nvkurbatova@sfedu.ru

Scientific Computer Software

(MatLab)

Topic 1. The MatLab environment.

Topic 2. Basic concepts

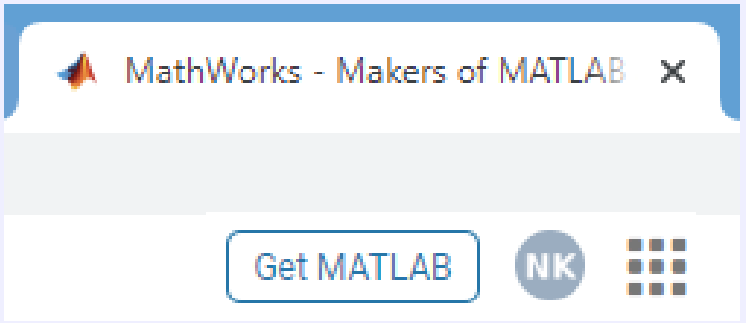
к.ф.-м. Курбатова Наталья Викторовна,
assistant professor, PhD, Kurbatova Natalia

nvkurbatova@sfedu.ru

MatLab on the site mathworks.com

You have 20 hours in the month!!!

- Log in to the site https://www.mathworks.com/
- Register on your own by mail *sfedu.ru
- Click the *Get MatLab* button



Good luck!

nvkurbatova@sfedu.ru

Table of contents

Topic 1

- Sources to study (resources)
- Multi-window interface
- ML Home and Edit Resource Guide
- Working with help interactively

Topic 2

- Basic ML functionality
- Creating script files and functions

nvkurbatova@sfedu.ru

We are learning MatLab

Useful resources:

- https://samoychiteli.ru/document21400.html
- https://exponenta.ru/academy/study_material
- Help MatLab (»*FunctionName+F1* or » *help*)
- Examples MatLab:

»help examples

or

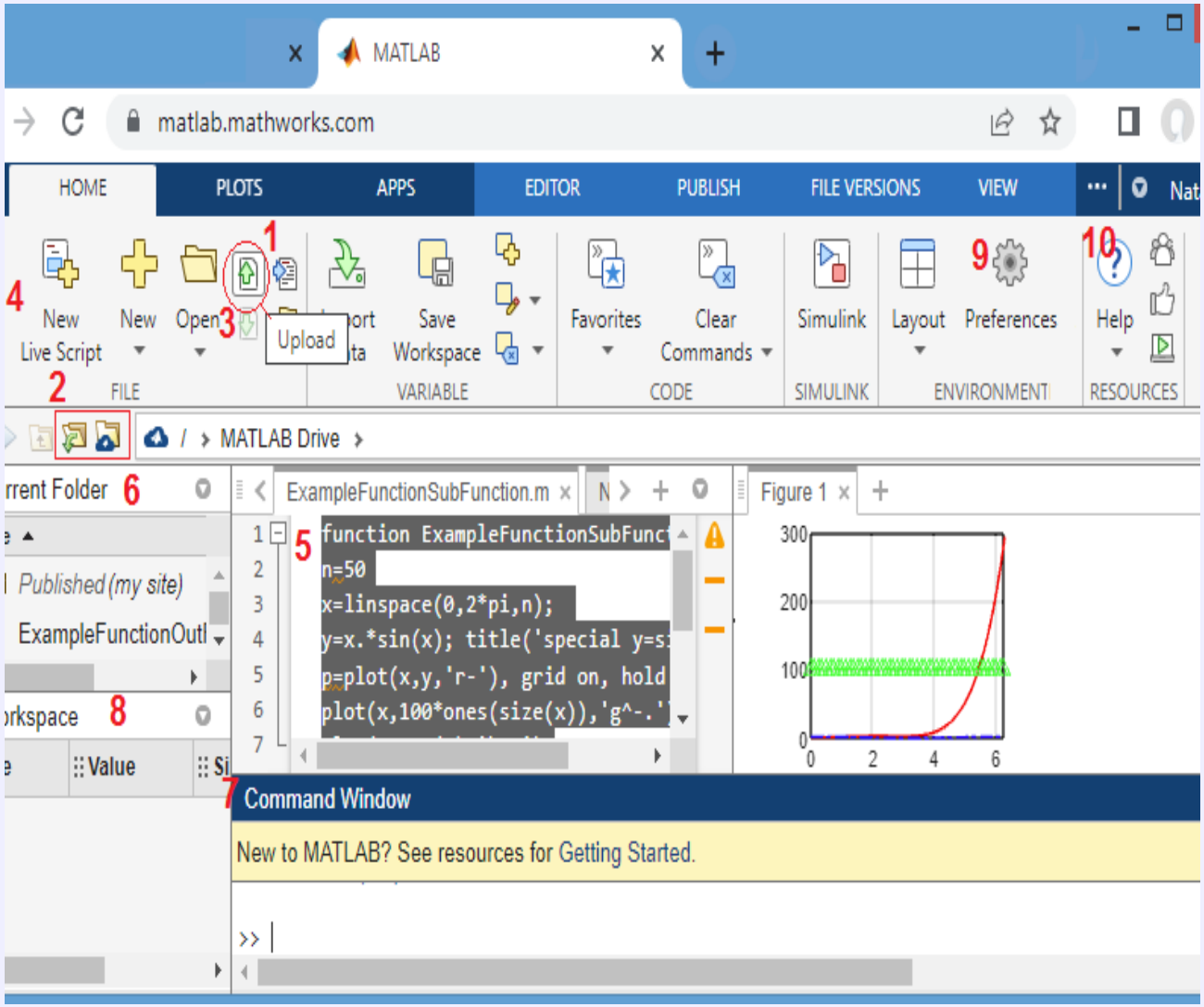
»demos)

nvkurbatova@sfedu.ru

MatLab Home

nvkurbatova@sfedu.ru

MatLab on the site https://www.mathworks.com/ (remotely)



Home

1 - downloading a file from a computer

2 - downloading a file to the cloud **MLDrive**

3 - (inactive down arrow) upload the file to the computer, after saving

4 - Creation empty Live script:*mlx ; New – empty script: *.m

5 - File editor

6 - Current Folder

7 - Command Window , (working window, I/O environment, interactive calculator)

8 - Workspace - information about variables (global)

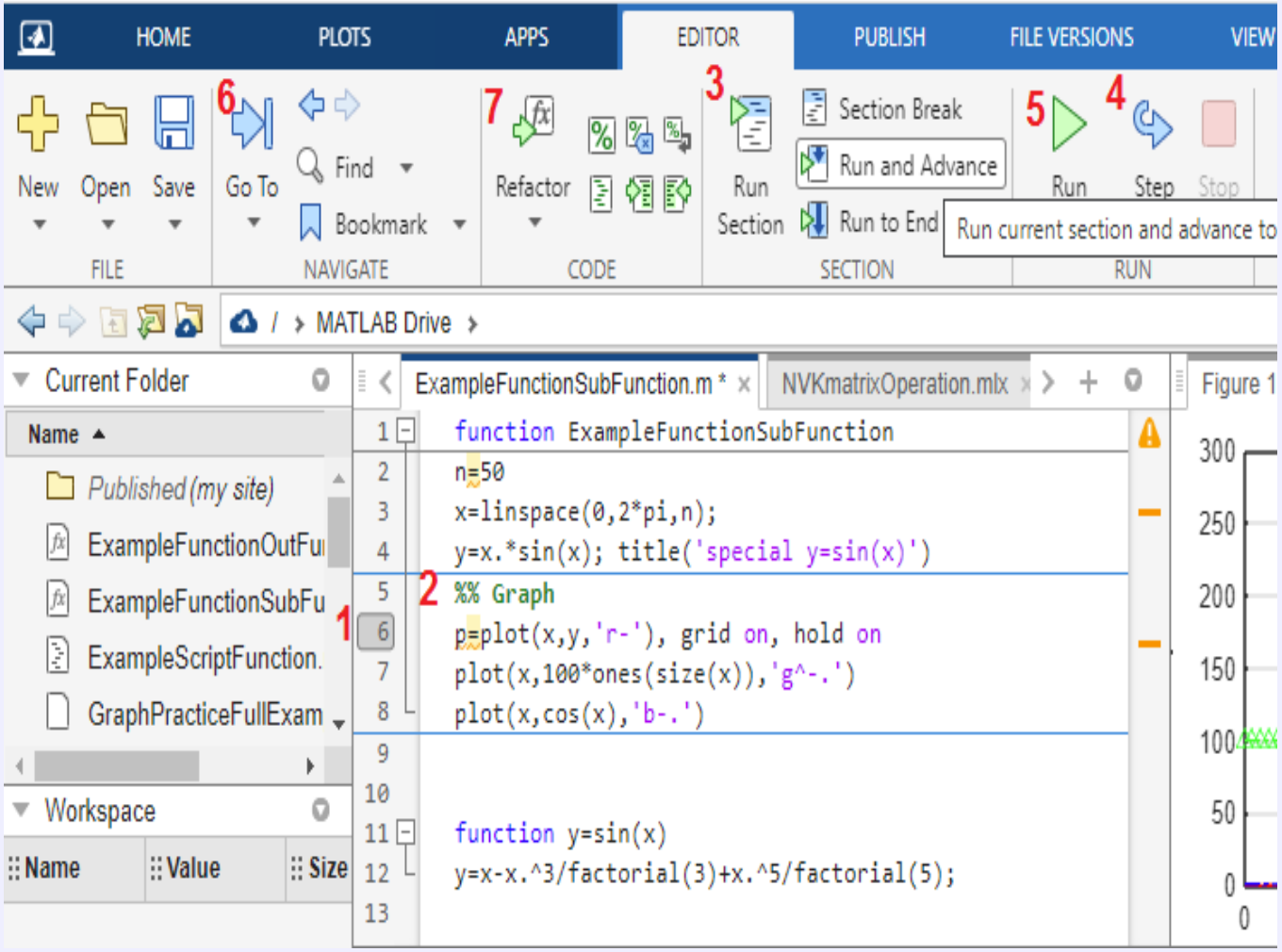
9 - Preferences – personal settings

10 - Help or ?

nvkurbatova@sfedu.ru

MatLab Editor

Working in the Editor



Editor

1 - Left–click on the line number - breakpoint

2 - %% - sign of the section beginning;
% - sign of comment

3 - execute a section

4 - execute line by line

5 - execute all file commands

6 - run everything down to the line with the cursor

7 - The selected part of the code is converted into a local function (in the same script body)

Stop - the debugger is interrupted

nvkurbatova@sfedu.ru

Local menu

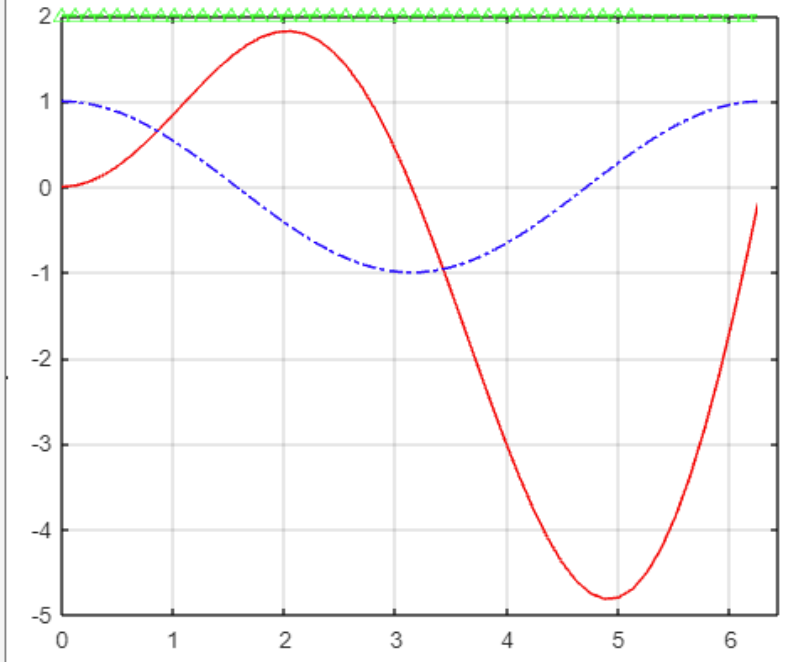
nvkurbatova@sfedu.ru

The effectiveness of the local menu

```
function ExampleFunctionSubFunction
n=50
x=linspace(0,2*pi,n);
y=x.*sin(x); title('special y=sin(x)')
%% Graph
figure
p=plot(x,y,'r-'), grid on, hold on
plot(x,2*ones(size(x)),'g
plot(x,cos(x),'b-.')


function y=sin(x)
y=x-x.^3/factorial(3)+x.^
```

1

2 Evaluate Selection in Command Window F9
Open Selection Ctrl+D
Help on Selection F1
Wrap Comments Ctrl+J
3 Comment Ctrl+R
4 Uncomment Ctrl+Shift+R
Duplicate Line(s) Ctrl+Shift+C
Change Case Ctrl+Shift+A
Smart Indent Ctrl+I
5 Convert to Function
Convert to Local Function
Section Break Ctrl+Alt+Enter
Run Section Ctrl+Enter
Split Screen



1 - select the code to be affected, right-click

2 - execute the selected

3 - comment out the highlighted

4 - uncomment the selected ones

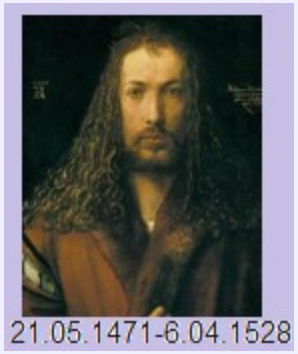
5 - convert to a local or external function

MATRIX CONSTRUCTORS

ONES – all elements are equal to one

EYE – identity matrix **ZEROS** – zero matrix

MAGIC – the magic square (the idea of Albrecht Durer)



Random Matrix Generator:

RAND – the elements are uniformly distributed over the segment [0,1]

RANDN – the elements have Normal Distribution

RANDI – integer elements on a given segment

spransym - symmetric matrix generation

The construction of matrices. Examples

nvkurbatova@sfedu.ru

NVKmatrixOperation.mlx *

Special type matrices

We can use functions to create special type matrices:

ones - to create a unit matrix

```
a0 = ones(1);  % a=ones(size), a=size(m,n)
a1 = ones(3)
```

zeros - to create a null matrix

```
b0 = zeros(2);
b1 = zeros(1,4)
```

rand - to create a random matrix, the elements of which belong the range from 0 to 1

```
c0 = rand(2) % rand(sizeMatrix)
c1 = rand(1,4)
```

randi - to create a random matrix, the elements of which belong the range from 0 to imax (the first meaning in the parenthesis)

```
d0 = randi(20, 2, 2) %X = randi([imin,imax],n,m) or X = randi(imax,n,m)
d1 = randi(10, 3)
d2 = randi(3, 2, 3)
```

eye - to create identity matrix

```
e0 = eye(2)
e1 = eye(2, 3)
```

magic - to create a Durer matrix. Matrix constructed from the integers 1 through n2 with equal row and column sums. The order n must be a scalar greater than or equal to 3 in order to create a valid magic square.

```
M = magic(5)
```

```
a1 = 3×3
     1     1     1
     1     1     1
     1     1     1

b1 = 1×4
     0     0     0     0

c0 = 2×2
    0.8258    0.9961
    0.5383    0.0782

c1 = 1×4
    0.4427    0.1067    0.9619    0.0046

d0 = 2×2
    16    18
    17     2

d1 = 3×3
     4     5     3
     3    10     2
     9     2     2

d2 = 2×3
     3     2     3
     2     1     2

e0 = 2×2
     1     0
     0     1

e1 = 2×3
     1     0     0
     0     1     0

M = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

nvkurbatova@sfedu.ru

MATFUN

Matrices function library

Examples of matrix functions

nvkurbatova@sfedu.ru

We can view all the functions for working with special matrices using the following command:

```
13  help matfun
```

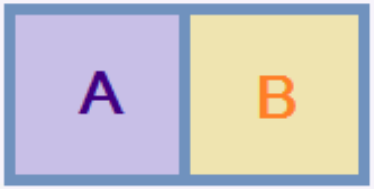
```
Matrix functions    - numerical linear algebra.

Arithmetic operators.
  mpower          - Matrix power ^

Matrix analysis.
  bandwidth       - Matrix bandwidth.
  isbanded        - Determine whether a matrix has certain bandwidth.
  isdiag          - Determine whether a matrix is diagonal.
  ishermitian     - Determine whether a matrix is Hermitian.
  issymmetric     - Determine whether a matrix is symmetric.
  istril          - Determine whether a matrix is lower triangular.
  istriu          - Determine whether a matrix is upper triangular.
  norm            - Matrix or vector norm.
  vecnorm         - Vector norm.
  normest         - Estimate the matrix 2-norm.
  rank            - Matrix rank.
```

nvkurbatova@sfedu.ru

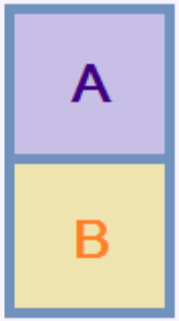
Concatenation

C=[A, B] ~ C=[A, B] ~

~ C=cat (2,A,B)

, C= 

D=[A;B], ~ D= cat (1,A,B)

; D = 

Operations on matrices

Concatenation. Examples

nvkurbatova@sfedu.ru

Let's consider some operations with matrices: concatenation

Concatenation is the process of joining arrays to make larger ones. The pair of square brackets [ ] is the concatenation operator.

```
14  f0 = [a1, zeros(size(a1))]
15  f1 = [[a0,a0]', b0]
```

Concatenating arrays next to one another using commas is called ***horizontal concatenation***. Each array must have the same number of rows.

You can concatenate *vertically* using semicolons:

```
16  f2 = [c0; d2(:,1:2)]
```

Each array must have the same number of columns.

We can also use the *cat* function to combine matrices:

```
17  M2 = [1 2 3];
18  M3 = [5 6 7; 8 9 10];
19  cat(1, M2, M3)  % along the first dimension
20  M4 = [1 2; 3 4];
21  M5 = [5 6; 8 9];
22  cat(2, M4, M5)  % along the second dimension
```

f0 = 3×6

	1	2	3	4	5	6
1	1	1	1	0	0	0
2	1	1	1	0	0	0
3	1	1	1	0	0	0

f1 = 2×3

```
     1     0     0
     1     0     0
```

f2 = 4×2

```
    0.5830    0.2904
    0.2518    0.6171
    2.0000    2.0000
    1.0000    2.0000
```

ans = 3×3

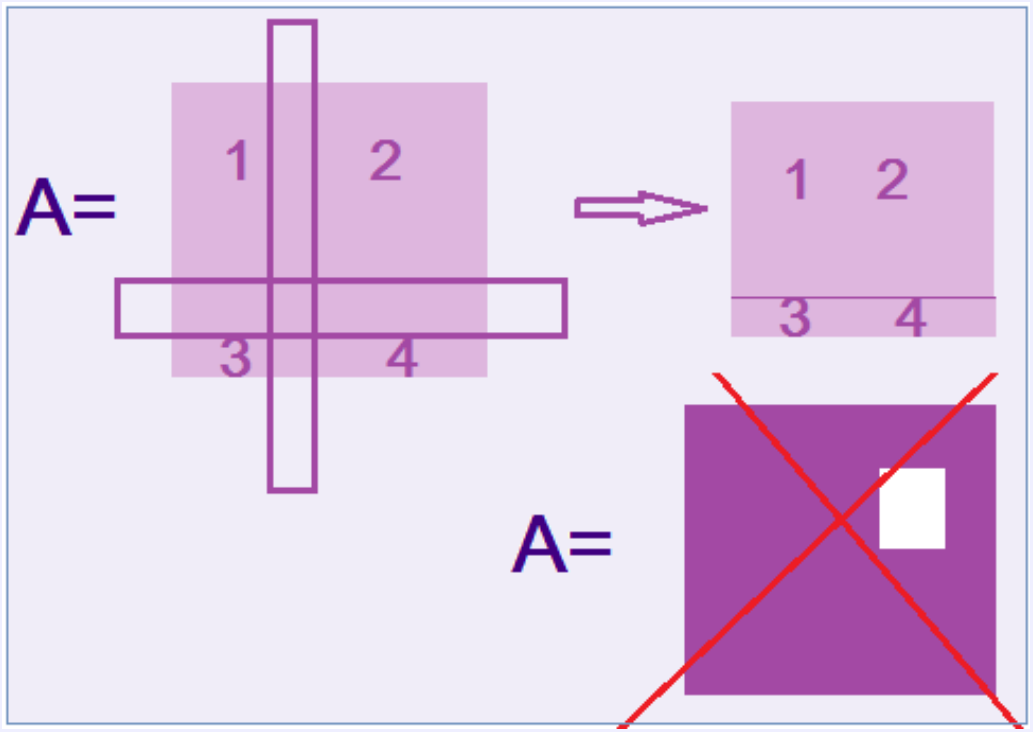
```
     1     2     3
     5     6     7
     8     9    10
```

ans = 2×4

```
     1     2     5     6
     3     4     8     9
```

nvkurbatova@sfedu.ru

Removing submatrices



Operations on matrices

nvkurbatova@sfedu.ru

Operation with submatrices. Examples

Removing submatrices and their elements:

We can delete either one specific element of the matrix, or several at once. We can even delete one or more columns (rows) and their individual sections at once.

What we can't delete?

```
23  M1(1, 1) = 0
24  M1(2:4, 2:3) = 0
25  M1(:, 2) = []
26  M1(2:end) = []
```

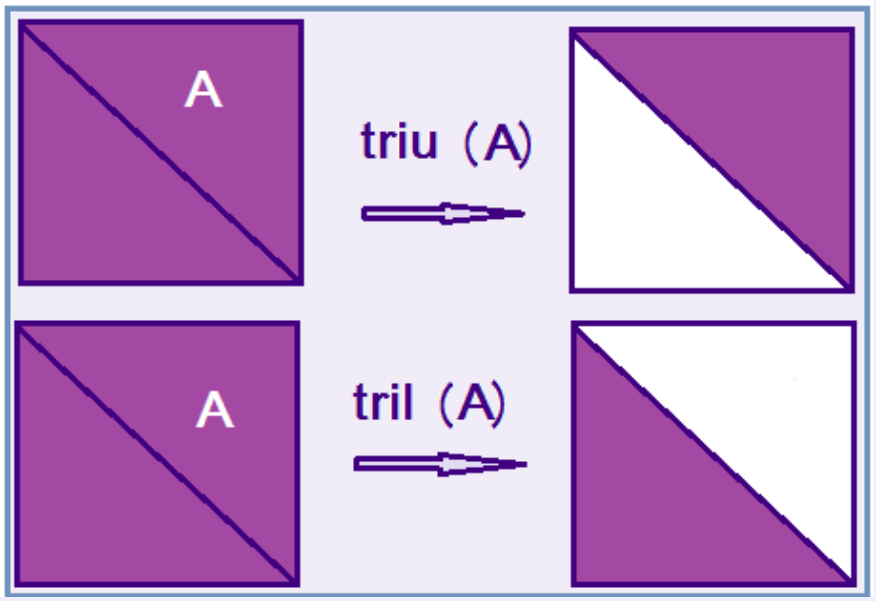
```
M1 = 0

M1 = 4×3
     0     0     0
     0     0     0
     0     0     0
     0     0     0

M1 = 4×2
     0     0
     0     0
     0     0
     0     0

M1 = 0
```

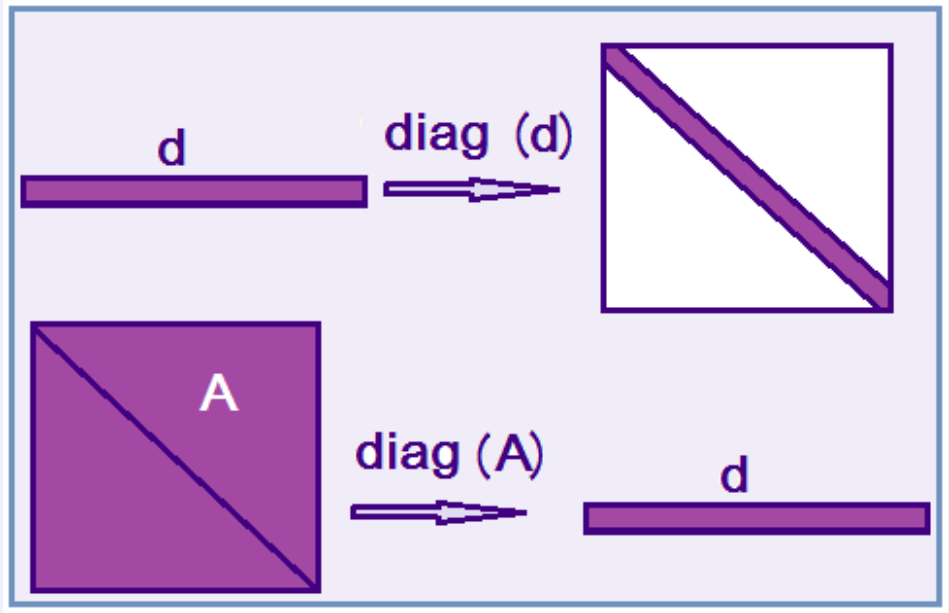
Is that right? A (1:2:end) =[]

And so? A (end:-2:1) =[]

nvkurbatova@sfedu.ru

Constructors of *diagonal* and *triangular* matrices





Operations on matrices

nvkurbatova@sfedu.ru

Special Matrix constructors

Diagonal matrices:

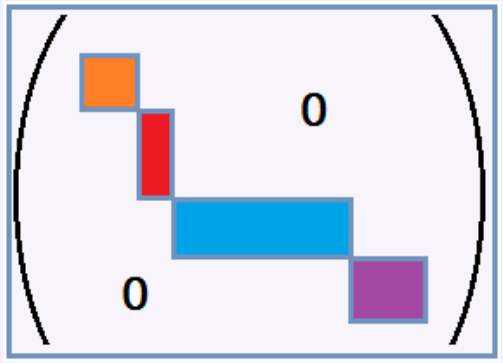
We can distinguish different diagonals of an already existing matrix.

```
A = [1 1 1; 2 2 2; 3 3 3]
d=diag(A)     % Highlighting % the main diagonal

diag(A, 1)  % Selection of the diagonal located above the main one
```

We can also create matrices based on the given diagonals.

```
diag(d)         % the d elements will be located on the main diagonal

diag(d,2)  % the d elements will be located above the main diagonal
```

```
A = 3×3
     1     1     1
     2     2     2
     3     3     3

d = 3×1
     1
     2
     3

ans = 2×1
     1
     2

ans = 3×3
     1     0     0
     0     2     0
     0     0     3

ans = 5×5
     0     0     1     0     0
     0     0     0     2     0
     0     0     0     0     3
```



Block-diagonal matrix

Triangular matrices:

We can find the upper or lower triangular matrix using the following functions.

```
T = randi(100, 5)
triu(T) % upper triangular part of matrix
tril(T) % lower triangular part of matrix
```

Block-diagonal matrices:

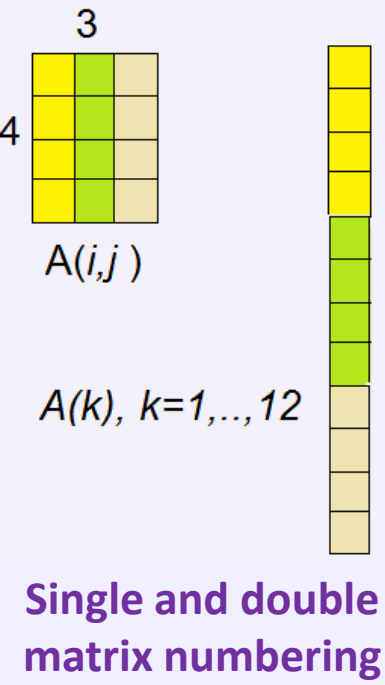
We can create block-diagonal matrices from existing ones.

```
A1 = ones(2,2);
A2 = 2*ones(3,2);
A3 = 3*ones(2,3);
B = blkdiag(A1,A2,A3)
```

```
T = 5×5
    48    16    25    29    43
     4    35    92    10    65
    18    61    27    58    65
    73    20    77    69    68
    48    74    19    55    64

ans = 5×5
    48    16    25    29    43
     0    35    92    10    65
     0     0    27    58    65
     0     0     0    69    68
     0     0     0     0    64

ans = 5×5
    48     0     0     0     0
     4    35     0     0     0
    18    61    27     0     0
    73    20    77    69     0
    48    74    19    55    64

B = 7×7
     1     1     0     0     0     0     0
     1     1     0     0     0     0     0
     0     0     2     2     0     0     0
     0     0     2     2     0     0     0
     0     0     2     2     0     0     0
     0     0     0     0     3     3     3
     0     0     0     0     3     3     3
```

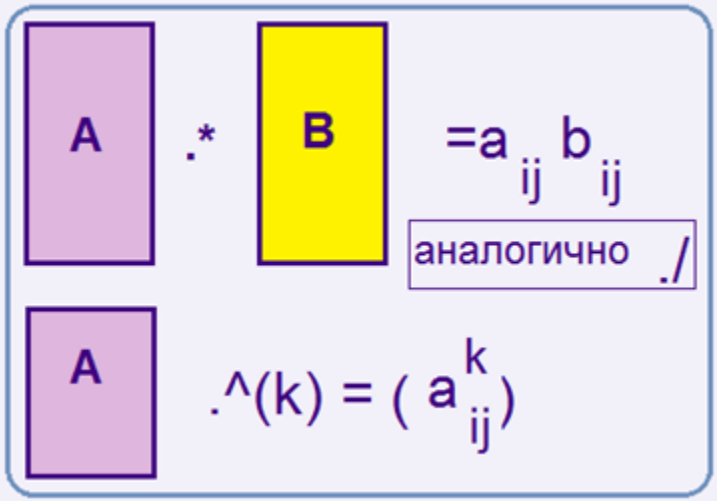
nvkurbatova@sfedu.ru

Editing the content and shape of the matrices

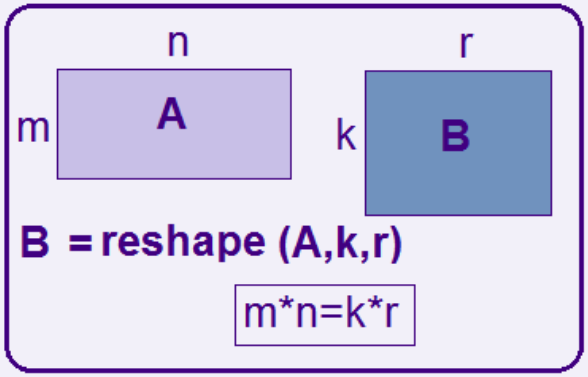


Single and double matrix numbering

Vector (element-by-element) operations:

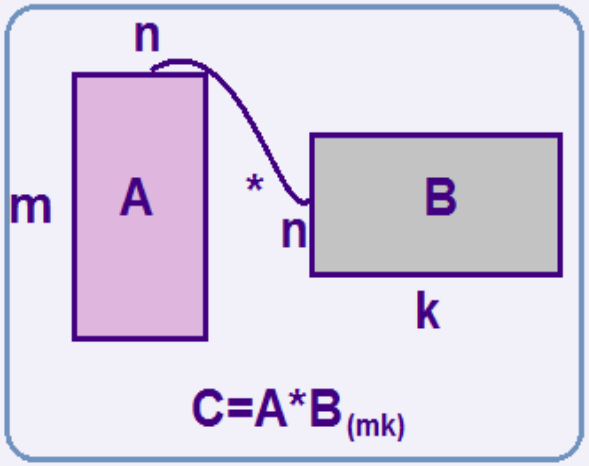


./ - similarly



Repmat – **by yourself!**

The rules of matrix algebra:



Operations on matrices

nvkurbatova@sfedu.ru

Examples of matrices editing

Editing

We can manually change individual elements, rows, columns, or some of these sections of the matrix, as we did in the examples above, or we can perform various arithmetic operations with the matrix, which will lead to a change in its elements.

Let's look at some examples:

```
44  v0 = 1:3;
45  v1 = 3:5;
46  v0 = v0*5
47  pv=v0.*v1
48  dv=v0./v1  % deviding left to right v0:v1
49  dvl=v0.\v1  % deviding  right to left v1:v0
50  v1 = v1.^3
51  v1.'        % what about v1 - complex? Explain result v1' for v1 complex!
```

Changing the structure

We can change the structure of the matrix not only by transposing, but also by using the built-in functions:

```
52  K1 = reshape(1:8, [], 2) % converting the matrix to a new size
53  K2 = linspace(10, 15, 6)
54  rK2=rot90(K2)  % rotate array 90 degrees
55  lrK2=fliplr(K2) % flip array left to right
56  K3 = K2.';
57  flipud(K3) % flip array up to down. Explain!
```

```
v0 = 1×3
     5    10    15

pv = 1×3
    15    40    75

dv = 1×3
    1.6667    2.5000    3.0000

dvl = 1×3
    0.6000    0.4000    0.3333

v1 = 1×3
    27    64   125

ans = 3×1
    27
    64
   125

K1 = 4×2
     1     5
     2     6
     3     7
     4     8

K2 = 1×6
    10    11    12    13    14    15

rK2 = 6×1
    15
    14
    13
    12
    11
    10

lrK2 = 1×6
    15    14    13    12    11    10
```

nvkurbatova@sfedu.ru

SPY

Visualization of the matrix structure

(non-zero elements)

nvkurbatova@sfedu.ru

Visualization of the matrix structure. What's wrong?

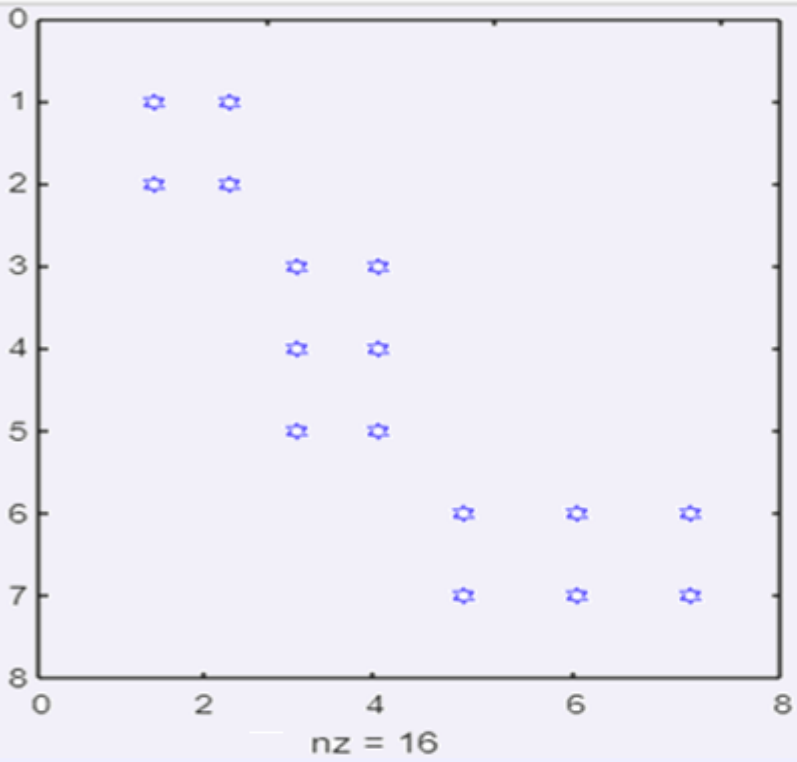
spy(B) – the graph is lost here! **Why?**

Visualization of non-zero elements

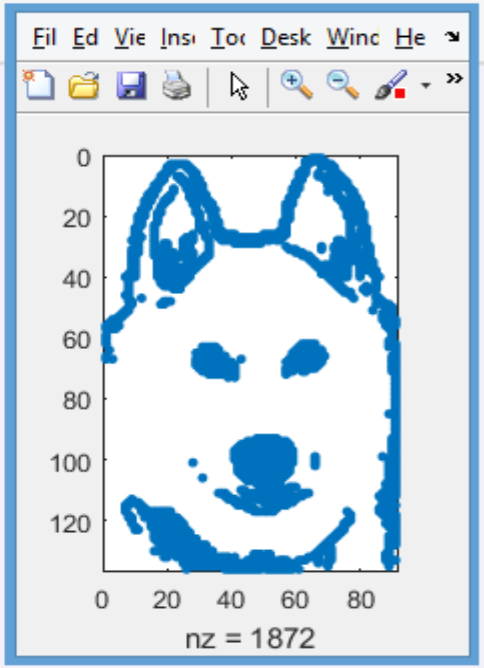
Function *spy* plots the sparsity pattern of matrix. Nonzero values are colored while zero values are white. We can change the color and shape using the corresponding function arguments:

```
spy(B)
spy(B, 'hb')
```

figure



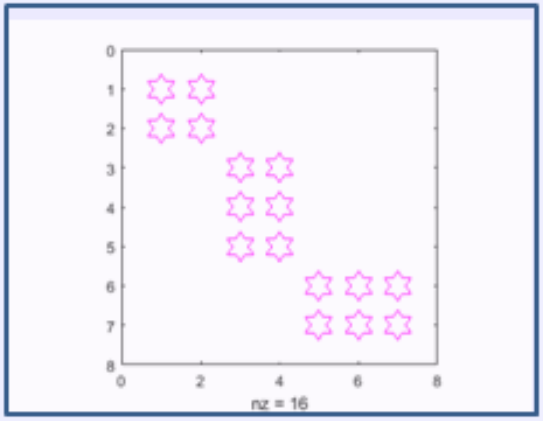
spy



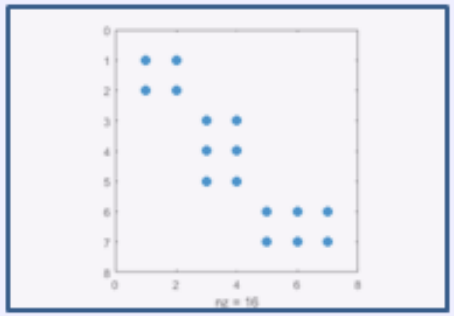
SPY – в деталях

nvkurbatova@sfedu.ru

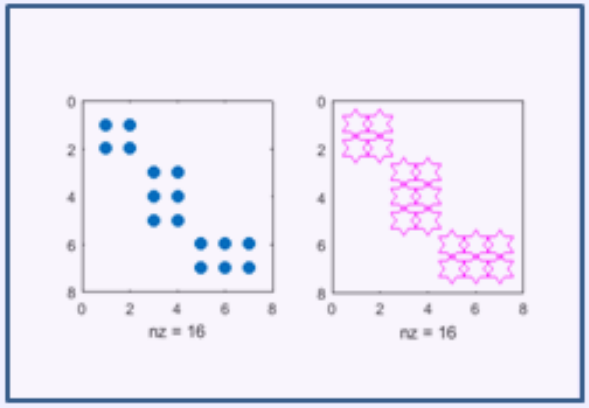
```
clear
% randi([vmin vmax],size)
B=blkdiag(rand(2), ones(3,2), ...
          randi([5,18],2,3)) % break line
nnz(B) % number of nonzero elements
spy(B,30)
spy(B,20, 'hm')  % 20 - markersize,
```




```
clear
B=blkdiag(rand(2), ones(3,2), randi([5,18],2,3))
spy(B,30)
figure
spy(B,20)
```




```
clear
B=blkdiag(rand(2),...
    ones(3,2), randi([5,18],2,3))
nnz(B)
subplot(1,2,1),  spy(B,30)
subplot(1,2,2),   spy(B,20,'hm')
```



blkdiag – конструктор блочно-диагональной матрицы

subplot(n,m,nc) – конструктор матрицы из графических окон, n – число строк, m – ?, nc – номер окна (счет идёт последовательно по строкам)

nvkurbatova@sfedu.ru

Elmat Library Functions

СИСТЕМНЫЕ ПЕРЕМЕННЫЕ И КОНСТАНТЫ MATLAB

clc - очистить Command Window
clear - удалить переменные Workspace
eps - Floating point relative accuracy
realmax - Largest positive floating point number
realmin - Smallest positive floating point number
intmax - Largest positive integer value
intmin - Smallest integer value
flintmax - Largest consecutive integer in floating point format 9.0072e+15
pi - 3.1415926535897....
i , j - Imaginary units
inf - Infinity
nan - Not-a-Number *(isnan(A));*
true - True array
false - False array
end - Last index

ОПЕРАЦИИ MATLAB

»help ops

Arithmetic operators:

plus - Plus +
uplus - Unary plus +
minus - Minus -
uminus - Unary minus -
mtimes - Matrix multiply *
times - Array multiply .*
mpower - Matrix power ^
power - Array power .^
mldivide - Backslash or left matrix divide \
mrdivide - Slash or right matrix divide /
ldivide - Left array divide .\
rdivide - Right array divide ./
idivide - Integer division with **rounding**option.
kron - Kronecker tensor product

Relational operators:

eq (A,B) - Equal **A == B**
ne (A,B) - Not equal **A~=B**
lt (A,B) - Less than **A < B**
gt (A,B) - Greater than **A> B**
le (A,B) - Less than or equal **A <= B**
ge (A,B) - Greater than or equal **A >=B**

Logical operators:

and (relop - Short-circuit logical) **A && B**
B - is not calculated if A consists of logical 0;
result - scalar *true* or *false*
or relop - Short-circuit logical **A || B**
B - is not calculated if A consists of logical 1 ;
result - scalar true or false
and - Element-wise logical **&**
or - Element-wise logical **or** **|**
not - Logical NOT **~**
punct - Ignore function argument or output **~**
xor - Logical EXCLUSIVE OR
any - True if any element of vector is nonzero
all - True if all elements of vector are nonzero

СПЕЦИАЛЬНЫЕ СИМВОЛЫ

Special characters:

colon	- Colon	**:**
paren	- Parentheses and subscripting	**( )**
paren	- Brackets	**[ ]**
paren	- Braces and subscripting	**{ }**
punct	- Function handle creation	**@**
punct	- Decimal point	**.**
punct	- Structure field access	**.**
punct	- Parent directory	**..**
punct	- Continuation	**...**
punct	- Separator	**,**
punct	- Semicolon	**;**

Special characters:

punct	- Comment	**%**
punct	- Invoke operating system command	**!**
punct	- Assignment	**=**
punct	- Quote	**'**
transpose	- Transpose	**.'**
ctranspose	- **Complex conjugate transpose**	**'**
horzcat	- Horizontal concatenation	**[,]**
vertcat	- Vertical concatenation	**[;]**
subsasgn	- Subscripted assignment	**( ),{ },.**
subsref	- Subscripted reference	**( ),{ },.**

nvkurbatova@sfedu.ru

Thanks for your attention!
See you later!

It's not a shame not to know, it's a shame not to want to know!