# Практикум №3: погрешности, метод наименьших квадратов

## 1 Погрешности

### 1.1

Найдем общую среднюю совокупности, состоящей из следующих трех групп:

| Группа | I | | II | | III | |
|---|---|---|---|---|---|---|
| Значение признака | 1 | 3 | 2 | 4 | 3 | 6 |
| Частота признака | 11 | 34 | 22 | 28 | 31 | 14 |
| Объем выборки | $11 + 34 = 45$ | | $22 + 28 = 50$ | | $31 + 14 = 45$ | |

Для начала найдем групповые средние: $<x_1>$, $<x_2>$ и $<x_3>$:

```
x1 = (11*1 + 34*3)/45
x2 = (22*2 + 28*4)/50
x3 = (31*3 + 14*6)/45
```

Теперь найдем их среднее:

```
X = (x1*45 + x2*50 + x3*45)/(45 + 50 + 45)
```

Однако, при работе с большими массивами данных лучше использовать преимущества матричной алгебры:

```
xi = [1 3 2 4 3 6];
ni = [11 34 22 28 31 14 ];
N = sum(ni)
X = sum(xi.*ni/N)
```

Найдем *генеральную дисперсию* и генеральное среднеквадратичное отклонение данной выборки:

```
D = sum(ni.*(xi-X).^2)/N
sigma=sqrt(D)
```

Кроме того, определить среднеквадратичное отклонение ряда $x$ можно при помощи команды `std(x)`.

## 1.2

Рассмотрим ряд измерений некоторой физической величины $x$. Результаты серии измерений заданы таблицей ($\nu_i$ – частота соответствующего значения $x_i$):

| $x_i$ | 31 | 28 | 34 | 26 | 35 | 30 | 34 | 32 | 40 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\nu_i$ | 20 | 12 | 10 | 5 | 7 | 20 | 12 | 19 | 4 | 2 |

Известно, что некоторые результаты могут быть заведомо ошибочными. Нам необходимо оценить среднее значение данной величины, исключив ошибочные результаты. Составим массивы величины $x$ и соответствующих частот $n$:

```
x = [ 31 28 34 26 35 30 34 32 40 20];
n = [ 20 12 10  5 7 20 12 19 4 2];
```

Отобразив данные на графике (`plot(x,n,'o')`) можно заметить, что действительно некоторые значения сильно отклоняются от положения, которое они занимали бы при нормальном распределении.

Найдем среднее значение величины $x$ и ее среднеквадратичное отклонение:

```
X = sum(x.*n)/sum(n)
sigma = sqrt(sum(n.*(x-X).^2)/sum(n))
```

Определим границы доверительного интервала $[a, b]$ в пределах трех $\sigma$:

```
a = X-3*sigma
b = X+3*sigma
```

Теперь исключим из выборки значения, выходящие за пределы интервала. При помощи функции `find` можно найти индексы членов массива, удовлетворяющих заданному условию. Исключить лишние элементы можно так:

```
idx = find(x < a);
x(idx) = []; n(idx) = [];
idx = find(x > b);
x(idx) = []; n(idx) = [];
```

Теперь повторим вычисление `X` и `sigma`, `a` и `b`. *Для того, чтобы вызвать из истории команд строку, начинающуюся с определенных символов, наберите один–два первых символа и нажмите клавишу «вверх».* Таким образом можно быстро вызвать из истории команд нужную вам команду, не перебирая все промежуточные.

Теперь проверим, не влияет ли «подозрительное» значение $x = 40$ на точность измерения. Найдем медиану нашего ряда и оценим доверительный интервал по медиане. Для этого нам необходимо построить новый вектор `newx`, в котором значения величины $x$ будут содержаться столько раз, какова их частота:

```
newx = []; for a = [1:length(n)]
newx = [newx ones(1,n(a)).*x(a)];
endfor
med = median(newx)
a = med-3*sigma
b = med+3*sigma
```

Действительно, значение $x = 40$ выбивается из доверительного интервала. Удалим его:

```
idx = find(x==40);
x(idx) = []; n(idx) = [];
```

и найдем $<x>$, близкое к истинному:

```
X = sum(x.*n)/sum(n)
sigma = sqrt(sum(n.*(x-X).^2)/sum(n))
a = X-3*sigma
b = X+3*sigma
find(x>b)
find(x<a)
```

Итак, все оставшиеся значения $x_i$ удовлетворяют критерию «трех сигм», следовательно, можно записать ответ: $x = 31.3 \pm 2.3$.

### 1.3

Теперь определим доверительный интервал величины $<x>$ с надежностью 95% при помощи распределения Стьюдента. Для этого в Octave существует функция `ttest`. В простейшем случае вида `h=ttest(x)` она возвращает вероятность отклонения гипотезы о нормальном распределении величины $x$ с математическим ожиданием $\overline{x} = 0$. Проверка даст результат: 1. Действительно, математическое ожидание нашей величины далеко не равно нулю. Второй аргумент функции `ttest` задает предполагаемое математическое ожидание. Проверим: `h=ttest(x,X)`. Получаем: `h=0`. Т.е., можно принять гипотезу о гауссовой форме распределения величины $x$ около ее среднего значения. Оценить 95%-й доверительный интервал величины $x$ можно при помощи расширенного вывода функции `ttest` в форме `[h,p,ci]=ttest(x,X)`. В этом случае параметр `h` сообщает о степени ненадежности гипотезы, `p` равен вероятности совпадения величины `X` с математическим ожиданием ряда `x`, `ci` сообщает границы 95%-го доверительного интервала. Определим доверительный интервал для нашего ряда без исключения заведомо ложных результатов и с их исключением:

```
x = [ 31 28 34 26 35 30 34 32 40 20];
n = [ 20 12 10  5 7 20 12 19 4 2];
newx = []; for a = [1:length(n)]
newx = [newx ones(1,n(a)).*x(a)];
endfor

[h,p,ci] = ttest(newx, X)
newx = []; for a = 1:8
newx = [newx ones(1,n(a)).*x(a)];
endfor
[h,p,ci] = ttest(newx, X)
```

Итак, в обоих случаях гипотеза о соответствии распределения величины $x$ нормальному распределению принимается, однако, во втором случае вероятность определения математического ожидания $\overline{x}$ выше, и доверительный интервал уже, что явно свидетельствует о большей надежности вычислений.

## 1.4

Octave предоставляет огромный набор инструментальных средств. Однако, при работе с большим количеством однообразных данных приходится много раз повторять одни и те же команды. Эту задачу можно упростить, создав **скрипт** (или m-файл). Скрипт представляет собой описание и реализацию пользовательской функции, которая вызывается из командной строки Octave аналогично любой команде, однако может содержать значительное количество инструкций, облегчающих работу пользователя.

M-файл может содержать любые инструкции. Если он не начинается со слова `function`, выполняется все его содержимое. Удобнее, однако, создать m-файл в виде функции, принимающей в качестве аргументов необходимые переменные и возвращающей определенные величины.

Заголовок файла функции имеет вид

```
%
% Комментарий, отображающийся при введении команды help имя_функции
%
function [возвращаемые величины] = имя_функции(входные, аргументы)
```

Далее следуют операторы, выполняемые в теле функции. Если после команды вы пропустите символ точки с запятой, ее вывод будет отображен на экране.

Итак, создадим m-файл, осуществляющий проверку выборки на корректность при помощи критерия «трех сигм»: `three_s.m`.

```
% three_s.m
% [ X sigma ] = three_s(x, n)
% Производит отбор выборки x с соответствующими частотами n
% при помощи критерия "трех сигм"
% результат: среднее значение X и его среднеквадратичное отклонение, sigma

function [ X sigma ] = three_s(x, n)
newx = []; % вспомогательный массив
Data = [x ; n]; % совмещенный массив данных
X = sum(x.*n)/sum(n); % среднее арифметическое
sigma = sqrt(sum(n.*(x-X).^2)/sum(n)); % среднеквадратичное отклонение
down = X-3*sigma; % нижняя граница доверительного интервала
up = X+3*sigma;   % верхняя граница -//-
a = find(x < down); % a и b - массив координат, выходящих за границы
b = find(x > up);
while (length(a) > 0) || (length(b) > 0) % пока есть неверные значения
Data = Data(:, find(Data(1, find(Data(1,:) >= down)) <= up)); % выбрасываем их
x = Data(1,:);
n = Data(2,:);
X = sum(x.*n)/sum(n);
for a = [1:length(n)]
newx = [newx ones(1,n(a)).*x(a)];
endfor
X = median(newx);
sigma = sqrt(sum(n.*(x-X).^2)/sum(n));
down = X-3*sigma;
```

```
up = X+3*sigma;
a = find(x < down);
b = find(x > up);
endwhile
endfunction
```

При запуске скрипта по умолчанию Octave его ищет в текущей директории. Однако, можно добавить любую директорию со скриптами в список поиска (`path`) при помощи команды `addpath`.

Запустить данный скрипт можно командой `[X sigma] = three_s(x,n)`.

### 1.5

Зачастую физику-экспериментатору приходится проверять нулевую гипотезу о равенстве средних двух независимых наборов данных. Пусть в результате одного измерения некоторой физической величины $x$ был получен ряд данных:

```
x1 = [ 47.78 36.40 35.66 8.93 40.42 54.16 51.76 44.32 46.19 50.75];
```

Затем было произведено независимое измерение этой же физической величины при других условиях эксперимента. При этом был получен ряд:

```
x2 = [ 44.09 46.75 44.20 7.99 47.74 75.07 62.48 44.43 34.73 55.26];
```

Требуется проверить нулевую гипотезу о равенстве математических ожиданий данных величин.

Для проверки данной гипотезы существует функция Octave `ttest2`, `ttest2(x1,x2)` даст ответ: `ans=0`, т.е. гипотеза о неравенстве математических ожиданий наших двух рядов отклонена на 95%-м уровне. Для определения доверительного интервала и вероятности равенства математических ожиданий воспользуемся расширенным выводом команды:

```
[h p ci] = ttest2(x1, x2)
```

Таким образом, вероятность того, что математические ожидания выборок равны, составляет лишь $p = 51\%$, при этом доверительный интервал математического ожидания разности $x_1 - x_2$ достаточно широк: $c_i = [-19.2, 9.9]$, т.е. математические ожидания данных рядов могут разниться на 4.6 со среднеквадратичным отклонением $\sigma = 14.6$.

Большая ширина доверительного интервала говорит о том, что данные в рядах $x_1$ и $x_2$ получены с низкой надежностью. Однако, найдя медианы рядов $x_1$, $x_2$ и совмещенного ряда $(x_1; x_2)$ можно попытаться с достаточно высокой степенью вероятности оценить математическое ожидание величины $x$.

## 2 Метод наименьших квадратов

### 2.1

Случайная погрешность физического измерения имеет природу, аналогичную белому шуму, поэтому для начала рассмотрим простейшие методы очистки одномерных сигналов вида $y = y(t)$ от шумов.

Используем массив из десяти сигналов, зашумленных с одинаковым уровнем SNR:

```
x=[0:0.05:20];
y=sin(x*10).*(0.5+sawtooth(x*pi/5)/2);
for a=[1:10]
    y1(a,:)=awgn(y,1,'measured');
endfor
```

Вид цикла `for` отличается от языков программирования вроде C: цикл поочередно перебирает все значения переменной `a`. Если бы мы заранее инициализировали ее массивом, можно было бы просто написать `for a`. Цикл `for` заканчивается командой `endfor`. Двоеточие в адресации `y(a,:)` означает, что мы выбираем **все** элементы по второй координате (т.е. приравнивание производится к целой строке). Еще одним отличием от языков программирования является динамическое расширение матриц: нет необходимости в начале работы с ней сообщать ее предельный размер.

Итак, мы получили массив `y1`, в строках которого содержатся зашумленные варианты одного и того же сигнала. Можно отобразить их все графически командой `plot(x,y1)`, а можно и с оригиналом: `plot(x,y,"linewidth",2, x, y1)`. Оценить зашумленность сигнала можно командой `plot(y,y1,'.')`. Если бы сигналы в `y1` совпадали с `y`, мы увидели бы отрезок с коэффициентом наклона 1. Чем дальше форма полученной фигуры от такого отрезка, тем больше зашумленность сигнала.

Для восстановления сигнала из десяти измерений попробуем усреднить наборы сигналов и найти их медиану:

```
y_mean = mean(y1);
y_med = median(y1);
plot(x,[y;y_mean;y_med]);
legend("original", "mean", "median");
```

Оба восстановленных сигнала имеют примерно одинаковые величины и довольно близки к реальной функции (особенно на участках с большой амплитудой сигнала). Однако, как мы увидим впоследствии, если к сигналу добавлен шум типа «соль/перец», медианная фильтрация будет работать намного эффективнее фильтрации по среднему арифметическому.

## 2.2

Рассмотрим линейную зависимость $y = ax + b$, заданную таблично в виде $y = y(x)$. Для определения методом наименьших квадратов коэффициентов линейной (а также высших степеней) зависимости служит функция `polyfit(x,y,n)`. Она содержит три аргумента: $x$ – вектор аргумента, $y$ – вектор функции, $n$ – степень аппроксимирующего полинома. Ее результат в простейшем случае представляет собой вектор коэффициентов (начиная со старшей степени). Если функцию вызвать как `[p,S] = polyfit(x,y,n)`, вектор $p$ будет содержать коэффициенты, а в структуре $S$ будут содержаться такие данные, как степени свободы (df) и норма отклонений данных от аппроксимирующей кривой (normr). Для восстановления полученной зависимости используется функция `polyval(p,x)`, где $p$ – полученный функцией `polyfit` вектор коэффициентов, $x$ – вектор аргумента. В таком виде функция возвращает вектор восстановленной функции. В виде `[y, delta] = polyval(p,x,S)` функция возвращает массив погрешностей (т.е. в каждой точке восстановленные значения функции можно представить в виде $y = y \pm delta$, т.е. оценить абсолютную погрешность восстановления можно при помощи команды `mean(delta)`.

Найдем коэффициенты модельной зависимости. Пусть $y = 7.15x + 4.22$. Построим векторы, соответствующие аргументу и функции:

```
x = [0:100]; y = 7.15*x + 4.22;
```

Зашумим сигнал для получения разброса точек $y_i$:

```
y1 = awgn(y,10,'measured');
```

Отобразим на экране оба ряда: `plot(x,y,x,y1,'o')` (запись `'o'` означает, что график будет отображаться кружка́ми). Разброс данных достаточно велик. Определим коэффициент корреляции: `corr(x,y1)`. Он довольно близок к единице, следовательно, мы можем попытаться получить коэффициенты линейной зависимости и восстановить функцию:

```
[p,S] = polyfit(x,y1,1);        % коэффициенты a и b
[y2, delta] = polyval(p,x,S);   % восстановленный вектор
plot(x,y1,'o',x,[y;y2])         % все три графика
legend("noicy", "original", "fitted");
mean(delta)                     % абсолютная ошибка
mean(delta)/mean(y)             % относительная ошибка
```

## 2.3

Можно найти приближение методом наименьших квадратов и другим способом. Пусть $Y$ – вектор–столбец значений функции, $A = (a, b)^{\mathrm{Tr}}$ – вектор–столбец коэффициентов разложения. Тогда условие $y_i = ax_i + b$ можно представить в виде матричного произведения $Y = XA$. Второй столбец матрицы $X$ целиком состоит из единиц, а в первом находится последовательность значений $x_i$. В этом случае нахождение коэффициентов сводится к решению системы линейных уравнений $y_i = ax_i + b$, дающему минимальную невязку. Такое решение находится при помощи операции левостороннего матричного деления: $X\backslash Y$. Решим предыдущий пример таким способом.

```
X = [x' ones(size(x'))]; % создаем матрицу аргумента
% (т.к. x и y1 - строки, транспонируем их)
A = X\y1'                % находим коэффициенты
% и отображаем их на экране
```

Полученные значения должны быть примерно равны найденным предыдущим способом. Как мы увидим далее, такой способ нахождения корней аппроксимации пригоден не только для полиномиальных, но и для многих других функций.

## 2.4

Попробуем создать квадратичную зависимость и аппроксимировать ее методом наименьших квадратов. Пусть зависимость на отрезке $[0, 100]$ имеет вид $y = 2.4x^2 - 0.87x + 2.13$. Создадим соответствующие массивы данных, добавим шум с SNR=20 дБ и отобразим оба сигнала на графике:

```
x = [1:100];
y = 2.4*x.^2-0.87*x+2.13;
y1 = awgn(y,20,'measured');
plot(x,[y;y1]);
```

Теперь создадим вектор коэффициентов аппроксимации полиномом второй степени восстановим функцию и отобразим на графике:

```
[p, S] = polyfit(x, y1, 2);
[y2, DELTA] = polyval(p, x, S);
plot(x,[y;y2]);
legend("original", "restored")
```

Отобразим на экране найденные коэффициенты: `p`. Рассчитаем среднее квадратичное отклонение аппроксимации (`mean(DELTA)`). Также рассчитаем относительную ошибку аппроксимации `mean(DELTA)/mean(y1)`.

И второй способ:

```
X = [(x.^2)' x' ones(size(x'))];
A = X\y1'
```

## 2.5

Однако, чаще всего функциональные зависимости имеют иные виды зависимости. Допустим, нам известно, что измеряемая величина изменяется по закону

$$y = a_0 + a_1\,\mathrm{e}^{-t} + a_2 t e^{-t}. \tag{2.1}$$

Для аппроксимации такой функцией можно представить уравнение (2.1) в матричном виде $Y = TA$, где $T$ – функциональная матрица, у которой в первом столбце размещены единицы (соответствует умножению на $a_0$), во втором — функция $\mathrm{e}^{-t}$, а в третьем — $t\,\mathrm{e}^{-t}$. Найти коэффициенты $A$ можно при помощи оператора левого деления: $A = T\backslash Y$.

```
t = [0 0.3 0.8 1.1 1.6 2.3]';      % сразу вводим данные в столбцах
y = [0.6 0.67 1.01 1.35 1.47 1.25]';
T = [ones(size(t))  exp(-t)  t.*exp(-t)];
A = T\y
```

Теперь отобразим данные на графике:

```
x = [0:0.1:2.5]';
Y = [ones(size(x))  exp(-x)  x.*exp(-x)]*A;
plot(x,Y, t,y,'o')
```

## 2.6

Для коррекции наведения и сопровождения телескопов используется СКН — система коррекции наведения, которая учитывает различного рода ошибки (гнутие осей и неперпендикулярность осей и т.п.). У БТА данные ошибки выражаются полиномами:

$$dA = K_0 + K_1\frac{1}{\operatorname{tg} Z} + K_2\frac{1}{\sin Z} - K_3\frac{\sin A}{\operatorname{tg} Z} + K_4\frac{\cos\delta\cos P}{\sin Z},$$

$$dZ = K_5 + K_6 \sin Z + K_7 \cos Z + K_3 \cos A + K_4 \cos\varphi \sin A.$$

Здесь:

$dA$, $dZ$ погрешности наведения по азимуту и зенитному расстоянию;

$K_x$ коэффициенты ошибок;

$\varphi$ широта места наблюдения;

$t$ часовой угол;

$P$ параллактический угол.

Для получения этих коэффициентов необходимо провести наблюдение в нескольких десятках равномерно распределенных по небесной полусфере точек (за исключением областей запрета, $Z < 5°$, и около горизонта, $Z > 70°$). Далее в каждом поле вычисляется астрометрия и определяется погрешность наведения. Составляется таблица (например, `2015_09_30_pf.tab`, по которой и необходимо вычислить коэффициенты. Вычислять коэффициенты будем следующим скриптом:

```
function SKN = getSKNcoeff(tabname, imprefix)
%
% SKN = getSKNcoeff(tabname)
%
% Calculate SKN coefficients & plot graphs
%
% parameters:
% tabname - filename with table of dA/dZ
%
% SKN:
% dA = K0 + K1/tg(Z) + K2/sin(Z) - K3*sin(A)/tg(Z) + K4 *cos(delta)*cos(P)/sin(Z)
% dZ = K5 + K6*siz(Z) + K7*cos(Z) + K3*cos(A) + K4*cos(phi)*sin(A)
%
% K0 = A0 - azimuth zero; K1 = L - horiz axe inclination; K2 = k - collimation error;
% K3 = F - lattitude error of vert. axe; K4 = dS - time error
% K5 = Z0 - zenith zero; K6 = d - tube bend; K7 = d1 - cos. tube bend
%
% phi = 43.6535278 - lattitude
% t = LST - Alpha  - hour angle
% P=atan(sin(t)/(tan(phi)*cos(Del)-sin(Del)*cos(t)))  - parallax angle
%
if(nargin == 1) imprefix = ""; endif
[Ald Alm Als Deld Delm Dels dAl_S dDel_S dA dZ A Z STh STm STs ] = ...
textread(tabname, "|%f:%f:%f %f:%f:%f|%f %f|%f %f|%f %f|%f:%f:%f|", ...
60, "headerlines", 8);
A = A*pi/180; % all angles here will be in radians
Z = Z*pi/180;
Al = pi*(Ald+Alm/60+Als/3600)/180; % right accession
Delsig = Deld./abs(Deld);   % declination sign
```

```
Del = pi*Delsig.*(abs(Deld)+Delm/60+Dels/3600)/180; % declination
phi = 43.6535278 * pi / 180; % lattitude
t = pi*(STh+STm/60+STs/3600)/12 - Al; % hour angle
P = atan(sin(t)./(tan(phi).*cos(Del)-sin(Del).*cos(t))); % parallax angle
cont = 1;
while cont
printf("\n\n\t\t\t\tIteration %d\n\n", cont);
onescol = ones(size(dA)); % column with ones - for less square method
cosZ = cos(Z);
sinZ = sin(Z);
cosA = cos(A);
sinA = sin(A);
tgZ  = tan(Z);
Xmatr = [onescol sinZ cosZ cosA cos(phi).*sinA];
K = Xmatr \ dZ;
K5 = K(1); K6 = K(2); K7 = K(3); K3 = K(4); K4 = K(5);
dZSKN = K5 + K6*sinZ + K7*cosZ + K3*cosA + K4*cos(phi)*sinA; % dZ by SKN
K4fr = cos(Del).*cos(P)./sinZ; % K4 multiplier
dASKN34 = dA + K3*sinA./tgZ - K4*K4fr; % dA components fixed by K3 & K4
Xmatr = [onescol 1./tgZ 1./sinZ];
K = Xmatr \ dASKN34;
K0 = K(1); K1 = K(2); K2 = K(3);
SKN = [K0, K1, K2, K3, K4, K5, K6, K7];
dASKN = K0 + K1./tgZ + K2./sinZ - K3*sinA./tgZ + K4*K4fr;
ddA = dA - dASKN;
ddZ = dZ - dZSKN;
sddA = std(ddA); sddZ = std(ddZ);
mddA = median(ddA); mddZ = median(ddZ);
printf("sigma(dda) = %f, sigma(ddZ) = %f\n", sddA, sddZ);
%printf("mean(dda) = %f, mean(ddZ) = %f\n", mean(ddA), mean(ddZ));
printf("median(dda) = %f, median(ddZ) = %f\n", mddA, mddZ);
surge = find(abs(ddA - mddA) > 2*sddA);
ssz = size(surge,1);
if(ssz != 0)
printf("Surges: \n")
for i = 1:ssz
idx = surge(i);
printf("%f (Z = %f, A = %f)\n", ddA(idx), Z(idx)*180/pi, A(idx)*180/pi);
endfor
Z(surge) = []; A(surge) = []; Al(surge) = []; Del(surge) = [];
t(surge) = []; P(surge) = []; dZ(surge) = []; dA(surge) = [];
++cont;
else
cont = 0;
endif
endwhile
printf("SKN coefficients: K0..K7: %f, %f, %f, %f, %f, %f, %f, %f\n", ...
```

```
K0, K1, K2, K3, K4, K5, K6, K7);
fg = figure;
plot(A*180/pi, [ddA ddZ], 'o');
legend("ddA", "ddZ");
xlabel("A, degr"); ylabel("Remaining error: real-model");
plotgr(sprintf("%s_%s", imprefix, "diff_vs_A"), fg);
fg = figure;
plot(Z*180/pi, [ddA ddZ], 'o');
legend("ddA", "ddZ");
xlabel("Z, degr"); ylabel("Remaining error: real-model");
plotgr(sprintf("%s_%s", imprefix, "diff_vs_Z"), fg);
endfunction

function plotgr(nm, fg)
print(fg, '-dpdf', sprintf("%s.pdf", nm));
print(fg, '-dpng', sprintf("%s.png", nm));
endfunction
```

Запускаем: `SKN = getSKNcoeff('2015_09_30_pf.tab')`. Строятся графики остаточных невязок и выводятся значения всех коэффициентов.

Из-за линейной зависимости коэффициентов задача их вычисления является некорректной, а ввиду малости объема экспериментального материала, решать задачу будем итерациями, на каждом шаге избавляясь от выбросов.

# 3 Задания для самостоятельного выполнения

1. Некоторая совокупность состоит из трех групп: $X_1$, $X_2$, и $X_3$. Группы имеют следующие значения:

   ```
   X1 =  35.04 35.45 35.01 34.94 34.63 35.11 34.41 35.29 35.69
   34.69 35.36 35.53 34.30 34.36 35.23
   X2 =  34.30 34.80 34.86 34.81 35.08 34.79 35.04 33.93 34.48
   34.41 33.74 34.60 34.00
   X3 =  35.17 34.21 34.78 34.65 34.16 33.62 34.53 34.12 34.82
   34.77 35.29 34.81 34.28 34.72 34.12 34.55 34.53 34.55
   ```

   Найдите: групповые средние (35,00, 34.53, 34.54), общее среднее (34.69), групповые дисперсии (0.19, 0.19, 0.16), генеральную дисперсию (0.22).
2. Усовершенствуйте скрипт `three_s.m` так, чтобы помимо основных вычислений на экране отображались среднее арифметическое значение массива с данными, а также 95%-й доверительный интервал по критерию Стьюдента.
3. Определите давление в цилиндре с газом, исходя из закона Менделеева–Клапейрона: $pV = mRT/\mu$, если известно, что масса газа $m = 2$ грамма, $\mu = 29$ г/моль, $R = 8.31$, а объем и температуру газа измеряли в течение минуты, получив следующие значения:

| Величина | Значение | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $V$, л | 2.27 | 2.27 | 2.26 | 2.25 | 2.26 | 2.27 | 2.29 | 2.28 | 2.25 | 2.28 |
| $T$, K | 399.4 | 399.1 | 399.3 | 396.8 | 399.5 | 400.2 | 400.6 | 403.0 | 399.2 | 401.3 |

Считайте, что за это время давление газа не успело сколь-нибудь значительно измениться. Определите погрешности измерения величин $V$ и $T$. Считая, что остальные величины являются постоянными, определите косвенную погрешность измерения $p$. Для удобства вычислений *создайте скрипт, позволяющий для заданного ряда данных получить математическое ожидание, среднеквадратичное отклонение и относительную ошибку.*
Запишите результат в виде $p = \overline{p} \pm \sigma_p$ ($p = 101 \pm 1$ кПа).

4. Для определения емкости $C$ неизвестного конденсатора при помощи осциллографа исследовали затухающий импульс, возникающий при разрядке конденсатора через резистор $R = 3$ кОм. По показаниям осциллографа были записаны следующие значения тока:

| $t$, с | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $I$, А | 1.00 | 0.72 | 0.52 | 0.37 | 0.26 | 0.19 | 0.14 | 0.10 | 0.07 | 0.04 | 0.03 |

Известно, что погрешность амперметра составляет $\sigma_I = 0.01$ А. Кроме того, известно что сопротивление резистора известно с точностью 5%. Из формулы $I = I_0 \exp(-t/[RC])$ определите погрешность измерения емкости конденсатора.
Методом наименьших квадратов определите значение емкости конденсатора, исходя из уравнения $t = -RC \ln I$ (97 мкФ). Запишите ответ в виде $C = <C> \pm \sigma_C$.
Для увеличения точности эксперимента было проведено еще одно измерение, результаты которого несколько отличались от предыдущих:

| $t$, с | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $I$, А | 1.00 | 0.75 | 0.56 | 0.41 | 0.30 | 0.23 | 0.17 | 0.12 | 0.10 | 0.07 | 0.05 |

Проверьте нулевую гипотезу о равенстве средних в обоих опытах. Определите величину емкости во втором случае (112 мкФ).
Столь большое различие емкостей, полученных в результате двух независимых экспериментов, заставило предположить, что в результате длительной эксплуатации резистор $R$ нагрелся, что вызвало увеличение его сопротивления. Считая емкость конденсатора прежней, определите сопротивление резистора во втором случае (3.5 кОм).

5. Найдите *обоими способами* коэффициенты $a$ и $b$ для таблично представленной зависимости $y(x)$, предполагая, что она имеет линейный вид. Найдите коэффициент корреляции $x$ и $y$, Данные представлены в таблице:

| **x** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **y** | 7.7 | 13.7 | 22.0 | 23.1 | 23.7 | 36.7 | 35.6 | 47.8 | 50.2 | 52.1 |

($a = 5.1$, $b = 3.4$).

6. Известно, что некоторая зависимость (см. таблицу ниже) имеет вид $y = ax\sin(x) - b\ln(x)$. Определите коэффициенты $a$ и $b$ и постройте данную кривую с более детальным отображением (на векторе `[1:0.05:10]`). Подсказка: сразу же задайте вектора $x$ и $y$ как столбцы; матрица $X$ задается командой `X=[x.*sin(x)  -log(x)]`.

| **x** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **y** | -0.68 | 8.41 | -23.0 | -37.2 | -73.2 | -39.7 | 9.14 | 21.0 | 7.97 | -72.5 |

($a = 7.72$, $b = 14.8$).

7. Составьте модель эксперимента по измерению амплитуды напряжения в контуре, испытывающем колебания с основной частотой $\Omega = 1000$ Гц и двумя гармониками $\Omega \pm \omega$, где $\omega = 74$ Гц. Известно, что суммарное колебание описывается приближенной форму-

лой $U = a\sin(\Omega t) + b\sin(\omega t) - c\cos(\omega t)$. Создайте интервал времен `t=[0: 0.06: 120]`. Для получения идеальных значений $U$ положите $a = 361$, $b = 117$, $c = 92$. Отношение сигнал/шум при получении зашумленного сигнала выберите равным 20 дБ.
Восстановите значения коэффициентов $a$, $b$ и $c$.