** % выводим матрицу

S **=** zeros**(**parallel\_tests**,**tests\_amount**);** % создаем матрицу из нулей

% заполняем главную диагональ значениями из вектора

**for** i **=** 1**:**min**(**parallel\_tests**,** tests\_amount**)**

S**(**i**,** i**)** **=** s\_i\_2**(**i**);**

**end**

disp**(**'Матрица оценок дисперсий S:'**);**

disp**(**S**);** % выводим матрицу

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 4. Выбор метода обработки

disp**(**'4. Выбор метода обработки'**);**

disp**(**' '**);**

q **=** 0**;**

**if** G **<** G\_critical

disp**(**"G < G\_critical => Случай равноточных измерений"**);**

**elseif** G **>** G\_critical

disp**(**'G > G\_critical => Случай неравноточных измерений'**);**

**else**

disp**(**"G = G\_critical"**);**

**end**

fprintf**(**'\nЗапишем условие полезной избыточности: k > p + 1 => '**);**

fprintf**(**'p < %d - 1 => p < %d\n'**,** tests\_amount**,** tests\_amount **-** 1**);**

p **=** tests\_amount**;**

% ищем p

**for** p\_counter **=** 1**:(**tests\_amount **-** 2**)**

p **=** p\_counter**;**

fprintf**(**'\n%d) Предположим, что p = %d:\n\n'**,** p\_counter**,** p**);**

X **=** zeros**(**tests\_amount**,** p**+**1**);** % создаем матрицу из нулей

% заполняем матрицу x

X **=** repmat**(**x**,** 1**,** p**+**1**);**

**for** X\_j **=** 1**:**p**+**1

% возводим каждый столбец в степень от 1 до p+1

X**(:,** X\_j**)** **=** X**(:,** X\_j**)** **.^** **(**X\_j **-** 1**);**

**end**

disp**(**' Матрица X:'**);**

disp**(**X**);** % выводим матрицу

disp**(**' Транспонированная матрица X\_transposed:'**);**

X\_transposed **=** transpose**(**X**);**

disp**(**X\_transposed**);** % выводим обратную матрицу

**if** G **<** G\_critical

% Найдём вектор оценок коэффициента A

% Найдем X\_transposed \* X

disp**(**' X\_transposed \* X:'**);**

XTX **=** X\_transposed **\*** X**;**

disp**(**XTX**);**

disp**(**' (X\_transposed \* X)^-1:'**);**

disp**(**inv**(**XTX**));**

% Найдем X\_transposed \* Y

disp**(**' X\_transposed \* Y:'**);**

XTY **=** X\_transposed **\*** Y**;**

disp**(**XTY**);**

% Теперь мы можем решить уравнение XTX\*A = XTY для A

% Используем функцию pinv() для вычисления псевдообратной матрицы XTX

% Далее умножим на XTY для получения A

disp**(**' Найдём вектор A:'**);**

A **=** XTX**\**XTY**;** % = pinv(XTX) \* XTY

disp**(**A**);** % выводим вектор

disp**([**' max s\_i^2 = '**,** sprintf**(**'%.6f'**,** max\_s\_i\_2**)]);**

disp**([**' n = '**,** sprintf**(**'%.0f'**,** parallel\_tests**)]);**

% Вычислим дисперсионную матрицу оценок коэффициентов

disp**(**' '**);**

disp**(**' Дисперсионная матрица оценок коэффициентов S\_a:'**);**

S\_a **=** max\_s\_i\_2**/**parallel\_tests **\*** pinv**(**XTX**);**

disp**(**S\_a**);**

**elseif** G **>** G\_critical

% Найдём вектор оценок коэффициента A

disp**(**' Обратная матрица: S^-1:'**);**

disp**(**pinv**(**S**));**

% Найдем (X\_transposed \* S^-1 \* X)^-1

disp**(**' X\_transposed \* S^-1 \* X:'**);**

XtSX **=** X\_transposed **\*** pinv**(**S**)** **\*** X**;**

disp**(**XtSX**);**

% Найдем X\_transposed \* S^-1 \* Y

disp**(**' X\_transposed \* S^-1 \* Y:'**);**

XtSY **=** X\_transposed **\*** pinv**(**S**)** **\*** Y**;**

disp**(**XtSY**);**

% Теперь мы можем решить уравнение A = pinv\_XtSX \* XtSY для A

% Используем функцию pinv() для вычисления псевдообратной матрицы XtSX

% Далее умножим на XTY для получения A

disp**(**' Найдём вектор A:'**);**

A **=** pinv**(**XtSX**)** **\*** XtSY**;** % = pinv(XtSX) \* XtSY

disp**(**A**);** % выводим вектор

% Вычислим дисперсионную матрицу оценок коэффициентов

disp**(**' '**);**

disp**(**' Дисперсионная матрица оценок коэффициентов S\_a:'**);**

S\_a **=** pinv**(**XtSX**)/**parallel\_tests**;**

disp**(**S\_a**);**

**else**

disp**(**" G = G\_critical"**);**

**end**

XAY **=** X **\*** A **-** Y**;**

disp**(**' Обратная матрица: S^-1:'**);**

disp**(**pinv**(**S**));**

R **=** parallel\_tests **\*** transpose**(**XAY**)** **\*** pinv**(**S**)** **\*** **(**XAY**);**

disp**([**' R^2 = '**,** sprintf**(**'%.16f'**,** R**)]);**

fprintf**(**'\n Проверим гипотезу с помощью упрощённого критерия Фишера:\n'**);**

% Зададим критические значения критерия Фишера в виде матрицы:

% \* берём за данное, что alpha = 0.05

Fisher **=** **[**

% n\k

1 3 4 5 6 7 8 9 10 11 12 **;**

5 5.41 5.19 5.05 4.95 4.88 4.82 4.77 4.73 4.68 4.62 **;**

7 4.35 4.12 3.97 3.87 3.79 3.72 3.68 3.63 3.57 3.51 **;**

9 3.86 3.63 3.48 3.37 3.29 3.23 3.18 3.14 3.07 3.00 **;**

11 3.58 3.36 3.20 3.09 3.01 2.95 2.90 2.85 2.79 2.72 **;**

13 3.41 3.18 3.02 2.91 2.83 2.77 2.71 2.67 2.63 2.60 **;**

15 3.29 3.05 2.90 2.79 2.71 2.64 2.59 2.54 2.50 2.47 **;**

17 3.19 2.96 2.81 2.70 2.61 2.55 2.49 2.45 2.41 2.38

**];**

alpha **=** 0.05**;**

fprintf**(**' k = %d\n'**,** tests\_amount**);**

fprintf**(**' p = %d\n'**,** p**);**

fprintf**(**' n = %d\n'**,** parallel\_tests**);**

Fisher\_k1 **=** tests\_amount **-** p **-** 1**;**

Fisher\_k2 **=** parallel\_tests **-** 1**;**

disp**([**' alpha = '**,** sprintf**(**'%.2f'**,** alpha**)]);**

fprintf**(**'\n k1 = k - p - 1 = %d\n k2 = n - 1 = %d\n'**,** Fisher\_k1**,** Fisher\_k2**);**

% Поиск крит. значения согласно таблице выше

F\_n **=** **-**1**;**

**for** n **=** 1**:**8

**for** k **=** 1**:**11

**if** Fisher**(**n**,**1**)** **==** Fisher\_k2 % ищем в 1 столбце

F\_n **=** n**;**

% fprintf('\nn = %d\n', F\_n);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** F\_n **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 8 **&&** k **==** 11

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

F\_k **=** **-**1**;**

**for** n **=** 1**:**8

**for** k **=** 1**:**11

**if** Fisher**(**1**,**k**)** **==** Fisher\_k1 % ищем в 1 строке

F\_k **=** k**;**

% fprintf('\nk = %d\n', F\_k);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** F\_k **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 8 **&&** k **==** 11

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

% Находим крит. значение критерия Фишера

F\_critical **=** Fisher**(**F\_n**,**F\_k**);** % выделяем нужный элемент

disp**([**' Крит. значение критерия Фишера: F\_critical = '**,** sprintf**(**'%.2f'**,** F\_critical**)]);**

fprintf**(**"\n Проверим, выполняется ли неравенство для R^2:\n"**);**

R\_Fisher **=** **(**tests\_amount **-** p **-** 1**)** **\*** F\_critical**;**

**if** R **<** R\_Fisher

disp**([**' R^2 = '**,** sprintf**(**'%.4f'**,** R**)]);**

disp**([**' R\_Fisher = '**,** sprintf**(**'%.4f'**,** R\_Fisher**)]);**

disp**(**' '**);**

disp**(**'Гипотеза не противоречит эксперментальным данным'**);**

disp**([**'степень полинома: q = '**,** sprintf**(**'%.0f'**,** p**)]);**

**break**

**else**

disp**(**'Гипотеза противоречит эксперментальным данным, повысим степень полинома'**);**

**end**

p **=** p **+** 1**;** % увеличиваем счётчик цикла

**end**

fprintf**(**'y = a0'**);**

**for** q\_counter **=** 1**:**q**+**1

fprintf**(**' + a%d \* x^%d'**,** q\_counter**,** q\_counter**);**

**end**

fprintf**(**'\ny = %.4f%d'**,** A**(**1**));**

**for** q\_counter **=** 1**:**q**+**1

fprintf**(**' + %.4f%d'**,** A**(**q\_counter**+**1**));**

fprintf**(**' \* x^%d\n'**,** q\_counter**);**

**end**

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 5. Определение коэффициента преобразования линейной

% статической характеристики преобразования.

disp**(**' '**);**

disp**(**'5. Определение коэффициента преобразования линейнойи'**);**

disp**(**'статической характеристики преобразования.'**);**

**for** p\_counter **=** 1**:(**tests\_amount **-** 2**)**

p **=** p\_counter**;**

disp**(**' '**);**

fprintf**(**'%d) Предположим, что p = %d:\n\n'**,** p\_counter**,** p**);**

% обнулим промежуточные значения

sum\_1 **=** 0**;**

sum\_2 **=** 0**;**

sum\_3 **=** 0**;**

**if** G **<** G\_critical

disp**(**' G < G\_critical => Случай равноточных измерений'**);**

% найдём оценку коэф. a1

**for** i **=** 1**:**tests\_amount

sum\_1 **=** sum\_1 **+** y\_average**(**i**)** **\*** x**(**i**);** % числитель дроби

sum\_2 **=** sum\_2 **+** x**(**i**)^**2**;** % знаменатель

a1 **=** sum\_1**/**sum\_2**;** % оценка коэф a1

**end**

disp**(**' Оценка коэф. a1:'**);**

disp**([**' a1 = '**,** sprintf**(**'%.6f'**,** a1**)]);**

disp**(**' '**);**

% среднеквадратическое значение погрешности

disp**(**' Среднеквадратическое значение его погрешности s\_a\_1:'**);**

s\_a\_1 **=** **(**max\_s\_i\_2 **/** **(**n **\*** sum\_2**))^-**2**;**

disp**([**' s\_a\_1 = '**,** sprintf**(**'%.6f'**,** s\_a\_1**)]);**

disp**(**' '**);**

**elseif** G **>** G\_critical

disp**(**' G > G\_critical => Случай неравноточных измерений'**);**

% найдём оценку коэф. a1

**for** i **=** 1**:**tests\_amount

sum\_1 **=** sum\_1 **+** **(**y\_average**(**i**)** **\*** x**(**i**))** **/** s\_i\_2**(**i**);** % числитель дроби

sum\_2 **=** sum\_2 **+** **(**x**(**i**)^**2 **/** s\_i**(**i**))^**2**;** % знаменатель

a1 **=** sum\_1**/**sum\_2**;** % оценка коэф a1

**end**

disp**(**' Оценка коэф. a1:'**);**

disp**([**' a1 = '**,** sprintf**(**'%.6f'**,** a1**)]);**

disp**(**' '**);**

% среднеквадратическое значение погрешности

disp**(**' Среднеквадратическое значение его погрешности s\_a\_1:'**);**

s\_a\_1 **=** **(**1 **/** **(**n **\*** sum\_2**))^-**2**;**

disp**([**' s\_a\_1 = '**,** sprintf**(**'%.6f'**,** s\_a\_1**)]);**

disp**(**' '**);**

**else**

disp**(**" G = G\_critical"**);**

**end**

% проверка гипотезы

disp**(**' Проверка статистической гипотезы:'**);**

**for** i **=** 1**:**tests\_amount

sum\_3 **=** sum\_3 **+** **((**a1 **\*** x**(**i**)** **-** y\_average**(**i**))^**2**)/**s\_i\_2**(**i**);**

**end**

R\_2 **=** tests\_amount **\*** sum\_3**;**

disp**([**' R^2 = '**,** sprintf**(**'%.6f'**,** R\_2**)]);**

**if** R **<** R\_Fisher

disp**([**' R\_Fisher = '**,** sprintf**(**'%.4f'**,** R\_Fisher**)]);**

disp**(**' '**);**

disp**(**'Фактическая нелинейность, если она есть, настолько мала, что не может быть'**);**

fprintf**(**'выявлена на фоне погрешностей измерений =>'**);**

disp**([**' Степень полинома: q = '**,** sprintf**(**'%.0f'**,** p**)]);**

**break;**

**else**

disp**(**'Гипотеза противоречит эксперментальным данным, повысим степень полинома'**);**

**end**

p **=** p **+** 1**;** % увеличиваем счётчик цикла

**end**

disp**(**' '**);**

disp**(**'Оценка статической характеристики преобразования:'**);**

fprintf**(**'y = %.4f%d'**,** a1**);**

fprintf**(**' \* x\n'**);**

disp**(**repmat**(**'-'**,** 1**,** 100**));**

% 6. Оценка характеристик погрешности средства измерений

% по результатам определения линейной статической характеристики

% преобразования.

disp**(**' '**);**

disp**(**'6. Оценка характеристик погрешности'**);**

disp**(**' '**);**

% Инициализация пустой матрицы delta для хранения решений

delta **=** zeros**(**tests\_amount**,**parallel\_tests**);**

% Инициализация пустого вектора delta\_i для хранения решений

delta\_i **=** zeros**(**tests\_amount**,**1**);**

**for** i **=** 1**:**tests\_amount

% сумма от 1 до n, обнуляем сумму после нахождения для i-го элемента

s\_delta\_i **=** 0**;**

**for** k **=** 1**:**parallel\_tests

delta**(**i**,**k**)** **=** y**(**i**,**k**)** **-** a1 **\*** x**(**i**);**

s\_delta\_i **=** s\_delta\_i **+** delta**(**i**,**k**);**

**end**

% Добавление значения в delta\_i

delta\_i**(**i**)** **=** s\_delta\_i**/**parallel\_tests**;**

**end**

disp**(**'Вычислим выборочные значения погрешности в каждой i-ой точке диапазона измерений:'**);**

disp**(**'delta:'**);**

disp**(**delta**);**

disp**(**'Оценки систематической составляющей погрешности:'**);**

disp**(**'delta\_i:'**);**

disp**(**delta\_i**);**

% Инициализация пустого вектора delta\_s\_i для хранения решений

delta\_s\_i **=** zeros**(**tests\_amount**,**1**);**

**for** i **=** 1**:**tests\_amount

% сумма от 1 до n, обнуляем сумму после нахождения для i-го элемента

s\_delta\_s\_i **=** 0**;**

**for** k **=** 1**:**parallel\_tests

s\_delta\_s\_i **=** s\_delta\_s\_i **+** **(**delta**(**i**,**k**)** **-** delta\_i**(**i**))^**2**;**

**end**

% Добавление значения в delta\_s\_i

delta\_s\_i**(**i**)** **=** **(**s\_delta\_s\_i**/(**parallel\_tests **-** 1**))^-**2**;**

**end**

disp**(**'Оценки среднеквадратического значения случайной составляющей погрешности:'**);**

disp**(**'s\_i:'**);**

disp**(**delta\_s\_i**);**

% Зададим толерантный множитель K(n,0.95,Q) в виде матрицы:

% \* берём за данное, что P = 0.95

K\_tolerant **=** **[**

% Q\n

1 3 4 5 6 7 8 9 10 **;**

0.80 5.090 3.935 3.440 3.167 2.980 2.860 2.770 2.690 **;**

0.95 9.916 6.370 5.079 4.414 4.007 3.732 3.532 3.379

**];**

disp**(**'Зададим коэффициенты для нахождения толерантного множителя:'**);**

P **=** 0.95**;**

Q **=** 0.80**;**

K\_n **=** 5**;**

disp**([**'P = '**,** sprintf**(**'%.2f'**,** P**)]);**

disp**([**'Q = '**,** sprintf**(**'%.2f'**,** Q**)]);**

disp**([**'n = '**,** sprintf**(**'%.0f'**,** K\_n**)]);**

% Поиск толерантного множителя согласно таблице выше

K\_i **=** **-**1**;**

**for** i **=** 1**:**3

**for** j **=** 1**:**9

**if** K\_tolerant**(**i**,**1**)** **==** Q % ищем строку элемента

K\_i **=** i**;**

% fprintf('\ni = %d\n', K\_i);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** K\_i **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 3 **&&** k **==** 9

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

K\_j **=** **-**1**;**

**for** i **=** 1**:**3

**for** j **=** 1**:**9

**if** K\_tolerant**(**1**,**j**)** **==** K\_n % ищем столбец элемента

K\_j **=** j**;**

% fprintf('\nj = %d\n', K\_j);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** K\_j **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 3 **&&** k **==** 9

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

% Находим толерантный множитель K

K **=** K\_tolerant**(**K\_i**,**K\_j**);** % выделяем нужный элемент

disp**([**'Толерантный множитель: K(n,P,Q) = '**,** sprintf**(**'%.2f'**,** K**)]);**

disp**(**' '**);**

disp**(**'Вычислим толерантные пределы в каждой i-ой точке:'**);**

lim\_min **=** zeros**(**1**,**tests\_amount**);**

lim\_max **=** zeros**(**1**,**tests\_amount**);**

**for** i **=** 1**:**tests\_amount

lim\_left **=** delta\_i**(**i**)** **-** **(**delta\_s\_i**(**i**)** **\*** K**);**

lim\_min**(**i**)** **=** lim\_left**;**

lim\_right **=** delta\_i**(**i**)** **+** **(**delta\_s\_i**(**i**)** **\*** K**);**

lim\_max**(**i**)** **=** lim\_right**;**

fprintf**(**'i = %d: [ %.8f%d'**,** i**,** lim\_left**);**

fprintf**(**' , %.8f%d'**,** lim\_right**);**

disp**(**' ]'**);**

diff **=** abs**(**lim\_left **-** lim\_right**);**

**end**

min\_delta **=** min**(**lim\_min**());**

max\_delta **=** max**(**lim\_max**());**

fprintf**(**'\nmax[ %.4f%d'**,** min\_delta**);**

fprintf**(**' , %.4f%d'**,** max\_delta**);**

disp**(**' ]'**);**

fprintf**(**'Максимальный разброс = %.4f%d'**,** max\_delta**-**min\_delta**);**

fprintf**(**'\n\nХарактеристика аддитивной погрешности:\n'**);**

fprintf**(**'delta\_a = g = max[|min|, |max|] = '**);**

**if** abs**(**min\_delta**)** **>=** abs**(**max\_delta**)**

delta\_a **=** min\_delta**;**

**else**

delta\_a **=** max\_delta**;**

**end**

disp**([**sprintf**(**'%.4f'**,** abs**(**delta\_a**))]);**

% Зададим Коэффициенты Стъюдента в виде матрицы:

t\_Student **=** **[**

% Q\n

1 4 5 6 8 10 12 **;**

0.80 1.64 1.53 1.48 1.41 1.38 1.36 **;**

0.95 3.18 2.78 2.57 2.36 2.26 2.20

**];**

disp**(**' '**);**

disp**(**'Зададим коэффициенты для нахождения Коэффициенты Стъюдента:'**);**

Q **=** 0.80**;**

disp**([**'Q = '**,** sprintf**(**'%.2f'**,** Q**)]);**

disp**([**'n = '**,** sprintf**(**'%.0f'**,** tests\_amount**)]);**

% Поиск толерантного множителя согласно таблице выше

t\_i **=** **-**1**;**

**for** i **=** 1**:**3

**for** j **=** 1**:**9

**if** t\_Student**(**i**,**1**)** **==** Q % ищем строку элемента

t\_i **=** i**;**

% fprintf('\ni = %d\n', K\_i);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** t\_i **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 3 **&&** k **==** 9

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

t\_j **=** **-**1**;**

**for** i **=** 1**:**3

**for** j **=** 1**:**9

**if** t\_Student**(**1**,**j**)** **==** K\_n % ищем столбец элемента

t\_j **=** j**;**

% fprintf('\nj = %d\n', K\_j);

**break;**

**else**

**continue;**

**end**

**end**

% Критерий для выхода из цикла, показывает, что элемент найден

**if** t\_j **>** 0

**break;**

**end**

% После прохождения по матрице не найден элемент -> ошибка

**if** n **==** 3 **&&** k **==** 9

disp**(**" Error: cell wasn't find"**);**

**end**

**end**

% Находим толерантный множитель K

t\_S **=** t\_Student**(**t\_i**,**t\_j**);** % выделяем нужный элемент

disp**([**'Коэффициент Стъюдента: t(n - 1) = '**,** sprintf**(**'%.2f'**,** t\_S**)]);**

delta\_K **=** s\_a\_1 **\*** t\_S**;**

fprintf**(**'\nС вероятностью %.2f%d'**,** Q**);**

fprintf**(**' модуль погрешности коэффициента преобразования не превосходит значения:\n'**);**

disp**([**'delta\_K = delta\_a1 = '**,** sprintf**(**'%.4f'**,** delta\_K**)]);**

disp**(**' '**);**

disp**(**'Определим коэфиициенты c и d:'**);**

d\_coef **=** delta\_a **/** abs**(**x**(**tests\_amount**))** **\*** 100**;**

fprintf**(**'d = %.4f%d'**,** d\_coef**);**

disp**(**' %'**);**

c\_coef **=** **((**delta\_K **/** a1**)** **+** d\_coef**)** **\*** 100**;**

fprintf**(**'c = %.4f%d'**,** c\_coef**);**

disp**(**' %'**);**

# Вывод программы

Ниже представлен вывод программы в окне Command Window для заданных значений x и y:

>> final

----------------------------------------------------------------------------------------------------

1. Вычисление мат. ожиданий

Вычислим мат. ожидания y:

y\_average\_1 = 29.46288333

y\_average\_2 = 29.40708333

y\_average\_3 = 29.72415000

y\_average\_4 = 29.96283333

y\_average\_5 = 29.58535000

y\_average\_6 = 29.50793333

----------------------------------------------------------------------------------------------------

2. Вычисление дисперсии

Множество решений s\_i^2:

s\_1 = 397.26425041

s\_2 = 386.98643742

s\_3 = 397.87462319

s\_4 = 393.60912415

s\_5 = 402.08461947

s\_6 = 403.00114495

----------------------------------------------------------------------------------------------------

3. Проверка выполнения гипотезы Кохрена

max s\_i^2 = 403.001145

Критерий Кохрена:

G = 0.16926988

alpha = 0.05

n = 6

k = 6

Крит. значение критерия Кохрена:

G\_critical = 0.4447

Вектор ср. значений Y:

29.4629

29.4071

29.7241

29.9628

29.5854

29.5079

Матрица оценок дисперсий S:

397.2643 0 0 0 0 0

0 386.9864 0 0 0 0

0 0 397.8746 0 0 0

0 0 0 393.6091 0 0

0 0 0 0 402.0846 0

0 0 0 0 0 403.0011

----------------------------------------------------------------------------------------------------

4. Выбор метода обработки

G < G\_critical => Случай равноточных измерений

Запишем условие полезной избыточности: k > p + 1 => p < 6 - 1 => p < 5

1) Предположим, что p = 1:

Матрица X:

1.0000 0.0712

1.0000 2.7897

1.0000 2.9809

1.0000 3.7974

1.0000 4.0810

1.0000 4.7488

Транспонированная матрица X\_transposed:

1.0000 1.0000 1.0000 1.0000 1.0000 1.0000

0.0712 2.7897 2.9809 3.7974 4.0810 4.7488

X\_transposed \* X:

6.0000 18.4690

18.4690 70.2992

(X\_transposed \* X)^-1:

0.8712 -0.2289

-0.2289 0.0744

X\_transposed \* Y:

177.6502

547.3854

Найдём вектор A:

29.4829

0.0408

max s\_i^2 = 403.001145

n = 6

Дисперсионная матрица оценок коэффициентов S\_a:

58.5167 -15.3735

-15.3735 4.9944

Обратная матрица: S^-1:

0.0025 0 0 0 0 0

0 0.0026 0 0 0 0

0 0 0.0025 0 0 0

0 0 0 0.0025 0 0

0 0 0 0 0.0025 0

0 0 0 0 0 0.0025

R^2 = 0.0028764532770471

Проверим гипотезу с помощью упрощённого критерия Фишера:

k = 6

p = 1

n = 6

alpha = 0.05

k1 = k - p - 1 = 4

k2 = n - 1 = 5

Крит. значение критерия Фишера: F\_critical = 5.19

Проверим, выполняется ли неравенство для R^2:

R^2 = 0.0029

R\_Fisher = 20.7600

Гипотеза не противоречит эксперментальным данным

степень полинома: q = 1

y = a0 + a1 \* x^1

y = 29.4829 + 0.0408 \* x^1

----------------------------------------------------------------------------------------------------

5. Определение коэффициента преобразования линейнойи

статической характеристики преобразования.

1) Предположим, что p = 1:

G < G\_critical => Случай равноточных измерений

Оценка коэф. a1:

a1 = 7.786513

Среднеквадратическое значение его погрешности s\_a\_1:

s\_a\_1 = 0.030429

Проверка статистической гипотезы:

R^2 = 15.081787

R\_Fisher = 20.7600

Фактическая нелинейность, если она есть, настолько мала, что не может быть

выявлена на фоне погрешностей измерений => Степень полинома: q = 1

Оценка статической характеристики преобразования:

y = 7.7865 \* x

----------------------------------------------------------------------------------------------------

6. Оценка характеристик погрешности

Вычислим выборочные значения погрешности в каждой i-ой точке диапазона измерений:

delta:

0.4260 18.0212 21.0059 35.3105 41.3853 57.3020

-20.2934 -3.1485 -0.5809 14.3632 20.3194 35.4507

-21.8633 -4.0523 -1.8324 12.6204 19.1690 35.0387

-27.7700 -10.7610 -7.3205 6.5276 12.9458 28.7441

-31.1262 -13.0805 -10.0637 4.6932 10.4080 26.0205

-35.9353 -18.7089 -15.3716 -0.9992 4.8349 21.3681

Оценки систематической составляющей погрешности:

delta\_i:

28.9085

7.6850

6.5133

0.3943

-2.1914

-7.4687

Оценки среднеквадратического значения случайной составляющей погрешности:

s\_i:

1.0e-05 \*

0.6281

0.6618

0.6265

0.6402

0.6128

0.6103

Зададим коэффициенты для нахождения толерантного множителя:

P = 0.95

Q = 0.80

n = 5

Толерантный множитель: K(n,P,Q) = 3.44

Вычислим толерантные пределы в каждой i-ой точке:

i = 1: [ 28.90846203 , 28.90850524 ]

i = 2: [ 7.68502647 , 7.68507200 ]

i = 3: [ 6.51331315 , 6.51335625 ]

i = 4: [ 0.39430850 , 0.39435255 ]

i = 5: [ -2.19142885 , -2.19138669 ]

i = 6: [ -7.46867853 , -7.46863654 ]

max[ -7.4687 , 28.9085 ]

Максимальный разброс = 36.3772

Характеристика аддитивной погрешности:

delta\_a = g = max[|min|, |max|] = 28.9085

Зададим коэффициенты для нахождения Коэффициенты Стъюдента:

Q = 0.80

n = 6

Коэффициент Стъюдента: t(n - 1) = 1.53

С вероятностью 0.80 модуль погрешности коэффициента преобразования не превосходит значения:

delta\_K = delta\_a1 = 0.0466

Посчитаем среднеквадратичный критерий p^2:

1) sum\_1 = 0.0005

2) sum\_2 = 0.0360

3) sum\_3 = 0.0143

4) sum\_4 = 0.1056

5) sum\_5 = 0.0041

6) sum\_6 = 0.0285

Ср. кв. критерий: p^2 = 0.1890

----------------------------------------------------------------------------------------------------

>>

# Вывод

Получен полином y = 29.4829 + 0.0408 \* x^1. Ниже показан график , на котором изображены исходные данные и получившийся полином:

Изображение выглядит как линия, График, число, диаграмма

Автоматически созданное описание

Рис. 4 – Полученный полином (красная линия) на фоне исходных значений

Изображение выглядит как линия, График, Параллельный, диаграмма

Автоматически созданное описание

Рис. 5 – Полученный полином (красная линия) на фоне средних

значений (фиолетовые точки на графике)

Для проверки результатов был посчитан среднеквадратичный критерий, чей результат оказался равен: . Это свидетельствует о точности полученного решения