

# Зачем нужны матрицы

Будем решать следующую задачу:

Как, имея таблицу данных с результатами измерения разных свойств, например, таблицу трендов по времени для свойств материала, вытащенных из базы данных за некоторый период времени, понять какие из свойств являются линейно коррелированными.

## PCA - principal component analysis (анализ главных компонент)

PCA - способ анализа степени "независимости" столбцов матрицы. Позволяет понять есть ли среди данных "лишние", то есть не несущие информации, понять как нужно преобразовать исходные данные, чтобы они были максимально линейно независимыми. Позволяет уменьшить размерность данных.

Применяется во многих разделах анализа данных, таких как, классификация, сжатие и распознавание изображений, корреляционный анализ и др. Часто используется в машинном обучении, когда надо "прорядить" исходные данные обучающей выборки для нейронных сетей.

## Немного о матрицах.

Матрица A - это:

1. Упорядоченная коллекция чисел:  $A(i, j) \Rightarrow$  индексирование (get\_index, set\_index)
2. Упорядоченная коллекция вектор-столбцов:  $A = [\vec{a}_1, \dots, \vec{a}_i, \dots, \vec{a}_m]$ ,  $A(:, i) = \vec{a}_i \Rightarrow$  итерирование по столбцам матрицы в цикле
3. Оператор линейного преобразования векторов, символ  $[A]$  - обозначает "действие" матрицы:  
 $\vec{A} \vec{b} = \vec{c} \Leftrightarrow$  подействовав матрицей  $A$  на вектор  $\vec{b}$  мы преобразовали его в вектор  $\vec{c}$
4. Совокупность  $m$  вектор-столбцов (в каждом из которых  $N$  координат)  $A = [\vec{a}_1, \dots, \vec{a}_i, \dots, \vec{a}_m]$  - это линейное подпространство векторного пространства  $R^N$  всех векторов размером  $N$ , обозначается как  $span\{A\}$

Сингулярная матрица - матрица, у которой колонки (или строки) являются зависимыми (либо равными нулю), то есть могут быть выражены как линейная комбинация других строк (колонок) этой же матрицы.

Ранг матрицы - количество ее линейно-независимых столбцов (строк)

Для сингулярной матрицы  $A$  размером  $[N \times M]$ , размерность линейного подпространства ее столбцов  $span\{A\}$  меньше размерности линейного пространства  $R^N$ .

**Как понять насколько сильно отличаются два вектора друг от друга?**

Косинус угла между ними!

Косинус угла между двумя векторами:

$$\cos(\alpha) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} = \hat{a} \cdot \frac{\vec{b}}{|\vec{b}|} = \hat{a}^T \frac{\vec{b}}{|\vec{b}|}$$

(\*)

$\hat{a}$  - единичный вектор, направленный вдоль  $\vec{a}$

Чем меньше косинус угла между векторами, тем они больше друг от друга отличаются (для ортогональных векторов он равен нулю), чем ближе косинус (по модулю) к единице, тем более "похожи"

$$\text{вектора друг на друга, так как } \cos(\alpha) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|}$$

### Проекция вектора на вектор

У вектора  $\vec{b}$  есть зависимая от  $\vec{a}$  компонента  $\vec{b}_{\parallel \vec{a}}$ , то есть та, которая может быть получена путем

умножения  $\vec{a}$  на число и полностью независимая компонента  $\vec{b}_{\perp \vec{a}}$  ортогональная вектору  $\vec{a}$ .

$$\vec{b} = \vec{b}_{\parallel \vec{a}} + \vec{b}_{\perp \vec{a}}$$

$\vec{b}_{\parallel \vec{a}} = \hat{a} |\vec{b}| \cos(\alpha) = \hat{a} |\vec{b}| \hat{a} \cdot \frac{\vec{b}}{|\vec{b}|} = \hat{a} (\hat{a}^T \vec{b})$  - вектор проекции вектора  $\vec{b}$  на направление вектора  $\vec{a}$  -

показывает зависимую от  $\vec{a}$  долю вектора  $\vec{b}$

$\vec{b}_{\perp \vec{a}} = \vec{b} - \vec{b}_{\parallel \vec{a}} = \vec{b} - \hat{a} (\hat{a}^T \vec{b})$  - характеризует независимую от вектора  $\vec{a}$  составляющую вектора  $\vec{b}$

(\*\*)

```
clearvars
a_vec = rand(2,1) % генерим случайный вектор на плоскости
```

```
a_vec = 2x1
    0.8147
    0.9058
```

```
% Проекция векторов
R =[78.511;0.73] % координаты вектора в сферической системе координат
```

```
R = 2x1
    78.5110
    0.7300
```

```
% угол - модуль
ang = deg2rad(R(1)); % угол вектора
module = R(2);% модуль вектора
```

```
b_vect = [module*cos(ang);module*sin(ang)];  
a_hat = a_vec/norm(a_vec) %- единичный вектор вдоль а
```

```
a_hat = 2x1  
0.6687  
0.7435
```

```
b_par = a_hat*(transpose(a_hat)*b_vect) % проекция вектора b на вектор а
```

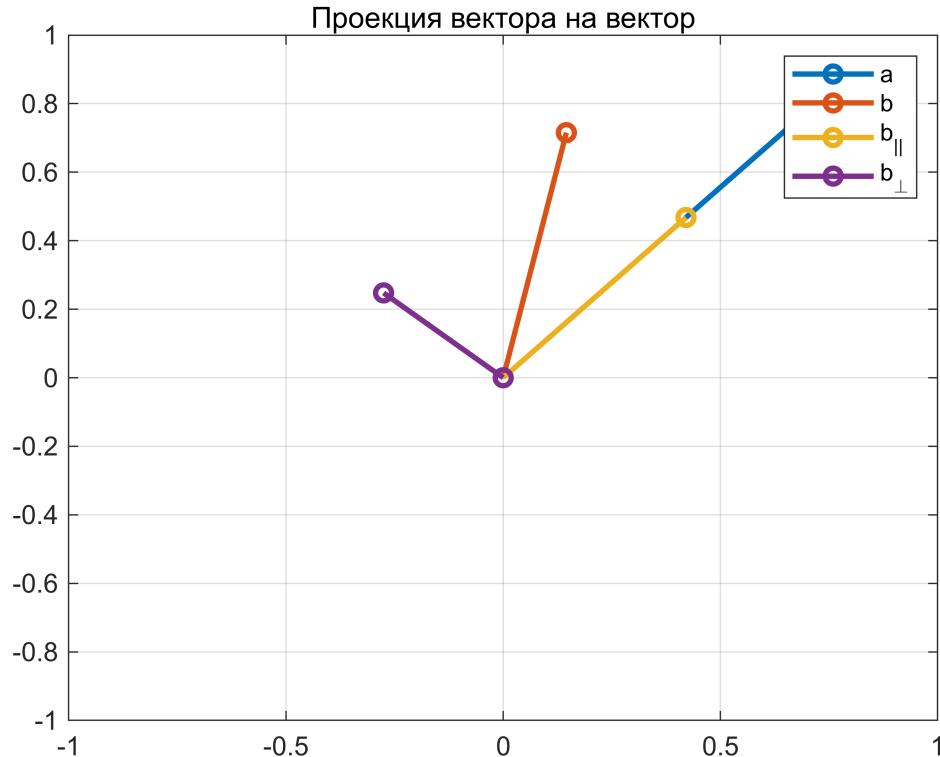
```
b_par = 2x1  
0.4207  
0.4677
```

```
b_per = b_vect - b_par % составляющая вектора b перпендикулярная вектору а
```

```
b_per = 2x1  
-0.2753  
0.2476
```

```
draw_vector([], 'Проекция вектора на вектор', ["a" "b" "b_||"  
"b_\perp"], "vector", a_vec, b_vect, b_par, b_per);
```

fig1



```
disp("(b_parallel^T)*b_perpendicular = " + transpose(b_per)*b_par)
```

```
(b_parallel^T)*b_perpendicular = -5.5511e-17
```

А что, если в выражении (\*\*) взять и "вынести вектор"  $\vec{b}$  за скобки

$\vec{b}_{\perp \vec{a}} = (\vec{I} - \hat{\vec{a}}\hat{\vec{a}}^T)\vec{b} = (\vec{I} - P_{\vec{a}})\vec{b}$  - составляющая вектора  $\vec{b}$ , перпендикулярная вектору  $\vec{a}$ , получается

действием матрицы  $T = (\vec{I} - P_{\vec{a}})$  на вектор  $\vec{b}$

$P_a = \hat{\vec{a}}\hat{\vec{a}}^T$  - оператор проецирования произвольного вектора на вектор  $\vec{a}$  - внешнее скалярное произведение

$$\vec{b}_{\parallel \vec{a}} = P_a \vec{b}$$

```
a = sym("a1", [3 1])
```

a =

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix}$$

```
a*transpose(a)
```

ans =

$$\begin{pmatrix} a_{11}^2 & a_{11}a_{12} & a_{11}a_{13} \\ a_{11}a_{12} & a_{12}^2 & a_{12}a_{13} \\ a_{11}a_{13} & a_{12}a_{13} & a_{13}^2 \end{pmatrix}$$

```
% посчитаем проекции через матричные операторы
```

```
P = a_hat*transpose(a_hat) % оператор проецирования
```

P = 2x2

$$\begin{matrix} 0.4472 & 0.4972 \\ 0.4972 & 0.5528 \end{matrix}$$

```
b_per2 = (eye(2)-P)*b_vect
```

$$\begin{matrix} b_{\text{per2}} = 2 \times 1 \\ -0.2753 \\ 0.2476 \end{matrix}$$

```
b_par2 = P*b_vect
```

$$\begin{matrix} b_{\text{par2}} = 2 \times 1 \\ 0.4207 \\ 0.4677 \end{matrix}$$

$$\vec{b}_{\parallel \vec{a}} \cdot \vec{b}_{\perp \vec{a}} = \vec{b}_{\parallel \vec{a}}^T \vec{b}_{\perp \vec{a}} = \vec{b}^T P_a^T (I - P_a) \vec{b} = 0$$

## Проекция вектора на векторное подпространство

Теперь рассмотрим два вектора  $\vec{a}_1$  и  $\vec{a}_2$ , они образуют плоскость (в трехмерном пространстве), на эту плоскость можно спроектировать вектор  $\vec{b}$

$\vec{b}_{\parallel \vec{a}_1} = P_{a_1} \vec{b}$  - проекция вектора  $\vec{b}$  на вектор  $\vec{a}_1$

$\vec{b}_{\parallel \vec{a}_2} = P_{a_2} \vec{b}$  - проекция вектора  $\vec{b}$  на вектор  $\vec{a}_2$

$\vec{\beta}_{\parallel \vec{a}_1, \vec{a}_2} = (P_{a_1} + P_{a_2}) \vec{b}$

Если ввести матрицу  $A = [\vec{a}_1, \vec{a}_2]$ , то оператор  $AA^T$  будет давать вектор  $\vec{\beta}_{\parallel \vec{a}_1, \vec{a}_2}$ , лежащий в плоскости вектор-столбцов этой матрицы:

$\vec{\beta}_{\parallel \vec{a}_1, \vec{a}_2} = (P_{a_1} + P_{a_2}) \vec{b} = (\hat{a}_1 \hat{a}_1^T + \hat{a}_2 \hat{a}_2^T) \vec{b} = AA^T \vec{b}$

$P_{a_1, a_2} = P_{a_1} + P_{a_2} = AA^T$

```
% Проверка формул выше через символьные вычисления
a1 = sym("a1",[3 1], 'real') % вектор-столбец
```

a1 =

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix}$$

```
a2 = sym("a2",[3 1], 'real') % вектор-столбец
```

a2 =

$$\begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \end{pmatrix}$$

```
b = sym("b",[3,1], 'real') % вектор-столбец
```

b =

$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

```
Pa1a2 = (a1*a1' + a2*a2') % сумма операторов проектирования
```

Pa1a2 =

$$\begin{pmatrix} a_{11}^2 + a_{21}^2 & a_{11}a_{12} + a_{21}a_{22} & a_{11}a_{13} + a_{21}a_{23} \\ a_{11}a_{12} + a_{21}a_{22} & a_{12}^2 + a_{22}^2 & a_{12}a_{13} + a_{22}a_{23} \\ a_{11}a_{13} + a_{21}a_{23} & a_{12}a_{13} + a_{22}a_{23} & a_{13}^2 + a_{23}^2 \end{pmatrix}$$

```
A = [a1,a2] % матрица из двух вектор-столбцов
```

A =

$$\begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{pmatrix}$$

```
Pa1a2_mat = A*A' % сумма матриц проектирования
```

Pa1a2\_mat =

$$\begin{pmatrix} a_{11}^2 + a_{21}^2 & a_{11}a_{12} + a_{21}a_{22} & a_{11}a_{13} + a_{21}a_{23} \\ a_{11}a_{12} + a_{21}a_{22} & a_{12}^2 + a_{22}^2 & a_{12}a_{13} + a_{22}a_{23} \\ a_{11}a_{13} + a_{21}a_{23} & a_{12}a_{13} + a_{22}a_{23} & a_{13}^2 + a_{23}^2 \end{pmatrix}$$

```
b_par = Pa1a2*b
```

b\_par =

$$\begin{pmatrix} b_2\sigma_3 + b_3\sigma_2 + b_1(a_{11}^2 + a_{21}^2) \\ b_1\sigma_3 + b_3\sigma_1 + b_2(a_{12}^2 + a_{22}^2) \\ b_1\sigma_2 + b_2\sigma_1 + b_3(a_{13}^2 + a_{23}^2) \end{pmatrix}$$

where

$$\sigma_1 = a_{12}a_{13} + a_{22}a_{23}$$

$$\sigma_2 = a_{11}a_{13} + a_{21}a_{23}$$

$$\sigma_3 = a_{11}a_{12} + a_{21}a_{22}$$

```
b_par2 = Pa1a2_mat*b
```

b\_par2 =

$$\begin{pmatrix} b_2 \sigma_3 + b_3 \sigma_2 + b_1 (a_{11}^2 + a_{21}^2) \\ b_1 \sigma_3 + b_3 \sigma_1 + b_2 (a_{12}^2 + a_{22}^2) \\ b_1 \sigma_2 + b_2 \sigma_1 + b_3 (a_{13}^2 + a_{23}^2) \end{pmatrix}$$

where

$$\sigma_1 = a_{12} a_{13} + a_{22} a_{23}$$

$$\sigma_2 = a_{11} a_{13} + a_{21} a_{23}$$

$$\sigma_3 = a_{11} a_{12} + a_{21} a_{22}$$

```
b_par - b_par2
```

ans =

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Вектор  $\vec{\beta}_{\parallel \vec{a}_1, \vec{a}_2}$  суммы проекций лежит в плоскости векторов  $\vec{a}_1$  и  $\vec{a}_2$ , но он, в общем случае, по

модулю не равен проекции  $\vec{b}_{\parallel \vec{a}_1, \vec{a}_2}$  вектора  $\vec{b}$  на плоскость векторов  $\vec{a}_1$  и  $\vec{a}_2$ .

Зато, мы теперь можем спроектировать исходный вектор на него:

$$\vec{b}_{\parallel \vec{a}_1} = P_\beta \vec{b}$$

Если бы исходные вектора были бы ортонормированы, то :

$$\vec{\beta}_{\parallel \vec{a}_1, \vec{a}_2} = \vec{b}_{\parallel \vec{a}_1, \vec{a}_2}$$

```
% Проекция вектора на пространство столбцов матрицы из двух столбцов
```

```
clearvars
% генерим случайные вектора в трехмерном пространстве
a1 = 2*(0.5 - rand(3,1));
a2 = 3*(0.5 - rand(3,1));
R = [0.72, -50.7, 161.3 ]; % модуль-тетта-фи - сферическая система координат для
трехмерного вектора
r = R(1);theta = R(2);phi = R(3);
b = [r*cosd(theta)*sind(phi);r*cosd(theta)*cosd(phi);r*sind(theta)] % некоторый
вектор
```

```
b = 3x1  
0.1462  
-0.4320  
-0.5572
```

```
A = [a1,a2]
```

```
A = 3x2  
0.7460 1.2074  
-0.8268 0.6645  
-0.2647 -0.1406
```

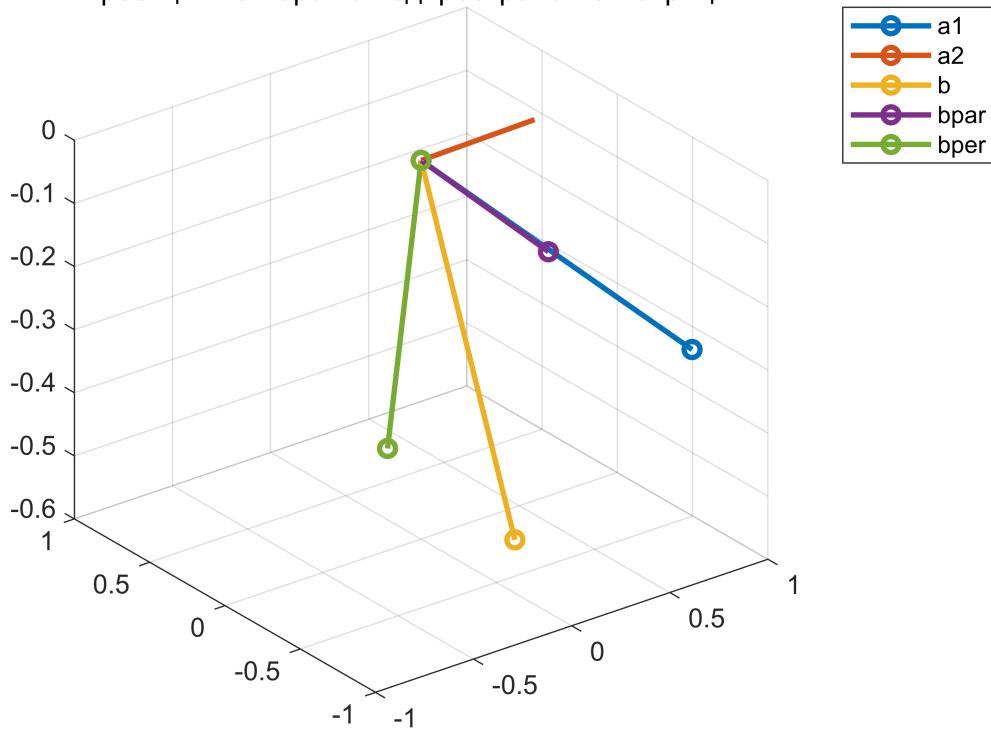
```
%norm(A)  
[bpar,bper,ang] = projection_matrix(A,b);  
disp("bpar'*bper =" + ...  
transpose(bpar)*bper)
```

```
bpar'*bper =4.1633e-17
```

```
draw_vector([], 'Проекция вектора на подпространство матрицы A', ["a1" "a2" "b"  
"bpar" "bper"], "vector", a1,a2, b,bpar,bper);
```

fig2

Проекция вектора на подпространство матрицы A



```
ang
```

```
ang = 41.1164
```

```
ang2 = rad2deg(subspace(A,b))
```

```
ang2 = 39.0227
```

```
% функция subspace позволяет посчитать угол между двумя подпространствами  
% для нее b - не обязательно вектор, но может быть и матрицей
```

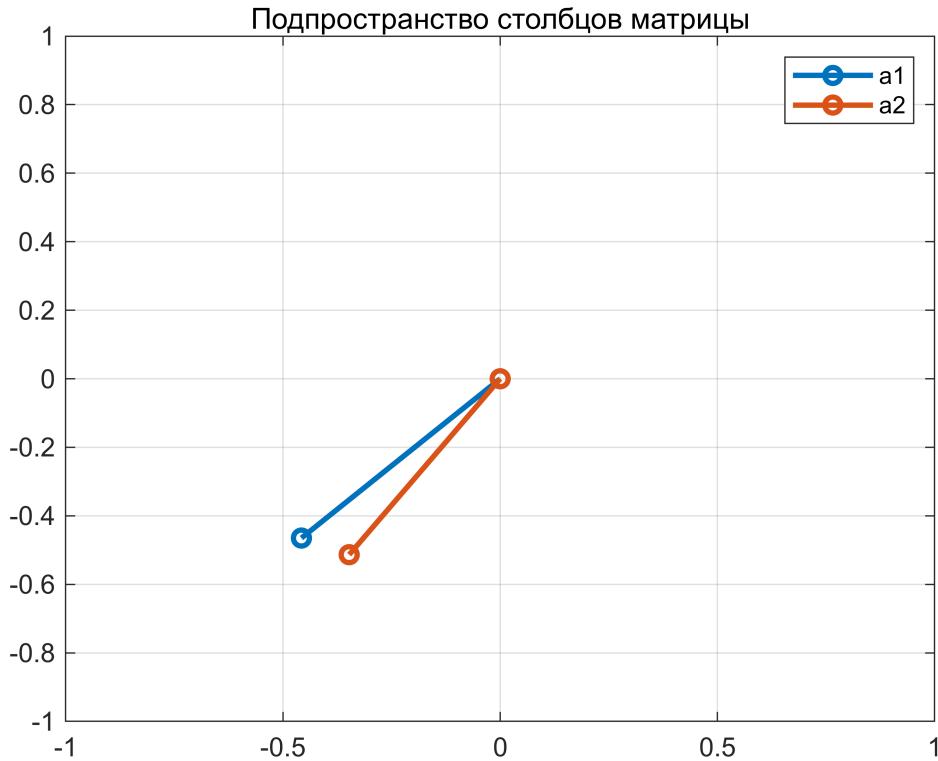
```
clearvars  
a1 = 0.5 - rand(2,1);  
%a2 = 0.5 - rand(2,1);
```

```
R1 = [0.62,235.9];  
r = R1(1);  
thetta = R1(2);  
%phi = R1(3);  
  
make_singular = false;  
if make_singular  
    a2=0.3*a1; % чтобы матриц стала сингулярной  
else  
    a2 = [r*cosd(thetta);r*sind(thetta)];  
  
end  
A = [a1,a2];  
theta = rad2deg(subspace(a1,a2)) % subspace - функция, которая возвращает угол  
между подпространствами
```

```
theta = 10.4415
```

```
ax1 = draw_vector([], 'Подпространство столбцов матрицы', ["a1" "a2"], "vector", a1, a2);
```

```
fig3
```



```
disp("Определитель матрицы A:" + det(A))
```

Определитель матрицы  $A$ : 0.073289

```
disp("SVD спектр матрицы A:" + join(string(svds(A,3))))
```

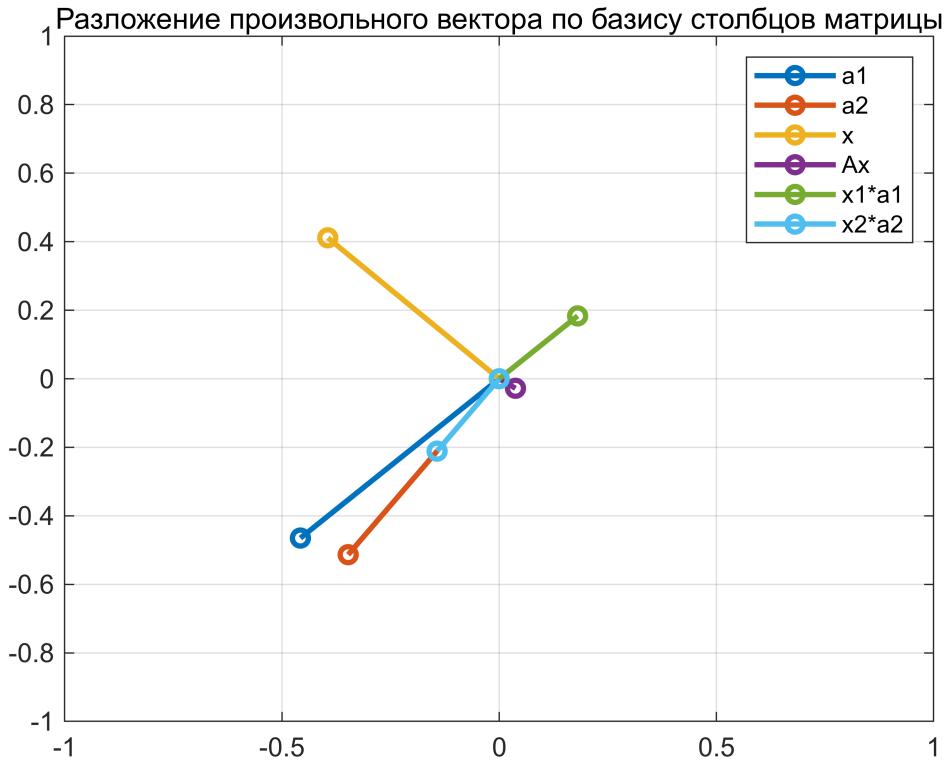
SVD спектр матрицы  $A$ : 0.89618 0.081779

```
disp("Спектр собственных значений матрицы A:" + join(string(eigs(A,3))))
```

Спектр собственных значений матрицы  $A$ : -0.88841 -0.082495

```
R = [0.57, 133.8];
r = R(1);theta = R(2);
X = [r*cosd(theta);r*sind(theta)];
draw_vector([], 'Разложение произвольного вектора по базису столбцов матрицы', ...
    ["a1" "a2" "x" "Ax" "x1*a1" "x2*a2"], ...
    "vector", a1, a2, X, A*X, X(1)*a1, X(2)*a2);
```

fig4



```
theta = subspace(A,X);
disp("Угол между подпространством столбцов матрицы и вектором X = " +
rad2deg(theta))
```

Угол между подпространством столбцов матрицы и вектором X = 2.0116e-14

```
disp("Угол между Ax и x: " + rad2deg(subspace(A*X,X)))
```

Угол между Ax и x: 9.6402

```
disp("Изменение длины вектора |Ax|/|x| = " + norm(A*X)/norm(X))
```

Изменение длины вектора |Ax|/|x| = 0.081893

## Два способа факторизации матриц (представления матрицы в виде произведения других матриц)

### I) Способы представления матриц

Прежде чем смотреть факторизации надо вспомнить общие правила обращения с матрицами

#### 1) Матрица как строка столбцов:

$A = [\vec{a}_1, \dots, \vec{a}_k]$  матрица размером  $[n \times k]$  - это строка из  $k$  векторов  $\vec{a}_i$  размером  $[n \times 1]$  каждый,

$B = [\vec{b}_1, \dots, \vec{b}_m]$  - размером  $[k \times m]$ , из  $m$  векторов  $\vec{b}_i$  - размером  $[k \times 1]$

## 2) Матрица как столбец строк:

$\overset{\rightarrow}{r}_1^T$   
 $A = \begin{matrix} \vdots & \end{matrix},$  где  $\overset{\rightarrow}{r}_n^T$  - вектор-строка (для определенности будем считать, что значок  $\rightarrow$  всегда обозначает столбец),  $\overset{\rightarrow}{r}_i^T : [1 \times k]$ .

$\overset{\rightarrow}{q}_1^T$   
 $B = \begin{matrix} \vdots & \end{matrix},$   $\overset{\rightarrow}{q}_i^T : [1 \times m]$   
 $\overset{\rightarrow}{q}_k^T$

$C = AB$  - матрица размером  $[n \times k]$

```
clearvars  
k=2
```

```
k = 2
```

```
n=3;  
m=2;  
a1 = sym("a1",[n 1], 'real')
```

```
a1 =
```

```

$$\begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix}$$

```

```
a2 = sym("a2",[n 1], 'real')
```

```
a2 =
```

```

$$\begin{pmatrix} a_{21} \\ a_{22} \\ a_{23} \end{pmatrix}$$

```

```
b1 = sym("b1",[k,1], 'real')
```

```
b1 =
```

```

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

```

```
b2 = sym("b2",[k,1], 'real')
```

```
b2 =
```

```

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

```

```

A = [a1,a2];
B = [b1,b2];
r1 = A(1,:)';
q1 = B(1,:)';
q2 = B(2,:)';
r2 = A(2,:)';
r3 = A(3,:)';

```

## II) "Способы" умножения матриц:

Как смотреть на умножение двух матриц

1) **Классический** (суммирование)  $C_{ij} = \sum_k \vec{a}_k(i)\vec{b}_j(k)$

```
C1 = A*B
```

$C_1 =$

$$\begin{pmatrix} a_{11}b_{11} + a_{21}b_{12} & a_{11}b_{21} + a_{21}b_{22} \\ a_{12}b_{11} + a_{22}b_{12} & a_{12}b_{21} + a_{22}b_{22} \\ a_{13}b_{11} + a_{23}b_{12} & a_{13}b_{21} + a_{23}b_{22} \end{pmatrix}$$

2) **Линейная комбинация столбцов** матрицы  $A$  с коэффициентами - координатами столбца матрицы  $B$ :

$C = [\vec{c}_1, \dots, \vec{c}_m] = [\sum_{i=1}^k \vec{a}_i \vec{b}_1(i), \dots, \sum_{i=1}^k \vec{a}_i \vec{b}_m(i)]$

```
C_Columns_Combination = [a1*b1(1)+a2*b1(2),a1*b2(1)+a2*b2(2)] % комбинация столбцов
```

$C\_Columns\_Combination =$

$$\begin{pmatrix} a_{11}b_{11} + a_{21}b_{12} & a_{11}b_{21} + a_{21}b_{22} \\ a_{12}b_{11} + a_{22}b_{12} & a_{12}b_{21} + a_{22}b_{22} \\ a_{13}b_{11} + a_{23}b_{12} & a_{13}b_{21} + a_{23}b_{22} \end{pmatrix}$$

3) **Матрица скалярных произведений (столбец строк на строку столбцов).** Скалярное произведение

$$\vec{r}_1^T \quad \vec{r}_1^T \vec{b}_1 \quad \dots \quad \vec{r}_1^T \vec{b}_m$$

строк матрицы  $A$  и столбцов матрицы  $B$ :  $C = AB = [\vec{r}_1 \dots \vec{r}_n][\vec{b}_1 \dots \vec{b}_m] = [\vec{r}_1^T \vec{b}_1 \dots \vec{r}_n^T \vec{b}_m]$  - это точно также

$$\vec{r}_n^T \quad \vec{r}_n^T \vec{b}_1 \quad \dots \quad \vec{r}_n^T \vec{b}_m$$

как умножать столбец на строку, только вместо скаляров - вектора

```
C_Column_Row = [r1'*b1 r1'*b2;r2'*b1 r2'*b2;r3'*b1 r3'*b2]
```

$C\_Column\_Row =$

$$\begin{pmatrix} a_{11}b_{11} + a_{21}b_{12} & a_{11}b_{21} + a_{21}b_{22} \\ a_{12}b_{11} + a_{22}b_{12} & a_{12}b_{21} + a_{22}b_{22} \\ a_{13}b_{11} + a_{23}b_{12} & a_{13}b_{21} + a_{23}b_{22} \end{pmatrix}$$

То есть, умножение матриц ведет себя точно также как умножение столбца на строку (внешнее произведение):

```
disp("column*row")
```

```
column*row
```

```
a1*b1'
```

```
ans =
```

$$\begin{pmatrix} a_{11} b_{11} & a_{11} b_{12} \\ a_{12} b_{11} & a_{12} b_{12} \\ a_{13} b_{11} & a_{13} b_{12} \end{pmatrix}$$

**4) Сумма матриц (диад) внешних произведений (строка столбцов на столбец строк) столбцов матрицы  $A$  и строк матрицы  $B$ :**

$$C = AB = [\vec{a}_1, \dots, \vec{a}_k] [\vec{q}_1^T, \dots, \vec{q}_k^T] = \vec{a}_1 \vec{q}_1^T + \dots + \vec{a}_k \vec{q}_k^T$$

```
C_Row_Column = a1*q1' + a2*q2' % Сумма внешних произведений
```

```
C_Row_Column =
```

$$\begin{pmatrix} a_{11} b_{11} + a_{21} b_{12} & a_{11} b_{21} + a_{21} b_{22} \\ a_{12} b_{11} + a_{22} b_{12} & a_{12} b_{21} + a_{22} b_{22} \\ a_{13} b_{11} + a_{23} b_{12} & a_{13} b_{21} + a_{23} b_{22} \end{pmatrix}$$

То есть, умножение матриц ведет себя точно также как умножение строки на столбец (скалярное произведение):

```
disp("Умножение столбца на строку:")
```

Умножение столбца на строку:

```
transpose(a2)*a1
```

```
ans = a11 a21 + a12 a22 + a13 a23
```

$$C = AA^T = [\vec{a}_1, \dots, \vec{a}_k] [\vec{a}_1^T, \dots, \vec{a}_n^T] = \vec{a}_1 \vec{a}_1^T + \dots + \vec{a}_k \vec{a}_k^T$$

Ранг каждой из диад  $\overset{\rightarrow}{a_k} \overset{\rightarrow}{a_k}^T$  равен единице!

Ранг матрицы  $C$  может быть равен сумме рангов этих матриц, но может быть и меньше.

$A^*A'$

ans =

$$\begin{pmatrix} a_{11}^2 + a_{21}^2 & a_{11}a_{12} + a_{21}a_{22} & a_{11}a_{13} + a_{21}a_{23} \\ a_{11}a_{12} + a_{21}a_{22} & a_{12}^2 + a_{22}^2 & a_{12}a_{13} + a_{22}a_{23} \\ a_{11}a_{13} + a_{21}a_{23} & a_{12}a_{13} + a_{22}a_{23} & a_{13}^2 + a_{23}^2 \end{pmatrix}$$

$a1*a1' + a2*a2'$

ans =

$$\begin{pmatrix} a_{11}^2 + a_{21}^2 & a_{11}a_{12} + a_{21}a_{22} & a_{11}a_{13} + a_{21}a_{23} \\ a_{11}a_{12} + a_{21}a_{22} & a_{12}^2 + a_{22}^2 & a_{12}a_{13} + a_{22}a_{23} \\ a_{11}a_{13} + a_{21}a_{23} & a_{12}a_{13} + a_{22}a_{23} & a_{13}^2 + a_{23}^2 \end{pmatrix}$$

### III) Первая факторизация. Спектральное разложение матрицы:

Ищем такие вектора, которые не меняют направление при действии на них квадратной матрицей размером  $n \times n$

$$A \vec{p}_i = \lambda_i \vec{p}_i \quad (1)$$

$\lambda_i$  - собственные значения,  $\vec{p}_i$  - собственные вектора. Если матрица несингулярна, то СВ линейно независимы и все СЗ различны. Если матрица  $A$  - положительно определена, то все СЗ больше нуля, если матрица симметрична - все СВ ортогональны.  $P = [\vec{p}_1 \dots \vec{p}_n]$  - матрица собственных векторов

$$AP = P[\begin{array}{ccc} \lambda_1 & \dots & 0 \\ & \ddots & \\ 0 & \dots & \lambda_n \end{array}] = P\Lambda$$

$$A = P\Lambda P^{-1} \text{ (если } A \text{ - симметрична, то } P^{-1} = P^T \text{)}$$

clearvars

% нажмите кнопку, чтобы построить вектора  
% можно посмотреть, что будет, если матрица диагональна (столбцы максимально независимы друг от друга)  
% А что если матрица сингулярна? (кстати, что это...)

$$A = [0.221, 0.699; \dots, 0.5, 0.721]$$

$$A = 2 \times 2$$

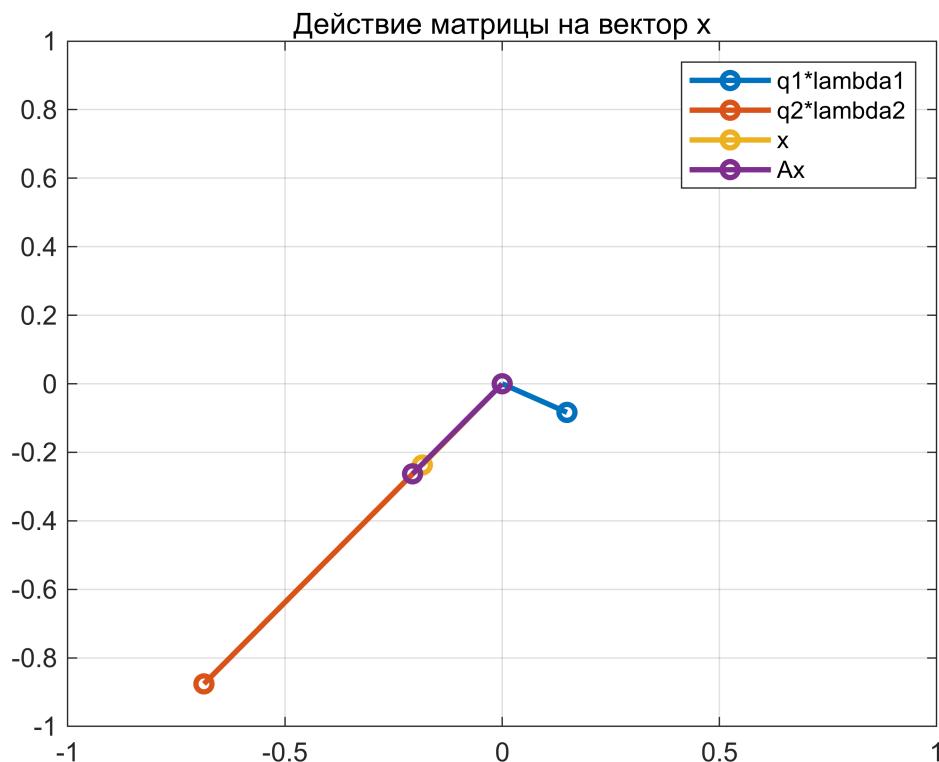
0.2210	0.6990
0.5000	0.7210

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

```
Q = 2x2
-0.8723 -0.6169
 0.4890 -0.7871
L = 2x1
-0.1709
 1.1129
```

```
R = [-127.787;0.3]; % это вектор (в полярных координатах), на который будет
% действовать матрица A
% можно покрутить вектор, чтобы убедиться, что если он совпадает по направлению
% с собственным вектором, действие на него матрицы не меняет его
% направление
ang = deg2rad(R(1)); %угол вектора
module = R(2);% модуль вектора
x = [module*cos(ang);module*sin(ang)];
q1 = Q(:,1)*L(1);q2 = Q(:,2)*L(2);
draw_vector([], 'Действие матрицы на вектор x', ["q1*lambda1" "q2*lambda2" "x"
"Ax"], "vector", q1,q2, x,A*x);
```

fig5



```
disp("угол между векторами A*x и x : "+ rad2deg(acos(x'*A*x/(norm(A*x)*norm(x)))) +
" deg")
```

угол между векторами A\*x и x : 0.34646 deg

```
disp("Q'*Q = ")
```

$Q^T Q =$

```
disp(transpose(Q)*Q)
```

$$\begin{matrix} 1.0000 & 0.1532 \\ 0.1532 & 1.0000 \end{matrix}$$

Спектральное разложение, незаменимо, когда нужны функции над матрицами:

$$A^2 = [P \Lambda P^{-1}] [P \Lambda P^{-1}] = P \begin{bmatrix} \lambda_1^2 & & \\ & \ddots & \\ 0 & \dots & \lambda_n^2 \end{bmatrix} P^{-1}$$

Аналогично в общем виде:

$$\Psi(A) = P \begin{bmatrix} \Psi(\lambda_1) & & 0 \\ & \ddots & \\ 0 & \dots & \Psi(\lambda_n) \end{bmatrix} P^{-1}$$

$\Psi$  - функция от матрицы, представимая в виде степенных рядов (голоморфная?)

#### IV) Вторая факторизация. Сингулярное разложение матрицы:

Ищем такой набор взаимоортогональных векторов  $v_i$  единичной длины, что действие на них матрицей  $A$  переводит их в другой набор взаимоортогональных векторов  $u_i$ :

$$A \vec{v}_i = \sigma_i \vec{u}_i$$

Или, в матричной форме:

$$AV = U\Sigma$$

Где  $V = [\vec{v}_1 \dots \vec{v}_n]$ ,  $U = [\vec{u}_1 \dots \vec{u}_m]$  - матрицы векторов

$$A = U\Sigma V^T \tag{2}$$

$[n \times m] = [n \times m] \cdot [n \times m] \cdot [m \times n]$  - размерность в выражении (2)

В (2)  $U$  и  $V$  - ортонормированные матрицы сингулярных векторов,  $\Sigma$  - матрица сингулярных значений, все сингулярные значения больше нуля и отсортированы в порядке убывания

Для ортонормированных матриц:  $V^T V = I$ , где  $I$  - единичная матрица, то есть транспонированная ортонормированная матрица является обратной к самой себе  $V^T = V^{-1}$ .

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ 0 & \ddots & 0 \\ \vdots & \dots & \sigma_m \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

$A^T \cdot A$ , а  $U$  и  $V$  - матрицы собственных векторов матриц  $A^T A$  и  $AA^T$  соответственно.

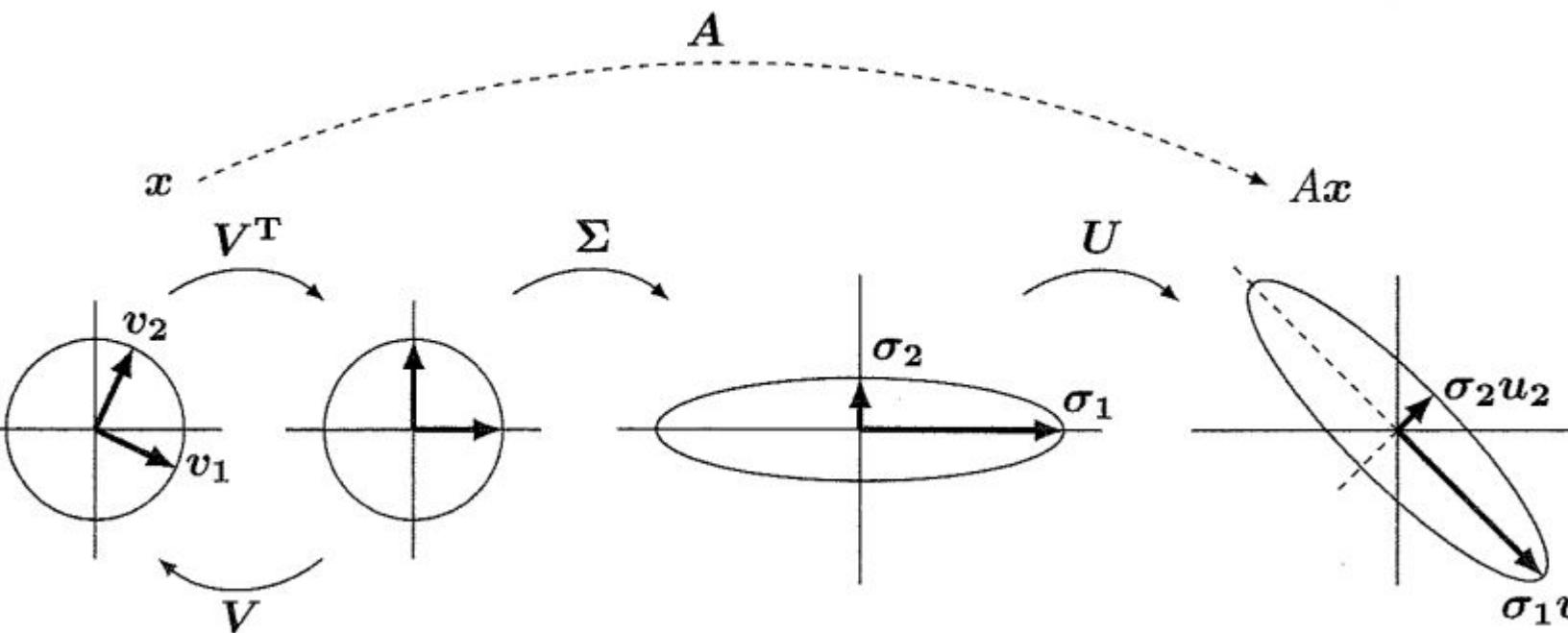
$$\sigma_1 \dots 0$$

Так как недиагональная часть матрицы  $\Sigma$  заполнена нулями,  $\Sigma$  можно записать как  $\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_m & \\ 0 & \dots & 0 \end{bmatrix}$

Важно, что сингулярные значения вдоль диагонали располагаются в порядке убывания

Геометрический смысл компонентов сингулярного разложения на примере двумерных матриц.

Картинка для пристального вглядывания:



(G.Strang. Linear algebra and learning from data. MIT (2019))

Таким образом. Ортонормированные матрицы  $U$  и  $V$  - это операции поворота. Диагональная матрица сингулярных значений характеризует масштабирование.

Чем больше разница между сингулярными значениями, тем более вытянут эллипс

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

, где  $\vec{u}_i$  и  $\vec{v}_i$  - вектор-столбцы матриц  $U$  и  $V$ .

Нужно обратить внимание, что  $\vec{u}_i \vec{v}_i^T$  - это внешнее произведение, то есть матрица (рангом один) с числом строк равным числу элементов первого вектора и числом колонок равным числу элементов второго вектора. То есть, сингулярное разложение представляет матрицу как сумму матриц того же размера, при этом "вклад" каждой из этих матриц пропорционален сингулярному значению.

Похоже на сумму операторов проецирования, хм...

Сингулярное разложение дает два ортонормированных базиса:

$U$  - базис пространства строк

$V$  - базис пространства столбцов

**Таким образом, сингулярное разложение, дает не только спектр, который характеризует степень сингулярности матрицы, но также и два ортонормированных базиса  $U$  и  $V$  первый - в пространстве столбцов, а второй - в пространстве строк**

### Сингулярное разложение матриц в matlab

```
% SVD разложение матрицы  
clearvars  
A = rand([5,3]) % матрица не квадратная, но для SVD это нормально!
```

```
A = 5x3  
0.1576 0.1419 0.6557  
0.9706 0.4218 0.0357  
0.9572 0.9157 0.8491  
0.4854 0.7922 0.9340  
0.8003 0.9595 0.6787
```

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

```
U = 5x5  
-0.2012 -0.4599 0.7532 -0.1948 0.3778  
-0.3163 0.7731 0.3895 0.3465 0.1748  
-0.5913 0.0357 0.1104 -0.3933 -0.6945  
-0.4768 -0.4311 -0.1654 0.7452 -0.0645  
-0.5315 0.0610 -0.4913 -0.3634 0.5834
```

```
S = 5x3  
2.6602 0 0  
0 0.8187 0  
0 0 0.2873  
0 0 0  
0 0 0
```

```
V = 3x3  
-0.5870 0.6737 0.4489  
-0.5981 0.0128 -0.8013  
-0.5456 -0.7389 0.3955
```

```
disp("U*U' = ")
```

```
U*U'=
```

```
disp(U*U') % ортонормированная матрица
```

```

1.0000 -0.0000 -0.0000 0.0000 -0.0000
-0.0000 1.0000 -0.0000 -0.0000 -0.0000
-0.0000 -0.0000 1.0000 -0.0000 -0.0000
0.0000 -0.0000 -0.0000 1.0000 0.0000
-0.0000 -0.0000 -0.0000 0.0000 1.0000

```

```
norm(U) % норма ортонормированной матрицы равна единице
```

```
ans = 1.0000
```

```
norm(A - U*S*V') % убеждаемся в правильности разложения
```

```
ans = 1.1560e-15
```

```
disp("A*v1:")
```

A\*v1:

A\*V(:,1) % действие матрицы на первый правый сингулярный вектор, дает первый левый сингулярный вектор умноженный на первое сингулярное значение

```

ans = 5x1
-0.5352
-0.8415
-1.5729
-1.2683
-1.4140

```

```
disp("u1*s1:")
```

u1\*s1:

```
U(:,1)*S(1,1)
```

```

ans = 5x1
-0.5352
-0.8415
-1.5729
-1.2683
-1.4140

```

## V) Сингулярное разложение (SVD) VS Спектральное разложение (EIG)

1. **SVD** применимо для любой матрицы, **EIG** только для квадратной
2. Сингулярные значения ( $\sigma_1 \dots \sigma_n$ ) всегда положительны и действительны, собственные значения ( $\lambda_1 \dots \lambda_n$ ) положительны только для положительно определенной матрицы, в общем случае, даже действительной матрицы могут "уходить" в комплексное пространство
3. Сингулярные векторы  $\vec{u}_1 \dots \vec{u}_n$  и  $\vec{v}_1 \dots \vec{v}_n$  ортогональны и нормированы на единицу (то есть  $\vec{u}_i \cdot \vec{u}_i = \vec{u}_i^T \vec{u}_i = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ ), то есть матрицы  $U$  и  $V$  - ортонормированы. Собственные вектора  $\vec{p}_1 \dots \vec{p}_n$  - не нормируются и ортогональны только для симметричной матрицы

4. Сингулярные значения ( $\sigma_1 \dots \sigma_n$ ) отсортированы в порядке убывания, собственные значения ( $\lambda_1 \dots \lambda_n$ ) не отсортированы
5. Для симметричной положительно определенной матрицы (например, такой как  $C = AA^T$ ), сингулярные значения равны квадратам соответствующих собственных значений, левые и правые сингулярные вектора совпадают  $V = U$ , сингулярные вектора совпадают по направлению с собственными векторами

Что быстрее SVD или EIG?

```
svd_test = @()svd(rand(1000));
eig_test = @()eig(rand(1000));
disp("svd_test:")
```

svd\_test:

```
timeit(svd_test,3)
```

ans = 0.3852

```
disp("eig_test:")
```

eig\_test:

```
timeit(eig_test,3)
```

ans = 0.9963

```
disp("SVD быстрее")
```

SVD быстрее

Действие матрицы на правый сингулярный вектор:

$$Av_1 = U\Sigma V^T \vec{v}_1 = \sum_{i=1}^m [\sigma_i \vec{u}_i \vec{v}_i^T] \vec{v}_1 = \sum_{i=1}^m [\sigma_i \vec{u}_i \vec{v}_i^T \vec{v}_1] = \sigma_1 \vec{u}_1$$

Так как:  $\vec{\sigma}_i \vec{u}_i \vec{v}_i^T \vec{v}_1 = \begin{cases} 1 & i = 1 \\ 0 & i \neq 1 \end{cases}$  вследствие ортогональности (и ортонормированности) сингулярных векторов.

```
clearvars
```

```
%M2x2 = 0.5 - rand(2)
M2x2 = [0.786, 0.417;
          0.67, 0.183]
```

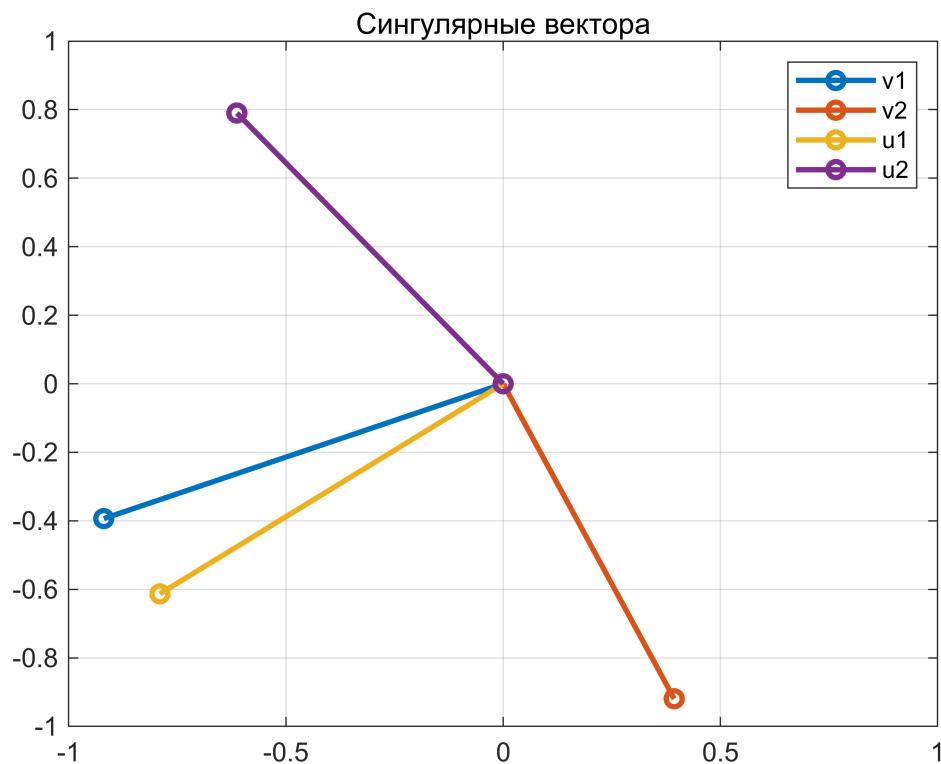
```
M2x2 = 2x2
      0.7860    0.4170
      0.6700    0.1830
```

```
[U,S,V] = svd(M2x2)
```

```
U = 2x2
-0.7901 -0.6130
-0.6130 0.7901
S = 2x2
1.1223 0
0 0.1208
V = 2x2
-0.9193 0.3935
-0.3935 -0.9193
```

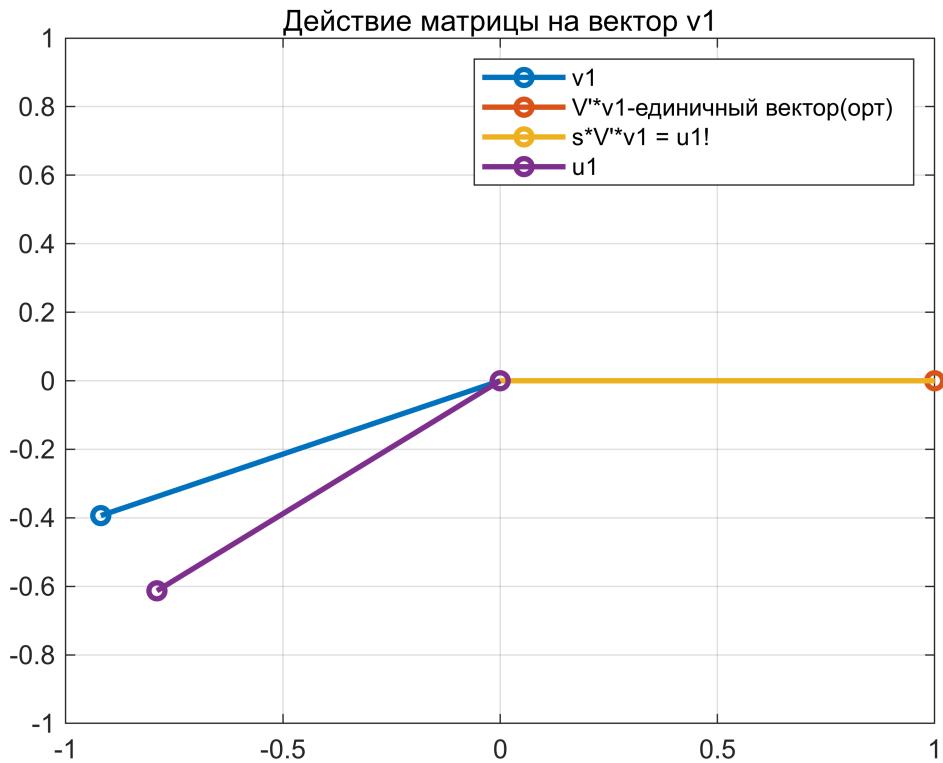
```
v1 = V(:,1); v2 = V(:,2);
u1 = U(:,1); u2 = U(:,2);
draw_vector([], 'Сингулярные вектора', ["v1" "v2" "u1" "u2"], "vector", v1, v2, u1, u2);
```

fig6



```
draw_vector([], 'Действие матрицы на вектор v1', ...
    ["v1" "V'*v1-единичный вектор(орт)" "s*V'*v1 = u1!" "u1"], "vector", v1, V'*v1,
    S(1,1)*V'*v1, u1);
```

fig7



```
disp("|A*v1|/|u1| = " + norm(M2x2*v1)/norm(u1))
```

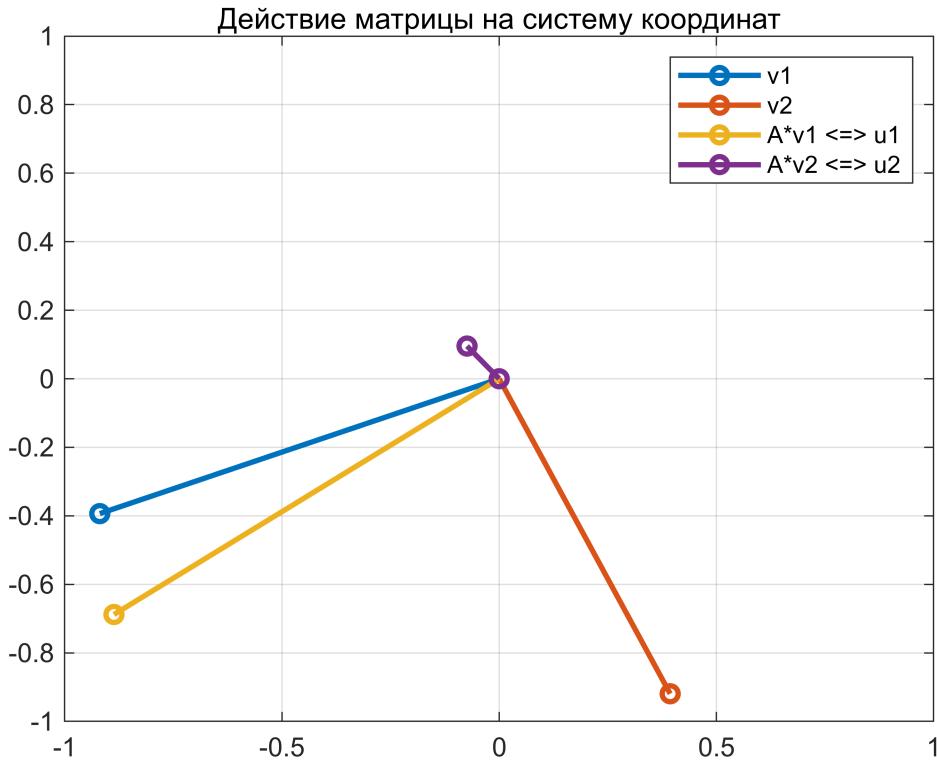
```
|A*v1|/|u1| = 1.1223
```

```
disp("S(1,1) = " + S(1,1))
```

```
S(1,1) = 1.1223
```

```
draw_vector([], "Действие матрицы на систему координат" , ...
    ["v1" "v2" "A*v1 <=> u1" "A*v2 <=> u2" ], ...
    "vector", v1, v2, M2x2*v1, M2x2*v2);
```

fig8



```
disp( "Вектор-столбцы матрицы V ортогональны: v1'*v2 = " + v1'*v2)
```

Вектор-столбцы матрицы V ортогональны:  $v1'*v2 = 0$

```
disp( "Вектора (A*v1) и A*v2 тоже ортогональны: (A*v1)'*A*v2 = v1'*A'A*v2 = "+ transpose(M2x2*v1)*M2x2*v2)
```

Вектора  $(A*v1)$  и  $A*v2$  тоже ортогональны:  $(A*v1)'*A*v2 = v1'*A'A*v2 = -5.5511e-17$

Действие матрицы на некоторый произвольный вектор:

Если вектор  $\vec{x} \in \text{span}\{V\}$ , то  $\vec{x} = P_{v_1} \vec{x} + \dots + P_{v_m} \vec{x} = \vec{v}_1(v_1^T \vec{x}) + \dots + \vec{v}_m(v_m^T \vec{x}) = x_1 \vec{v}_1 + \dots + x_m \vec{v}_m$  ( $P_{v_i}$  - операторы проецирования)

$$A \vec{x} = U \Sigma V^T \vec{x} = \sum_{i=1}^m [\sigma_i \vec{u}_i \vec{v}_i^T] [\sum_{i=1}^m [x_1 \vec{v}_1 + \dots + x_m \vec{v}_m]] = \sum_{i=1}^m [\sigma_i x_i \vec{u}_i]$$

Так как:  $\vec{u}_i \vec{v}_i^T \vec{v}_1 = \begin{cases} 1 & : i = 1 \\ 0 & : i \neq 1 \end{cases}$  вследствие ортогональности (и ортонормированности) сингулярных векторов.

```
R =[75.957;1]
```

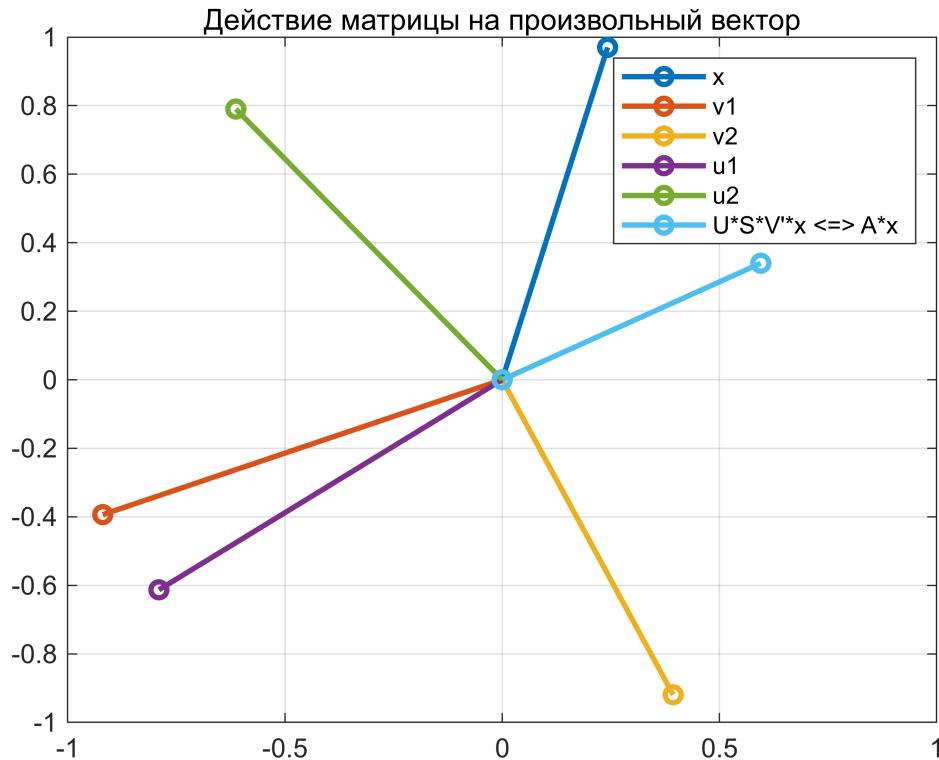
```
R = 2×1
75.9570
1.0000
```

```

ang = deg2rad(R(1)); %угол вектора
module = R(2);% модуль вектора
x = [module*cos(ang);module*sin(ang)];
%draw_vector("Действие матрицы на произвольный вектор",["x" "v1" "v2" "u1" "u2"
"V'*x" "S*V'*x" "U*S*V'*x <=> A*x"],x,v1,v2,u1,u2,v'*x, s*v'*x,u*s*v'*x);
draw_vector([], "Действие матрицы на произвольный вектор",["x" "v1" "v2" "u1" "u2"
...
"U*S*V'*x <=> A*x"], "vector",x,v1,v2,u1,u2,U*S*V'*x);

```

fig9



```

cos_alfa = transpose(U*S*V'*x)*x;
cos_alfa = cos_alfa/(norm(U*S*V'*x)*norm(x));
disp("Скалярное произведение acos(alfa) = acos((x'*Ax)/(|x|*|Ax|)) = " +
rad2deg(acos(cos_alfa)))

```

Скалярное произведение  $\text{acos}(\alpha) = \text{acos}((\mathbf{x}' \cdot \mathbf{Ax}) / (\|\mathbf{x}\| \cdot \|\mathbf{Ax}\|)) = 46.2152$

```

clearvars
% матрица диагональная
A = diag(rand(5,1))

```

```

A = 5x5
0.6302      0      0      0      0
    0   0.0953      0      0      0
    0      0   0.6620      0      0
    0      0      0   0.6232      0
    0      0      0      0   0.6833

```

```
svds(A)
```

```
ans = 5x1
0.6833
0.6620
0.6302
0.6232
0.0953
```

```
% матрица с большой асимметрией
```

```
A = eye(5);
A(5) = 1e6
```

```
A = 5x5
```

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
1000000	0	0	0	1

```
S = svds(A)
```

```
S = 5x1
10^6 *
1.0000
0.0000
0.0000
0.0000
0.0000
```

```
% Последнее сингулярное значение близко к нулю, почему так?
```

## 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 определенное значение из этого дискретного набора характеризуется набором вероятностей:

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

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

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

Среднее значение результатов набора из  $M$  испытаний:

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

Разброс данных характеризуется вариацией:

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

Для экспериментальной оценки вариации:

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

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

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

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

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

$p_{ik}$  - совместная вероятность

**Матрица ковариации:**

$\vec{X}_1 \dots \vec{X}_m \dots \vec{X}_M$  - вектора результатов испытания (определения  $N$  свойств) "образца" размером  $N \times 1$  каждый (вектора состояния некоторого случайного процесса).

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

$\vec{\mu} = \mu_1 \dots \mu_n \dots \mu_N$  - 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  $[N \times M]$ :

$A = [\vec{x}_1 - \vec{\mu}, \dots, \vec{x}_m - \vec{\mu}, \dots, \vec{x}_N - \vec{\mu}]$  - 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  $N \times N$ ):

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

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

C =

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

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 ( $n = 1 \dots N$ ), 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:

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

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:

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

(4)

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

произведения различных столбцов матрицы  $B$ :  $C_{nk} = \frac{\overrightarrow{b_n} \cdot \overrightarrow{b_k}}{M - 1}$  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 $\Leftrightarrow$ 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 векторы матрицы испытаний есть нормированные собственные векторы 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 = 2x500  
-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 свойств
```

```
B = 500x2  
-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 приклейны 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 склеенных монет: ")

```

Матрица испытаний для склеенных монет:

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

```

Acor = 2x500
 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 = 2x2
 0.6940   -0.7200
 0.7200    0.6940
S = 2x500
 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 = 2x2
 -0.7200    0.6940
 0.6940    0.7200
L = 2x2
 123.0578      0
      0   126.4922

```

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

Сингулярное разложение матрицы ковариации для склеенных монеток

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

U = 2x2

-0.7071	0.7071
-0.7071	-0.7071

S = 2x500

15.8102	0	0	0	0	0	0	0	...
0	0.0000	0	0	0	0	0	0	

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

Спектральное разложение матрицы ковариации для склеенных монеток

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

Q = 2x2

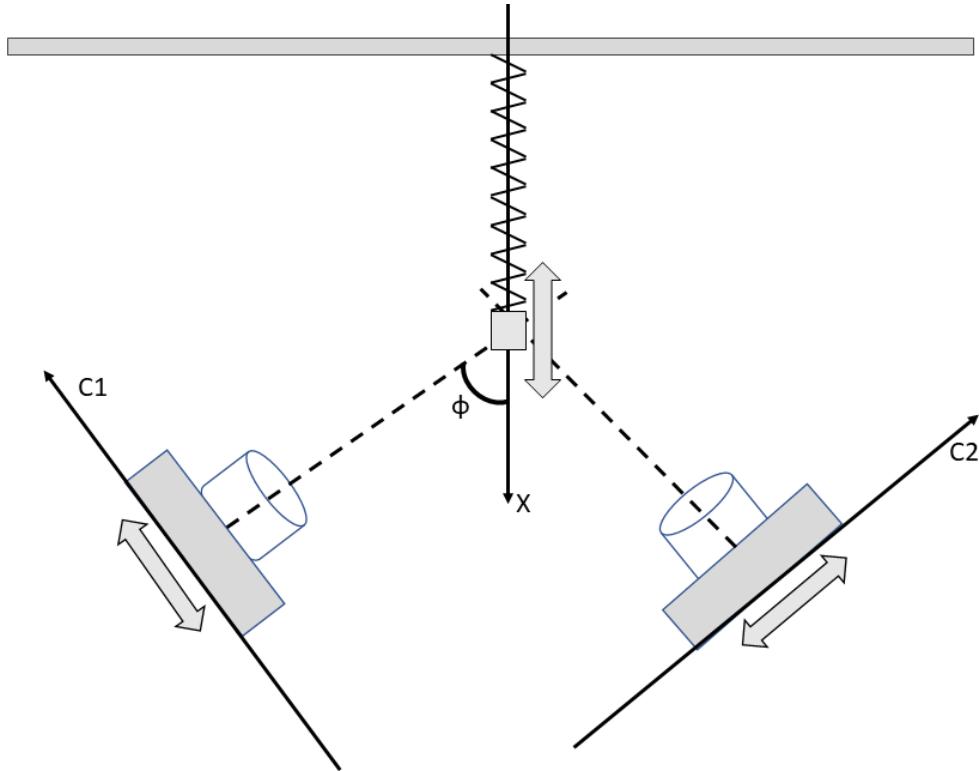
-0.7071	0.7071
0.7071	0.7071

L = 2x2

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 = 1x100
-3.1416 -3.0781 -3.0147 -2.9512 -2.8877 -2.8243 -2.7608 -2.6973 ...
```

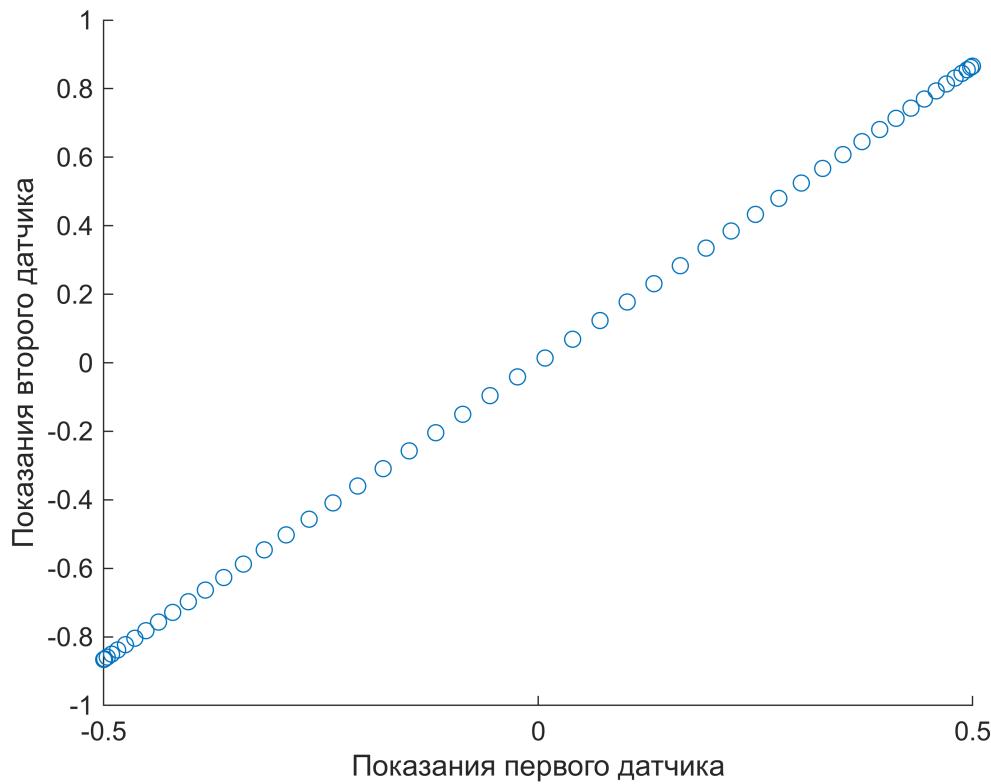
```
X = cos(t) % закон движения грузика на пружине
```

```
X = 1x100
-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 = 2x100
-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 = 2x2
0.8660 0.5000
0.5000 -0.8660
S = 2x100
```

```

7.1063      0      0      0      0      0      0      0      0 ...
0      0.0000      0      0      0      0      0      0      0
U = 100x100
-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 = 2x2
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 угол между поодпространством ее
% столбцов и вектором будет угол фи
disp("Матрица ковариации:")

```

Матрица ковариации:

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

```
C = 2x2
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 свойства  
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
```

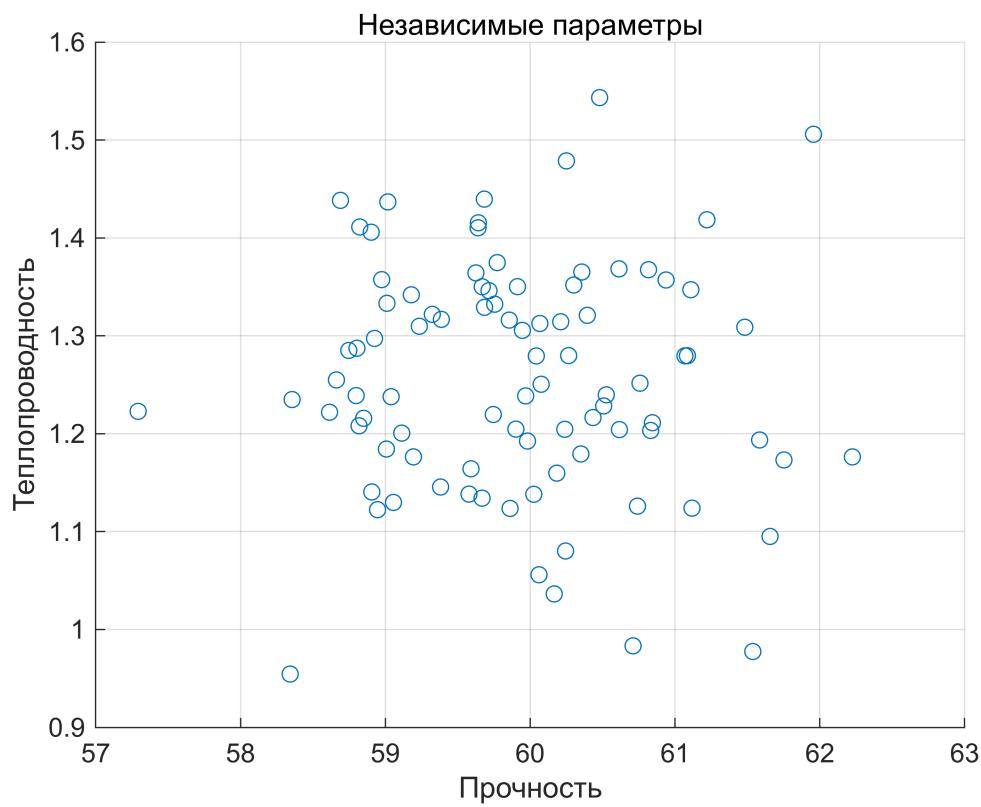
```
disp("Матрица ковариации для трех параметров {X=rand,rand,Y="+alfa_cor+"*X +(1-  
alfa_cor)*rand}")
```

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

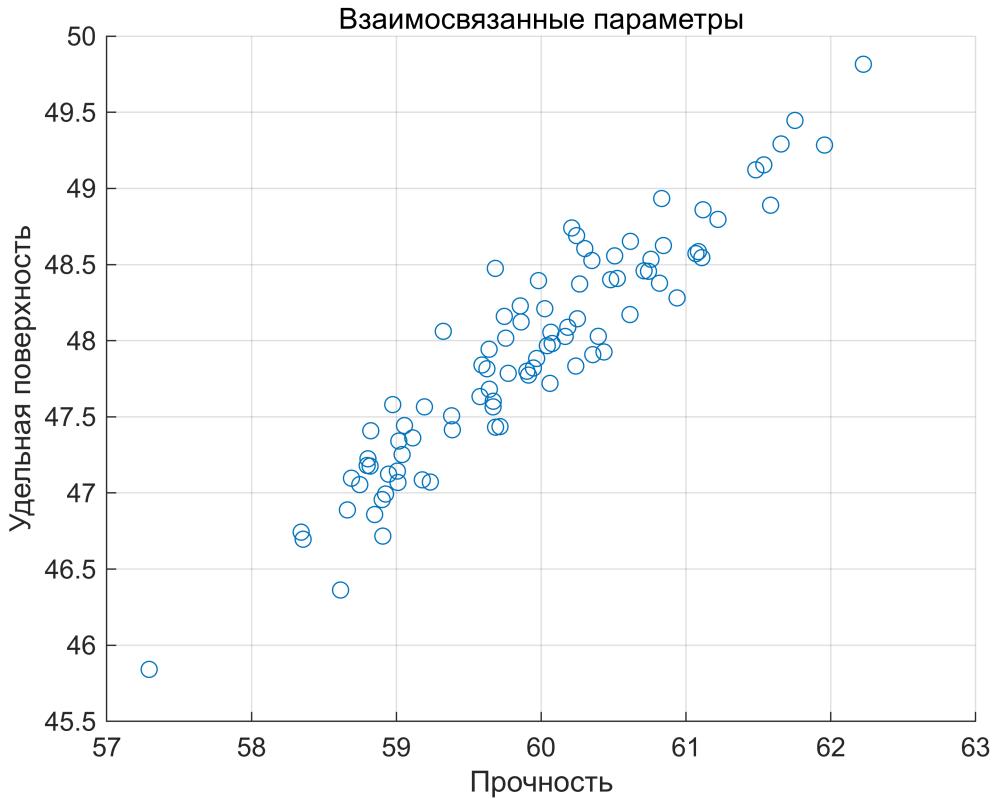
```
disp(covMAT3)
```

```
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 = 2x2
 0.6170 -0.7869
 -0.7869 -0.6170
L = 2x1
 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 = 2x2
 0.7869  0.6170
 0.6170 -0.7869
S = 2x90
 11.0612      0      0      0      0      0      0      0     ...
      0  1.8517      0      0      0      0      0      0
V = 90x90
 -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 = 90x2
 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 = 2x1
 -0.0000
  0.6430

```

```

a_lsqr = [cos(lsqr_fit(2));sin(lsqr_fit(2))]

```

```

a_lsqr = 2x1
 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_lsqr)

```

```

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'

```

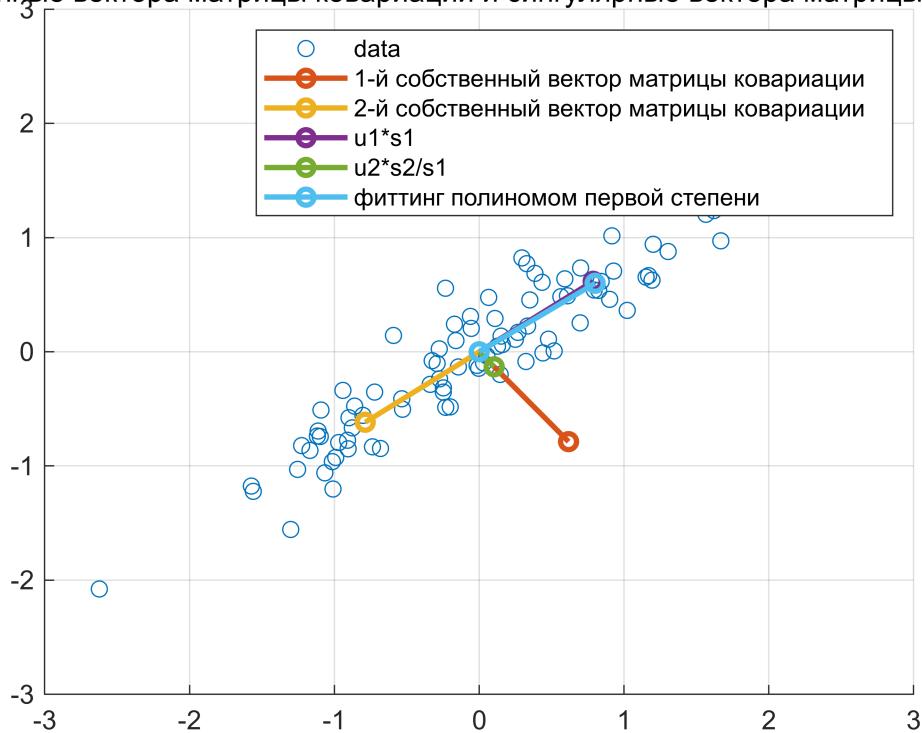
Show all properties

```

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" ])

```

:обственые вектора матрицы ковариации и сингулярные вектора матрицы испыт



```

% 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 соответствует направлению, на которое сумма проекций максимальна.

Проекция столбцов матрицы  $A$  на вектор  $\vec{u}_1$ :

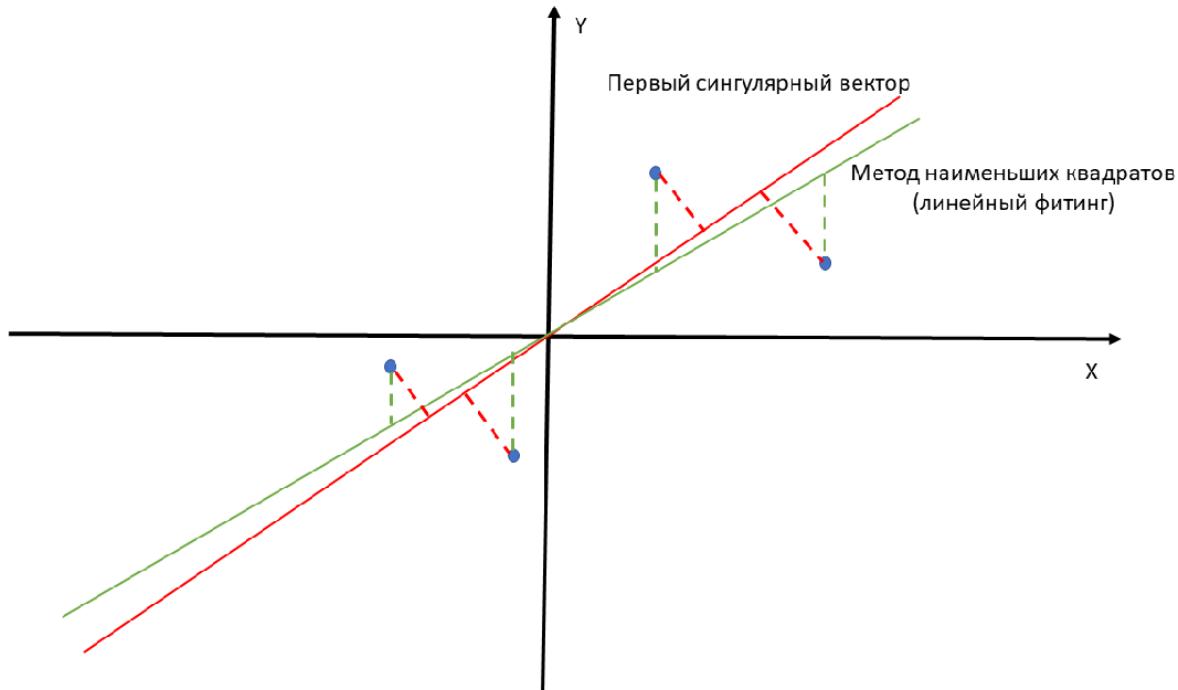
$$\vec{u}_1^T A = \vec{u}_1^T U \Sigma V^T = \vec{u}_1^T \sum_{i=1}^m [\sigma_i \vec{u}_i \vec{v}_i^T] = \sigma_1 \vec{v}_1^T$$

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 проблем экстраполировалась на пространства большей размерности.

Три ...

```
covMAT3
```

```
covMAT3 = 3x3
 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 = 3x3
 0.7858 -0.6030  0.1378
 -0.0018  0.2206  0.9754
  0.6185  0.7667 -0.1722
```

```
S = 3x90
```

```
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 = 90x90
```

```
-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
 ...

```

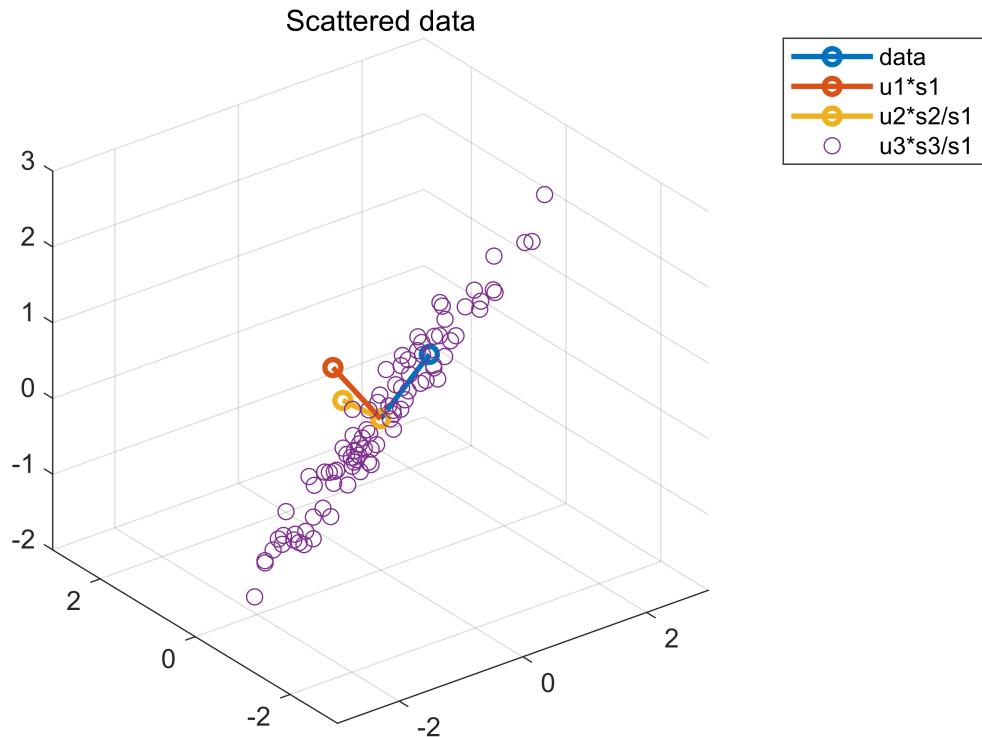
```
U(:,3)'*U(:,2)
```

```
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 = 5x5
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 = 5x1
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 Machine Learning (2018)
5. youtube: канал AMATH 301, лектор Nathan Kutz, лекции: The singular Value Decomposition and Principal Component Analysis

## BONUS с картинками!

Теорема Экхарта-Янга:

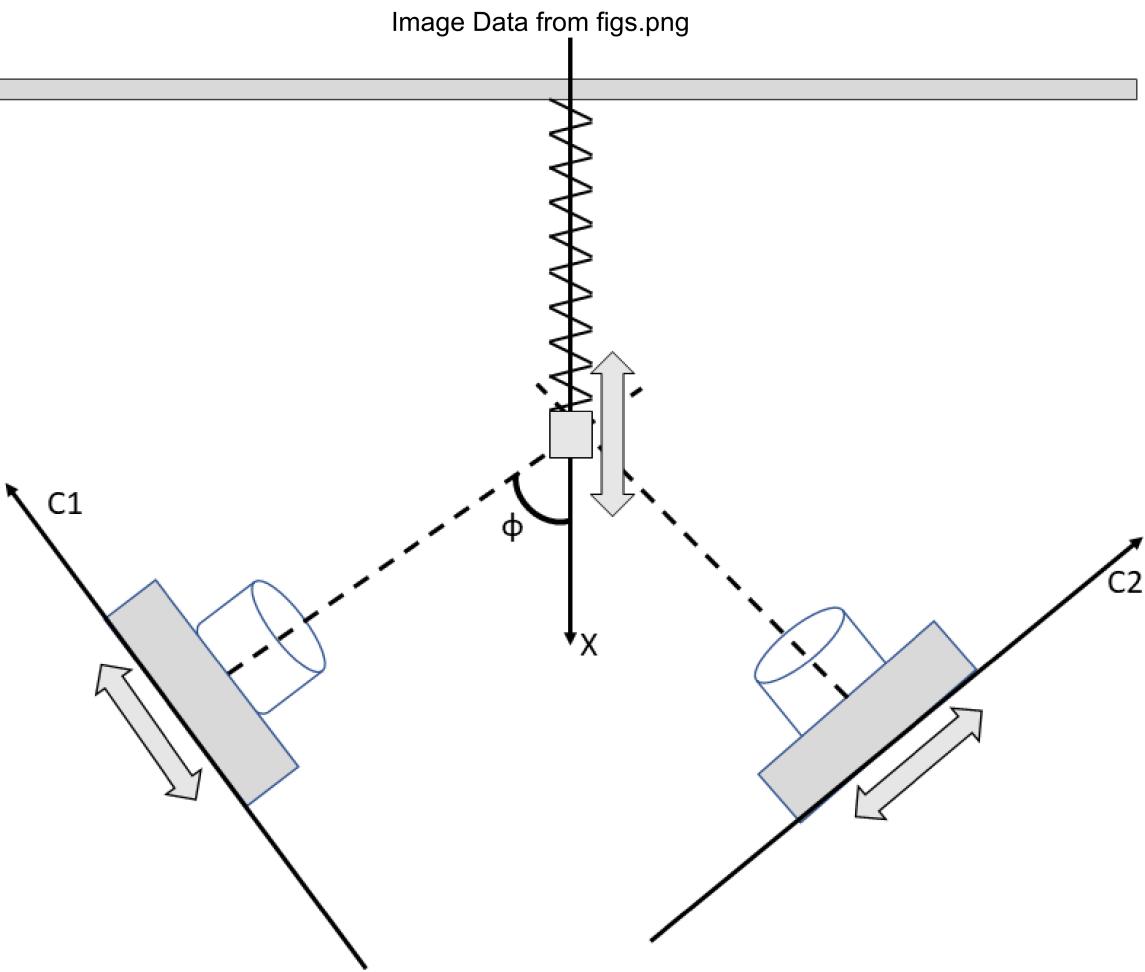
Пусть  $A_k = \sum_{i=1}^k [\sigma_i \vec{u}_i \vec{v}_i^T]$ , где  $\forall k \leq m$ , тогда для  $\forall$  матрицы  $B$  ранга  $k$ :  $\|A - B\| \geq \|A - A_k\|$ .

Иными словами, матрица  $A_k$  является наилучшей аппроксимацией матрицы  $A$  среди всех матриц ранга  $k$ .

Картинка - это же матрица!

```
clearvars
% Import image
folder = get_folder();
selected_image= "figs.png";
figs_folder = folder + "\figs";
full_file = fullfile(figs_folder,selected_image);
figs = imread(full_file);
```

```
% Display results
imshow(figs);
title("Image Data from figs.png");
```



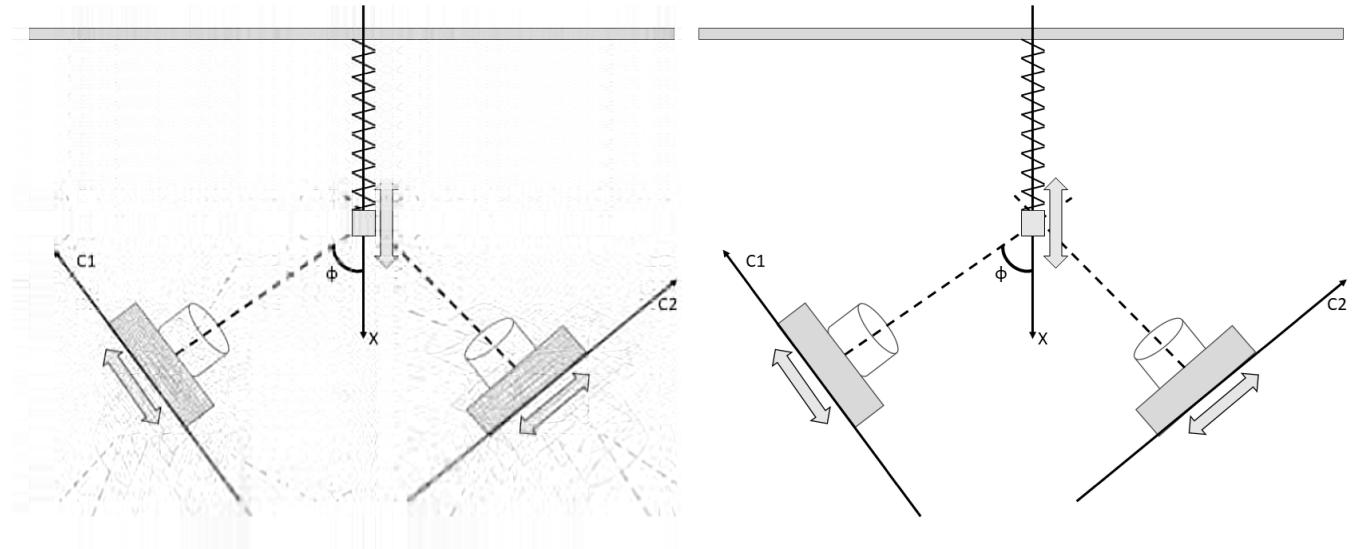
```
A = double((im2gray(figs)));
[U,S,V] = svd(A); % сингулярное разложение исходной картинки
```

```
number_of_dimentions=60; % ранг матрицы Ak
if number_of_dimentions>size(U,2)
    number_of_dimentions = size(U,2);
end
% уменьшаем размерность исходной картинки
Uk = U(:,1:number_of_dimentions);
Sk = diag(S(1:number_of_dimentions,1:number_of_dimentions));
Vk = V(:,1:number_of_dimentions);
```

```
Ak = Uk.*Sk'*Vk'; % матрица уменьшенной размерности
disp("Слева-сжатая, справа - исходная:")
```

Слева-сжатая, справа - исходная:

```
imshow(uint8([Ak,A]))
```



```
display("Ранг исходной картинки:" + rank(A))
```

"Ранг исходной картинки:415"

```
display("Ранг сжатой картинки:" + rank(Ak))
```

"Ранг сжатой картинки:60"

```
wA = mem_size_summ("A"); % объем памяти в байтах, занимаемой исходной матрицей
wAk = mem_size_summ("Uk", "Sk", "Vk"); % объем памяти в байтах
disp("Объем исходной матрицы в байтах :" + wA)
```

Объем исходной матрицы в байтах :4592240

```
disp("Объем сжатой матрицы в байтах :" + wAk)
```

Объем сжатой матрицы в байтах :731520

```
disp("Относительный объем памяти сжатый/несжатый:" + wAk/wA)
```

Относительный объем памяти сжатый/несжатый:0.15929

```
ax = get_next_ax();
```

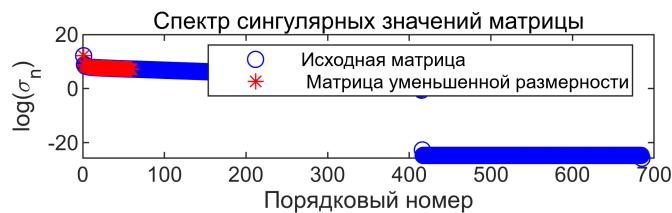
fig1

```
plot(ax, 1:size(S, 2), log(sum(S)), 'ob', 1:number_of_dimentions, log(Sk), '*r')
title(ax, "Спектр сингулярных значений матрицы");
xlabel(ax, "Порядковый номер");
```

```

ylabel(ax,"log(\sigma_n)");
legend(ax,[ "Исходная матрица" " Матрица уменьшенной размерности"]);

```



```

% загружаем картинки написанных от руки цифр и пытаемся сравнить спектры их
% сингулярных значений
check_numbers_flag = true;
if check_numbers_flag

path = get_folder + "\figs\";
svd_struct(10) = struct("V",[],"S",[],"U",[],"M",[]);
for ii=0:9
    cur_im = imread(fullfile(path,ii+".png"));
    mat = double((im2gray(cur_im)));
    i = ii +1;
    svd_struct(i).M = (mat);
    [svd_struct(i).V,svd_struct(i).S,svd_struct(i).U] = svd(mat);
end
imshow(uint8([svd_struct(:).M]))
ranks = arrayfun(@(X)rank(X.M),svd_struct)
sig_mat = ones(size(mat,1),10); % матрица спектров сингулярных значений
for iii = 1:numel(svd_struct)
    sig_mat(:,iii) = sum(svd_struct(iii).S);
end
p = plot(get_next_ax(),log(sig_mat), 'LineWidth',3,'MarkerSize',6);
ylim([-1,10])
legend(string(0:9) + " rank=" + string(ranks), 'FontSize',12);
mark = ["+", "o", "*", ".", "square", "diamond", "v", "^", "pentagram", "hexagram"];
arrayfun(@(ii)set(p(ii),"Marker",mark(ii)),1:10);
title("Спектр сингулярных значений циферок");
xlabel("Порядковый номер");
ylabel("log(\sigma_n)");
end

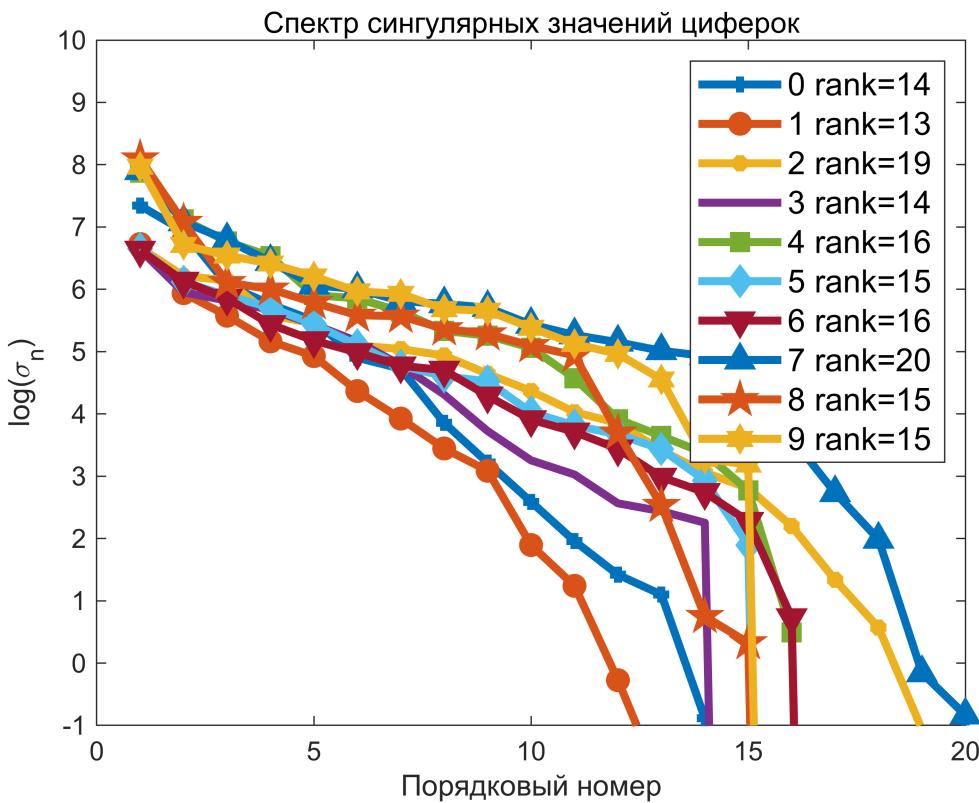
```



```

ranks = 1x10
14    13    19    14    16    15    16    20    15    15
fig2

```



```

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 д
%                 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{i});
            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{i});
            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
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 нескольких 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

```