Seminář 1

Matlab jako kalkulačka

práce s Command Window

Aritmetické operátory +, -, *, /

```
3 + 5
```

```
ans = 8
```

přiřazení hodnoty (operátor =)

```
m = 3 + 5
```

```
m = 8
```

Vytvořte proměnnou n a přiřaďte do ní hodnotu součinu 2 a proměnné m.

Potlačení výpisu výsledku příkazu ;

```
o = m + 5;
```

Historie příkazů - šipky nahoru a dolů.

Zeditujte předchozí příkaz na o = m + 2;

Výpis hodnoty proměnné.

```
o
```

```
o = 13
```

```
disp(o);
```

```
    13
```

```
display(o);
```

```
o = 13
```

Proměnné

Matlab je case sensitive.

Vytvořte proměnné a a A (přiřaďte jim nějakou hodnotu) a spočítejte jejich průměr meanA.

Zkuste vytvořit proměnnou 3a. A podívejte se na chybovou hlášku.

Každá proměnná má v matlabu přiřazen nějaký datový typ (třídu).

Podívejte se na výpis příkazu `whos`

```
n = 1.5
```

```
n = 1.5000
```

```
whos n
```

```
  Name      Size            Bytes  Class     Attributes

  n         1x1                 8  double
```

Datový typ proměnné `m` je `double` (číslo s plovoucí čárkou s dvojitou přesností).

`double` je defaultní datový typ číselných hodnot.

Datové typy:

- číselné: `float`, `double`, `int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`, `uint64`, ...

- logické: `logical` (`true`, `false`)

- znaky: `char` (řetězec je pole znaků), string

Pro převod mezi jednotlivými datovými typy můžeme použít název třídy jako funkci.

```
u = 10
```

```
u = 10
```

```
whos u
```

```
  Name      Size            Bytes  Class     Attributes

  u         1x1                 8  double
```

```
u2 = uint8(u)
```

```
u2 = uint8

    10
```

```
whos u2
```

```
  Name      Size            Bytes  Class    Attributes

  u2        1x1                 1  uint8
```

Případně speciální funkce jako jsou num2str(), str2num() a jiné.

```
c = num2str(3)
```

```
c =
'3'
```

```
whos c
```

```
  Name      Size            Bytes  Class    Attributes

  c         1x1                 2  char
```

Složitějším datovým typům (strukturám, objektům, ...) se budeme věnovat později.

Uložení Workspace

```
save nazev
```

Vyčištění Workspace

```
clear
```

Načtení Workspace

```
load nazev
```

Případně 2x poklepat na soubor ve File browseru, nebo import ikona.

Je možné načít nebo uložit jen některé proměnné

```
save nazev m o
```

Matlab obsahuje vestavěné konstanty i, j (imaginární jednotka), pi, Inf (nekonečno), NaN (not a number) a jiné

```
x = pi/2
```

```
x = 1.5708
```

Ukazují se jen 4 desetinná místa, ale v paměti je uloženo více.

Přesnost můžeme upravit pomocí příkazu `format, např.`

```
format long
x
```

```
x =
   1.570796326794897
```

defaultní formát je short

Pro více informací o příkazu `format` (ale i jiných funkcí) použijeme `help`

```
help format
```

```
  format Set output display format.
     fmt = format returns the current display format.

     format style changes the output display format. For a description of
     possible values for style, see details below.

     fmt = format(style) stores the current display format in fmt and then
     changes the display format to the specified style.

     format(fmt) changes the output display format to the fmt returned by a
     previous call to format.

     format does not affect how MATLAB computations are done. Computations
     on float variables, namely single or double, are done in appropriate
     floating point precision, no matter how those variables are displayed.
     Computations on integer variables are done natively in integer. Integer
     variables are always displayed to the appropriate number of digits for
     the class, for example, 3 digits to display the INT8 range -128:127.
     format SHORT and LONG do not affect the display of integer variables.

     format DEFAULT may be used to restore the default display format,
     which is SHORT for numeric format and LOOSE for line spacing.

     format may be used to switch between different output display formats
     of all float variables as follows:
       format SHORT     Short fixed point format with 4 digits after the
                        decimal point.
       format LONG      Long fixed point format with 15 digits after the
                        decimal point for double values and 7 digits after
                        the decimal point for single values.
       format SHORTE    Short scientific notation with 4 digits after the
                        decimal point.
       format LONGE     Long scientific notation with 15 digits after the
                        decimal point for double values and 7 digits after
                        the decimal point for single values.
       format SHORTG    Short fixed format or scientific notation,
                        whichever is more compact, with a total of 5 digits.
       format LONGG     Long fixed format or scientific notation, whichever
                        is more compact, with a total of 15 digits for
                        double values and 7 digits for single values.
       format SHORTENG  Engineering format with 4 digits after the decimal
                        point and a power that is a multiple of three.
       format LONGENG   Engineering format that has exactly 15 significant
                        digits and a power that is a multiple of three.

     format may be used to switch between different output display formats
     of all numeric variables as follows:
       format HEX       Hexadecimal format.
       format +         The symbols +, - and blank are printed
                        for positive, negative and zero elements.
                        Imaginary parts are ignored.
       format BANK      Currency format with 2 digits after the decimal
                        point.
       format RATIONAL  Approximation by ratio of small integers. Numbers
                        with a large numerator or large denominator are
                        replaced by *.

     format may be used to affect the spacing in the display of all
     variables as follows:
```

```
  format COMPACT  Suppress excess blank lines to show more output.
  format LOOSE    Add blank lines to make output more readable.

Example:
   format short, pi, single(pi)
displays both double and single pi with 5 digits as 3.1416 while
   format long, pi, single(pi)
displays pi as 3.141592653589793 and single(pi) as 3.1415927.

   format, intmax('uint64'), realmax
shows these values as 18446744073709551615 and 1.7977e+308 while
   format hex, intmax('uint64'), realmax
shows them as ffffffffffffffff and 7fefffffffffffff respectively.
   fmt=format("hex"), intmax("uint64"), format(fmt)
shows ffffffffffffffff and restores the previous display format.
The HEX display corresponds to the internal representation of the value
and is not the same as the hexadecimal notation in the C programming
language.

See also disp, display, isnumeric, isfloat, isinteger.

Documentation for format
```

Vraťte formát výpisu do původního stavu.

```
format short
```

V matlabu exisuje velké množství vestavěných funkcí. Pro výpis základních matematických funkcí použijeme příkaz

```
help elfun
```

```
Elementary math functions.

Trigonometric.
  sin         - Sine.
  sind        - Sine of argument in degrees.
  sinh        - Hyperbolic sine.
  asin        - Inverse sine.
  asind       - Inverse sine, result in degrees.
  asinh       - Inverse hyperbolic sine.
  cos         - Cosine.
  cosd        - Cosine of argument in degrees.
  cosh        - Hyperbolic cosine.
  acos        - Inverse cosine.
  acosd       - Inverse cosine, result in degrees.
  acosh       - Inverse hyperbolic cosine.
  tan         - Tangent.
  tand        - Tangent of argument in degrees.
  tanh        - Hyperbolic tangent.
  atan        - Inverse tangent.
  atand       - Inverse tangent, result in degrees.
  atan2       - Four quadrant inverse tangent.
  atan2d      - Four quadrant inverse tangent, result in degrees.
  atanh       - Inverse hyperbolic tangent.
  sec         - Secant.
  secd        - Secant of argument in degrees.
  sech        - Hyperbolic secant.
  asec        - Inverse secant.
  asecd       - Inverse secant, result in degrees.
```

```
  asech       - Inverse hyperbolic secant.
  csc         - Cosecant.
  cscd        - Cosecant of argument in degrees.
  csch        - Hyperbolic cosecant.
  acsc        - Inverse cosecant.
  acscd       - Inverse cosecant, result in degrees.
  acsch       - Inverse hyperbolic cosecant.
  cot         - Cotangent.
  cotd        - Cotangent of argument in degrees.
  coth        - Hyperbolic cotangent.
  acot        - Inverse cotangent.
  acotd       - Inverse cotangent, result in degrees.
  acoth       - Inverse hyperbolic cotangent.
  hypot       - Square root of sum of squares.
  deg2rad     - Convert angles from degrees to radians.
  rad2deg     - Convert angles from radians to degrees.

Exponential.
  exp         - Exponential.
  expm1       - Compute exp(x)-1 accurately.
  log         - Natural logarithm.
  log1p       - Compute log(1+x) accurately.
  log10       - Common (base 10) logarithm.
  log2        - Base 2 logarithm and dissect floating point number.
  pow2        - Base 2 power and scale floating point number.
  realpow     - Power that will error out on complex result.
  reallog     - Natural logarithm of real number.
  realsqrt    - Square root of number greater than or equal to zero.
  sqrt        - Square root.
  nthroot     - Real n-th root of real numbers.
  nextpow2    - Next higher power of 2.

Complex.
  abs         - Absolute value.
  angle       - Phase angle.
  complex     - Construct complex data from real and imaginary parts.
  conj        - Complex conjugate.
  imag        - Complex imaginary part.
  real        - Complex real part.
  unwrap      - Unwrap phase angle.
  isreal      - True for real array.
  cplxpair    - Sort numbers into complex conjugate pairs.

Rounding and remainder.
  fix         - Round towards zero.
  floor       - Round towards minus infinity.
  ceil        - Round towards plus infinity.
  round       - Round towards nearest integer.
  mod         - Modulus (signed remainder after division).
  rem         - Remainder after division.
  sign        - Signum.
```

Pro výpis speciálních funkcí

```
help specfun
```

```
Specialized math functions.

Specialized math functions.
  airy        - Airy functions.
  besselj     - Bessel function of the first kind.
  bessely     - Bessel function of the second kind.
```

```
  besselh     - Bessel functions of the third kind (Hankel function).
  besseli     - Modified Bessel function of the first kind.
  besselk     - Modified Bessel function of the second kind.
  beta        - Beta function.
  betainc     - Incomplete beta function.
  betaincinv  - Inverse incomplete beta function.
  betaln      - Logarithm of beta function.
  ellipj      - Jacobi elliptic functions.
  ellipke     - Complete elliptic integral.
  erf         - Error function.
  erfc        - Complementary error function.
  erfcx       - Scaled complementary error function.
  erfinv      - Inverse error function.
  erfcinv     - Inverse complementary error function.
  expint      - Exponential integral function.
  gamma       - Gamma function.
  gammainc    - Incomplete gamma function.
  gammaincinv - Inverse incomplete gamma function.
  gammaln     - Logarithm of gamma function.
  psi         - Psi (polygamma) function.
  legendre    - Associated Legendre function.
  cross       - Vector cross product.
  dot         - Vector dot product.

Number theoretic functions.
  factor      - Prime factors.
  isprime     - True for prime numbers.
  primes      - Generate list of prime numbers.
  gcd         - Greatest common divisor.
  lcm         - Least common multiple.
  rat         - Rational approximation.
  rats        - Rational output.
  perms       - All possible permutations.
  nchoosek    - All combinations of N elements taken K at a time.
  factorial   - Factorial function.

Coordinate transforms.
  cart2sph    - Transform Cartesian to spherical coordinates.
  cart2pol    - Transform Cartesian to polar coordinates.
  pol2cart    - Transform polar to Cartesian coordinates.
  sph2cart    - Transform spherical to Cartesian coordinates.
  hsv2rgb     - Convert hue-saturation-value colors to red-green-blue.
  rgb2hsv     - Convert red-green-blue colors to hue-saturation-value.
```

Spočítejte sinus proměnné x a výslednou hodnotu uložte do y.

Spočítejte druhou odmocninu čísla -9 a výsledek uložte do proměnné z.

Kromě příkazu `help` můžeme použít stejným způsobem `doc`. Případně prohledat nápovědu.

```
doc floor
```

Zjistěte k čemu slouží funkce `randn` a jak se používá.

Vektory a matice

Všechny proměnné v matlabu jsou matice. Každá proměnná může obsahovat více hodnot.

skalár - matice obsahující jeden prvek

```
a = 10
```

```
a = 10
```

řádkový vektor

prvky píšeme do hranatých závorek a oddělujeme je mezerou (případně čárkou)

```
v1 = [1 2 3]
```

```
v1 = 1×3
     1     2     3
```

sloupcový vektor

oddělíme-li prvky `;` vynikne sloupcový vektor

```
v2 = [1; 2; 3]
```

```
v2 = 3×1
     1
     2
     3
```

matice

oddělení prvků mezerou a `;` můžeme kombinovat a tím vytvářet matice.

Matice vytváříme řádek po řádku, ty oddělujeme `;`

```
m = [1 2; 3 4]
```

```
m = 2×2
     1     2
     3     4
```

Uvnitř závorek můžeme používat výpočty, proměnné a jiné.

Vytvořte proměnné x a y a přiřaďte jim nějakou hodnotu. Poté vytvořte matici M, kde první řádek bude obsahovat hodnoty x a absolutní hodnotu x a druhý y a absolutní hodnotu y.

Speciální vektory a matice

vektory s prvky, které se liší o stejnou hodnotu

např. [1 2 3 4]

operátor `:`

`prvni_prvek : krok : posledni_prvek`

pokud `krok` vynecháme, je krok roven 1

```
v1 = 1 : 10
```

```
v1 = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```
v2 = 1 : 2 : 10
```

```
v2 = 1×5
     1     3     5     7     9
```

Co nejelegantněji vytvořte vektor v = [10 9 8 7 ... 1]

Jak bychom vytvořili co nejelegantněji vektor, který má 5 prvků, první prvek roven 0, poslední 100?

Pokud víme počet prvků vektoru, první a poslední prvek, ale nevíme, jaký je krok mezi prvky, můžeme použít příkaz `linspace(prvni_prvek, posledni_prvek, pocet_prvku)`

```
v3 = linspace(0, 100, 5)
```

```
v3 = 1×5
     0    25    50    75   100
```

Sloupcový vektor můžeme z řádkového (a naopak) vytvořit pomocí operátoru `'` (transponování).

```
v4 = v3'
```

```
v4 = 5×1
     0
    25
    50
    75
   100
```

Speciální matice vytvoříme například pomocí následujících příkazů.

- `rand(m)` - náhodná čtvercová matice o velikosti m x m

- `rand(m, n)` - náhodná matice velikosti m x n (m řádků, n sloupců)

```
m1 = rand(2)
```

```
m1 = 2×2
    0.8147    0.1270
    0.9058    0.9134
```

- `randi(max, m)`, `randi(max, m, n)` - matice obsahující náhodná celá čísla od 0 do max

```
m2 = randi(3,2)
```

```
m2 = 2×2
     2     1
     1     2
```

- `ones(m)`, `ones(m, n)` - matice obsahující samé 1

```
m3 = ones(2)
```

```
m3 = 2×2
     1     1
     1     1
```

- `zeros(m)`, `zeros(m, n)` - nulová matice

```
m4 = zeros(2)
```

```
m4 = 2×2
     0     0
     0     0
```

- `eye(m)`, `eye(m, n)` - jednotková matice

```
m5 = eye(2)
```

```
m5 = 2×2
     1     0
     0     1
```

Pomocí hranatých závorek můžeme spojovat vektory a matice. Je nutné dávat pozor na rozměry.

Matice se stejným počtem řádků můžeme spojit následujícím způsobem.

```
m6 = [m3 m4]
```

```
m6 = 2×4
     1     1     0     0
     1     1     0     0
```

Případně matice se stejným počtem sloupců můžeme sloučit do jedné následovně.

```
m7 = [m3; m4]
```

```
m7 = 4×2
     1     1
     1     1
     0     0
     0     0
```

Případně můžeme použít příkazy `horzcat()`, `vertcat()` a `cat()`. Pro více informací použijte `help`.

Co nejelegantněji vytvořte následující matici A.

A = [1 1 0 0;

1 1 0 0;

0 0 1 1;

0 0 1 1]

Práce s vektory a maticemi

V Matlabu indexujeme od 1 a indexy píšeme do kulatých závorek.

```
v = 0 : 5
```

```
v = 1×6
     0     1     2     3     4     5
```

```
v(1)
```

```
ans = 0
```

Vyzkoušejte

```
v(0)
```

```
Array indices must be positive integers or logical values.
```

Změna hodnoty

```
v(5) = 10
```

```
v = 1×6
     0     1     2     3    10     5
```

Můžeme indexovat více prvků zároveň.

První tři hodnoty vektoru získáme následujícím způsobem

```
v(1:3)
```

```
ans = 1×3
     0     1     2
```

Případně výpisem indexů (ty zadáváme jako vektor).

```
v([1 3 5])
```

```
ans = 1×3
     0     2    10
```

Pokud chceme získat poslední prvek, můžeme místo inxedu použít `end`

```
v(end)
```

```
ans = 5
```

Je možné použít i `end - 1` pro předposlední prvek a pod.

Indexovat můžeme i pomocí logického vektoru stejné velikosti.

Logický vektor můžeme získat relačními operacemi, například `v > 2`

```
w = v(v > 2)
```

```
w = 1×3
     3    10     5
```

Porovnání >, <, >=, <=, ==, ~=

Logické spojky ~, | a &

Při indexaci matice zadáváme indexy dva (řádek, sloupec).

```
M = [1 2; 3 4]
```

```
M = 2×2
     1     2
     3     4
```

```
M(2, 1)
```

```
ans = 3
```

Indexovat se dá i pomocí jediného indexu. Pak se prvky počítají po sloupcích.

Pro výběr celého řádku (všechny sloupce řádku) můžeme místo druhého indexu použít `:`

Obdobně pro výběr celého sloupce.

```
M(1, :)
```

```
ans = 1×2
     1     2
```

```
M(:, 1)
```

```
ans = 2×1
     1
     3
```

Nahraďte první sloupec matice M druhým sloupcem. Výsledná matice by měla vypadat:

[2 2;

4 4]

Nastavením hodnoty prvku matice, který je mimo její rozměr, dojde k jejímu zvětšení.

```
M(3,3) = 5
```

```
M = 3×3
     1     2     0
     3     4     0
     0     0     5
```

Všechny nulové hodnoty v matici M nahrďte číslem 3.

Velikost matice (vektoru) získáme pomocí funkce

```
size(M)
```

```
ans = 1×2
     3     3
```

Uložení výsledku do jedné proměnné

```
velikost = size(M)
```

```
velikost = 1×2
     3     3
```

Uložení do více proměnných

```
[radky, sloupce] = size(M)
```

```
radky = 3
sloupce = 3
```

Pokud nás zajímá jen počet sloupců, můžeme použít jednu z následujcích možností

```
sloupce = size(M,2)
```

```
sloupce = 3
```

```
[~, sloupce] = size(M)
```

```
sloupce = 3
```

Matice mohou být i více rozměrné

```
M3 = zeros(4,3,2)
```

```
M3 =
M3(:,:,1) =

     0     0     0
     0     0     0
     0     0     0
```

```
     0     0     0


M3(:,:,2) =

     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

```
size(M3)
```

```
ans = 1×3
     4     3     2
```

Operace s maticemi (vektory)

Aritmetické operace matice a skaláru

```
A = [1 2; 3 4]
```

```
A = 2×2
     1     2
     3     4
```

```
B = A + 2
```

```
B = 2×2
     3     4
     5     6
```

Aritmetické operace (+, -, *, /) matic

```
C = A + B
```

```
C = 2×2
     4     6
     8    10
```

```

```

* a / je klasické maticové násobení a dělení. Pokud chceme násobit nebo dělit prvek po prvku, musíme použít operátory .* a ./

```
D = A * B
```

```
D = 2×2
    13    16
    29    36
```

```
E = A .* B
```

```
E = 2×2
     3     8
    15    24
```

Další funkce, které se nám mohou při práci s maticemi hodit jsou `max()` a `min()` pro hledání největšího, respektive nejmenšího prvku.

```
v = [1 10 8 4 1]
```

```
v = 1×5
     1    10     8     4     1
```

```
max_v = max(v)
```

```
max_v = 10
```

Funkce může kromě hodnoty vracet také index, kde se tato hodnota nachází.

```
[hodnota, index] = max(v)
```

```
hodnota = 10
index = 2
```

Pokud je prvků s maximální hodnotou více, vrací první z nich. Pokud bychom potřebovali všechny indexy, použijeme funkci `find()`.

Zjistěte, jak se funkce `find()` *používá. Najděte nejmenší hodnotu a indexy této hodnoty ve vektoru* `v`*.*

Pro matice vrací funkce největší (respektive nejmenší) prvek v každém sloupci. Pokud chceme největší prvek, použijeme funkci dvakrát

```
max_M = max(max(M))
```

```
max_M = 5
```

Úkoly k procvičení

- *Vytvořte co nejelegantněji následující matici*

```
B = [1 2 3 1 4 7 1 2 3;
4 5 6 2 5 8 4 5 6;
7 8 9 3 6 9 7 8 9]
```

- *Určete počet prvků* `s` *větších než 3 v matici* `B` *(je jich 18). Zjistěte logické a maticové souřadnice těchto prvků.*

- *Všechny číslice 5 v matici* `B` *nahraďte 0.*

- *Najděte funkci, pomocí které můžeme spočítat součet všech prvků vektoru (matice). Spočítejte součet všech prvků v matici* `B`.

- *Matlab obsahuje několik testovacích datasetů. Například* `patients.mat` *obsahující data 100 smyšlených pacientů. Z tohoto datasetu načtěte pouze proměnné* `Height` a `Weight` *a spočítejte BMI index pro všech 100 pacientů.* `BMI` *proměnnou spolu s* `Height` *a* `Weight` *uložte do souboru* `bmi.mat`.

- *Nastavte formát výpisu tak, aby hodnotu 0.5 vypsal jako zlomek 1/2.*

- *Vytvořte vektor* `hody` *obsahující 100 hodnot neprezentujících hody kostkou (obsahuje hodnoty 1, ..., 6). Vytvořte vektor* `statistika`*, který bude obsahovat 6 hodnot. První představuje počet 1 ve vektoru* `hody`*, druhý počet 2, ...*

- *Zeptejte se MATLABu proč* (`why`).

```
why
```

```
The not very terrified mathematician wanted it that way.
```