Семинар 13. Сингулярное разложение матрицы эксперимента и спектральное разложении матрицы ковариции

Table of Contents

Немного теорвера и матстата

У нас есть случаная переменная, которая может принимать некоторый (дискретный) набор возможных значений:

$$x \sim x_1 \dots x_N$$

Вероятность переменной иметь некоторое определенноей значение из этого дискретного набора характеризуется набором вероятностей:

$$P \sim p_1 \dots p_N,\ \Sigma_i p_i = 1$$

Для некоторой случайно переменной, которая получается из некоторого распределения P, математическое ожидание будет:

$$m = \mathbb{E}(x \sim P) = \Sigma_i p_i x_i$$

Среднее значение результатов набора из M испытаний:

$\mu = \frac{\Sigma_{j=1}^{M} x_j}{M}$ - применяется для экспериментальной оценки мат. ожидания.

Разброс данных характеризуется вариацией:

$$\nu = \mathbb{E}[(x-m)^2] = \Sigma_i p_i (x_i - m)^2$$

Для экспериментальной оценки вариации:

$$\sigma = \frac{\Sigma_{j=1}^{M}(x_j - \mu)^2}{M-1}$$

Если случаная величина характеризуется векторов, то есть состояний несколько, например, $\begin{bmatrix} x \\ y \end{bmatrix}$, каждая из компонент идет из своего распределения, то :

$$\sigma_x = \frac{\Sigma_{j=1}^{M}(x_j - \mu_x)^2}{M-1}, \sigma_y = \frac{\Sigma_{j=1}^{M}(y_j - \mu_y)^2}{M-1}$$

Если есть две переменных, то появляется ковариация, которая характеризуется совместной вероятностью события с вектором состояний $\begin{bmatrix} x_i \\ y_j \end{bmatrix}$.

$$\nu = \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])] = \Sigma_{ik} p_{ik}(x_i - m_x)(y_k - m_y)$$

p_{ik} - совместная вероятность

Матрица ковариации:

$\vec{X_1}...\vec{X_m}...\vec{X_M}$ - вектора результатов испытания (определения N свойств) "образца" размером $N \times 1$ каждый (вектора состояния некоторого случайного процесса).

M - число испытаний, N - число свойств ($N < M$)

$\vec{\mu} = \mu_1...\mu_n...\mu_N$ - средние значения по всем испытаниям для каждого из свойств: $\mu_n = \frac{\sum_{i=1}^{M} x_{ni}}{M}$

```
clearvars
x1 = sym("x1",[2 1])
```

```
x1 =
```

$$\begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix}$$

```
x2 = sym("x2",[2 1])
```

```
x2 =
```

$$\begin{pmatrix} x_{21} \\ x_{22} \end{pmatrix}$$

```
x3 = sym("x3",[2 1])
```

```
x3 =
```

$$\begin{pmatrix} x_{31} \\ x_{32} \end{pmatrix}$$

```
mu = sym("mu",[2,1])
```

```
mu =
```

$$\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}$$

```
% три измерения двух свойств N = 2, M = 3
```

Матрица матрица испытаний $[N \times M]$:

$A = [\vec{X_1} - \vec{\mu}, ..., \vec{X_m} - \vec{\mu}, ..., \vec{X_N} - \vec{\mu}]$ - каждый столбец - вектор результатов измерений (вектор состояния), смещенный на среднее значение ($[...]$ - операция конкатенации)

```
A = [x1-mu,x2-mu,x3-mu]
```

```
A =
```

$$\begin{pmatrix}x_{11}-\mu_1 & x_{21}-\mu_1 & x_{31}-\mu_1\\ x_{12}-\mu_2 & x_{22}-\mu_2 & x_{32}-\mu_2\end{pmatrix}$$

Экспериментальная оценка матрицы ковариации (симметричная матрица размером $N \times N$):

$$C = \frac{AA^T}{(M-1)} \qquad (3)$$

```
C = A*transpose(A)*sym("1/2")
```

```
C =
```

$$\begin{pmatrix}\frac{(\mu_1-x_{11})^2}{2}+\frac{(\mu_1-x_{21})^2}{2}+\frac{(\mu_1-x_{31})^2}{2} & \sigma_1\\ \sigma_1 & \frac{(\mu_2-x_{12})^2}{2}+\frac{(\mu_2-x_{22})^2}{2}+\frac{(\mu_2-x_{32})^2}{2}\end{pmatrix}$$

where

$$\sigma_1 = \frac{(\mu_1-x_{11})\ (\mu_2-x_{12})}{2}+\frac{(\mu_1-x_{21})\ (\mu_2-x_{22})}{2}+\frac{(\mu_1-x_{31})\ (\mu_2-x_{32})}{2}$$

Диагональные элементы матрицы C_{nn}- коэффициенты вариации ($n = 1...N$), для n - го свойства :

$$C_{nn} = \frac{\sum_{i=1}^{M}(x_{ni}-\mu_n)^2}{M-1}$$

Если из диагональных элементов извлечь корень и поделить на количество экспериментов L, то получим стандартное отклонение среднего арифметического.

Элементы матрицы C, стоящие вне диагонали, - коэффициенты ковариации n - го и m - го свойств:

$$C_{nk} = \frac{\sum_{i=1}^{M}[(x_{ni} - \mu_n)(x_{ki} - \mu_k)]}{M-1} \ (n,k = 1...N, n \neq k)$$

Эти формулы понятнее, если посмотреть не на матрицу A, на транспонированную матрицу $B = A^T$,

в матрице B, каждый столбец ($\vec{b}_n$), размером $[M \times 1]$ будет соответствовать какому-то свойству,

измеренному M раз. Тогда вектор средних $\vec{\mu}$ - это вектор средних значений по каждому из столбцов.

То есть, хотя табличка данных одна и та же в зависимости от того как на нее смотреть ее интерпретация различна:

матрица A - это набор испытаний, в каждом из которых был определен некоторый набор свойств,

матрица $B = A^T$ - это набор свойств, для каждого из которых было выполнено некоторое число испытаний.

```
B = transpose(A)
```

B =

$$\begin{pmatrix} x_{11} - \mu_1 & x_{12} - \mu_2 \\ x_{21} - \mu_1 & x_{22} - \mu_2 \\ x_{31} - \mu_1 & x_{32} - \mu_2 \end{pmatrix}$$

```
b1 = B(:,1)
```

b1 =

$$\begin{pmatrix} x_{11} - \mu_1 \\ x_{21} - \mu_1 \\ x_{31} - \mu_1 \end{pmatrix}$$

```
b2 = B(:,2)
```

b2 =

$$\begin{pmatrix} x_{12} - \mu_2 \\ x_{22} - \mu_2 \\ x_{32} - \mu_2 \end{pmatrix}$$

В соответствии с (3), матрица ковариации через матрицу свойств B выражается в виде:

$$C = AA^T/(M-1) = B^TB/(M-1)$$

(4)

То есть, диагональные элементы матрицы ковариации (вариация) $C_{nn} = \frac{\vec{b}_n^T \vec{b}_n}{M-1}$ - это просто скалярное произведение столбца матрицы B самого на себя, а элементы, стоящие вне диагонали - это скалярные

произведения различных столбцов матрицы B: $C_{nk} = \frac{\vec{b}_n^T \vec{b}_k}{M-1}$ скалярное произведение двух векторов характеризует то насколько один вектор "отстоит" от другого, по сути это проекция одного вектора на другой. Если оба вектора единичные по амплитуде, то это косинус угла между ними. Если скалярное произведение равно нулю, то вектора сонаправлены, то есть максимально "зависимы".

```
C_b = transpose(B)*B
```

C_b =

$$\begin{pmatrix} (\mu_1 - x_{11})^2 + (\mu_1 - x_{21})^2 + (\mu_1 - x_{31})^2 & \sigma_1 \\ \sigma_1 & (\mu_2 - x_{12})^2 + (\mu_2 - x_{22})^2 + (\mu_2 - x_{32})^2 \end{pmatrix}$$

where

$$\sigma_1 = (\mu_1 - x_{11})\ (\mu_2 - x_{12}) + (\mu_1 - x_{21})\ (\mu_2 - x_{22}) + (\mu_1 - x_{31})\ (\mu_2 - x_{32})$$

Спектральное разложение матрицы ковариации <=> сингулярное разложение матрицы измерений:

Пусть матрица измерений A имеет следующее сингулярное разложение:

$A = U\Sigma V^T$

Тогда матрица ковариации может быть представлена в виде:

$C = AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V\Sigma U^T) = U\Sigma^2 U^T$
(5)

$C = U\Sigma^2 U^T = U\Sigma^2 U^{-1}$

$C = P\Lambda P^{-1}$ - спектральное разложение матрицы ковариации

Столбцы матрицы собственных векторов не ортонормированы, однако, для симметричной положительно определенной матрицы C, которой является матрица ковариации, они ортогональны, то есть: $P^{-1} = P^T$, таким образом, левые сингулярные вектора матрицы испытаний есть нормированные собственные вектора матрицы ковариации, а собственные значения - квадраты сингулярных значений.

Пример корреляции при бросании двух монеток.

Испытание - однократное бросание двух монеток (вектор состояния системы из двух монеток -1/2 - решка, +1/2 - орел).

Свойство - состояние одной монетки.

Если монетки некоррелированы, то в каждом испытании состояние каждой из монет не зависит от состояния другой.

Монетки максимально коррелированы, когда одна приклеена к другой.:

```
clearvars
points_number = 500; % число бросаний
% монетки независимы друг от друга
first_coin = (rand(points_number,1)>=0.5) - 0.5; % +-1/2 орел/решка, среднее - ноль
second_coin = (rand(points_number,1)>=0.5) -0.5;
disp("Матрица испытаний для независимых монет: ")
```

```
Матрица испытаний для независимых монет:
```

```
A = transpose([first_coin-mean(first_coin),second_coin-mean(second_coin)]) %
матрица испытаний
```

```
A = 2×500
   -0.5240   -0.5240    0.4760    0.4760    0.4760    0.4760   -0.5240    0.4760 ···
    0.5180   -0.4820    0.5180   -0.4820   -0.4820   -0.4820   -0.4820   -0.4820
```

```
disp("Матрица свойств: ")
```

```
Матрица свойств:
```

```
B = A' % матрица свойтсв
```

```
B = 500×2
   -0.5240    0.5180
   -0.5240   -0.4820
    0.4760    0.5180
    0.4760   -0.4820
    0.4760   -0.4820
    0.4760   -0.4820
   -0.5240   -0.4820
    0.4760   -0.4820
   -0.5240   -0.4820
    0.4760   -0.4820
      ⋮
```

```
ncorrelated_coins_covariance_matrix =
transpose([first_coin,second_coin])*[first_coin,second_coin]/(points_number-1);
disp("Монетки бросаются независимо друг от друга:")
```

```
Монетки бросаются независимо друг от друга:
```

```
disp(ncorrelated_coins_covariance_matrix)
```

```
    0.2505    0.0030
    0.0030    0.2505
```

```
disp("Матрица ковариации близка к диагональной - максимально несингулярна, колонки
полностью независимы")
```

```
Матрица ковариации близка к диагональной - максимально несингулярна, колонки полностью независимы
```

```
% монетки приклеены друг к другу одноименными концами
first_coin_dependent = (rand(points_number,1)>=0.5) - 0.5;
first_coin_dependent = first_coin_dependent-mean(first_coin_dependent);
disp("Матрица испытаний для склееных монет: ")
```

Матрица испытаний для склееных монет:

```
Acor=transpose([first_coin_dependent,first_coin_dependent]) % матрица испытаний
```

```
Acor = 2×500
    0.4940   -0.5060    0.4940   -0.5060    0.4940    0.4940    0.4940   -0.5060 ···
    0.4940   -0.5060    0.4940   -0.5060    0.4940    0.4940    0.4940   -0.5060
```

```
correlated_coins_covariance_matrix =
transpose([first_coin_dependent,first_coin_dependent])*[first_coin_dependent,first_c
oin_dependent]/(points_number-1);
disp("Монетки склеены одноименными концами, матрица ковариации:")
```

Монетки склеены одноименными концами, матрица ковариации:

```
disp(correlated_coins_covariance_matrix)
```

```
    0.2505    0.2505
    0.2505    0.2505
```

```
disp("Матрица ковариации сингулярна, так как колонки одинаковы, то есть линейно
зависимы")
```

Матрица ковариации сингулярна, так как колонки одинаковы, то есть линейно зависимы

```
disp("Сингулярное разложение матрицы ковариации для независимых монеток")
```

Сингулярное разложение матрицы ковариации для независимых монеток

```
[U,S] = svd(A)
```

```
U = 2×2
    0.6940   -0.7200
    0.7200    0.6940
S = 2×500
   11.2469         0         0         0         0         0         0         0 ···
         0   11.0931         0         0         0         0         0         0
```

```
disp("Спектральное разложение матрицы ковариации для независимых монеток")
```

Спектральное разложение матрицы ковариации для независимых монеток

```
[Q,L] = eig(A*A')
```

```
Q = 2×2
   -0.7200    0.6940
    0.6940    0.7200
L = 2×2
  123.0578         0
         0  126.4922
```

```
disp("Сингулярное разложение матрицы ковариации для склееных монеток")
```

Сингулярное разложение матрицы ковариации для склееных монеток

```
[U,S] = svd(Acor)
```

```
U = 2×2
   -0.7071    0.7071
   -0.7071   -0.7071
S = 2×500
   15.8102         0         0         0         0         0         0         0 ···
         0    0.0000         0         0         0         0         0         0
```

```
disp("Спектральное разложение матрицы ковариации для склееных монеток")
```

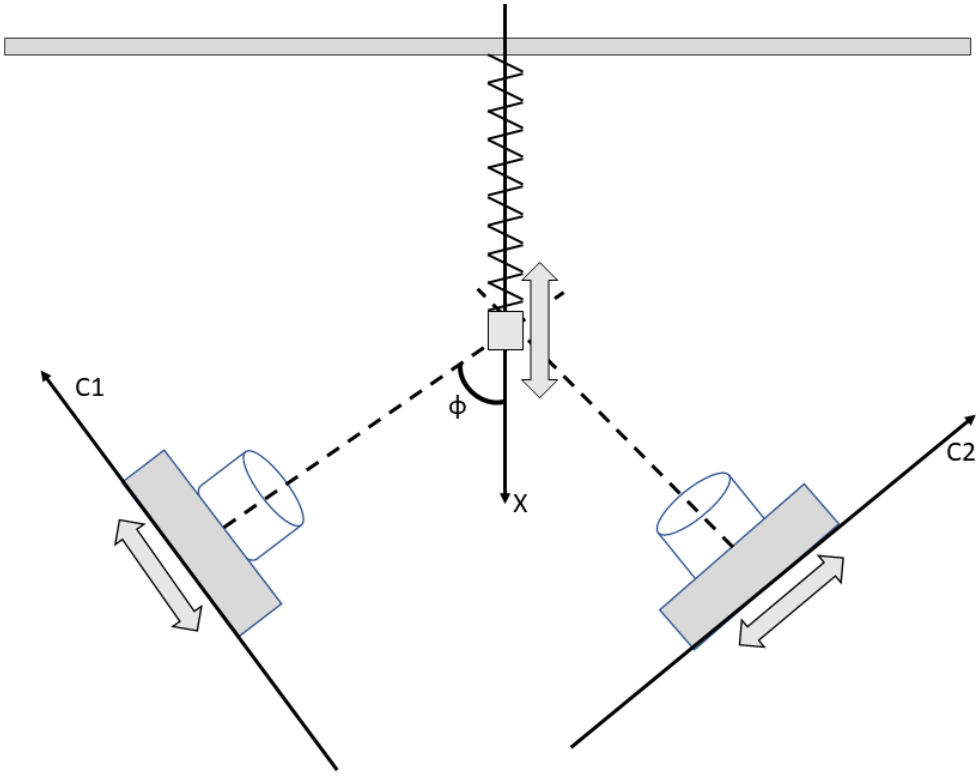
```
Спектральное разложение матрицы ковариации для склееных монеток
```

```
[Q,L] = eig(Acor*Acor')
```

```
Q = 2×2
   -0.7071    0.7071
    0.7071    0.7071
L = 2×2
         0         0
         0  249.9640
```

Примеры использования

Пример использования - детерминированные данные: колебание грузика на пружине и два датчика




```
clearvars
t = linspace(-pi,pi,100) % фаза движения
```

```
t = 1×100
```

```
   -3.1416   -3.0781   -3.0147   -2.9512   -2.8877   -2.8243   -2.7608   -2.6973 ···
```

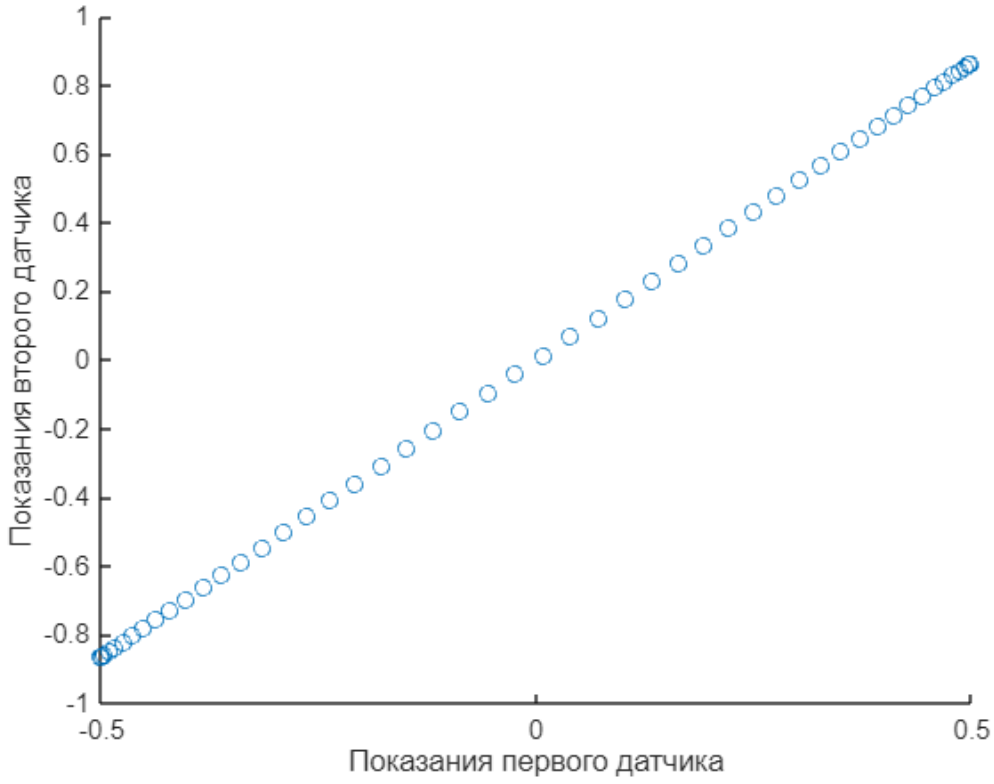
```
X = cos(t) % закон движения грузика на пружине
```

```
X = 1×100
   -1.0000   -0.9980   -0.9920   -0.9819   -0.9679   -0.9501   -0.9284   -0.9029 ···
```

```
phi = 30; % угол ориентации камер относительно оси колебания грузика
c1 = X*cosd(phi); % показания камеры 1
c2=X*sind(phi); % показания камеры 2
A=[c1;c2] % матрица эксперимента
```

```
A = 2×100
   -0.8660   -0.8643   -0.8591   -0.8504   -0.8383   -0.8228   -0.8040   -0.7820 ···
   -0.5000   -0.4990   -0.4960   -0.4910   -0.4840   -0.4750   -0.4642   -0.4515
```

```
scatter(c2,c1);
xlabel("Показания первого датчика")
ylabel("Показания второго датчика")
```




```
[V,S,U] = svd(A) % считаем сингулярное разложение
```

```
V = 2×2
    0.8660    0.5000
    0.5000   -0.8660
S = 2×100
    7.1063         0         0         0         0         0         0         0 ···
         0    0.0000         0         0         0         0         0         0
U = 100×100
   -0.1407    0.9839    0.0059    0.0197   -0.0036   -0.0173   -0.0002   -0.0182 ···
```

```
   -0.1404   -0.0174   -0.1466   -0.1575   -0.1347   -0.1200   -0.1321   -0.1124
   -0.1396   -0.0269    0.9817   -0.0183   -0.0178   -0.0174   -0.0171   -0.0165
   -0.1382   -0.0407   -0.0200    0.9799   -0.0193   -0.0186   -0.0186   -0.0177
   -0.1362   -0.0168   -0.0167   -0.0165    0.9837   -0.0160   -0.0156   -0.0152
   -0.1337   -0.0025   -0.0146   -0.0142   -0.0143    0.9858   -0.0137   -0.0135
   -0.1306   -0.0194   -0.0164   -0.0163   -0.0160   -0.0157    0.9847   -0.0149
   -0.1271   -0.0006   -0.0136   -0.0133   -0.0134   -0.0133   -0.0128    0.9873
   -0.1230   -0.0243   -0.0162   -0.0162   -0.0158   -0.0154   -0.0151   -0.0146
   -0.1184   -0.0043   -0.0132   -0.0129   -0.0129   -0.0128   -0.0124   -0.0122
     ⋮
```

```
% спектр сингулярных значений состоит из одного значения, это значит, что
% нам достаточно одного датчика!
cosd(phi) % косинус угла ориентации датчиков
```

```
ans = 0.8660
```

```
sind(phi) % синус угла ориентации датчиков
```

```
ans = 0.5000
```

```
V2 = [cosd(phi) sind(phi);...
     sind(phi) -cosd(phi)] % матрица вращения на угол фи в двумерном пространстве
```

```
V2 = 2×2
    0.8660    0.5000
    0.5000   -0.8660
```

```
% Таким образом, V - матрица поворота
rad2deg(subspace(A,[1;0])) % угол между подпространством матрицы измерений
единичным вектором (вектор X рисунке), так как матрица
```

```
ans = 30.0000
```

```
% A - сингулярна (измерения полностью коррелированы), ее подпространство
% состоит из одного вектора, поэтому углом между поодпространством ее
% столбцов и вектором будет угол фи
disp("Матрица ковариации:")
```

```
Матрица ковариации:
```

```
C = A*A'/(numel(t)-1)
```

```
C = 2×2
    0.3826    0.2209
    0.2209    0.1275
```

Пример использования: "случаные" данные

Генерация исходных данных по трем свойствам (два из них - коррелированы)

```
clearvars
rand_fun = "randn"; % выбираем статистику из которой набираются "экспериментальные
точки" randn - Гауссово распределение, rand - равномерное распределение от 0 до 1
rand_fun_handle = str2func(rand_fun); % форма распределения
points_number = 90; % число экспериментальных точек
```

```
X = 60 + rand_fun_handle(points_number,1); % генерим массив случайных точек
(результаты измерения первого параметра)
Y = 1.26+ 0.1*rand_fun_handle(points_number,1); % генерим массив случайных точек
(результаты измерения второго параметра) полностью случайны, не зависят от X
alfa_cor =0.75;
Ycor = 3 + (alfa_cor*X + (1-alfa_cor)*rand_fun_handle(points_number,1));% Ycor -
коррелирует с X
disp("Матрица свойств:")
```

```
Матрица свойств:
```

```
t1 = table(X,Y,Ycor,'VariableNames',["Прочность" "Теплопроводность" "Удельная
поверхность"]);
disp(t1)
```

Прочность	Теплопроводность	Удельная поверхность
58.356	1.2349	46.696
60.48	1.5433	48.401
59.324	1.3219	48.062
60.844	1.2112	48.625
60.061	1.0559	47.72
58.85	1.2157	46.858
59.683	1.4396	48.475
59.969	1.2386	47.883
58.798	1.2389	47.181
61.537	0.97752	49.155
60.71	0.98332	48.459
59.591	1.1642	47.84
59.772	1.3748	47.786
60.166	1.0364	48.028
58.925	1.2973	46.993
59.669	1.3501	47.602
61.956	1.5059	49.285
61.068	1.2795	48.572
59.578	1.1383	47.633
59.056	1.1298	47.443
60.525	1.2397	48.409
61.22	1.4186	48.796
58.818	1.2081	47.177
58.823	1.4112	47.409
58.975	1.3574	47.58
59.855	1.316	48.228
60.938	1.357	48.281
61.109	1.3471	48.545
58.804	1.2873	47.224
60.025	1.1381	48.21
58.615	1.2219	46.363
59.912	1.3502	47.774
59.642	1.4155	47.681
60.184	1.1597	48.088
59.716	1.3461	47.436
60.832	1.2033	48.934
59.685	1.3291	47.432
59.668	1.1342	47.565
60.067	1.3126	48.056
60.35	1.1793	48.527
60.507	1.2283	48.557
58.662	1.2549	46.889
58.946	1.1224	47.124

```
    60.239          1.2045           47.833
    59.901          1.2047           47.799
     59.04          1.2378           47.252
    59.234          1.3097           47.072
     60.25          1.4787           48.144
    61.086          1.2797           48.585
    59.194          1.1763           47.566
    60.741           1.126           48.456
    59.011          1.3332           47.069
    60.616          1.2042           48.652
    57.293          1.2229           45.842
    59.981          1.1925           48.394
    61.752          1.1732           49.446
    60.211          1.3143            48.74
     59.64          1.4104           47.944
    59.007          1.1843           47.144
    60.817          1.3675           48.378
    59.861          1.1237           48.123
    59.625          1.3642           47.816
    60.434          1.2165           47.926
    60.394          1.3209           48.029
    61.482          1.3088           49.122
    59.946          1.3055           47.821
    60.076          1.2505           47.982
    60.758          1.2516           48.535
    58.906          1.1405           46.717
    58.747          1.2851           47.055
    60.356          1.3652           47.908
    60.042          1.2794           47.967
    62.225          1.1763           49.815
    59.179          1.3419           47.086
    60.301          1.3519           48.605
     58.69          1.4384           47.097
    60.613          1.3684           48.172
    59.017          1.4368           47.341
    58.902          1.4058           46.957
    61.584          1.1936            48.89
    61.117           1.124            48.86
    60.244          1.0802            48.69
    60.265          1.2798           48.372
    61.656          1.0951           49.292
    59.755          1.3323           48.017
    59.386          1.3168           47.415
    59.113          1.2006           47.362
    59.745          1.2196            48.16
    58.343         0.95449           46.743
    59.381          1.1455           47.507
```

```
disp("Матрица испытаний:")
```

```
Матрица испытаний:
```

```
disp([X';Y';Ycor'])
```

```
   58.3560   60.4796   59.3239   60.8438   60.0607   58.8500   59.6832   59.9688   58.7983   61.5371   60.7104   59
    1.2349    1.5433    1.3219    1.2112    1.0559    1.2157    1.4396    1.2386    1.2389    0.9775    0.9833    1
   46.6964   48.4006   48.0615   48.6250   47.7197   46.8584   48.4747   47.8833   47.1807   49.1545   48.4590   47.
```

```
% СЧИТАЕМ СРЕДНИЕ И НОРМИРУЕМ

Xn = (X-mean(X)); % смещаем среднее для первого свойства
Yn = (Y - mean(Y));% смещаем среднее для второго свойсва
```

```
Ycorn = ((Ycor - mean(Ycor))); % нормированные данные

% вначале рассмотрим матрицы 2x2
A2 = transpose([Xn,Yn]); % матрица испытаний для некоррелированных параметров (два
параметра)
A2cor = transpose([Xn,Ycorn]); % матрица испытаний для коррелированных параметров
(два параметра)
A3 = transpose([Xn,Yn,Ycorn]); % матрица испытаний для всех трех свойств
covMATuncor = A2*transpose(A2)/(points_number-1); % матрица ковариации для
некоррелированных данных
covMATcor =  A2cor*transpose(A2cor)/(points_number-1);% матрица ковариации для
коррелированных данных
covMAT3 = A3*transpose(A3)/(points_number-1);
disp("Матрица ковариации независимых друг от друга данных:")
```

```
Матрица ковариации независимых друг от друга данных:
```

```
disp(covMATuncor)
```

```
    0.8660   -0.0041
   -0.0041    0.0138
```

```
disp("Матрица ковариации взаимосвязанных данных Y="+alfa_cor+"*X +"+(1-alfa_cor)
+"*rand")
```

```
Матрица ковариации взаимосвязанных данных Y=0.75*X +0.25*rand
```

```
disp(covMATcor)
```

```
    0.8660    0.6488
    0.6488    0.5473
```

```
disp("Матрица ковариации для трех параметров {X=rand,rand,Y="+alfa_cor+"*X +"+(1-
alfa_cor)+"*rand}")
```

```
Матрица ковариации для трех параметров {X=rand,rand,Y=0.75*X +0.25*rand}
```

```
disp(covMAT3)
```

```
    0.8660   -0.0041    0.6488
   -0.0041    0.0138   -0.0033
    0.6488   -0.0033    0.5473
```

```
scatter(t1.("Прочность"),t1.("Теплопроводность"));
xlabel(t1.Properties.VariableNames{1});
ylabel(t1.Properties.VariableNames{2});
title("Независимые параметры")
grid on
```


```
scatter(t1.("Прочность"),t1.("Удельная поверхность"));
xlabel(t1.Properties.VariableNames{1});
ylabel(t1.Properties.VariableNames{3});
title("Взаимосвязанные параметры")
grid on
```

Собственные значения и собственные вектора матрицы ковариации:

```
[Q,L] = eig(covMATcor,'vector')
```

```
Q = 2×2
    0.6170   -0.7869
   -0.7869   -0.6170
L = 2×1
    0.0385
    1.3747
```

```
ax = draw_vector([],"Scattered data","","point",A2cor(1,:),A2cor(2,:));
```

```
fig13
```

```
draw_vector(ax,"Собственные вектора матрицы ковариации","","vector",Q(:,1),Q(:,2))
```

```
ans =
  Axes (Собственные вектора матрицы ковариации) with properties:

            XLim: [-1 1]
            YLim: [-1 1]
          XScale: 'linear'
          YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1300 0.1100 0.7750 0.8150]
            Units: 'normalized'

  Show all properties
```

```
[U,S,V] = svd(A2cor) % SVD разложение
```

```
U = 2×2
    0.7869    0.6170
    0.6170   -0.7869
S = 2×90
   11.0612         0         0         0         0         0         0         0 ···
         0    1.8517         0         0         0         0         0         0
V = 90×90
   -0.1792   -0.0004   -0.0776    0.1056    0.0218   -0.1170   -0.0391    0.0073 ···
    0.0670   -0.0170   -0.2485   -0.0085    0.1309    0.1162   -0.3115    0.0318
   -0.0341   -0.2580    0.9332    0.0054    0.0338    0.0204   -0.0796    0.0084
    0.1054    0.0090    0.0063    0.9905   -0.0016    0.0108    0.0027   -0.0006
   -0.0008    0.1328    0.0337   -0.0011    0.9827   -0.0125    0.0410   -0.0043
   -0.1350    0.0954    0.0191    0.0112   -0.0119    0.9762    0.0296   -0.0027
    0.0145   -0.3139   -0.0791    0.0015    0.0409    0.0310    0.9031    0.0100
    0.0018    0.0326    0.0084   -0.0005   -0.0043   -0.0029    0.0101    0.9989
   -0.1207   -0.0588   -0.0195    0.0113    0.0082   -0.0078   -0.0180    0.0022
    0.1843    0.0150    0.0108   -0.0166   -0.0027    0.0189    0.0044   -0.0010
     ⋮
```

```
I = ones(size(A2cor,2),1);
% если данные линейно коррелированы, почему бы не попробовать зафитить их
% полиномом первой степени
% A2cor(2,:) = lsqr_fit(1) + lsqr_fit(2)*A2cor(1,:)
vandermonde_matrix = [I A2cor(1,:)'] % матрица Вадермонда (что это?)
```

```
vandermonde_matrix = 90×2
    1.0000   -1.5599
    1.0000    0.5637
    1.0000   -0.5920
    1.0000    0.9279
    1.0000    0.1448
    1.0000   -1.0659
    1.0000   -0.2327
    1.0000    0.0529
    1.0000   -1.1176
    1.0000    1.6212
     ⋮
```

```
lsqr_fit = atan(vandermonde_matrix\A2cor(2,:)') % решаем задачу метода наименьших
квадратов (
```

```
lsqr_fit = 2×1
   -0.0000
    0.6430
```

```
a_lsqrt = [cos(lsqr_fit(2));sin(lsqr_fit(2))]
```

```
a_lsqrt = 2×1
    0.8003
    0.5996
```

```
draw_vector(ax,"Собственные вектора матрицы ковариации и сингулярные вектора
матрицы испытаний",["u1" "u2"],"vector",U(:,1),U(:,2)*S(2,2)/S(1,1),a_lsqrt)
```

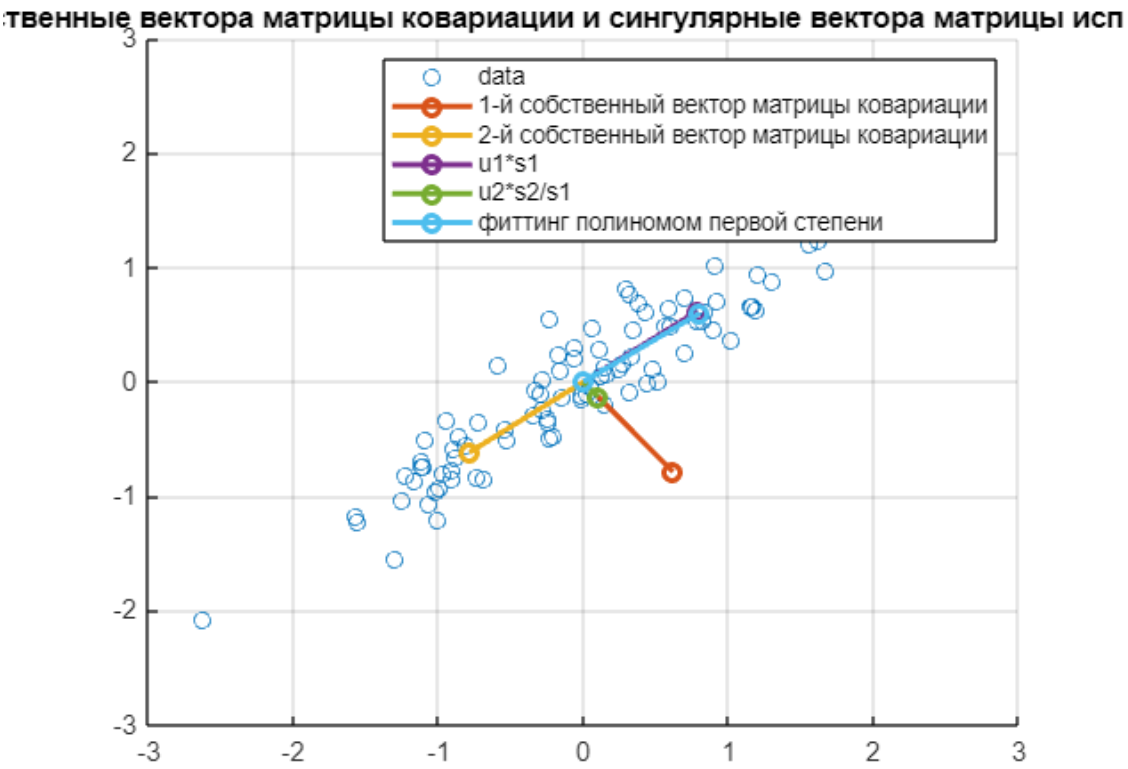
```
ans =
  Axes (Собственные вектора матрицы ковариации и сингулярные вектора матрицы испытаний) with properties:

            XLim: [-1 1]
            YLim: [-1 1]
```

```
          XScale: 'linear'
          YScale: 'linear'
    GridLineStyle: '-'
         Position: [0.1300 0.1100 0.7750 0.8150]
            Units: 'normalized'

  Show all properties
```

```
xlim(ax,[-3,3])
ylim(ax,[-3,3])
legend(ax,["data" "1-й собственный вектор матрицы ковариации" ...
    "2-й собственный вектор матрицы ковариации" "u1*s1" "u2*s2/s1" "фиттинг
полиномом первой степени "])
```




```
% transpose(Q)*U
disp("Матрица сингулярных значений матрицы испытаний (в квадрате и нормированы на
(1-N)):")
```

```
Матрица сингулярных значений матрицы испытаний (в квадрате и нормированы на (1-N)):
```

```
(S.^2)/(points_number-1)
```

```
ans = 2×90
    1.3747         0         0         0         0         0         0         0 ···
         0    0.0385         0         0         0         0         0         0
```

```
disp(" Матрица собственных значений матрицы ковариации : ")
```

```
 Матрица собственных значений матрицы ковариации :
```

```
diag(eig(covMATcor))
```

```
ans = 2×2
    0.0385         0
         0    1.3747
```

Таким образом, левые сингулярные вектора дают направление векторов вдоль главных осей эллипса, первая главная ось соотвествует направлению, на которое сумма проекций максимальна.

Проекция столбцов матрицы A на вектор $\vec{u_1}$:

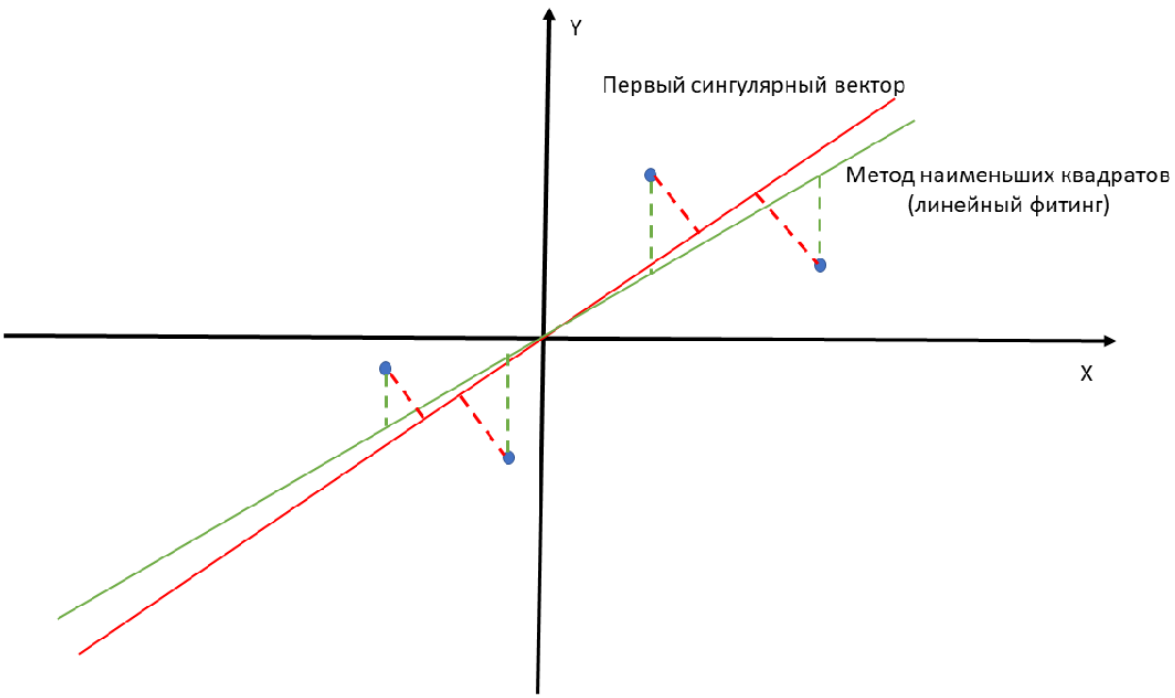
$$\vec{u}_1^T A = \vec{u}_1^T U \Sigma V^T = \vec{u}_1^T \sum_{i=1}^{m} [\sigma_i \vec{u}_i \vec{v}_i^T] = \sigma_1 \vec{v}_1^T$$

Амплитуда суммы проекций столбцов матрицы это длина этого вектора:

$$||\vec{u}_1^T A|| = ||\sigma_1 \vec{v}_1^T|| = \sigma_1$$

Направление первого левого сингулярного вектора показывает вектор, сумма проекций векторов состояния (столбцы матрицы измерений) на который максимальна!

Разница между методом наименьших квадратов и сингулярным разложением показана на рисунке:



Линейная алгебра построена так, чтобы она без проблемы экстраполировалась на пространства б**о**льшей размерности.

Три ...

```
covMAT3
```

```
covMAT3 = 3×3
    1.0916   -0.0051    0.8389
   -0.0051    0.0076    0.0015
    0.8389    0.0015    0.6862
```

```
%ax2 = draw_vector([],"Scattered data","","point",A3(1,:),A3(2,:),A3(3,:));
[U,S,V] = svd(A3)
```

```
U = 3×3
    0.7858   -0.6030    0.1378
   -0.0018    0.2206    0.9754
    0.6185    0.7667   -0.1722
S = 3×90
   12.4868         0         0         0         0         0         0         0 ···
         0    1.5448         0         0         0         0         0         0
         0         0    0.7698         0         0         0         0         0
V = 90×90
   -0.0537   -0.0577   -0.0005    0.0930   -0.1385   -0.0205    0.2151    0.0160 ···
   -0.0471    0.0921    0.0738   -0.0041    0.0301    0.0198    0.0214   -0.2072
   -0.1837    0.0753   -0.0500   -0.1249   -0.1424   -0.0607    0.1278   -0.1466
    0.0733    0.1268   -0.0574    0.9760   -0.0073   -0.0063   -0.0008   -0.0209
   -0.1531    0.0915   -0.0893   -0.0006    0.9645   -0.0101    0.0424   -0.0055
   -0.0274    0.0416   -0.0454   -0.0044   -0.0111    0.9959    0.0109   -0.0029
    0.2279   -0.0910    0.0348   -0.0093    0.0406    0.0092    0.9443    0.0098
   -0.0130    0.2292    0.1145   -0.0186   -0.0141   -0.0049    0.0212    0.9417
   -0.1475   -0.0396    0.0023    0.0183   -0.0149   -0.0009    0.0258    0.0108
    0.2338   -0.1356    0.0254   -0.0053    0.0448    0.0107   -0.0609    0.0196
      ⋮
      ⋮
```

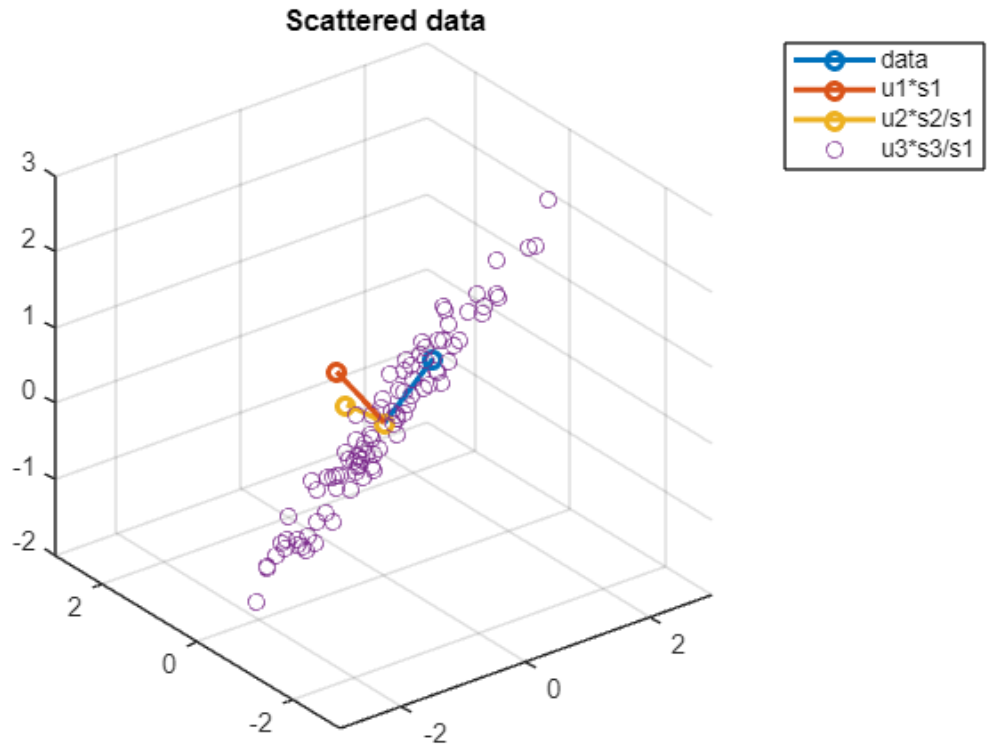
```
U(:,3)'*U(:,2)
```

```
ans = -1.3878e-16
```

```
ax2 =draw_vector([],"Scattered data","","vector",U(:,1),U(:,2),U(:,3));
```

```
fig11
```

```
draw_vector(ax2,"Scattered data","","point",A3(1,:),A3(2,:),A3(3,:));
xlim(ax2,[-3,3]);
ylim(ax2,[-3,3]);
legend(ax2,["data" "u1*s1" "u2*s2/s1" "u3*s3/s1"]);
```



Пять...

```
clearvars -except points_number rand_fun_handle alfa_cor
X1 = randn(points_number,1);
X2 = randn(points_number,1);
X3 = randn(points_number,1);
alfas1 = [0.01,0.01,0.07];
alfas2 = [0,0,0.08];
alfas2 = alfas2/sum(alfas2);alfas1 = alfas1/sum(alfas1);

X4 = (1-alfa_cor)*rand_fun_handle(points_number,1) +
alfa_cor*sum([X1,X2,X3].*alfas1,2);
X5 = (1-alfa_cor)*rand_fun_handle(points_number,1) +
alfa_cor*sum([X1,X2,X3].*alfas2,2);

C = transpose([X1,X2,X3,X4,X5])*[X1,X2,X3,X4,X5]/(points_number-1)
```

```
C = 5×5
    1.2861    0.2160    0.0660    0.1608    0.0737
    0.2160    1.0458   -0.0964    0.0713   -0.0456
    0.0660   -0.0964    1.1366    0.7230    0.8458
    0.1608    0.0713    0.7230    0.5497    0.5387
    0.0737   -0.0456    0.8458    0.5387    0.7022
```

```
SIG = svds(C)/(points_number-1)
```

```
SIG = 5×1
    0.0258
    0.0159
    0.0102
    0.0008
    0.0004
```

ВЫВОД:

Метод анализа главный компонент позволяет установить насколько коррелированы исходные данные, а также дает линейное преобразование исходных данных в "новые", в которых базис соответствует собственным векторам матрицы ковариации.

M - матрица эксперимента

S = svds(M,N) - первые N ее сингулярных значений, чем ближе они друг к другу, тем более некоррелированными являются колонки в таблице

Литература

1. Gilbert Strang. Linear algebra and learning from data. MIT (2019)
2. Gilbert Strang. Introduction to linear algebra. 2016 (Есть перевод старой версии книги : Г.Стрэнг Введение в линейную алгебру)
3. youtube: канал MIT OpenCourseWare, курс лекций MIT: 18.06SC Linear Algebra (2011)
4. youtube: канал MIT OpenCourseWare, курс лекций: MIT 18.065 Matrix methods in Data Analysis, Signal processing, and Mechine Learning (2018)
5. youtube: канал AMATH 301, лектор Nathan Kutz, лекции: The singular Value Decomposition and Principal Componen Analysis


```
function ax = draw_vector(ax,ttl,names,type,varargin)
% функция строит двух- и трех-мерные вектора, а также рассеянные данные из
% матрицы
% ax - оси (если пустые, то создаются новые)
% ttl - заголовок картинки
% names - имена векторов
% type:
%       "vector" - аргументы, которые передаются после интерпретируются
%                  как отдельные вектора
%       "point"  - в этом случае передается матрица в качестве аргумента и
%       столбцы матрицы строятся при помощи функций scatter и scatter3 d
%       в зависимости от размерности массива
    arguments
        ax =[]
        ttl string =strings(0,1)
        names string =strings(0,1)
        type string {mustBeMember(type,["vector" "point"])}="vector"
```

```
    end
    arguments (Repeating)
        varargin double
    end
    was_empty = isempty(ax); % это признак того, что все строится на новых осях
    if was_empty
        ax = get_next_ax();
    else
        hold(ax,"on");
        % if ~isempty(ax.Legend)
        %     leg_before = ax.Legend.String;
        % else
        %     leg_before = strings(0,1);
        % end
    end

    if strcmp(type,"vector")
        is_3D = numel(varargin{1})==3;
            if is_3D
                [x,y,z]  = make_xy(varargin{1});
                plot3(ax,x,y,z,'LineWidth',2,'Marker','o');
                hold on
                for iii = 2:numel(varargin)
                        [x,y,z] = make_xy(varargin{iii});
                        plot3(ax,x,y,z,'LineWidth',2,'Marker','o');
                end
                grid on
                hold off
            else
                [x,y]  = make_xy(varargin{1});
                plot(ax,x,y,'LineWidth',2,'Marker','o');
                hold on
                for iii = 2:numel(varargin)
                        [x,y]  = make_xy(varargin{iii});
                        plot(ax,x,y,'LineWidth',2,'Marker','o');
                end
                grid on
                hold off
            end
            if isempty(names)||(numel(names)~=numel(varargin))
                legend(ax,string(1:numel(varargin)));

            else
                % if ~was_empty
                %      names= [names(:);leg_before(:)];
                % end
                legend(ax,names);
            end
            xlim(ax,[-1 1]);
            ylim(ax,[-1 1]);
```

```
            if ~isempty(ttl)
                title(ax,ttl);
            end
    else
        %data_number = numel(varargin); % число массивов данных
        is_3D = numel(varargin)==3;
        data = varargin{1};
        if size(data,2)>1
            data = transpose(data);
            is_transpose = true;
        else
            is_transpose = false;
        end
        if ~is_transpose
            for iii = 2:numel(varargin)
                data = [data,varargin{iii}];
            end
        else
            for iii = 2:numel(varargin)
                data = [data,transpose(varargin{iii})];
            end
        end

        if is_3D
            scatter3(ax,data(:,1),data(:,2),data(:,3));
        else
            scatter(ax,data(:,1),data(:,2));
        end

    end
    if ~was_empty
            hold(ax,"off");
    end
end
function [x,y,z]  = make_xy(col)
% добавляет к координатам вектора нули так, чтобы при помощи функции plot
% строилась линия
    switch numel(col)
        case 1
            x = [col(1)];
            y = 0;
            z = 0;
        case 2
            x = [0 col(1)];
            y = [0 col(2)];
            z = zeros(1,2);
        case 3
            x = [0 col(1)];
            y = [0 col(2)];
            z = [0 col(3)];
```

```
    end
end
function [bpar,bper,ang] = projection_matrix(A,b)
% функция считает угол между вектором и пространством столбцов матрицы A
    for ii =1:size(A,2)
        A(:,ii) = A(:,ii)/norm(A(:,ii));
    end
    beta = A*transpose(A)*b;
    beta = beta/norm(beta); % нормируем вектор beta
    Pbeta = beta*transpose(beta); % оператор проектирования вектора на
    bpar = Pbeta*b;
    bper = b-bpar;
    ang = rad2deg(acos(norm(bpar)/norm(b)));
end
function [new_ax,fig_handle] = get_next_ax(index)
% функция, которая возвращает новые оси на новой фигуре
    arguments
        index = []
    end
    persistent N;
    if isempty(index)
        if isempty(N)
            N=1;
        else
            N  = N+1;
        end
        fig_handle = figure(N);
        clf(fig_handle);
        new_ax = axes(fig_handle);
        disp("fig"+ N)
    else
        fig_handle = figure(index);
        clf(fig_handle);
        new_ax = axes(fig_handle);
    end
end
function mem_size = mem_size_summ(varargin)
% функция считает объем несокльких переменных в памяти base workspace
    names = string(varargin);
    mem_size = 0;
    base_vars = evalin("base","whos");
    flag = arrayfun(@(X)any(strcmp(X.name,names)),base_vars);
    if ~any(flag)
        return
    end
    mem_size = sum(arrayfun(@(X)X.bytes,base_vars(flag)));
end
function folder = get_folder()
% текущая папка
folder = fileparts(matlab.desktop.editor.getActiveFilename);
```

```
end
```