


FACULTY OF MATHEMATICS,
PHYSICS AND INFORMATICS
Comenius University
Bratislava

Rozpoznávanie Obrazcov

0. Cvičenie - Matlab

Ing. Viktor Kocur, PhD.

16.2.2022

Prostredie Matlab

- Matrix Laboratory of Mathworks
- Od roku 1984
- Originálne prostredie na využitie LINPACK-u a EISPACK-u bez znalosti Fortranu
- Optimalizovaý na mnohé druhy výpočtov - hlavne lin. algebra
- Jednoduchá implementácia a testovanie algoritmov spracovania obrazu

Webové zdroje

Prezentácie a podklady k cvičeniam:

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

Externé:

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

Help v Matlabe

Ako nájsť pomoc priamo v Matlabe

- help command
- lookfor keyword
- F1

Úloha

Otestujte pre príkaz/heslo ’edge’

Poznámka

V matlabe môžete používať štandardné unixové príkazy ako cd, ls, mkdir, ...

Skalárne premenné a aritmetika

Priradʼovanie premenných

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

Skalárne premenné a aritmetika

Prirad'ovanie premenných

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

Názvy

Názvy su case sensitive! Musia začínať písmenom (potom môžu byť aj čísla) a mať max 63 znakov.

Skalárne premenné a aritmetika

Aritmetika

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

Skalárne premenné a aritmetika

Aritmetika

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

Inf a NaN

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

Matematická vsuvka - Maticové násobenie

Definícia

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

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

Stĺpce vs riadky

Značíme $\mathbb{R}^{\text{počet riadkov} \times \text{počet stĺpcov}}$ a $\mathbb{A}_{\text{riadok,stĺpec}}$

Vektorové premenné

Priradʼovanie vektorových premenných

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

Vektorové premenné

Prirad'ovanie vektorových premenných

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

Stĺpcové vs riadkové vektory

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

Vektorové premenné

Prirad'ovanie vektorových premenných

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

Stĺpcové vs riadkové vektory

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

Generovanie vektorov

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

Maticové premenné

Priradʼovanie matíc

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

Maticové premenné

Priradʼovanie matíc

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

Funkcie na generáciu matíc

- zeros(n), zeros(sz), zeros(s1,...,sn)
- ones(n)
- eye(n) - Matica identity
- rand(n) - Náh. matica s rovnomernou dist.
- randn(n) - Náh. matica s norm. dist.
- magic(n) - Magická matica

Operácie s poliami

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

Operácie s maticami

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

Relačné operácie

Relačné operátory - vracajú typ logical

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

Relačné operácie

Relačné operátory - vracajú typ logical

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

Porovnávať môžeme aj vektory a matice

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

Logické operácie

Logické funkcie a operátory - vracajú a používajú typ logical

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

Short-circuit operátory - iba pre skaláry

```
&&, ||
```

Logické operácie

Logické funkcie a operátory - vracajú a používajú typ logical

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

Short-circuit operátory - iba pre skaláry

```
&&, ||
```

Redukcia na jednu hodnotu

- any(a) - Vráti True ak aspoň jeden prvok v a je True
- all(a) - Vráti True ak všetky prvky v a sú True

Funkcie na prácu s maticami

Užitočné funkcie

- flip(A) - Pretočenie matice
- rot90(A) - Otočenie matice
- transpose(A), A’ - Transpozícia matice
- inv(A) - Inverzná matica k A
- repmat(A,n) - Matica s $n \times n$ podmaticami A
- reshape(A,s1,..,sn) - Zmena tvaru matice
- squeeze(A) - Odstránenie ’singleton’ dimenzie
- size(A) - Vel’kosť matice
- numel(A) - Počet prvkov matice

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

Úloha na prácu s maticami

Zadanie

Riešte rovnicu $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

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

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

Úloha na prácu s maticami

Zadanie

Riešte rovnicu $\mathbb{A}\boldsymbol{x} = \boldsymbol{b}$

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

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

Riešenie napr.

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

Vektorová indexácia

Pozor

Indexy začínajú od 1!

Indexácia

```
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]
```

Zápis pomocou indexácie

Zápis prostredníctvom indexácie

```
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
```

Zápis pomocou indexácie

Zápis prostredníctvom indexácie

```
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
```

Za koniec vektoru môžeme zapisovať, ale nie čítať

```
v = [1 2 3]
v(4) %nebude fungovať
v(4) = 4 %bude fungovať
```

Maticová indexácia

Tri spôsoby indexácie

Je potrebné rozlišovať medzi troma spôsobmi indexácie matíc!

- jedným indexom
- dvojicou (riadok, stĺpec) - obecne n-ticou
- logickou maticou

Maticová indexácia - jedným indexom

Pri použití jedného indexu začneme vľavo hore a idem najprv dole po stĺpci, na konci prejdeme na vrch nasledujúceho stĺpca.

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

Maticová indexácia - jedným indexom

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 ...]
```

Maticová indexácia jedným indexom

Zápis

```
    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]
```

Maticová indexácia - dvoma indexmi

Čítanie

```
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])
```

Zápis

```
    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]
```

Maticová indexácia - logickou maticou

Čítanie a zápis

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

Maticová indexácia - logickou maticou

Čítanie a zápis

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

Pozor na rozmery

Logická matica musí mať rovnaký rozmer ako matica s ktorou operujeme

Maticová indexácia - prechody

Funkcie na prechod

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

Maticová indexácia - prechody

Funkcie na prechod

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

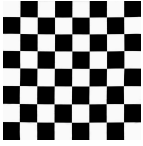
Zápis do prázdneho indexu mimo matice

```
    A = magic(5)
!!! A(26) = 1  % nefunguje - nejednoznačné
    A(6,1) = 1 % funguje
```

Úloha na indexáciu 1

Zadanie

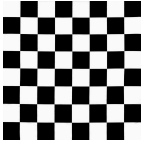
Vygenerujte maticu pomocou rand(8). Premente všetky prvky ktoré by boli na šachovnici na čiernom políčku na 1. Následne premente všetky prvky menšie ako 0.3 na 0.



Úloha na indexáciu 1

Zadanie

Vygenerujte maticu pomocou rand(8). Premente všetky prvky ktoré by boli na šachovnici na čiernom políčku na 1. Následne premente všetky prvky menšie ako 0.3 na 0.



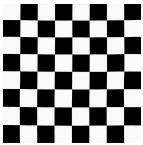
Riešenie napr:

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

Úloha na indexáciu 2

Zadanie

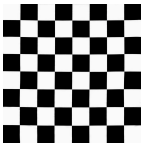
Vygenerujte maticu pomocou magic(8) a z nej vytvorte maticu 8x4 len z prvkov na bielych políčkach.



Úloha na indexáciu 2

Zadanie

Vygenerujte maticu pomocou magic(8) a z nej vytvorte maticu 8x4 len z prvkov na bielych políčkach.



Riešenie napr:

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

Matematické funkcie

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`

Treba čítať dokumentáciu

Napríklad príkaz sum aplikovaný na maticu vráti riadkový vektor so súčtami hodnôt v jednotlivých stĺpcoch. Ak chceme sčítať všetky prvky matice musíme použiť sum(sum(A)), alebo sum(A(:)). Toto platí aj pre min, max, mean, var, diff...

Dátové typy

Numerické typy

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

Dátové typy

Numerické typy

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

Ostatné typy

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

Numerické typy

Zistenie typu

```
class(a)
whos a
```

Zmena typov

```
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]
```

Numerické typy

Integer overflow

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

Numerické typy

Integer overflow

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

Zmena vypisovania čísel

```
format long
format shortEng
```

Krátko k niektorým iným typom

Cell array

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

Krátko k niektorým iným typom

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)
```

Skripty

Skripty

Skripty ukladáme do samostatného súboru s príponou .m

Spúštanie

- Spúštanie z command window - príkaz je totožný s názvom súboru (musíme byť v správnej zložke, resp. mať zložku v PATH)
- Spúštanie z editoru - možnosť debugovania

Skripty

Skripty

Skripty ukladáme do samostatného súboru s príponou .m

Spúštanie

- Spúštanie z command window - príkaz je totožný s názvom súboru (musíme byť v správnej zložke, resp. mať zložku v PATH)
- Spúštanie z editoru - možnosť debugovania

Pozor!

Skript má prístup k premenným z workspace!

Funkcie

Funkcie

Funkcie ukladáme obdobne ako skripty do samostatného súboru s príponou .m.
Na rozdiel od skriptov maju funkcie vstupy a výstupy.

Funkcie

Funkcie

Funkcie ukladáme obdobne ako skripty do samostatného súboru s príponou .m. Na rozdiel od skriptov maju funkcie vstupy a výstupy.

Spúšťanie

Funkciu voláme príkazom podľa názvu SÚBORU. Súbor musí byť uložený v pracovnej zložke, alebo zložke ktorá je v Matlabovskom PATH.

Funkcie - štruktúra

Funkcie

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

Funkcie - štruktúra

Viac vstupov a výstupov

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

Variabilný počet vstupov a výstupov

Pre variabilný počet vstupov môžeme použiť špeciálne premenné nargin (počet vstupov) a varargin (pole vstupov variabilnej dĺžky) pre vstupy a nargout pre výstupy.

Funkcie - nested a local

Funkcie

```
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
```

Nested Funkcie

Majú prístup k premenným parent funkcie a naopak. Ak však použijú premennú, ktorá nieje definovaná v parent funkcii, tak táto hodnota sa stratí po konci volania nested funkcie.

Local funkcie

Nemajú prístup k premenným parent funkcie.

Anonymné funkcie

Príklad

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

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

If

Štruktúra

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

Switch

Štruktúra

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

For cyklus

Štruktúra

```
for index = values
    statements
end

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

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

While cyklus

Štruktúra

```
while expression
   statements
end
```

While cyklus

Štruktúra

```
while expression
   statements
end
```

Predčasné ukončenie while a for cyklu

- break - ukončí celý cyklus
- continue - prejde na d'alšiu iteráciu cyklu

While cyklus

Štruktúra

```
while expression
   statements
end
```

Predčasné ukončenie while a for cyklu

- break - ukončí celý cyklus
- continue - prejde na d'alšiu iteráciu cyklu

Terminácia programu

Akýkoľvek skript a funkciu môžeme ukončiť stlačením Ctrl-C

Úloha

Zadanie

Napíšte funkciu fib(n), ktorá vráti n-tý člen Fibonacciho postupnosť i.

Zadanie - ťažká verzia

Napíšte funkciu na lineárne rekurentnú postupnosť linrek(n,$\boldsymbol{a}$,$\boldsymbol{b}$), ktorá vráti n-tý člen lineárnej rekurentnej postupnosti v tvare

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

s počiatočnými hodnotami $f_i = b_i$ pre $i \leq dim(\boldsymbol{a}) = dim(\boldsymbol{b})$.

Úloha - riešenie

Riešenie napr.:

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

Úloha - riešenie

Riešenie napr.:

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

Alebo:

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

Riešenie ťažšej verzie napr:

```
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
```

Úloha - bez rekurzie

Riešenie napr:

```
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
```

Meranie doby behu

Tic Toc

```
tic
statements
toc
```

Cputime

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

Optimalizácia matlabu - porovnajte doby behov

Bez alokácie

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

S alokáciou

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

Vektorovo

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

Plot

Jednoduchý plot

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

Viac argumentov

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

Linespec - príklady

- Linestyle - -, –, :, -.
- Marker - o, +, *, ., x, s, d
- Farby - y, m, c, r, g, b, w, k $\rightarrow$ kombinujeme napr. r*-

Hold on a hold off

```
x = linspace(0,10,1000);
plot(x,sin(x))
hold on
plot(x,cos(x)) % druhy plot sa nakresli do prveho
hold off
plot(x,cos(x)) % tento plot prekresli prve dva
```

Ploty

Hold on a hold off

```
x = linspace(0,10,1000);
plot(x,sin(x))
hold on
plot(x,cos(x)) % druhy plot sa nakresli do prveho
hold off
plot(x,cos(x)) % tento plot prekresli prve dva
```

Figure a Axes

```
fig1 = figure % vytvori okno
ax1 = axes % vytvori kartezsky podklad
```

Ploty

Kam sa kreslia grafy

Grafy sa kreslia do aktívnej figure (gcf) a aktívnych axes (gca). Ak chceme kresliť inam použijeme napr. plot(____,'Parent',ax4)

Ploty

Kam sa kreslia grafy

Grafy sa kreslia do aktívnej figure (gcf) a aktívnych axes (gca). Ak chceme kresliť inam použijeme napr. plot(___,'Parent',ax4)

Viac plotov v jednej figure

```
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)
```

Ine grafy a mazanie

Ostatné druhy grafov

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

Ine grafy a mazanie

Ostatné druhy grafov

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

Mazanie

- cla, clf - mažú aktívne axis/figure
- close all - zavri všetky figure okná

Načítanie obrázkov

Obrázky

Stiahnite si zip z githubu kocurvik/edu/PSO/supplementary

Načítanie obrázkov

Obrázky

Stiahnite si zip z githubu kocurvik/edu/PSO/supplementary

Načítanie súborov

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

Zobrazovanie

Zobrazovanie pomocou imshow

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

Zobrazovanie

Zobrazovanie pomocou imshow

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

Zobrazovanie pomocou image

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

Zápis

Zápis pomocou imwrite

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

Pozor!

Ak je obrázok v uint8, tak očakávaný rozsah je 0-255. Ak je obrázok v double, tak očakávaný rozsah je 0-1. Väčšinou, čo ukáže imshow to zapíše imread. O image a hlavne imagesc to obecne neplatí!

Úloha

Zrušenie farebnej zložky

Vo farebnom obrázku zátišia znížte červenú zložku RGB na pätinu. Hint: pole je tvaru riadky x stĺpce x farba.

Úloha

Zrušenie farebnej zložky

Vo farebnom obrázku zátišia znížte červenú zložku RGB na pätinu. Hint: pole je tvaru riadky x stĺpce x farba.

Riešenie

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

Úloha

Scale

Vytvorte funkciu myimresize(I,s), ktorá vráti obrázok zväčšený pomerom s pomocou metódy nearest point interpolation. Porovnajte s funkciu imresize.

Úloha

Scale

Vytvorte funkciu myimresize(I,s), ktorá vráti obrázok zväčšený pomerom s pomocou metódy nearest point interpolation. Porovnajte s funkciu imresize.

Riešenie

```
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
```