POČÍTAČOVÁ PODPORA V ELEKTROTECHNICE

Ing. Lenka Šroubová, Ph.D.
lsroubov@kte.zcu.cz

Ing. Petr Kropík, Ph.D.
pkropik@kte.zcu.cz

Katedra teoretické elektrotechniky
Fakulta elektrotechnická
Západočeská univerzita v Plzni

Místnost: EK602

FACULTY OF ELECTRICAL ENGINEERING
UNIVERSITY OF WEST BOHEMIA

Polynomiální regrese

Příklad: Vytvoření uživatelské funkce **prolozeni_bodu** pro polynomiální regresi:
Vstupní data: x-ové a y-ové souřadnice bodů (2 vektory x, y), stupeň polynomu n, kterým budou body proloženy.

Nejprve provedeme jemnější dělení osy x pro zobrazení křivky (od minima z vektoru x do maxima z vektoru x s krokem vypočteným jako (max(x)-min(x))/100), poté získáme koeficienty polynomu n-tého stupně pro polynomiální regresi příkazem **polyfit** a vyčíslíme hodnoty polynomu pro všechny body z jemnějšího dělení osy x příkazem **polyval**. Nakonec vykreslíme graf.

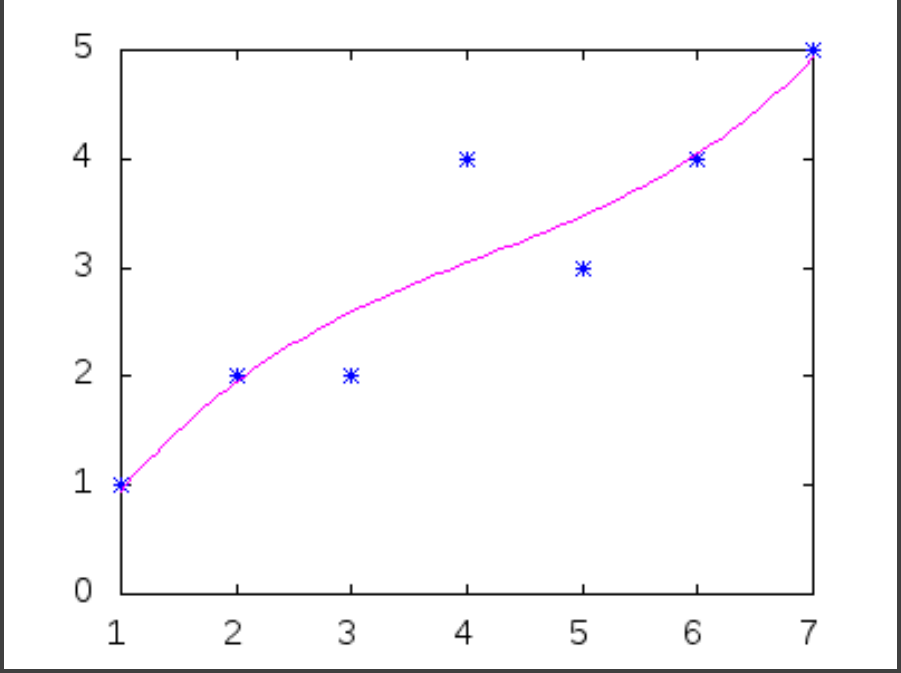
```
function prolozeni_bodu(x,y,n)
xp = [min(x):((max(x)-min(x))/100):max(x)];
yn = polyval(polyfit(x,y,n),xp);
plot(x,y,'*',xp,yn,'m')
end
```

Polynomiální regrese

Pokračování příkladu: Volání funkce **`prolozeni_bodu`** pro body s x-ovými a y-ovými souřadnicemi danými tabulkou. Stupeň polynomu, kterým budou body proloženy, je **3**.

x	1	2	3	4	5	6	7
y	1	2	2	4	3	4	5

```
prolozeni_bodu([1:7],[1,2,2,4,3,4,5],3)
```



Polynomiální regrese

Příklad: Vytvoření funkce `proloz()`, která proloží body se souřadnicemi x, y polynomem zadaného stupně. Funkce umožní uživateli prokládat body pouze polynomem 2., 3., 5. nebo 7. stupně, v jiných případech bude uživateli oznámeno, že data proložit nelze a funkce bude ukončena. Polynom (křivka) 2. stupně bude vykreslen červeně, 3. stupně zelenou barvou, 5. stupně černě a polynom 7. stupně žlutou barvou.
Vstupní parametry: souřadnice x, y (2 vektory x, y) a stupeň polynomu st.

Připomeňme:
`return` – ukončení funkce.

Pokud není zvolena možnost, která je v nabídce (2., 3., 5. nebo 7. stupeň polynomu), musíme ukončit funkci `proloz()` včas, aby se nezačal kreslit graf a vypisovat chybné hlášky, protože nebude známo, jakou křivku kreslit …

```
function proloz(x,y,st)
plot(x,y,'*');
switch st
    case 2
        pol=polyfit(x,y,2); barva='r';
    case 3
        pol=polyfit(x,y,3); barva='g';
    case 5
        pol=polyfit(x,y,5); barva='k';
    case 7
        pol=polyfit(x,y,7); barva='y';
    otherwise
        disp('Tato možnost v nabídce není');
        return
end
xx=linspace(min(x),max(x),1000);
hp=polyval(pol,xx);
hold on
plot(xx,hp,barva);
hold off
end
```

Polynomiální regrese

Pokračování příkladu:

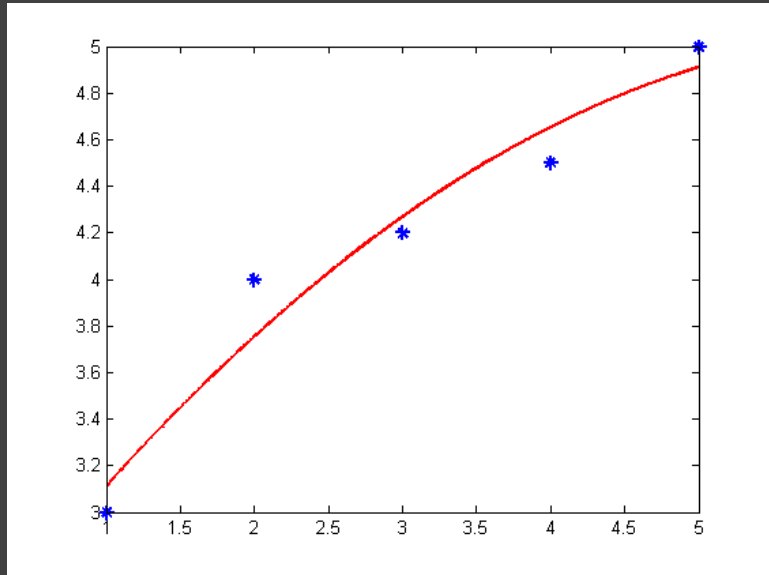
Volání funkce **proloz()** pro body s x-ovými souřadnicemi od **1** do **10** a y-ovými souřadnicemi – náhodnými čísly od **0** do **10**. Stupeň polynomu je **4**.

```
proloz([1:10],round(rand(1,10).*10),4)
Tato možnost v nabídce není
```

Volání funkce **proloz()** pro body se souřadnicemi danými tabulkou. Stupeň polynomu je **2**.

x	1	2	3	4	5
y	3	4	4,2	4,5	5

```
proloz(1:5,[3,4,4.2,4.5,5],2)
```



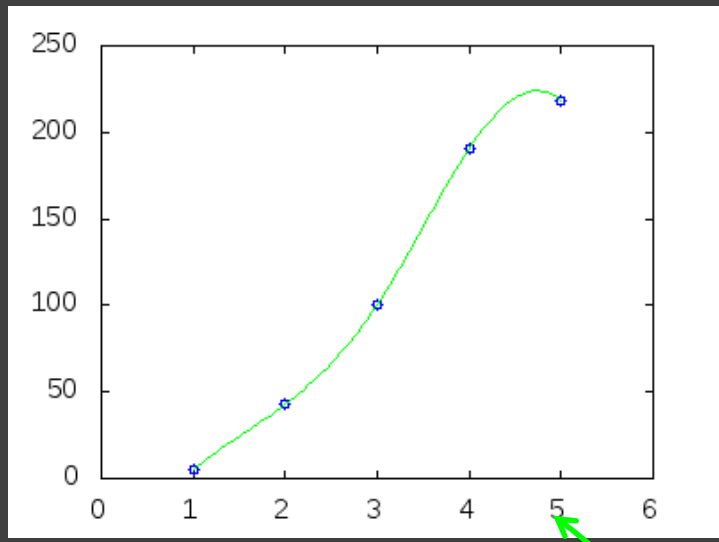
Interpolace

Lze použít i další typy křivek pro proložení bodů, nejen výše uvedené polynomy

Např.: **spline**

Použitelné funkce:

interp1 – 1Dimenzionální interpolace
interp2 – 2D interpolace
interp3 – 3D interpolace
interpn – nDimenzionální interpolace
interpft – interpolace s využitím FFT



Např. 1D interpolace dat v tabulce pomocí spline funkcí:

t[s]	1	2	3	4	5
u[V]	5.5	43.1	100.2	190.7	218.4

```
t = [1:5]; u = [5.5,43.1,100.2,190.7,218.4];
rt = [0:0.05:6];
spl_int = interp1(t,u,rt,'spline');
plot(t,u,'o',rt,spl_int, 'g')
```

Interpolace

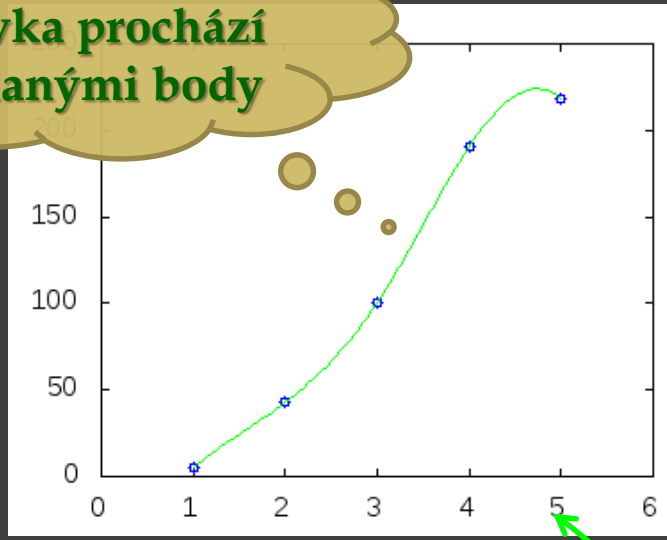
Lze použít i další typy křivek pro proložení bodů, nejen výše uvedené polynomy

Např.: **spline**

Použitelné funkce:

interp1 – 1Dimenzionální interpolace
interp2 – 2D interpolace
interp3 – 3D interpolace
interpn – nDimenzionální interpolace
interpft – interpolace s využitím FFT

křivka prochází zadanými body



Např. 1D interpolace dat v tabulce pomocí spline funkcí:

t[s]	1	2	3	4	5
u[V]	5.5	43.1	100.2	190.7	218.4

```
t = [1:5]; u = [5.5,43.1,100.2,190.7,218.4];
rt = [0:0.05:6];
spl_int = interp1(t,u,rt,'spline');
plot(t,u,'o',rt,spl_int, 'g')
```

Grafy

`subplot(m,n,p)` – rozdělení grafického okna na *m* * *n* částí, kde

`m` – počet řádků v grafickém okně,

`n` – počet sloupců v grafickém okně,

`p` – číslo obrázku v grafickém okně,

číslování jednotlivých obr. – zleva doprava, shora dolů, tedy:

```
subplot(kolik_radku_s_grafy,kolik_sloupcu_s_grafy,ktery_graf)
```

Např. rozdělení grafického okna na 6 částí: 2 řádky, 3 sloupce (2*3=6)

`subplot(2,3,1)` a něco vykreslíme do 1. části grafického okna;

`subplot(2,3,2)` a něco vykreslíme do 2. části grafického okna;

`subplot(2,3,3)` a něco vykreslíme do 3. části grafického okna;

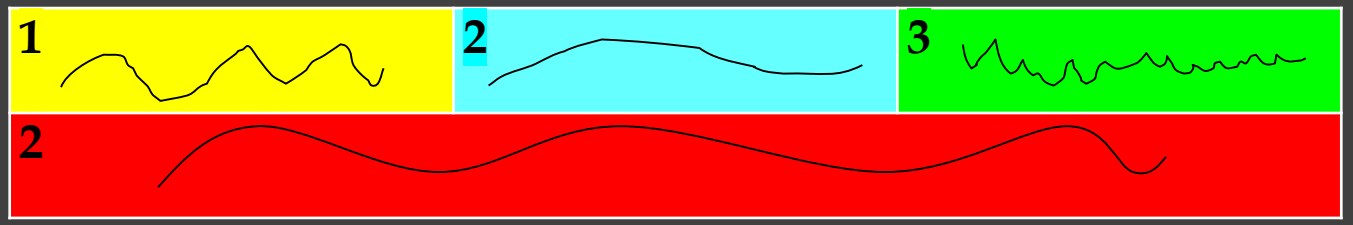
`subplot(2,3,4)` a něco vykreslíme do 4. části grafického okna;

`subplot(2,3,5)` a něco vykreslíme do 5. části grafického okna;

`subplot(2,3,6)` a něco vykreslíme do 6. části grafického okna.

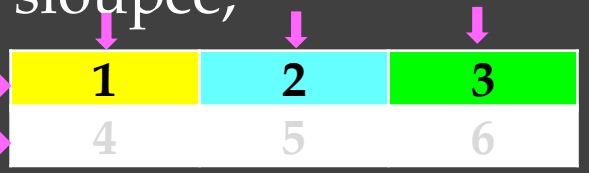
Grafy

Grafické okno lze rozdělit i např. takto:

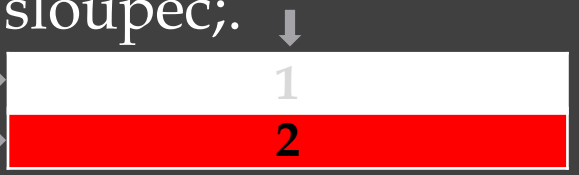


Jak na to?

`subplot(2,3,1)` a něco vykreslíme do 1. části grafického okna;
`subplot(2,3,2)` a něco vykreslíme do 2. části grafického okna;
`subplot(2,3,3)` a něco vykreslíme do 3. části grafického okna;
jako kdyby bylo rozděleno na 6 částí: 2 řádky, 3 sloupce;



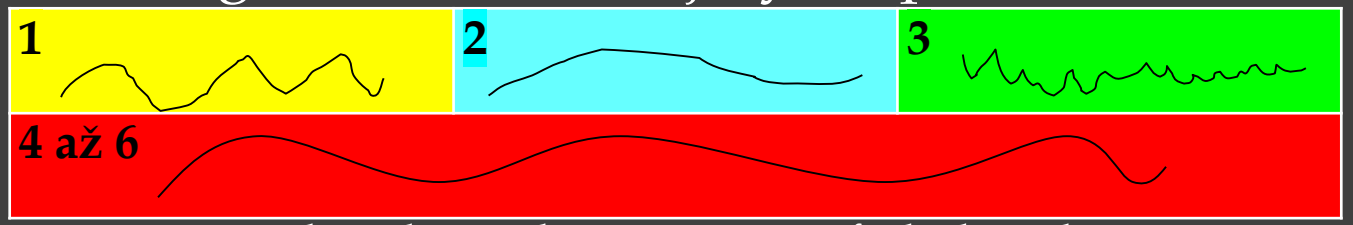
`subplot(2,1,2)` a něco vykreslíme do 2. části grafického okna,
jako kdyby bylo rozděleno na 2 částí: 2 řádky, 1 sloupec;.



Subgrafy se chovají jako samostatné – mají svůj titulek, popisy, své nastavení `hold`, `grid` atd.

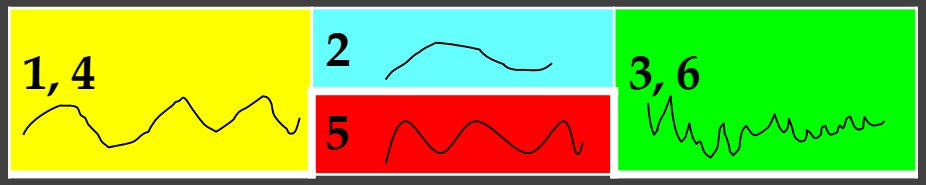
Grafy

Nebo stejné rozdělení grafického okna jiným způsobem:

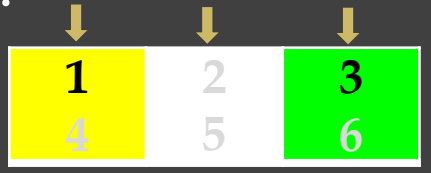


`subplot(2,3,1)` a něco vykreslíme do 1. části grafického okna;
`subplot(2,3,2)` a něco vykreslíme do 2. části grafického okna;
`subplot(2,3,3)` a něco vykreslíme do 3. části grafického okna;
`subplot(2,3,4:6)` a něco vykreslíme do 4. - 6. části grafického okna;

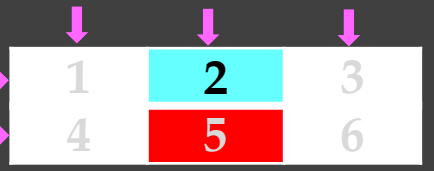
A jiný příklad: rozdělení grafického okna na 4 části:



`subplot(1,3,1)`
`subplot(1,3,3)`



`subplot(2,3,2)`
`subplot(2,3,5)`



nebo

`subplot(2,3,[1,4])` a něco vykreslíme do 1. a 4. části grafického okna;
`subplot(2,3,2)` a něco vykreslíme do 2. části grafického okna;
`subplot(2,3,[3,6])` a něco vykreslíme do 3. a 6. části grafického okna;
`subplot(2,3,5)` a něco vykreslíme do 5. části grafického okna;

Grafy

`plot(x,y,S)` – rovinný graf s **lineárním** dělením na osách x, y (tj. „klasický“ graf x, y),), jsou-li x a y vektory o stejné délce, pak **`plot(x,y)`** vykreslí x-y graf,
`semilogx(x,y,S)` – graf jako **`plot`**, ale s **logaritmickou** stupnicí (o základu 10) **na ose** x,
`semilogy(x,y,S)` – graf jako **`plot`**, ale s **logaritmickou** stupnicí (o základu 10) **na ose** y,
`loglog(x,y,S)` – graf jako **`plot`**, ale s **logaritmickou** stupnicí (o základu 10) **na obou osách** x, y.

Parametr **`S`** je **řetězec**, který musí být v apostrofech (některé výpočetní systémy připouštějí i uvozovky) a specifikuje barvu, způsob vykreslení a styl křivky nebo typ značky bodu, např.
`plot(x,y,'ro')` – body zobrazeny jako červená kolečka,
`semilogx(x,y,'y--')` – křivka zobrazena čárkovaně žlutou barvou,
`loglog(x,y,'g*-')` – body zobrazeny jako zelené hvězdičky spojené plnou čarou.

Pozn. viz **`help plot`** (barvy a typy bodů, styly čar – viz předchozí přednášky)

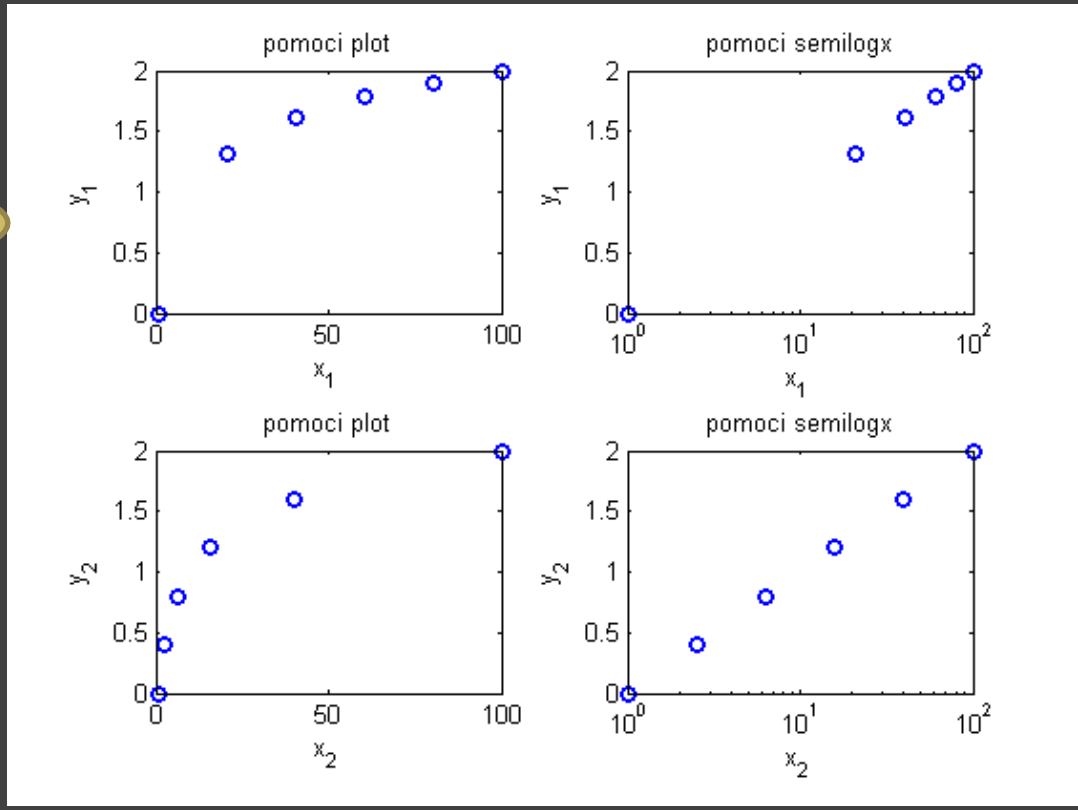
Grafy

Příklad: Vykreslení grafu funkce $y = \log(x)$ pro x od 1 do 100

```
x1=linspace(1,100,6);
y1=log10(x1);
x2=logspace(0,2,6);
y2=log10(x2);
subplot(2,2,1)
plot(x1,y1,'o')
xlabel('x_1')
ylabel('y_1')
title('pomoci plot')
subplot(2,2,2)
semilogx(x1,y1,'o')
xlabel('x_1')
ylabel('y_1')
title('pomoci semilogx')
subplot(2,2,3)
plot(x2,y2,'o')
xlabel('x_2')
ylabel('y_2')
title('pomoci plot')
```

$10^2=100$

$10^0=1$

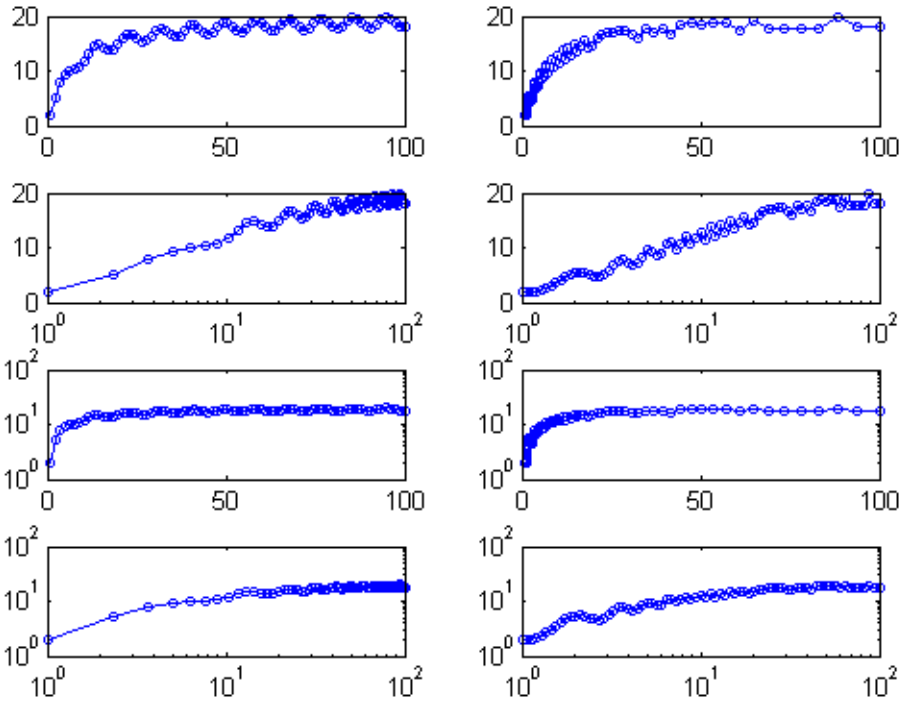



```
subplot(2,2,4)
semilogx(x2,y2,'o')
xlabel('x_2')
ylabel('y_2')
title('pomoci semilogx')
```

Grafy

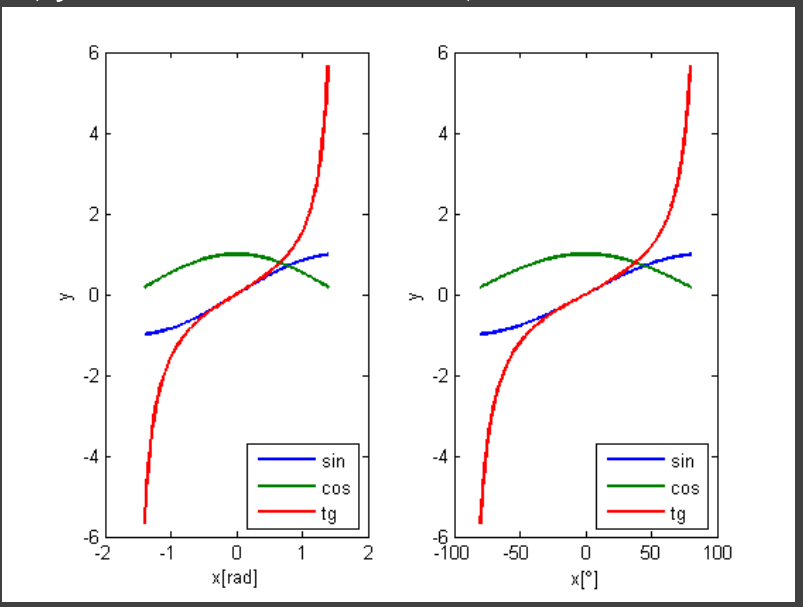
Příklad: Vykreslení grafu funkce $y = e^{(3x)/(x+2)} + \sin(4x)$ pro x od 1 do 100 s lineárním i logaritmickým dělením na osách

```
x1 = linspace(1,100,75);
x2 = logspace(0,2,75);
y1 = exp((3.*x1)./(x1+2))+sin(4.*x1);
y2 = exp((3.*x2)./(x2+2))+sin(4.*x2);
subplot(4,2,1)
plot(x1,y1,'o-')
subplot(4,2,2)
plot(x2,y2,'o-')
subplot(4,2,3)
semilogx(x1,y1,'o-')
subplot(4,2,4)
semilogx(x2,y2,'o-')
subplot(4,2,5)
semilogy(x1,y1,'o-')
subplot(4,2,6)
semilogy(x2,y2,'o-')
subplot(4,2,7)
loglog(x1,y1,'o-')
subplot(4,2,8)
loglog(x2,y2,'o-')
```



Grafy

Příklad: Vykreslení grafu goniometrických funkcí $y_1 = \sin(t)$, $y_2 = \cos(t)$, $y_3 = \mathrm{tg}(t)$ pro t od $-4\pi/9$ do $4\pi/9$ (tj. od -80° do 80°).



Funkce `sin`, `cos`, `tan`, `cot` – argument v radiánech, funkce `sind`, `cosd`, `tand`, `cotd` – argument ve stupních.

```
xr = linspace(-4*pi/9,4*pi/9,100);
y1r = sin(xr);
y2r = cos(xr);
y3r = tan(xr);
subplot(1,2,1)
plot(xr,y1r,xr,y2r,xr,y3r)
xlabel('x[rad]')
ylabel('y')
legend('sin','cos','tg',...
'Location','SouthEast')
xd = linspace(-80,80,100);
y1d = sind(xd);
y2d = cosd(xd);
y3d = tand(xd);
subplot(1,2,2)
plot(xd,y1d,xd,y2d,xd,y3d)
xlabel('x[\circ]')
ylabel('y')
legend('sin','cos','tg',...
'Location','SouthEast')
```

argument v radiánech

argument ve stupních

Funkce pro práci s maticemi

det() – **determinant** matice (lze jen pro čtvercové matice)
rank() – **hodnost** matice, stanoví odhad počtu lineárně nezávislých řádků nebo sloupců matice
cond() – číslo podmíněnosti, charakterizuje **podmíněnost** matice, je-li veliké, je matice špatně podmíněná
rcond() – číslo, které také charakterizuje **podmíněnost** matice, je-li nulové nebo velmi blízké nule, je matice špatně podmíněná
eig() – **vlastní čísla** matice

```
T = [1,2,3;2,4,6;3,6,9]
T =
     1     2     3
     2     4     6
     3     6     9
```

```
L = eig(T)
L =
     0
     0
    14
```

Pro vlastní čísla platí

```
det(T - L(1)*eye(3))
ans = 0
det(T - L(3)*eye(3))
ans = 1.12e-013 → 0
```

```
det(T)
ans =   0
```

```
rank(T)
ans =   1 < 3
```

```
cond(T)
ans =   Inf → ∞
```

```
rcond(T)
ans =   0
```

Funkce pro práci s maticemi

Regulární matice (např. `R = [3,2;4,1]`)
je taková čtvercová matice,

- jejíž determinant je různý od nuly,
- její řádky jsou lineárně nezávislé,
- její sloupce jsou lineárně nezávislé,
- hodnost čtvercové regulární matice o velikosti $n \times n$ je právě n,
- existuje k ní inverzní matice,
- všechna její vlastní čísla jsou nenulová.

Singulární matice (např. `S = [9,3;6,2]`)
je taková čtvercová matice,

- jejíž determinant je roven nule,
- její řádky nejsou lineárně nezávislé (jsou lineárně závislé) ,
- její sloupce nejsou lineárně nezávislé (jsou lineárně závislé),
- hodnost čtvercové singulární matice o velikosti $n \times n$ je menší než n,
- neexistuje k ní inverzní matice.

```
R =
     3     2
     4     1
```

```
eig(R)
ans =
     5  ≠ 0
    -1  ≠ 0
```

```
det(R)
ans = -5 ≠ 0
```

```
cond(R)   ≠ ∞
ans = 5.83
```

```
rank(R)
ans = 2
```

```
rcond(R)
ans = 0.14  ≠ 0
```

```
S =
     9     3
     6     2
```

```
eig(S)
ans =
    11    ≠ 0
     0    = 0
```

```
det(S)
ans = 0
```

```
cond(S)  → ∞
ans = 2.71e+016
```

```
rank(S)
ans =   1 < 2
```

```
rcond(S)
ans =   0
```

Řešení soustavy lineárních rovnic

Příklad: řešení soustavy lineárních algebraických rovnic:

$$3x_1 + 4x_2 = 11$$
$$2x_1 - 5x_2 = -8$$

```
det([3,4;2,-5])
ans = -23
```
$\neq 0$

```
cond([3,4;2,-5])
ans =  1.7888
```
$\neq \infty$
– OK, dobře podmíněná matice

```
eig([3,4;2,-5])
ans = 3.8990
     -5.8990
```
$\neq 0$
$\neq 0$

Hodnost matice koeficientů soustavy

```
rank([3,4;2,-5])
ans =  2
```
– OK, máme 2 rovnice

Hodnost rozšířené matice (včetně pravých stran)

```
rank([3,4,11;2,-5,-8])
ans =  2
```
– OK, máme 2 rovnice, 2 = 2

```
rcond([3,4;2,-5])
ans = 0.36508
```
$\neq 0$

```
x = [3,4;2,-5] \ [11;-8]
x =
   1.00
   2.00
```

Je-li matice koeficientů soustavy **regulární**, potom má soustava **právě jedno řešení**.

Řešení soustavy lineárních rovnic

Příklad: řešení soustavy lineárních algebraických rovnic:

$$x_1 + x_2 = 11$$
$$2x_1 + 2x_2 = -8$$

```
det([1,1;2,2])
ans = 0
```

```
cond([1,1;2,2])
ans =  3.8442e+16 → ∞
```
– špatně podmíněná matice

```
rcond([1,1;2,2])
ans = 0
```

```
eig([1,1;2,2])
ans = 0    = 0
      3    ≠ 0
```

Hodnost matice koeficientů soustavy

```
rank([1,1;2,2])
ans =  1
```
– pouze 1 lineárně nezávislý řádek a 2 rovnice

Hodnost rozšířené matice (včetně pravých stran)

```
rank([1,1,11;2,2,-8])
ans =  2
```
1 < 2

```
x = [1,1;2,2] \ [11;-8]
Warning: Matrix is singular
to working precision.
```

```
x =
   -Inf
    Inf
```

Je-li matice koeficientů soustavy **singulární** a vektor pravých stran je takový, že rovnice jsou **lineárně nezávislé**, potom soustava **nemá řešení**. Vypočte-li výpočetní systém řešení a provedeme-li zkoušku, nevychází.

Řešení soustavy lineárních rovnic

Příklad: řešení soustavy lineárních algebraických rovnic:

$$x_1 + x_2 = 10$$
$$2x_1 + 2x_2 = 20$$

```
det([1,1;2,2])
ans = 0
```

```
rcond([1,1;2,2])
ans = 0
```

```
inv([1,1;2,2])
ans =
   Inf   Inf
   Inf   Inf
```

```
eig([1,1;2,2])
ans = 0    = 0
      3    ≠ 0
```

Hodnost matice koeficientů soustavy

```
rank([1,1;2,2])
ans =  1
```

Hodnost rozšířené matice (včetně pravých stran)

```
rank([1,1,10;2,2,20])
ans =  1          1 = 1 < 2
```

```
cond([1,1;2,2])
ans =  3.8442e+16 → ∞
```

– špatně podmíněná matice

```
x = [1,1;2,2] \ [10;20]
Warning: Matrix is singular
to working precision.
```

```
x =
   NaN
   NaN
```

Je-li matice koeficientů soustavy **singulární** a vektor pravých stran je takový, že rovnice jsou **lineárně závislé**, potom soustava **má nekonečně mnoho řešení**. Výpočetní systém může vypočíst jedno z mnoha řešení a provedeme-li zkoušku, vychází správně.

Řešení soustavy lineárních rovnic

Příklad: řešení soustavy lineárních algebraických rovnic:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 15 \\ 4x_1 + 5x_2 + 6x_3 &= 20 \\ 7x_1 + 8x_2 + 9x_3 &= -16 \end{aligned}$$

```
L = [1,2,3;4,5,6;7,8,9];
p = [15;20;-16]
```

```
det(L)
ans = 6.6613e-016
```
→ 0

```
cond(L)
ans = 3.8131ee+16
```
→ ∞

– špatně podmíněná matice

```
rank(L)
ans =  2
```
– pouze 2 lineárně nezávislé řádky a 3 rovnice

```
rank([L,p])
ans =  3
```
2 < 3

```
eig(L)
ans =
   16.1168
   -1.1168
    0.0000
```
≠ 0, ≠ 0, = 0

```
rcond(L)
ans = 1.5420e-018
```
→ 0

```
x = L \ p
Warning: Matrix is close
to singular or badly
scaled.
Results may be inaccurate.
```

```
x =
1.0e+017 *
    1.8465
   -3.6930
    1.8465
```

Soustava **nemá řešení**.

Vypočte-li výpočetní systém řešení a provedeme-li zkoušku, nevychází.

Řešení soustavy lineárních rovnic

Příklad: řešení soustavy lineárních algebraických rovnic:

$$x_1 + 5x_2 + 8 = 40$$
$$x_3 - 9 = 2 - 2x_1$$
$$3x_3 + 36 = 9x_2$$

$$1x_1 + 5x_2 + 0x_3 = 32$$
$$2x_1 + 0x_2 - 1x_3 = 11$$
$$0x_1 - 9x_2 + 3x_3 = -36$$

```
G = [ 1, 5, 0; 2, 0,-1; 0,-9, 3]; h = [32; 11; -36];
```

```
det(G)
ans = -39
```
$\neq 0$

```
cond(G)
ans = 6.4693
```
$\neq \infty$

– dobře podmíněná matice

```
rcond(G)
ans = 0.084416
```
$\neq 0$

```
rank(G)
ans =  3
```
$3 = 3$
```
rank([G,h])
ans =  3
```

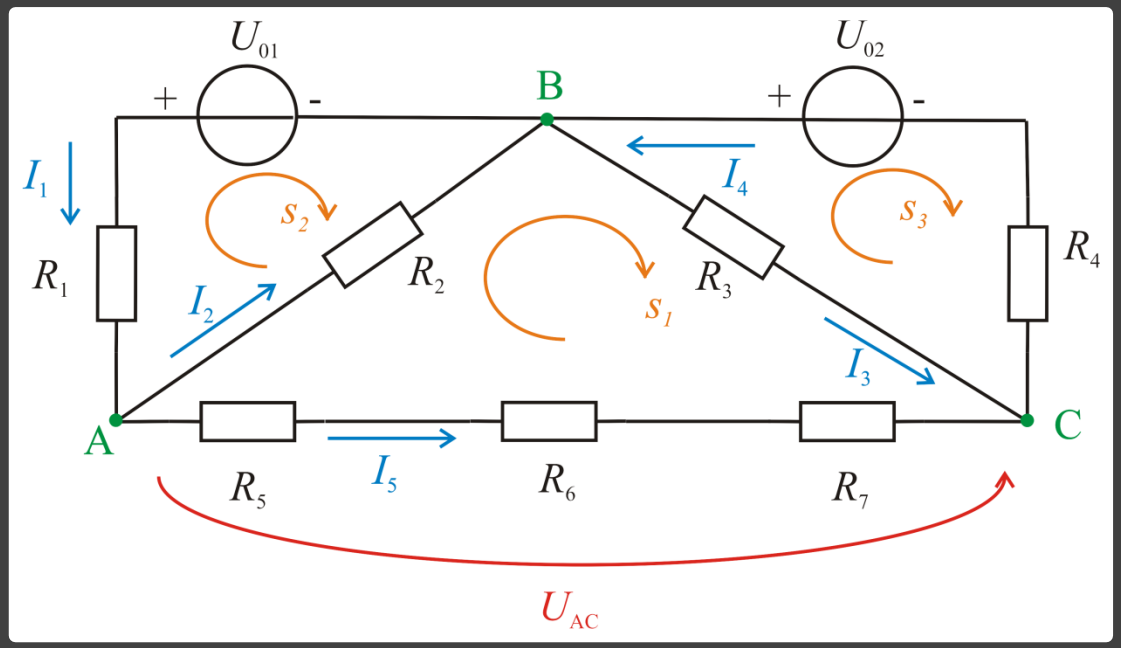
```
eig(G)
ans =
   -3.5629
    1.9502
    5.6127
```
$\neq 0$ $\neq 0$ $\neq 0$

```
x = G \ h
x =
    7.0000
    5.0000
    3.0000
```

Matice G je **regulární** (není singulární) => soustava rovnic má **právě jedno řešení**.

Řešení soustavy lineárních rovnic

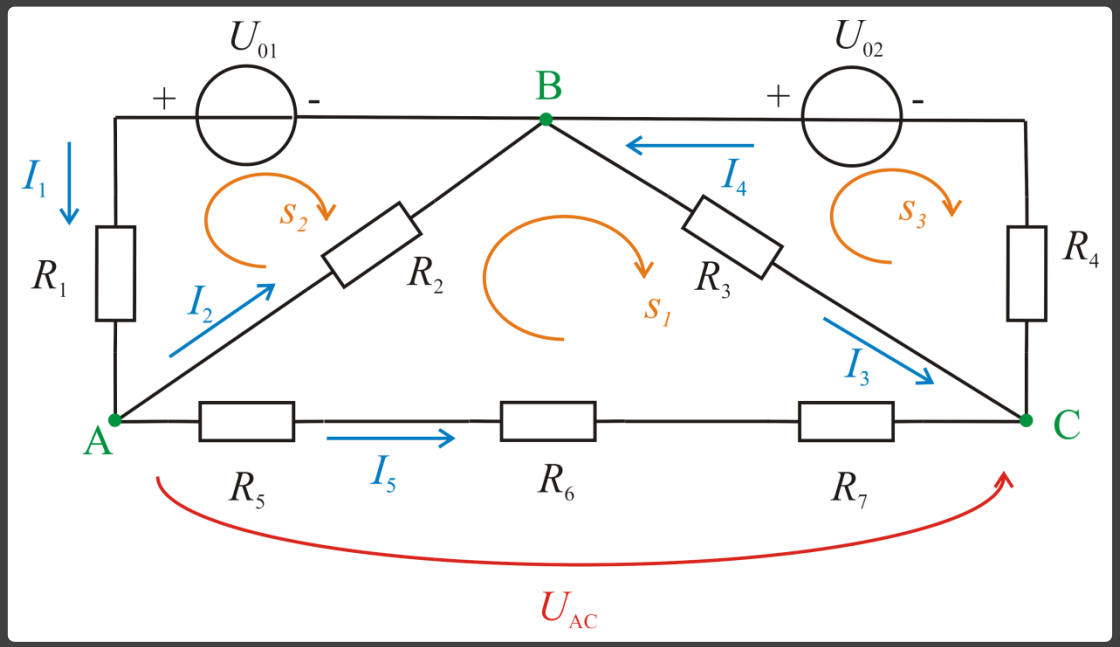
Příklad: v elektrickém obvodu se dvěma stejnosměrnými zdroji napětí na obr. určete proudy I_1, I_2, I_3, I_4, I_5, je-li dáno:
$R_1 = 3\ \Omega$, $R_2 = 2\ \Omega$, $R_3 = 3\ \Omega$, $R_4 = 6\ \Omega$, $R_5 = 5\ \Omega$, $R_6 = 6\ \Omega$, $R_7 = 5\ \Omega$,
$U_{01} = 48$ V, $U_{02} = 43{,}2$ V.



Řešte pomocí přímé aplikace Kirchhoffových zákonů:

Řešení soustavy lineárních rovnic

Pokračování příkladu:



1. Kirchhoffův zákon pro **uzel A**: $I_1 - I_2 - I_5 = 0$
1. Kirchhoffův zákon pro **uzel C**: $I_3 - I_4 + I_5 = 0$
2. Kirchhoffův zákon pro **smyčku s_1**: $R_2 I_2 + R_3 I_3 - R_5 I_5 - R_6 I_5 - R_7 I_5 = 0$
2. Kirchhoffův zákon pro **smyčku s_2**: $-R_1 I_1 - R_2 I_2 + U_{01} = 0$
2. Kirchhoffův zákon pro **smyčku s_3**: $-R_3 I_3 - R_4 I_4 + U_{02} = 0$

Řešení soustavy lineárních rovnic

Pokračování příkladu:

1. Kirchhoffův zákon pro **uzel A**: $I_1 - I_2 - I_5 = 0$
1. Kirchhoffův zákon pro **uzel C**: $I_3 - I_4 + I_5 = 0$
2. Kirchhoffův zákon pro **smyčku s_1**: $R_2 I_2 + R_3 I_3 - R_5 I_5 - R_6 I_5 - R_7 I_5 = 0$
2. Kirchhoffův zákon pro **smyčku s_2**: $-R_1 I_1 - R_2 I_2 + U_{01} = 0$
2. Kirchhoffův zákon pro **smyčku s_3**: $-R_3 I_3 - R_4 I_4 + U_{02} = 0$

Řešíme soustavu 5 rovnic o 5 neznámých:

$$1I_1 - 1I_2 + 0I_3 + 0I_4 - 1I_5 = 0$$
$$0I_1 + 0I_2 + 1I_3 - 1I_4 + 1I_5 = 0$$
$$0I_1 + R_2 I_2 + R_3 I_3 + 0I_4 + (-R_5 - R_6 - R_7) I_5 = 0$$
$$-R_1 I_1 - R_2 I_2 + 0I_3 + 0I_4 + 0I_5 = -U_{01}$$
$$0I_1 + 0I_2 - R_3 I_3 - R_4 I_4 + 0I_5 = -U_{02}$$

Zapsáno maticově:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & R_2 & R_3 & 0 & (-R_5 - R_6 - R_7) \\ -R_1 & -R_2 & 0 & 0 & 0 \\ 0 & 0 & -R_3 & -R_4 & 0 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -U_{01} \\ -U_{02} \end{bmatrix}$$

Řešení soustavy lineárních rovnic

Pokračování příkladu:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 1 \\ 0 & R_2 & R_3 & 0 & (-R_5 - R_6 - R_7) \\ -R_1 & -R_2 & 0 & 0 & 0 \\ 0 & 0 & -R_3 & -R_4 & 0 \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -U_{01} \\ -U_{02} \end{bmatrix}$$

```
R1 = 3; R2 = 2; R3 = 3; R4 = 6; R5 = 5; R6 = 6;
R7 = 5;
U01 = 48; U02 = 43.2;
A = [ 1,    -1,    0,  0,          -1;…
      0,     0,    1, -1,           1;…
      0,    R2,   R3,  0,-R5-R6-R7;…
    -R1,   -R2,    0,  0,           0;…
      0,     0, -R3,-R4,           0];
b = [0;0;0;-U01;-U02];
```

Řešení soustavy lineárních rovnic

Pokračování příkladu:

```
R1 = 3; R2 = 2; R3 = 3; R4 = 6; R5 = 5; R6 = 6; R7 = 5;
U01 = 48; U02 = 43.2;
A = [ 1,-1,0,0,-1;0,0,1,-1,1;0,R2,R3,0,-R5-R6-R7;…
     -R1,-R2,0,0,0;0,0,-R3,-R4,0];
b = [0;0;0;-U01;-U02];
```

```
det(A)
ans = 864
```
$\neq 0$

```
cond(A)
ans = 12.106
```
$\neq \infty$

– dobře podmíněná matice

```
rcond(A)
ans = 0.052288
```
$\neq 0$

Matice **A** je **regulární** => soustava rovnic má **právě jedno řešení**.

```
rank(A)
ans =  5
rank([A,b])
ans =  5
```
$5 = 5$

```
x = A \ b
x =
   10.3000
    8.5500
    3.6333
    5.3833
    1.7500
```

Řešením této soustavy rovnic jsou proudy:

$I_1 = 10{,}300\text{A}$

$I_2 = 8{,}550\text{A}$

$I_3 = 3{,}633\text{A}$

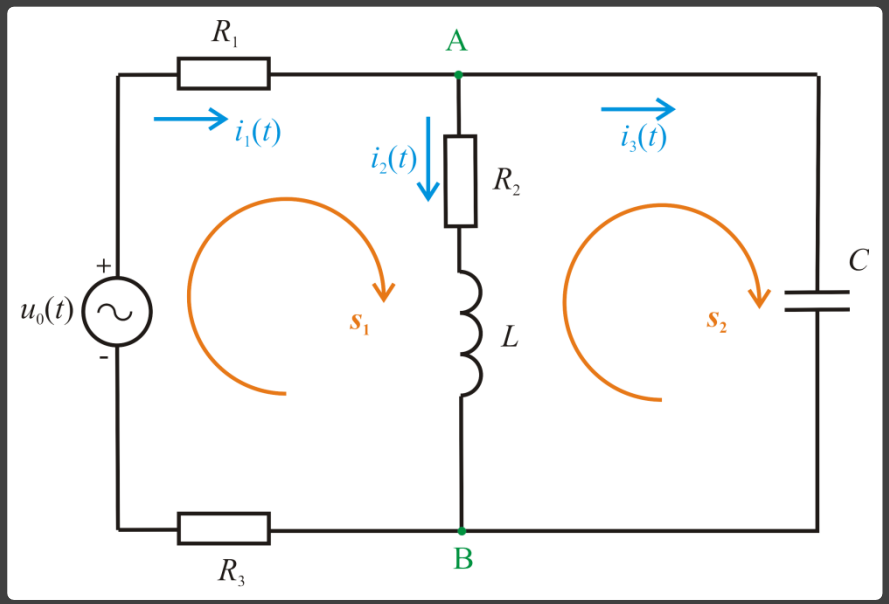
$I_4 = 5{,}383\text{A}$

$I_5 = 1{,}750\text{A}$

Řešení soustavy lineárních rovnic

Příklad: Stanovení časových průběhů větvových proudů i_1, i_2, i_3 v elektrickém obvodu na obrázku pomocí přímé aplikace Kirchhoffových zákonů s využitím symbolicko-komplexního zobrazení harmonických veličin, tj.
$u_0(t) = U_m \sin(\omega t + \varphi)$ odpovídá fázor max. hodnoty $\underline{U_0} = U_m e^{j\varphi}$.

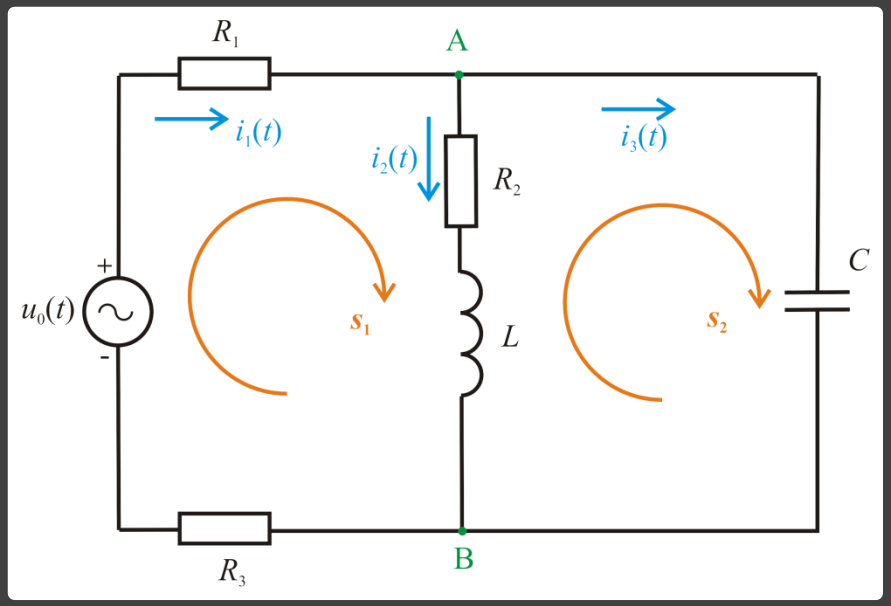
Dáno: $R_1 = 40\ \Omega$; $R_2 = 30\ \Omega$; $R_3 = 10\ \Omega$; $L = 0,1\text{mH}$; $C = 200\mu\text{F}$;
$u_0(t) = 10 \sin(\omega t + 30°)$, $\omega = 2\pi f$; $f = 50$ Hz; $\underline{U_0} = 10\, e^{j\,30°}$ V.



Fázory veličin označíme podtrženou kurzívou (např. $\underline{U}$, $\underline{I}$)

Řešení soustavy lineárních rovnic

Pokračování příkladu:



Pozn.:

1/j = –j

```
1/j
ans =
 0 - 1.00i
```

Kirchhoffovy zákony pro daný obvod:

1. Kirchhoffův z. pro **uzel A**: $\underline{I}_1 - \underline{I}_2 - \underline{I}_3 = 0$
2. Kirchhoffův z. pro **smyčku s_1**: $R_1\,\underline{I}_1 + R_2\,\underline{I}_2 + \mathrm{j}\omega L\,\underline{I}_2 + R_3\,\underline{I}_1 = \underline{U}_0$
2. Kirchhoffův z. pro **smyčku s_2**: $-R_2\,\underline{I}_2 - \mathrm{j}\omega L\,\underline{I}_2 - \mathrm{j}/(\omega C)\,\underline{I}_3 = 0$

Rovnice upravíme:

$$1\underline{I}_1 - 1\underline{I}_2 - 1\underline{I}_3 = 0$$

$$(R_1 + R_3)\,\underline{I}_1 + (R_2 + \mathrm{j}\omega L)\underline{I}_2 = \underline{U}_0$$

$$-(R_2 + \mathrm{j}\omega L)\underline{I}_2 + 1/(\mathrm{j}\omega C)\,\underline{I}_3 = 0$$

Řešení soustavy lineárních rovnic

Pokračování příkladu:

Řešíme soustavu 3 rovnic o 3 neznámých:

$$1\underline{I}_1 - 1\underline{I}_2 - 1\underline{I}_3 = 0$$
$$(R_1 + R_3)\,\underline{I}_1 + (R_2 + j\omega L)\underline{I}_2 = \underline{U}_0$$
$$-(R_2 + j\omega L)\underline{I}_2 + 1/(j\omega C)\,\underline{I}_3 = 0$$

```
R1 = 40; R2 = 30; R3 = 10;
L = 0.1e-3;  % převod na z mH na H
C = 200e-6;  % převod na z µF na F
f = 50;
w = 2*pi*f;  % znak w odpovídá úhlové frekvenci ω
Uo=10*exp(j*30*pi/180);  % převod stupňů na radiány nezbytný

A = [      1,          -1,              -1; ...
     (R1+R3), (R2+j*w*L),               0; ...
           0,  -R2-j*w*L, (1./(j*w*C))];
b = [0;Uo;0];
```

Řešení soustavy lineárních rovnic

Pokračování příkladu:

```
A = [1,-1,-1; (R1+R3),(R2+j*w*L),0; ...
     0, -R2-j*w*L,(1./(j*w*C))];
b = [0;Uo;0];
x = A \ b;  % výpočet fázorů maximálních hodnot proudů
```

```
det(A)
ans = 1.5005e+003-1.2717e+003i
```
$\neq 0$

```
rank(A)
ans =  3
rank([A,b])
ans =  3
```
$3 = 3$

```
cond(A)
ans = 54.7243
```
$\neq \infty$

– dobře podmíněná matice

```
rcond(A)
ans = 0.0116
```
$\neq 0$

Fázory maximálních hodnot proudů (komplexní čísla):

```
x = A \ b
x =
   0.1275 + 0.1163i
   0.0762 - 0.0273i
   0.0513 + 0.1436i
```

Matice A je **regulární** => soustava rovnic má **právě jedno řešení**.

Řešení soustavy lineárních rovnic

Pokračování příkladu:

```
x = A \ b
```

```
x =
    0.1275 + 0.1163i
    0.0762 - 0.0273i
    0.0513 + 0.1436i
```

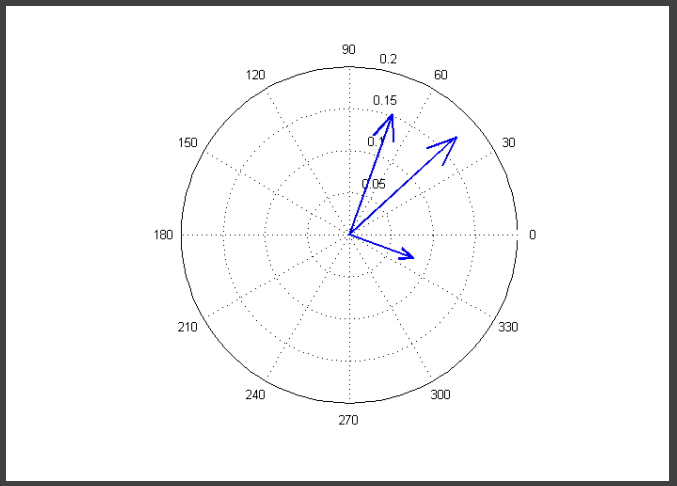
```
compass(x)
```
% zobrazení proudů v komplexní rovině

```
Im = abs(x)
```
% maximální hodnoty proudů

```
Im =
    0.1726
    0.0809
    0.1525
```

```
fi = angle(x)*180/pi
```
% fázové posuny proudů převedené na stupně

```
fi =
    42.3813
   -19.7188
    70.3412
```




```
for n=1:length(x)
      disp(['i',num2str(n),' = ',num2str(Im(n)),...
            ' sin(wt + ',num2str(fi(n)),'°)'])
end
```

```
i1 = 0.1726 sin(wt + 42.3813°)
i2 = 0.0809 sin(wt + -19.7188°)
i3 = 0.1525 sin(wt + 70.3412°)
```

% výpis okamžitých hodnot proudů

(o `num2str` viz dále)

Funkce pro vstup a výstup

Pozn.:

disp() – jednoduchý **výstup bez názvu proměnné**, nemá možnost formátovat text, formát výstupu je dán pouze nastavením pomocí příkazu **format**.

```
promenna = 7;
disp(promenna);
   7
```

```
disp([1,8,7])
     1     8     7
```

```
disp([1:5])
     1     2     3     4     5
```

```
disp('textovy retezec 1');
   textovy retezec 1
```

```
disp(['textovy ','retezec ', '2'])
   textovy retezec 2
```

Funkce pro vstup a výstup

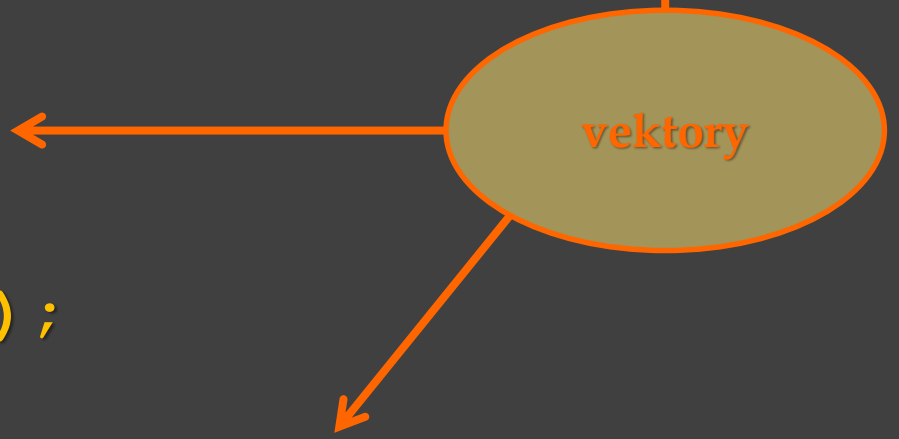
Pozn.:

disp() – jednoduchý **výstup bez názvu proměnné**, nemá možnost formátovat text, formát výstupu je dán pouze nastavením pomocí příkazu **format**.

```
promenna = 7;
disp(promenna);
   7
```

```
disp([1,8,7])
     1     8     7
```

```
disp([1:5])
     1     2     3     4     5
```




```
disp('textovy retezec 1');
   textovy retezec 1
```

```
disp(['textovy ','retezec ', '2'])
   textovy retezec 2
```

Funkce pro vstup a výstup

Pozn.:
`num2str()` – převod čísel na řetězec, např.

```
num2str(rand)
ans =  0.81472
```

```
whos
  Name        Size                    Bytes  Class
  ans         1x7                        14  char
```

Výstup na obrazovku (bez možnosti formátování textu) s využitím `disp` a `num2str` (výpis vektoru řetězců):

```
x = 5;
disp(['Zvetsime-li cislo ', num2str(x),...
  ' o jednicku, dostaneme ', num2str(x+1)]);
```

```
Zvetsime-li cislo 5 o jednicku, dostaneme 6
```

Řešení soustavy lineárních rovnic

Příklad: Funkce pro řešení soustavy lineárních algebraických rovnic s otestováním řešitelnosti soustavy. Vstupní parametry: matice koeficientů soustavy **A** a vektor pravých stran **b**, výstup: řešení soustavy x.

```
function [x] = testSoustavy(A,b)
 if (size(A,2) ~= rank(A))
    if (rank(A) ~= rank([A,b]))
      disp('Soustava nema reseni...')
      x = [];
    else
      disp('Soustava ma nekonecne mnoho reseni')
      x = [];
    end
  else
    x = A \ b;
    disp('Reseni soustavy:');
    disp(x);
  end
end
```

Řešení soustavy lineárních rovnic

Pokračování příkladu: Volání funkce `testSoustavy`

$$3x_1 + 4x_2 = 11$$
$$2x_1 - 5x_2 = -8$$

```
x = testSoustavy([3,4;2,-5],[11;-8]);
Reseni soustavy:
     1
     2
```

$$y_1 + y_2 = 11$$
$$2y_1 + 2y_2 = -8$$

```
y = testSoustavy([1,1;2,2],[11;-8]);
Soustava nema reseni...
```

$$z_1 + z_2 = 10$$
$$2z_1 + 2z_2 = 20$$

```
z = testSoustavy([1,1;2,2],[10;20]);
Soustava ma nekonecne mnoho reseni
```

Řešení soustavy lineárních rovnic

Pokračování příkladu:

Volání funkce **testSoustavy** pro soustavu:

$$I_1 - I_2 - I_5 = 0$$
$$I_3 - I_4 + I_5 = 0$$
$$R_2 I_2 + R_3 I_3 - R_5 I_5 - R_6 I_5 - R_7 I_5 = 0$$
$$-R_1 I_1 - R_2 I_2 + U_{01} = 0$$
$$-R_3 I_3 - R_4 I_4 + U_{02} = 0$$

```
R1 = 3; R2 = 2; R3 = 3; R4 = 6; R5 = 5; R6 = 6;
R7 = 5; U01 = 48; U02 = 43.2;
A = [ 1,-1,0,0,-1;0,0,1,-1,1;0,R2,R3,0,-R5-R6-R7;...
    -R1,-R2,0,0,0;0,0,-R3,-R4,0];
b = [0;0;0;-U01;-U02];
I = testSoustavy(A,b)
Reseni soustavy:
   10.3000
    8.5500
    3.6333
    5.3833
    1.7500
```

$$I_1 = 10{,}300\,\mathrm{A}$$
$$I_2 = 8{,}550\,\mathrm{A}$$
$$I_3 = 3{,}633\,\mathrm{A}$$
$$I_4 = 5{,}383\,\mathrm{A}$$
$$I_5 = 1{,}750\,\mathrm{A}$$

Řešení soustavy lineárních rovnic

Pokračování příkladu:

Volání funkce **testSoustavy** pro soustavu:

$$\underline{I}_1 - \underline{I}_2 - \underline{I}_3 = 0$$
$$R_1\,\underline{I}_1 + R_2\,\underline{I}_2 + \mathrm{j}\omega L\,\underline{I}_2 + R_3\,\underline{I}_1 = \underline{U}_0$$
$$-R_2\,\underline{I}_2 - \mathrm{j}\omega L\,\underline{I}_2 + 1/(\mathrm{j}\omega C)\,\underline{I}_3 = 0$$

```
R1 = 40; R2 = 30; R3 = 10;
L = 0.1e-3; C = 200e-6; f = 50; w = 2*pi*f;
Uo=10*exp(j*30*pi/180);
A = [1,-1,-1; (R1+R3),(R2+j*w*L),0; ...
      0, -R2-j*w*L,(1./(j*w*C))];
b = [0;Uo;0];
I = testSoustavy(A,b);
Reseni soustavy:
   0.1275 + 0.1163i
   0.0762 - 0.0273i
   0.0513 + 0.1436i
```

Funkce pro vstup a výstup

Vstup

`input()` – zobrazí uživateli textový řetězec, a pak čeká na vstup z klávesnice, např.:

```
prom = input('Vyzva pro uzivatele: ');
```

Uživatel zadá číslo (příp. vektor, matici) nebo matematický výraz, který se uloží do proměnné `prom`. Zadání potvrdí stiskem klávesy Enter.
Pokud uživatel stiskne klávesu Enter, aniž by něco zadal, `input` vrátí prázdnou matici.

Vstup znaku nebo textu:

```
zadany_text = input('Zadejte text: ', 's')
```

– zobrazí uživateli textový řetězec, a pak čeká na vstup řetězce znaků z klávesnice, který se uloží do proměnné `zadany_text`.

Funkce pro vstup a výstup

Např.:

```
n = input('Zadej cislo: ')
Zadej cislo: 2+4
n =
     6

m = input('Zadej cislo: ');
Zadej cislo: 7

v = input('Zadej cislo: ');
Zadej cislo: [5,8,9]

W = input('Zadej cislo: ');
Zadej cislo: rand(3,5)
```

Funkce pro vstup a výstup

Např.:

```
n = input('Zadej cislo: ')
Zadej cislo: 2+4
n =
     6

m = input('Zadej cislo: ');
Zadej cislo: 7

v = input('Zadej cislo: ');
Zadej cislo: [5,8,9]

W = input('Zadej cislo: ');
Zadej cislo: rand(3,5)
```

Vstup z klávesnice, není ukončen středníkem, `n` se po zadání uživatelem vypíše

Funkce pro vstup a výstup

Např.:

```
n = input('Zadej cislo: ')
Zadej cislo: 2+4
n =
     6

m = input('Zadej cislo: ');
Zadej cislo: 7

v = input('Zadej cislo: ');
Zadej cislo: [5,8,9]

W = input('Zadej cislo: ');
Zadej cislo: rand(3,5)
```

Vstup z klávesnice, není ukončen středníkem, `n` se po zadání uživatelem vypíše

Vstup z klávesnice, je ukončen středníkem, `m`, `v` (vektor) ani `W` (matice) se po zadání uživatelem nevypíší

Funkce pro vstup a výstup

Např.:

```
n = input('Zadej cislo: ')
Zadej cislo: 2+4
n =
     6

m = input('Zadej cislo: ');
Zadej cislo: 7

v = input('Zadej cislo: ');
Zadej cislo: [5,8,9]

W = input('Zadej cislo: ');
Zadej cislo: rand(3,5)

whos
  Name      Size              Bytes  Class     Attributes
  W         3x5                 120  double
  m         1x1                   8  double
  n         1x1                   8  double
  v         1x3                  24  double
```

Vstup z klávesnice, není ukončen středníkem, `n` se po zadání uživatelem vypíše

Vstup z klávesnice, je ukončen středníkem, `m`, `v` (vektor) ani `W` (matice) se po zadání uživatelem nevypíší

`W` je matice náhodných čísel s 3 řádky a 5 sloupci

Funkce pro vstup a výstup

Např.:

```
z = input('Zadej znak: ','s')
Zadej znak: n
z =
     n

t = input('Zadej znak: ','s');
Zadej znak: Ahoj!

r = input('Zadej znak: ','s');
Zadej znak: 78

c = input('Zadej znak: ');
Zadej znak: 78
```

Funkce pro vstup a výstup

Např.:

```
z = input('Zadej znak: ','s')
Zadej znak: n
z =
     n

t = input('Zadej znak: ','s');
Zadej znak: Ahoj!

r = input('Zadej znak: ','s');
Zadej znak: 78

c = input('Zadej znak: ');
Zadej znak: 78
```

parametr `'s'` značí, že je očekáván znak (příp. řetězec znaků)

Funkce pro vstup a výstup

Např.:

```
z = input('Zadej znak: ','s');
Zadej znak: n
z =
     n

t = input('Zadej znak: ','s');
Zadej znak: Ahoj!

r = input('Zadej znak: ','s');
Zadej znak: 78

c = input('Zadej znak: ');
Zadej znak: 78
```

parametr `'s'` značí, že je očekáván znak (příp. řetězec znaků)

uživatel zadá dva znaky - číslice 7 a 8, které se uloží do proměnné `r`

Funkce pro vstup a výstup

Např.:

```
z = input('Zadej znak: ','s');
Zadej znak: n
z =
     n

t = input('Zadej znak: ','s');
Zadej znak: Ahoj!

r = input('Zadej znak: ','s');
Zadej znak: 78

c = input('Zadej znak: ');
Zadej znak: 78
```

parametr `'s'` značí, že je očekáván znak (příp. řetězec znaků)

uživatel zadá dva znaky - číslice 7 a 8, které se uloží do proměnné `r`

parametr `'s'` zde není, je očekávána numerická hodnota (bez ohledu na výzvu pro uživatele)

Funkce pro vstup a výstup

Např.:

```
z = input('Zadej znak: ','s');
Zadej znak: n
z =
     n

t = input('Zadej znak: ','s');
Zadej znak: Ahoj!

r = input('Zadej znak: ','s');
Zadej znak: 78

c = input('Zadej znak: ');
Zadej znak: 78

whos
  Name    Size            Bytes  Class    Attributes
  c       1x1                 8  double
  r       1x2                 4  char
  t       1x5                10  char
  z       1x1                 2  char
```

parametr 's' značí, že je očekáván znak (příp. řetězec znaků)

uživatel zadá dva znaky - číslice 7 a 8, které se uloží do proměnné r

parametr 's' zde není, je očekávána numerická hodnota (bez ohledu na výzvu pro uživatele)

c je číslo 78 - typ double, r je řetězec znaků - char

Funkce pro vstup a výstup

Příklad: Vytvoření funkce pro porovnání dvou čísel

```
function porovnani
while(1)
 a = input('Zadej první cislo: ');
 b = input('Zadej druhe cislo: ');
 if(a < b)    % porovnani dvou cisel
     disp([num2str(a),' je mensi nez ',num2str(b)]);
 elseif(a == b)
     disp([num2str(a),' je rovno ',num2str(b)]);
 else
     disp([num2str(a),' je vetsi nez ',num2str(b)]);
 end
 v = input('Chces pokracovat? (A - ano): ','s');
 if (v ~= 'A')        % zadá-li uživatel jiný znak než 'A',
     disp('Konec')    % funkce bude ukončena
     return           % ukončení běhu funkce
 end
end
end
```

Funkce pro vstup a výstup

Pokračování příkladu: Volání funkce **porovnání**

```
Zadej první cislo: 6
Zadej druhe cislo: 7
6 je mensi nez 7
Chces pokracovat? (A - ano): A
Zadej první cislo: 9
Zadej druhe cislo: 9
9 je rovno 9
Chces pokracovat? (A - ano): A
Zadej první cislo: -1
Zadej druhe cislo: 2
-1 je mensi nez 2
Chces pokracovat? (A - ano): n
Konec
```

Pokud uživatel zadá více čísel (vektor, matici), budeme porovnávat jen jedno, proto se musíme přebytečných zbavit, přidáme do funkce ošetření, např. takto:

```
function porovnani
while(1)
 a = input('Zadej první cislo: ');
 if (length(a) > 1)
     a = a(1,1);
 end
 b = input('Zadej druhe cislo: ');
 if (length(b) > 1)
     b = b(1,1);
 end
 if(a < b)    % porovnani dvou cisel
     disp([num2str(a),' je mensi nez ',num2str(b)]);
 elseif(a == b)
     disp([num2str(a),' je rovno ',num2str(b)]);
 else
     disp([num2str(a),' je vetsi nez ',num2str(b)]);
 end
 v = input('Chces pokracovat? (A - ano): ','s');
 if (v ~= 'A')       % zadá-li uživatel jiný znak než 'A',
     disp('Konec')   % funkce bude ukončena
     return          % ukončení běhu funkce
 end
end
end
```

je uvažováno vždy první číslo z matice či vektoru

Funkce pro vstup a výstup

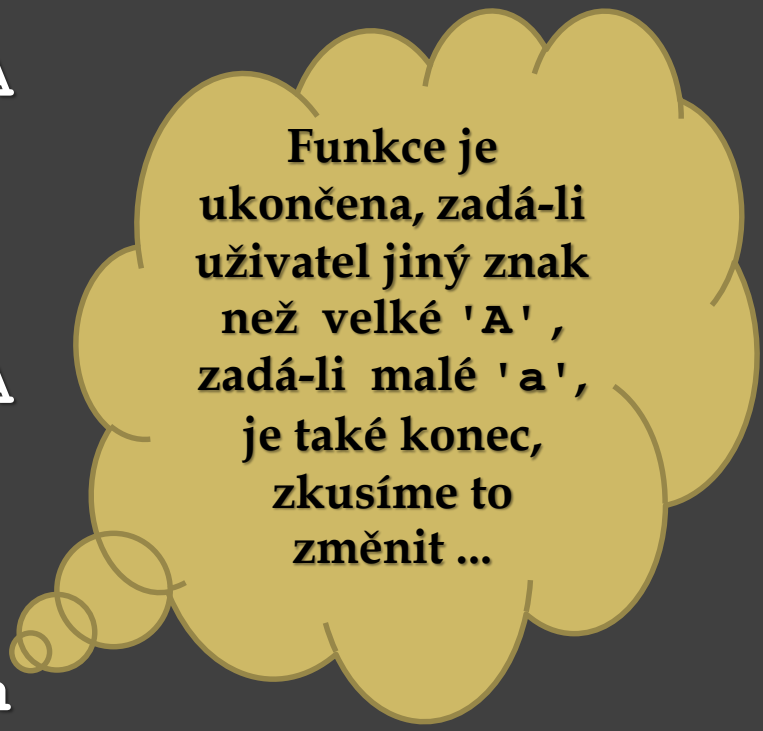
Pokračování příkladu:
Volání funkce `porovnani`

```
Zadej první cislo: 6
Zadej druhe cislo: [7,2,6]
6 je mensi nez 7
Chces pokracovat? (A - ano): A
Zadej první cislo: [1,2,3]
Zadej druhe cislo: [4,5,6,9]
1 je mensi nez 4
Chces pokracovat? (A - ano): A
Zadej první cislo: 9:-1:1
Zadej druhe cislo: [8,3;7,5]
9 je vetsi nez 8
Chces pokracovat? (A - ano): a
Konec
```

Funkce pro vstup a výstup

Pokračování příkladu:
Volání funkce **porovnani**

```
Zadej první cislo: 6
Zadej druhe cislo: [7,2,6]
6 je mensi nez 7
Chces pokracovat? (A - ano): A
Zadej první cislo: [1,2,3]
Zadej druhe cislo: [4,5,6,9]
1 je mensi nez 4
Chces pokracovat? (A - ano): A
Zadej první cislo: 9:-1:1
Zadej druhe cislo: [8,3;7,5]
9 je vetsi nez 8
Chces pokracovat? (A - ano): a
Konec
```



```
function porovnani
while(1)
 a = input('Zadej první cislo: ');
 if (length(a) > 1)
     a = a(1,1);
 end
 b = input('Zadej druhe cislo: ');
 if (length(b) > 1)
     b = b(1,1);
 end
 if(a < b)    % porovnani dvou cisel
     disp([num2str(a),' je mensi nez ',num2str(b)]);
 elseif(a == b)
     disp([num2str(a),' je rovno ',num2str(b)]);
 else
     disp([num2str(a),' je vetsi nez ',num2str(b)]);
 end
 v = input('Chces pokracovat? (A - ano): ','s');
 if ((v ~= 'A') &&(v ~= 'a'))
     disp('Konec')
     return
 end
end
end
```

Nezadal-li uživatel znak 'A' ani 'a', funkce bude ukončena

Funkce pro vstup a výstup

Pokračování příkladu:

Volání funkce **porovnani**

```
Zadej první cislo: 152
Zadej druhe cislo: 47
152 je vetsi nez 47
Chces pokracovat? (A - ano): A
Zadej první cislo: 6
Zadej druhe cislo: 8
6 je mensi nez 8
Chces pokracovat? (A - ano): a
Zadej první cislo: 1
Zadej druhe cislo: 2
1 je mensi nez 2
Chces pokracovat? (A - ano): x
Konec
```

Zadá-li uživatel znak `'A'` ,příp. `'a'`, funkce pokračuje.

Nezadá-li uživatel znak `'A'` ani `'a'`, funkce je ukončena.

Funkce pro vstup a výstup

Pozn.:
jiná možnost řešení pomocí funkcí **`upper`**, příp. **`lower`**:

- **`upper`** – převádí v řetězci malá písmena na velká písmena, ostatní znaky zůstávají beze změny.
- **`lower`** – převádí v řetězci velká písmena na malá písmena, ostatní znaky zůstávají beze změny.

Tedy,
`upper('a')` – vrátí **`'A'`** , tj. z malých písmen dělá velká,
`lower('A')` – vrátí **`'a'`** , tj. z velkých písmen dělá malá.

Příklad:

```
upper('Ahoj! Jak se mas?')
ans =
AHOJ! JAK SE MAS?
```

```
lower('Ahoj! Jak se mas?')
ans =
ahoj! jak se mas?
```

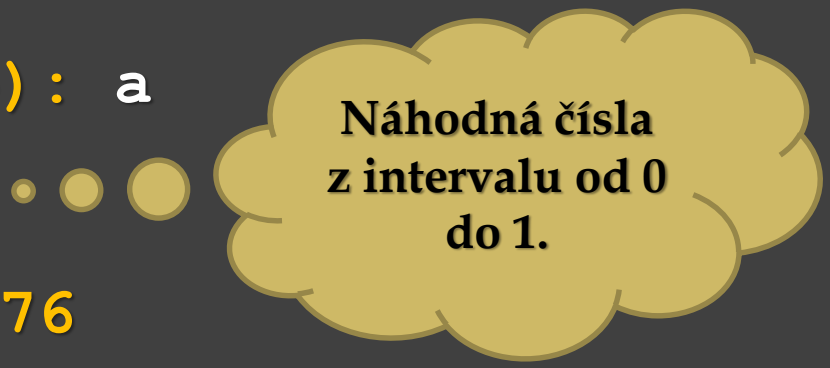
```
function porovnani
while(1)
 a = input('Zadej první cislo: ');
 if (length(a) > 1)
     a = a(1,1);
 end
 b = input('Zadej druhe cislo: ');
 if (length(b) > 1)
     b = b(1,1);
 end
 if(a < b)    % porovnani dvou cisel
     disp([num2str(a),' je mensi nez ',num2str(b)]);
 elseif(a == b)
     disp([num2str(a),' je rovno ',num2str(b)]);
 else
     disp([num2str(a),' je vetsi nez ',num2str(b)]);
 end
 v = input('Chces pokracovat? (A - ano): ','s');
 v = upper(v);
 if (v ~= 'A')
     disp('Konec')
     return
 end
end
end
```

Z každého malého písmene, které zadá uživatel se stane velké, tj. např. z malého 'a' bude 'A', a to je porovnáno.

Funkce pro vstup a výstup

Pokračování příkladu:
Volání funkce **porovnani**

```
Zadej první cislo: 43
Zadej druhe cislo: 71
43 je mensi nez 71
Chces pokracovat? (A - ano): A
Zadej první cislo: 66
Zadej druhe cislo: 66
66 je rovno 66
Chces pokracovat? (A - ano): a
Zadej první cislo: rand
Zadej druhe cislo: rand
0.14189 je mensi nez 0.42176
Chces pokracovat? (A - ano): x
Konec
```



Funkce pro vstup a výstup

Příklad: Vytvořte uživatelskou funkci **s2cos_graf** bez parametrů. Tato funkce **s2cos_graf** bude vykreslovat graf křivky dané rovnicí

$$y = \sin^2(x)\cos(x)$$

pro x z intervalu, jehož dolní mez, horní mez a krok zadá uživatel z klávesnice.

Je třeba ošetřit, aby dolní mez byla menší než horní, aby krok byl kladný a nepřevyšoval hodnotu **1/10** z rozdílu mezi horní a dolní mezí intervalu.

Pro výpočet $y = \sin^2(x)\cos(x)$ bude vytvořena další funkce **s2cos** s jedním vstupním parametrem x a jedním výstupním parametrem y. Funkce **s2cos** bude volána ve funkci **s2cos_graf**.

Funkce mohou být napsány každá v samostatném souboru a uloženy ve stejné složce nebo mohou být napsány v jednom souboru (nejprve hlavní funkce **s2cos_graf** a potom funkce **s2cos**). Nejdříve vytvoříme funkci **s2cos_graf**:

```
function s2cos_graf
dolni = input('Zadejte pocatecni hodnotu: ');
horni = input('Zadejte koncovou hodnotu: ');
krok = input('Zadejte krok: ');
if (horni <= dolni)
    disp('Dolni mez je mensi nebo stejna s horni!');
    return;
end
if (krok <= 0)
    disp('Je pripustny pouze kladny krok...');
    return;
elseif (krok > ((horni - dolni) ./ 10))
    disp('Krok je prilis velky...');
    return;
end;
osa_x = dolni:krok:horni;
osa_y = s2cos(osa_x);
plot(osa_x, osa_y)
end
```

krok musí být kladný

je požadován krok max 1/10 ze zadaneho intervalu

volání funkce s2cos se vstupem osa_x pro výpočet *y*

Tuto funkci s2cos musíme nyní vytvořit :

Funkce pro vstup a výstup

Pokračování příkladu: funkce **s2cos** pro výpočet $y = \sin^2(x)\cos(x)$

```
function y = s2cos(x)
y = (sin(x).^ 2).* cos(x);
end
```

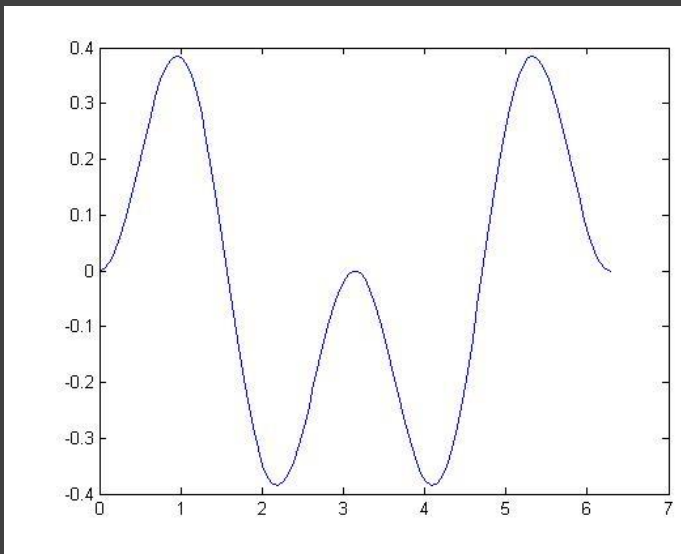
operace prvek po prvku nutné

Funkce **s2cos** je využita ve funkci **s2cos_graf**.

Příklady volání funkce **s2cos_graf** jejím názvem :

```
s2cos_graf
Zadejte pocatecni hodnotu: 0
Zadejte koncovou hodnotu: 2*pi
Zadejte krok: 100
Krok je prilis velky...
```

```
s2cos_graf
Zadejte pocatecni hodnotu: 0
Zadejte koncovou hodnotu: 2*pi
Zadejte krok: 2*pi/100
```



Funkce pro vstup a výstup

Pokračování příkladu:

Funkci **s2cos** lze volat i z příkazového řádku, pokud je uložena v samostatném souboru.

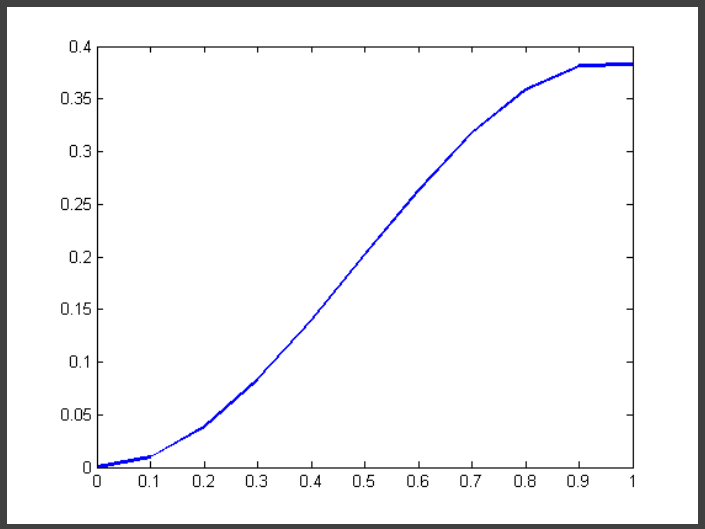
```
function y = s2cos(x)
y = (sin(x).^ 2).* cos(x);
end
```

Příklady volání funkce **s2cos** jejím názvem :

```
vstup = 0:0.1:1;
kolik = s2cos(vstup);
```

Funkce **s2cos** slouží pro výpočet, pak vykreslíme graf

```
plot(vstup, kolik)
```

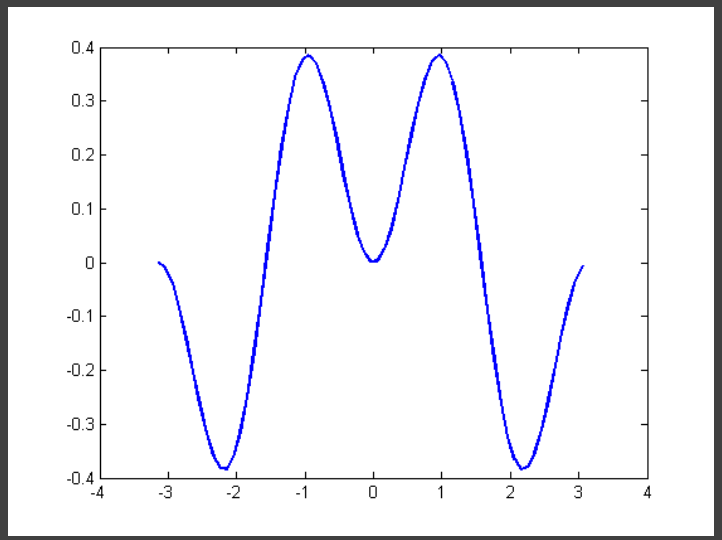


Pro jiný vstup tatáž funkce:

```
x = [-pi:0.1:pi];
vysledek = s2cos(x);
```

A ještě vykreslíme graf

```
plot(x,vysledek)
```



Pro vstup 0:

```
y = s2cos(0)
y =
     0
```

Funkce pro vstup a výstup

Příklad: Vytvořte uživatelskou funkci **s2cos_fajn** bez parametrů. Tato funkce **s2cos_fajn** bude vykreslovat graf křivky dané rovnicí

$$y = \sin^2(x)\cos(x)$$

pro x z intervalu, jehož dolní mez, horní mez a krok zadá uživatel z klávesnice.

Je třeba ošetřit, aby dolní mez byla menší než horní, aby krok byl kladný a nepřevyšoval hodnotu **1/10** z rozdílu mezi horní a dolní mezí intervalu. Pokud se uživatel zmýlí, bude mu umožněno dolní mez, horní mez a krok zadat znovu.

Pro výpočet $y = \sin^2(x)\cos(x)$ bude využita funkce **s2cos**.

Takže vylepšíme uživatelskou funkci `s2cos_graf`, která byla uvedena dříve, a přidáme cykly.

```
function s2cos_fajn
while(1)
 dolni = input('Zadejte pocatecni hodnotu: ');
 horni = input('Zadejte koncovou hodnotu: ');
 if (horni > dolni)
    break;
 end
 disp('Dolni mez je mensi nebo stejna s horni!');
end
while(1)
    krok = input('Zadejte krok: ');
    if (krok <= 0)
        disp('Je pripustny pouze kladny krok...');
        continue;
    elseif (krok > ((horni - dolni)./10))
        disp('Krok je prilis velky...');
        continue;
    end;
    break;
end;
osa_x = dolni:krok:horni;
osa_y = s2cos(osa_x);
plot(osa_x, osa_y)
end
```

```
function y = s2cos(x)
y = (sin(x).^ 2).* cos(x);
end
```

```
function s2cos_fajn
while(1)
 dolni = input('Zadejte pocatecni hodnotu: ');
 horni = input('Zadejte koncovou hodnotu: ');
 if (horni > dolni)
    break;
 end
 disp('Dolni mez je mensi nebo stejna s horni!');
end
while(1)
    krok = input('Zadejte krok: ');
    if (krok <= 0)
        disp('Je pripustny pouze kladny krok...');
        continue;
    elseif (krok > ((horni - dolni)./10))
        disp('Krok je prilis velky...');
        continue;
    end;
    break;
end;
osa_x = dolni:krok:horni;
osa_y = s2cos(osa_x);
plot(osa_x, osa_y)
end
```

break - ukončí běh cyklu a vyskočí z něj za jeho koncový **end**, **continue** - vyvolá okamžitě další otočku cyklu

```
function y = s2cos(x)
y = (sin(x).^ 2).* cos(x);
end
```

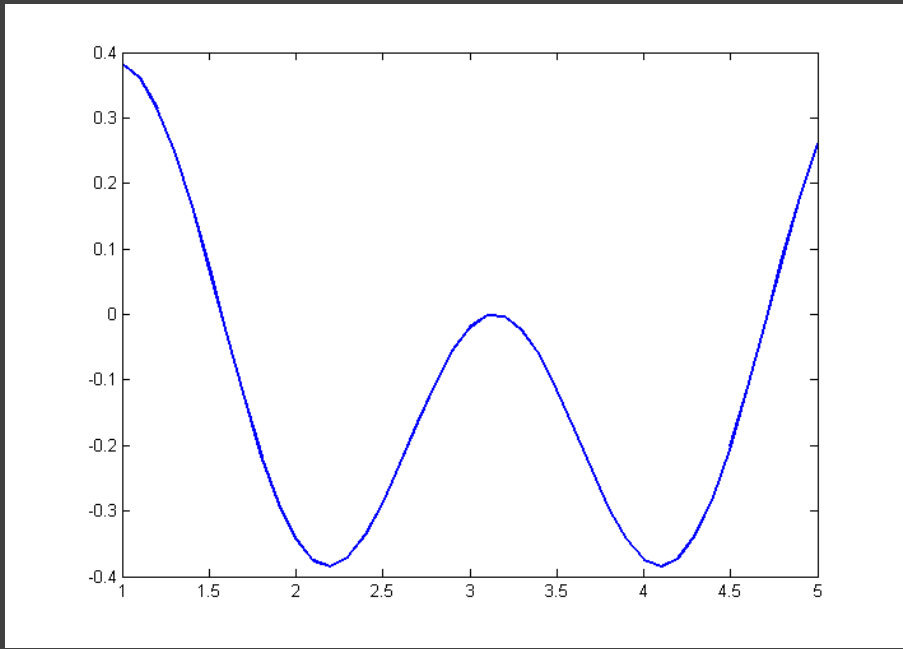
Funkce pro vstup a výstup

Pokračování příkladu:

Volání funkce jejím názvem:

```
s2cos_fajn
Zadejte pocatecni hodnotu: 10
Zadejte koncovou hodnotu: 0
Dolni mez je mensi nebo stejna s horni!
Zadejte pocatecni hodnotu: 1
Zadejte koncovou hodnotu: 5
Zadejte krok: 20
Krok je prilis velky...
Zadejte krok: 0.1
```

a už se vykreslí graf se zadanými hodnotami ...



Funkce pro vstup a výstup

Příklad: Vytvořte uživatelskou funkci **s2cos_super** bez parametrů. Tato funkce **s2cos_super** bude vykreslovat graf křivky dané rovnicí

$$y = \sin^2(x)\cos(x)$$

pro x z intervalu, jehož dolní mez, horní mez a krok zadá uživatel z klávesnice. Je třeba ošetřit, aby dolní mez byla menší než horní, aby krok byl kladný a nepřevyšoval hodnotu **1/10** z rozdílu mezi horní a dolní mezí intervalu. Pokud se uživatel zmýlí, bude mu umožněno dolní mez, horní mez a krok zadat znovu. Ve funkci bude provedeno ošetření počtu chyb uživatele, maximální počet chyb, které může uživatel udělat, bude **10**.

Pro výpočet $y = \sin^2(x)\cos(x)$ bude využita funkce **s2cos**.

Takže k funkci `s2cos_fajn()`, která byla uvedena dříve, přidáme počítání chyb.

```
function s2cos_super
kolik_chyb = 0;
while(1)
 dolni = input('Zadejte pocatecni hodnotu: ');
 horni = input('Zadejte koncovou hodnotu: ');
 if (horni > dolni)
    break;
 end
 kolik_chyb = kolik_chyb + 1;
 switch kolik_chyb
     case 0
        disp('Chybka programu, nula chyb, tady nemam byt?!?!');
     case 1
        disp('Tak to byla prvni chyba...');
        disp('Dolni mez je mensi nebo stejna s horni!');
     case 2
        disp('Tak to byla druha chyba...');
        disp('Dolni mez je mensi nebo stejna s horni!');
     case 3
        disp('Tak to bylo do tretice...');
        disp('Dolni mez je mensi nebo stejna s horni!');
     case {4, 5, 6, 7, 8, 9}
        disp(['Uz ',num2str(kolik_chyb), ...
        '. chyba, to snad neni mozne!']);
```

Pokračování …

```
        disp('Dolni mez je mensi nebo stejna s horni!');
    case 10
        disp('Jubileum, 10. chyba! Bude konec!');
        disp('Dolni mez je mensi nebo stejna s horni!');
    otherwise
        disp('Uz si nemame co rici... Jsi beznadejny pripad.');
        return;
    end
end;
while(1)
    krok = input('Zadejte krok: ');
    if (krok <= 0)
        disp('Je pripustny pouze kladny krok...');
        continue;
    elseif (krok > ((horni - dolni)./10))
        disp('Krok je prilis velky...');
        continue;
    end;
    break;
end;
osa_x = dolni:krok:horni;
osa_y = s2cos(osa_x);
plot(osa_x, osa_y)
end
```

Ošetření počtu chyb u zadávání kroku bychom provedli podobně

```
function y = s2cos(x)
y = (sin(x).^ 2).* cos(x);
end
```

Funkce pro vstup a výstup

Pokračování příkladu:

Volání funkce jejím názvem:

```
s2cos_super
Zadejte pocatecni hodnotu: 3
Zadejte koncovou hodnotu: 0
Tak to byla prvni chyba...
Dolni mez je mensi nebo stejna s horni!
Zadejte pocatecni hodnotu: 5
Zadejte koncovou hodnotu: 0
Tak to byla druha chyba...
Dolni mez je mensi nebo stejna s horni!
Zadejte pocatecni hodnotu: 6
Zadejte koncovou hodnotu: -1
Tak to bylo do tretice...
Dolni mez je mensi nebo stejna s horni!
Zadejte pocatecni hodnotu: 0
Zadejte koncovou hodnotu: 20
Zadejte krok: 0.1
```

a už se vykreslí graf se zadanými hodnotami ...

Uživatel v tomto případě udělal **3** chyby při zadávání.