PV, PSO - základy matlabu

Ing. Viktor Kocur
viktor.kocur@fmph.uniba.sk

DAI FMFI UK

25.9.2019

Matlab Environment

- Matrix Laboratory of Mathworks
- Since 1984
- Original IDE for LINPACK a EISPACK libs without Fortran
- Optimized for various computations - especially Linear Algebra
- Simple and fast implementation of computer vision applications

Web resources

Lab and lecture study materials:

- https://dai.fmph.uniba.sk/w/Image_Processing_Fundamentals/
- https://github.com/kocurvik/edu/

External:

- https://www.mathworks.com/help/matlab/
- https://www.mathworks.com/matlabcentral/answers/
- https://stackoverflow.com/

Help in Matlab

How to find help in Matlab

- help command
- lookfor keyword
- F1

Exercise

Test this for the function/keyword 'edge'

Note

You can use standard unix commands within Matlab (cd, ls, mkdir, ...)

Scalar variables and arithmetic

Variable assignment

```
a = 1
a = 1;
```

Scalar variables and arithmetic

Variable assignment

```
a = 1
a = 1;
```

Named

Names are case sensitive! They have to start with a letter and need to have less than 63 characters.

Scalar variables and arithmetic

Variable assignment

```
a = 1
a = 1;
```

Named

Names are case sensitive! They have to start with a letter and need to have less than 63 characters.

Arithmetic

```
a = 1 * 2 + 8/9 - 4^(3/2)
b = a - 1 + 54*24
a = b*a
```

Scalar variables and arithmetic

Variable assignment

```
a = 1
a = 1;
```

Named

Names are case sensitive! They have to start with a letter and need to have less than 63 characters.

Arithmetic

```
a = 1 * 2 + 8/9 - 4^(3/2)
b = a - 1 + 54*24
a = b*a
```

Inf and NaN

```
1/0 == Inf
0/0 == NaN
```

Mathematics - Matrix multiplication

Definícia

$$\mathbb{A} \in \mathbb{R}^{m \times n}, \mathbb{B} \in \mathbb{R}^{n \times l}, \mathbb{C} \in \mathbb{R}^{m \times l}, \mathbb{A}\mathbb{B} = \mathbb{C} \iff$$

$$\forall i \in \hat{m}, \forall j \in \hat{l}, \mathbb{C}_{i,j} = \sum_{k=1}^{n} \mathbb{A}_{i,k} \cdot \mathbb{B}_{k,j}$$

Columns vs rows

We denote $\mathbb{R}^{\text{num rows} \times \text{num columns}}$ and $\mathbb{A}_{\text{row,column}}$

Vector variables

Assignment of vector variables

```
v = [1 2 3]
w = [1; 2; 3]
```

Vector variables

Assignment of vector variables

```
v = [1 2 3]
w = [1; 2; 3]
```

Column vs row vectors

```
w*v != v*w
v+w != v+w'
```

Vector variables

Assignment of vector variables

```
v = [1 2 3]
w = [1; 2; 3]
```

Column vs row vectors

```
w*v != v*w
v+w != v+w'
```

Generating vectors

```
r = start:step:end
r = linspace(start,end,n)
```

Matrix variables

Matrix assignment

```
A = [1 2 3; 4 5 6]
B = [v; 2*v - 1]
C = [w w]
D = [A; B]
```

Matrix variables

Matrix assignment

```
A = [1 2 3; 4 5 6]
B = [v; 2*v - 1]
C = [w w]
D = [A; B]
```

Matrix generation functions

- zeros(n), zeros(sz), zeros(s1,...,sn)
- ones(n)
- eye(n) - Identity matrix
- rand(n) - Random matrix with uniform distribution
- randn(n) - Random matrix with normal distribution
- magic(n) - Magic matrix

Array operations

Operator	Purpose	Description	Reference Page
+	Addition	A+B adds A and B.	plus
+	Unary plus	+A returns A.	uplus
-	Subtraction	A-B subtracts B from A	minus
-	Unary minus	-A negates the elements of A.	uminus
.*	Element-wise multiplication	A.*B is the element-by-element product of A and B.	times
.^	Element-wise power	A.^B is the matrix with elements A(i,j) to the B(i,j) power.	power
./	Right array division	A./B is the matrix with elements A(i,j)/B(i,j).	rdivide
.\	Left array division	A.\B is the matrix with elements B(i,j)/A(i,j).	ldivide
.'	Array transpose	A.' is the array transpose of A. For complex matrices, this does not involve conjugation.	transpose

Matrix operations

Operator	Purpose	Description	Reference Page
*	Matrix multiplication	`C = A*B` is the linear algebraic product of the matrices `A` and `B`. The number of columns of `A` must equal the number of rows of `B`.	`mtimes`
\	Matrix left division	`x = A\B` is the solution to the equation $Ax = B$. Matrices `A` and `B` must have the same number of rows.	`mldivide`
/	Matrix right division	`x = B/A` is the solution to the equation $xA = B$. Matrices `A` and `B` must have the same number of columns. In terms of the left division operator, `B/A = (A'\B')'`.	`mrdivide`
^	Matrix power	`A^B` is `A` to the power `B`, if `B` is a scalar. For other values of `B`, the calculation involves eigenvalues and eigenvectors.	`mpower`
'	Complex conjugate transpose	`A'` is the linear algebraic transpose of `A`. For complex matrices, this is the complex conjugate transpose.	`ctranspose`

https://www.mathworks.com/help/matlab/matlab_prog/array-vs-matrix-operations.html

Relation operators

Relation operators - return the logical type

`<, <=, >, >=, ==, ~=`

Relation operators

Relation operators - return the logical type

```
<, <=, >, >=, ==, ~=
```

We can compare vectors and matrices

```
A = rand(5)
B = rand(5)
A > B
A > 0.5
```

Logical operations

Logical functions and operators - return and use the logical type

`and (&), or (|), not (~), xor`

Short-circuit operators - for scalars only

`&&, ||`

Logical operations

Logical functions and operators - return and use the logical type

`and (&), or (|), not (~), xor`

Short-circuit operators - for scalars only

`&&, ||`

Reduction to one value

- any(a) -Returns true if any element is true
- all(a) - Returns true if all elements are true

Matrix operation functions

Užitočné funkcie

- flip(A)
- rot90(A)
- transpose(A), A' - transpose of matrix
- inv(A) - inverse matrix
- repmat(A,n) - Matrix with $n \times n$ submatrices A
- reshape(A,s1,..,sn) - Reshapes the matrix
- squeeze(A) - Removes 'singleton' dimensions
- size(A)
- numel(A) - Number of elements

Zoznam funkcií na prácu s maticami a poliami:
`https://www.mathworks.com/help/matlab/matrices-and-arrays.html`

Exercise for matrix operations

Assignment

Solve the equation $\mathbb{A}\vec{x} = \vec{b}$

$$\mathbb{A} \in \mathbb{R}^{4\times4}, \mathbb{A}_{i,j} = i \cdot (j+2)$$

$$\vec{b} \in \mathbb{R}^4, \vec{b_i} = i^2$$

Exercise for matrix operations

Assignment

Solve the equation $\mathbb{A}\vec{x} = \vec{b}$

$$\mathbb{A} \in \mathbb{R}^{4\times 4}, \mathbb{A}_{i,j} = i \cdot (j+2)$$

$$\vec{b} \in \mathbb{R}^4, \vec{b_i} = i^2$$

Example solution

```
A = (1:4)'*(3:6)
b = (1:4).^2
x = A\b'
```

Vector indexing

Alert

Indices start at 1!

Indexing

```
v = [7 8 5 2 4 6 5 2]
v(2) == 8
v(4:6) == [2 4 6]
v(1:2:end) == [7 5 4 5]
v([3 6 2]) == [5 6 8]
```

Writing using index

Writing using index

```
v = [7 8 5 2 4 6 5 2]
v(2) = 4
v(4:6) = [1 2 3]
v(1:2:end) = [1 3 5 7]
v([3 6 2]) = 1
v(70) = 10000
```

Writing using index

Writing using index

```
v = [7 8 5 2 4 6 5 2]
v(2) = 4
v(4:6) = [1 2 3]
v(1:2:end) = [1 3 5 7]
v([3 6 2]) = 1
v(70) = 10000
```

We can write beyond the end of a vector, but we cannot read

```
v = [1 2 3]
v(4) %works
v(4) = 4 %doesn't work
```

Maticová indexation

Three options for indexation

It is necessary to know the difference between these approaches

- with one index
- with a pair (row, col) - generally with an n-tuple
- with a logical matrix

Matrix indexation - one index

When using one index we start at the top left and we first traverse downwards through the column. At the end of the column we move onto the top of the next column.

$$\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$$

Matrix indexation - one index

Čítanie

```
A = magic(5)
A(4) == 10
A([4 5 6]) == [10 11 24]
A([4; 5; 6]) == [10; 11; 24]
A([10 25; 11 15]) == [18 9; 1 25]
A(4:4:20) == [10 6 7 8 2]
A(:) == [17 23 4 10 11 24 5 6 12 ...]
```

Assignment

```
    A = magic(5)
    A(4) = 10
    A([4 5 6]) = [10 11 24]
    A([10 25; 11 15]) = [100 200; 300 400]
!!! A([10 25; 11 15]) = [100 200 300 400]
```

Matrix indexation - index pair

Reading

```
    A = magic(5)
    A(2,2) == 5
    A(:,2) == [24; 5; 6; 12; 18]
    A(1:2:end,1:3)
    A([3 5],3:5)
    A([5 5 4 2 1],[2 4 5])
```

Writing

```
    A = magic(5)
    A(2,2) = 1000
    A(1:2:5,1:end-2) = eye(3)
!!! A(1:2:5,1:3) = [1 2 3 4 5 6 7 8 9]
!!! A([5 5],1) = [1 2]
```

Matrix indexation - logical matrix

Reading and writing

```
A = rand(5)
L = A>0.5
A(L)
A(L) = 0
B = magic(5)
B(A < 0.3 | L) = 50
```

Matrix indexation - logical matrix

Reading and writing

```
A = rand(5)
L = A>0.5
A(L)
A(L) = 0
B = magic(5)
B(A < 0.3 | L) = 50
```

Beware of dimensions

The logical matrix needs to have matching dimensions to what we attempt to index.

Matrix indexation - changing the indexation approach

Funkcie for change

```
[r,c] = ind2sub(sz,idx)
 idx  = sub2ind(sz,r,c)
 idx  = find(logicalMatrix)
```

Matrix indexation - changing the indexation approach

Funkcie for change

```
[r,c] = ind2sub(sz,idx)
 idx  = sub2ind(sz,r,c)
 idx  = find(logicalMatrix)
```

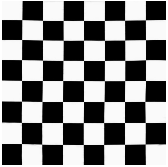
Writing beyond the end of matrix

```
    A = magic(5)
!!! A(26) = 1  % doesn't work not well defined
    A(6,1) = 1 % works
```

Indexation exercise 1

Assigment

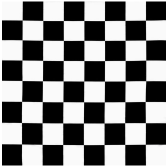
Generate a matrix with rand(8). Change all the elements that would be black on a chekerboard to 1. Afterwards change all of the elements smaller than 0.3 to 0.



Indexation exercise 1

Assigment

Generate a matrix with rand(8). Change all the elements that would be black on a chekerboard to 1. Afterwards change all of the elements smaller than 0.3 to 0.



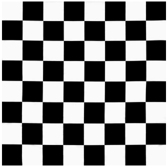
Example solution:

```
R = rand(8)
R(1:2:7,2:2:8) = 1
R(2:2:8,1:2:7) = 1
R(R<0.3) = 0
```

Indexation exercise 2

Assignment

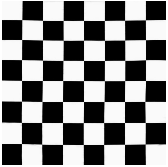
Generate a matrix using magic(8) and create a matrix of shape 8x4 from just the elements that would be on the white tiles in a checkerboard.



Indexation exercise 2

Assignment

Generate a matrix using magic(8) and create a matrix of shape 8x4 from just the elements that would be on the white tiles in a checkerboard.



Example solution:

```
A = magic(8)
s = [1 0;0 1]
I = repmat(s,4)
B = reshape(A(I == 1),[8 4])
```

Mathematical functions

Príklady funkcií

- mod, round, floor, ceil
- abs, sgn, exp, log, sin, cos, tan, asin...
- min, max
- sum, diff, mean, var

Viac na `https://www.mathworks.com/help/matlab/functionlist.html`

Read the documentations

For example the function sum applied to a matrix returns a row vector with sums of values for each column. If we want to add all of the elements of the matrix we need to use sum(sum(A)) or sum(A(:)). Same rule applies for mean, var, min, max, ...

Data types

Numeric types

- single, double
- int8, int16, int32, int64
- uint8, uint16, uint32, uint64

Data types

Numeric types

- single, double
- int8, int16, int32, int64
- uint8, uint16, uint32, uint64

Other types

- char, string
- cell array, map, table, categorical array, struct, logical
- date, time, time series, timetable
- function handle, handle

Data types

Numeric types

- single, double
- int8, int16, int32, int64
- uint8, uint16, uint32, uint64

Other types

- char, string
- cell array, map, table, categorical array, struct, logical
- date, time, time series, timetable
- function handle, handle

Type recognition

```
class(a)
```

Numerical types

Type change

```
a = 150
class(a) == 'double'
b = uint16(a)
class(b) == 'uint16'
cast(int8(-50),'uint8') == 0
typecast(int8(-50),'uint8') == 206
typecast(-50,'int16') == [0 0 0 0 0 0 73 192]
```

Numerical types

Type change

```
a = 150
class(a) == 'double'
b = uint16(a)
class(b) == 'uint16'
cast(int8(-50),'uint8') == 0
typecast(int8(-50),'uint8') == 206
typecast(-50,'int16') == [0 0 0 0 0 0 73 192]
```

Integer overflow

```
uint8(200) + uint8(200) == 255
```

Numerical types

Type change

```
a = 150
class(a) == 'double'
b = uint16(a)
class(b) == 'uint16'
cast(int8(-50),'uint8') == 0
typecast(int8(-50),'uint8') == 206
typecast(-50,'int16') == [0 0 0 0 0 0 73 192]
```

Integer overflow

```
uint8(200) + uint8(200) == 255
```

Displaying numbers

```
format long
format shortEng
```

Other types

Cell array

```
c = {45, ones(5), 'hello', [1 2 3]}
c(1) == {[45]}
c{1} == 45
c{3} = 5.24754
```

Other types

Cell array

```
c = {45, ones(5), 'hello', [1 2 3]}
c(1) == {[45]}
c{1} == 45
c{3} = 5.24754
```

Struct

```
s.a = 1;
s.b = {'A','B','C'}
s =
 struct with fields:
  a: 1
  b: {'A'  'B'  'C'}
p = struct('fieldName',fieldVal)
```

Scripts

Scripts

We save scripts to separate .m files,

Running the script

- We can call the name of the m-file in the command window provided the script is in path or current folder
- Running from the editor - option to debug

Scripts

Scripts

We save scripts to separate .m files,

Running the script

- We can call the name of the m-file in the command window provided the script is in path or current folder
- Running from the editor - option to debug

Alert!

Scripts have access to the variables in the workspace

Functions

Functions

Functions are saved in a same fashion as scripts. The difference is that functions have inputs and outputs.

Functions

Functions

Functions are saved in a same fashion as scripts. The difference is that functions have inputs and outputs.

Running

Functions are called by the FILENAME of the m-files. The file has to be in the current directory or somewhere in PATH.

Functions - structure

Functions

```
function output = functionName(input)
% comment - bude sa zobrazovat v helpe
    output = 2*input;
% tento comment sa uz nezobrazi v helpe
end
```

Functions - structure

Multiple inputs and outputs

```
function [out1,out2] = functionName(in1,in2)
    out1 = 2*in1;
    out2 = in1*in2;
end
```

Variable inputs and outputs

For variable amount of inputs and outputs we can use special variables nargin (number of inputs) and vararging (array of the inputs) for inputs and nargout for the number of outputs.

Functions - nested a local

Functions

```
function parent
    disp('This is the parent function')
    nestedfx
    localfx

    function nestedfx
        disp('This is the nested function')
    end
end

function localfx
    disp('This is a local function')
end
```

Funkcie

Nested functions

Nested functions have access to the variables of the parent function and vice versa. If we use a variable which is not defined in the parent function this variable is then lost after calling the nested function.

Local functions

Local functions do not have access to the variables of the parent function.

Anonymous functions

Example - essentially a wrapper

```
sqr = @(x) x.^2;
sqr(5) == 25

q = integral(sqr,0,1);
q = integral(@(x) x.^2,0,1);
```

If

Structure

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

Switch

Structure

```
switch switch_expression
    case case_expression
        statements
    case case_expression
        statements
        ...
    otherwise
        statements
end
```

For cycle

Structure

```
for index = values
    statements
end

for i = 1:n
     r(i) = ...
end

for i = [5 0 2 7 5 1]
     statements
end
```

While cycle

Structure

```
while expression
   statements
end
```

While cycle

Structure

```
while expression
   statements
end
```

Ending while a for cycle

- break - ends the whole cycle
- continue - skips to the next iteration

While cycle

Structure

```
while expression
   statements
end
```

Ending while a for cycle

- break - ends the whole cycle
- continue - skips to the next iteration

Terminating the program

Any script or function can be terminated by pressing Ctrl + C

Exercise

Assignment

Write a function fib(n), which returns the n-th element of the Fibonacci sequence.

Assignment - hard version

Write a function for a linear recurrent sequence linrek(n,$\vec{a}$,$\vec{b}$), which returns the n-th elenent of the sequence defined as

$$f_n = \sum_{i=1}^{dim(\vec{a})} a_i \cdot f_{n-i},$$

with initial values $f_i = b_i$ pre $i \leq dim(\vec{a}) = dim(\vec{b})$.

Exercise

Example solution:

```
function out = fib(n)
    if n <= 0
        out = 0;
    elseif n == 1
        out = 1;
    else
        out = fib(n-1) + fib(n-2)
    end
end
```

Exercise

Example solution:

```
function out = fib(n)
    if n <= 0
        out = 0;
    elseif n == 1
        out = 1;
    else
        out = fib(n-1) + fib(n-2)
    end
end
```

Or:

```
function out = fib(n)
    out = round(1.61803398875^(n-1))
end
```

Hard version solution:

```
function rek = linrek(n,a,b)
    if length(b) >= n
        rek = b(n);
    else
        rek = 0;
        for i=1:length(b)
            rek = rek + a(i) * linrek(n-i,a,b);
        end
    end
end
```

Exercise - no recursion

Solution example:

```
function rek = linrekaprox(n,a,b)
    pol = [-1 a];
    r = roots(pol)';
    m = zeros(size(r));
    for i = 1:numel(a)
        m(i) = sum(r(1:i) == r(i)) - 1;
    end
    rowmat = repmat((1:numel(a))',1,numel(a));
    A = (r.^rowmat).*(rowmat.^m);
    k = A\b';
    rek = round(real(((n.^m).*(r.^n))*k));
end
```

Profiling

Tic Toc

```
tic
statements
toc
```

Cputime

```
t1= cputime
statements
t2 = cputime
disp(t2 - t1)
```

Matlab optimization compare the runtimes

No allocation

```
p = 0;
for k=1:10000
    p(k) = k/(sin(k)+2)
end
```

With allocation

```
p = zeros(1,10000)
for k=1:10000
    p(k) = k/(sin(k)+2)
end
```

With vector operations

```
k = 1:10000
p = k./(sin(k)+2)
```

Plot

Simple plot

```
x = linspace(0,10,1000);
plot(x,sin(x))
```

More arguments

```
plot(X,Y,LineSpec)
plot(X1,Y1,...,Xn,Yn)
plot(____, Name, Value)
```

Linespec - examples

- Linestyle - -, –, :, -.
- Marker - o, +, *, ., x, s, d
- Colors - y, m, c, r, g, b, w, k → can be combined e.g. r*-

Images

Hold on a hold off

```
x = linspace(0,10,1000);
plot(x,sin(x))
hold on
plot(x,cos(x)) % the second plot is drawn into the firs
hold off
plot(x,cos(x)) % this one redraws the whole plot
```

Images

Hold on a hold off

```
x = linspace(0,10,1000);
plot(x,sin(x))
hold on
plot(x,cos(x)) % the second plot is drawn into the firs
hold off
plot(x,cos(x)) % this one redraws the whole plot
```

Figure a Axes

```
fig1 = figure % generates a figure window
ax1 = axes % generates axes
```

Images

Hold on a hold off

```
x = linspace(0,10,1000);
plot(x,sin(x))
hold on
plot(x,cos(x)) % the second plot is drawn into the firs
hold off
plot(x,cos(x)) % this one redraws the whole plot
```

Figure a Axes

```
fig1 = figure % generates a figure window
ax1 = axes % generates axes
```

Where are plots drawn

Plots are drawn into the active figure (gcf) and active axes (gca). If you want to draw somewhere else you can use
plot(..., 'Parent',ax4)

Subplot

Multiple plots in one

```
ax1 = subplot(2,2,1);
plot(x,sin(x))
ax2 = subplot(2,2,2);
plot(x,cos(x))
ax3 = subplot(2,2,3);
ax4 = subplot(2,2,4);
plot(x,x.^2,'Parent',ax3)
plot(x,x.^3,'Parent',ax4)
```

Other plot types

Other plot types

- plot3, loglog, semilogx, semilogy, errorbar
- bar, bar3, barh, barh3, histogram, pie, pie3
- stem, stairs, scatter
- countour, countourf, surf, ezsurf
- feather, quiver, compass

Other plot types

Other plot types

- plot3, loglog, semilogx, semilogy, errorbar
- bar, bar3, barh, barh3, histogram, pie, pie3
- stem, stairs, scatter
- countour, countourf, surf, ezsurf
- feather, quiver, compass

Removing plots

- cla, clf - clears active axis/figure
- close all - closes all figure windows

Reading images

Images

Download the zip from github kocurvik/edu/PSO/supplementary

Reading images

Images

Download the zip from github kocurvik/edu/PSO/supplementary

Reading

```
rgb = imread('zatisie.jpg');
whos rgb
d = im2double(rgb)
whos d
bw = imread('zatisie.pgm');
whos bw
```

Displaying

imshow

```
imshow(rgb)
imshow(d)
imshow(bw)
```

Displaying

imshow

```
imshow(rgb)
imshow(d)
imshow(bw)
```

image

```
image(rgb)
image(d)
imagesc(bw)
colormap(gray)
```

Writing

imwrite

```
imwrite(rgb, 'filename.png')
imwrite(d, 'filename.jpg')
imwrite(bw, 'filename.png')
```

Alert!

If the image is in uint8, then the values are expected to be in the range 0-255. If the image is in double the range is 0.0 - 1.0. Usually what imshow displays that will get saved using imwrite. For other display options this may not be the case!

Exercise

Reducing a color channel

In the color image of the still (zatisie) reduce the red element of RGB to one fifth. Hint: the array is in shape rows x cols x channels.

Exercise

Reducing a color channel

In the color image of the still (zatisie) reduce the red element of RGB to one fifth. Hint: the array is in shape rows x cols x channels.

Solution

```
rgb(:,:,1) = 0.2 * rgb(:,:,1);
imshow(rgb)
```

Exercise - Hard

Scale

Create a function myimresize(I, s), which returns the image scaled by the values s using the nearest point interpolation.

Exercise - Hard

Scale

Create a function myimresize(I, s), which returns the image scaled by the values s using the nearest point interpolation.

Solution

```
function I = myimresize(I, s)
    oldrows = size(I,1);
    oldcols = size(I,2);
    r = round(linspace(1,oldrows,round(s*oldrows)));
    c = round(linspace(1,oldcols,round(s*oldcols)));
    I = I(r,c);
end
```