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

Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

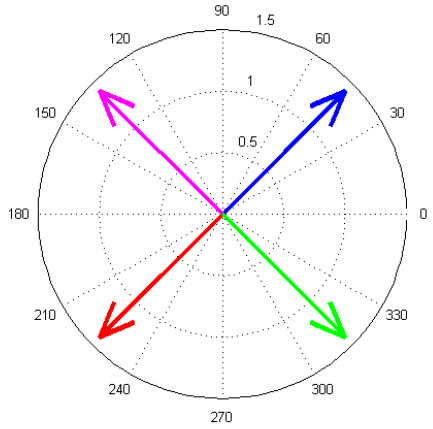
```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```

POZOR!!!

atan (imag(c)./real(c)) – počítá **úhel** v **1.** a **4. kvadrantu**, podle vzorce $\varphi = \arctan \frac{\mathrm{Im}(c)}{\mathrm{Re}(c)}$

tj. úhel vychází od **$-\pi/2$** do **$\pi/2$**, např:

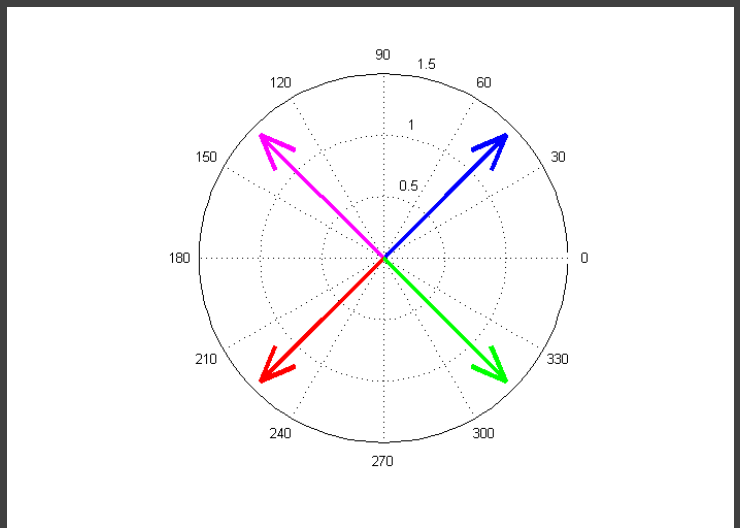
```
atan(1./1)*180/pi
ans =    45
atan((1)./-1)*180/pi
ans =   -45
```



Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```



POZOR!!!

atan (imag(c)./real(c)) – počítá **úhel** v **1.** a **4. kvadrantu**, podle vzorce $\varphi = \arctan \frac{\mathrm{Im}(c)}{\mathrm{Re}(c)}$

tj. úhel vychází od **–π/2** do **π/2**, např:

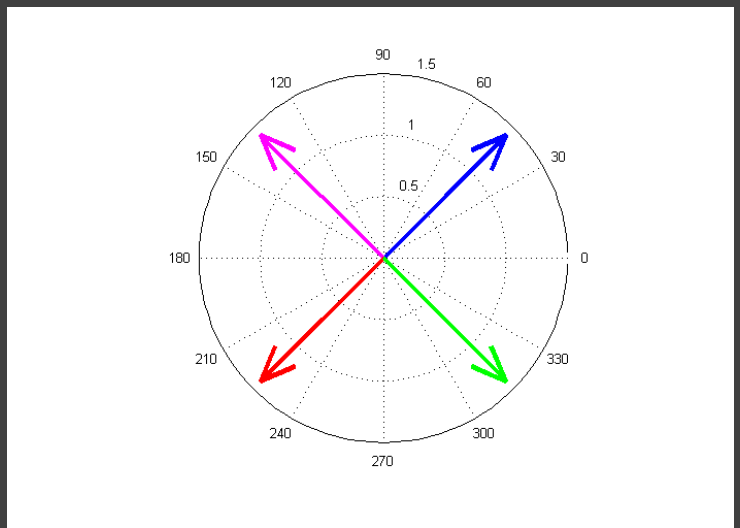
```
atan(1./1)*180/pi
ans =    45
atan((1)./-1)*180/pi
ans =   -45
```

úhel není vypočten správně

Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```



POZOR!!!

atan (imag(c)./real(c)) – počítá **úhel** v **1.** a **4. kvadrantu**, podle vzorce $\varphi = \arctan \frac{\mathrm{Im}(c)}{\mathrm{Re}(c)}$

tj. úhel vychází od **–π/2** do **π/2**, např:

```
atan(1./1)*180/pi
ans =    45
atan((1)./-1)*180/pi
ans =   -45
```

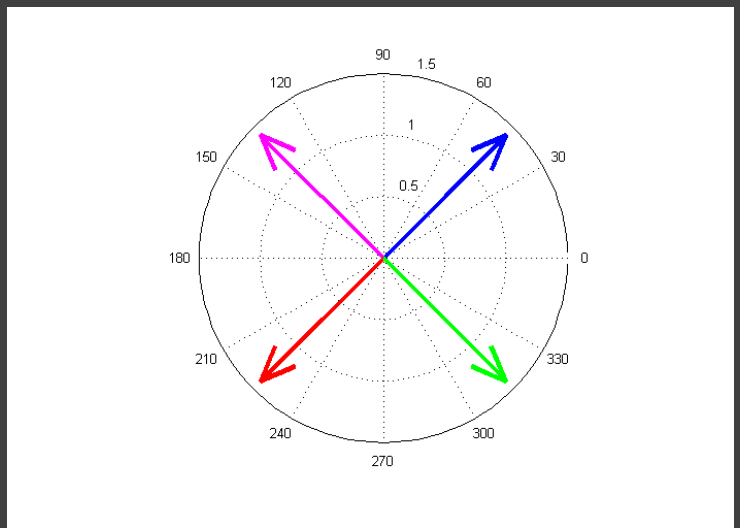
úhel není vypočten správně

```
atan((-1)./(-1))*180/pi
ans =    45
```

Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```



POZOR!!!

atan (imag(c)./real(c)) – počítá **úhel** v **1.** a **4. kvadrantu**, podle vzorce $\varphi = \arctan \frac{\mathrm{Im}(c)}{\mathrm{Re}(c)}$

tj. úhel vychází od **–π/2** do **π/2**, např:

```
atan(1./1)*180/pi
ans =    45
atan((1)./-1)*180/pi
ans =   -45
```

úhel není vypočten správně

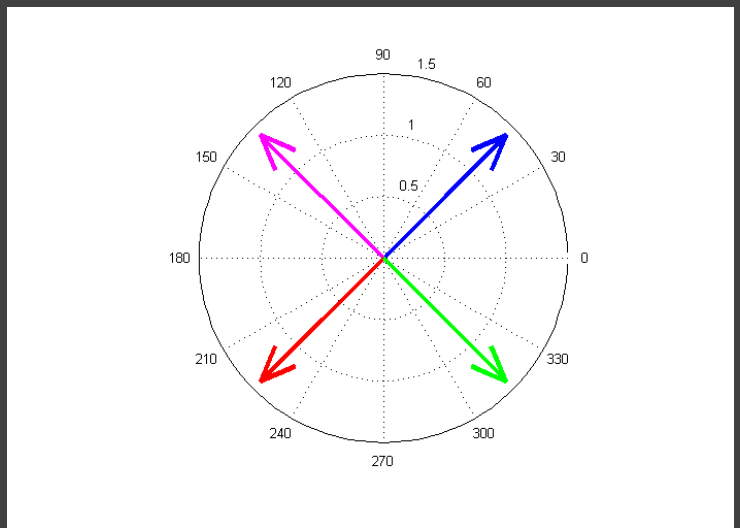
```
atan((-1)./(-1))*180/pi
ans =    45
```

úhel není vypočten správně

Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```



POZOR!!!

atan (imag(c)./real(c)) – počítá **úhel** v **1.** a **4. kvadrantu**, podle vzorce $\varphi = \arctan \frac{\mathrm{Im}(c)}{\mathrm{Re}(c)}$

tj. úhel vychází od **–π/2** do **π/2**, např:

```
atan(1./1)*180/pi
ans =    45
atan((1)./-1)*180/pi
ans =   -45
```

úhel není vypočten správně

```
atan((-1)./(-1))*180/pi
ans =    45
```

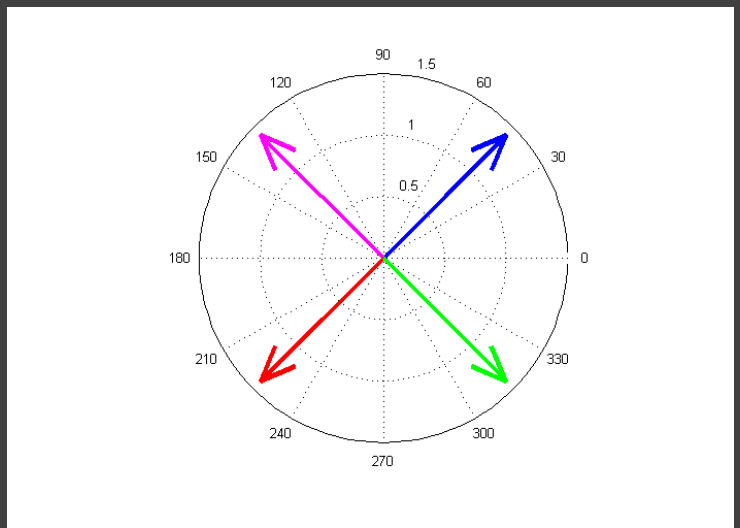
úhel není vypočten správně

```
atan(-1./(1))*180/pi
ans =   -45
```

Práce s komplexními čísly

angle(c) – počítá **úhel** ve všech **4 kvadrantech**, např:

```
angle(1+i)*180/pi
      ans =    45
angle(-1+i)*180/pi
      ans =   135
angle(-1-i)*180/pi
      ans =  -135
angle(1-i)*180/pi
      ans =   -45
```



atan2 (imag(c),real(c)) – počítá úhel také ve **všech 4 kvadrantech**, tj. od **–π** do **π** , např:

```
atan2(1,1)*180/pi
      ans =    45
atan2(1,-1)*180/pi
      ans =   135
atan2(-1,-1)*180/pi
      ans =  -135
atan2(-1,1)*180/pi
      ans =   -45
```

Pomocí příkazu atan2() jsou úhly správně, OK.

Práce s komplexními čísly

Vše platí i pro vektory a matice, např:

```
C = [1+2i, -2+i, -3-4i; 4-3i, 1, i]
C =
   1.00 + 2.00i  -2.00 + 1.00i  -3.00 - 4.00i
   4.00 - 3.00i   1.00                0 + 1.00i
```

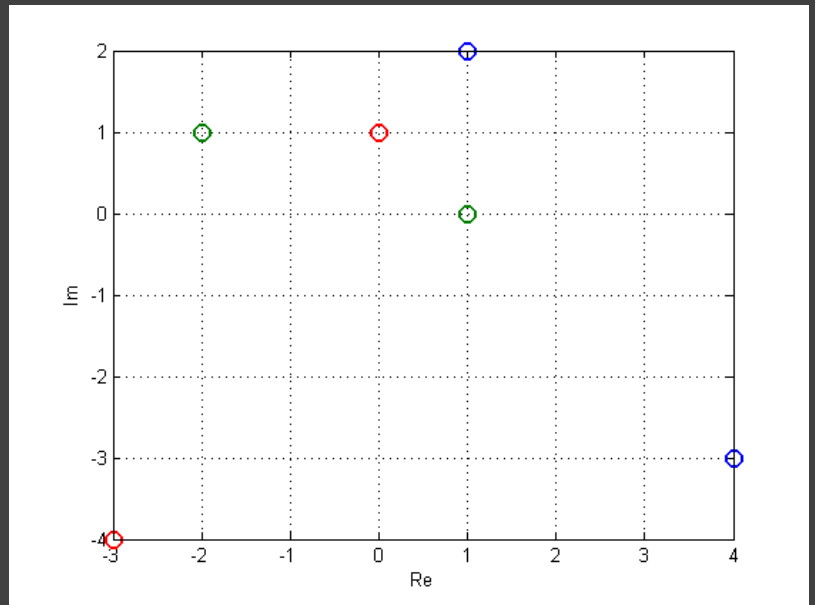
`real(C)` – reálné části komplexních čísel (prvků matice C)

```
ans =
     1    -2    -3
     4     1     0
```

`imag(C)` – imaginární části komplexních čísel (prvků matice C)

```
ans =
     2     1    -4
    -3     0     1
```

```
plot(C,'o')
grid
xlabel('Re')
ylabel('Im')
```



Práce s komplexními čísly

Vše platí i pro vektory a matice, např:

```
C = [1+2i, -2+i, -3-4i; 4-3i, 1, i]
C =
   1.00 + 2.00i  -2.00 + 1.00i  -3.00 - 4.00i
   4.00 - 3.00i   1.00                0 + 1.00i
```

`isreal(C)` – test, **jsou**-li prvky matice **C** **reálná čísla** (nepravda)

```
ans  =  0
```

`conj(C)` – vypíše čísla **komplexně sdružená**
(u imaginární části s opačnými znaménky)

```
ans =
   1.00 - 2.00i   -2.00 - 1.00i   -3.00 + 4.00i
   4.00 + 3.00i    1.00                0 - 1.00i
```

Pozn. `C'` – transponovaná matice s **komplexně sdruženými** čísly

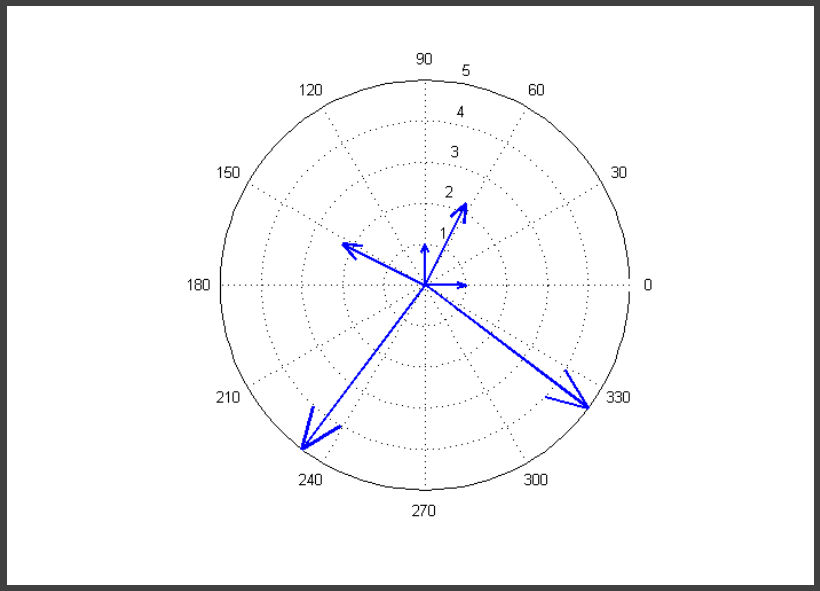
```
ans =
   1.00 - 2.00i   4.00 + 3.00i
  -2.00 - 1.00i   1.00
  -3.00 + 4.00i      0 - 1.00i
```

Práce s komplexními čísly

Vše platí i pro vektory a matice, např:

```
C = [1+2i, -2+i, -3-4i; 4-3i, 1, i]
C =
   1.00 + 2.00i  -2.00 + 1.00i  -3.00 - 4.00i
   4.00 - 3.00i   1.00                0 + 1.00i
```

```
compass(C)
```




```
abs(c)
ans  =
    2.24   2.24   5.00
    5.00   1.00   1.00
```

`angle(C)` – v radiánech

```
ans =
    1.11   2.68  -2.21
   -0.64      0   1.57
```

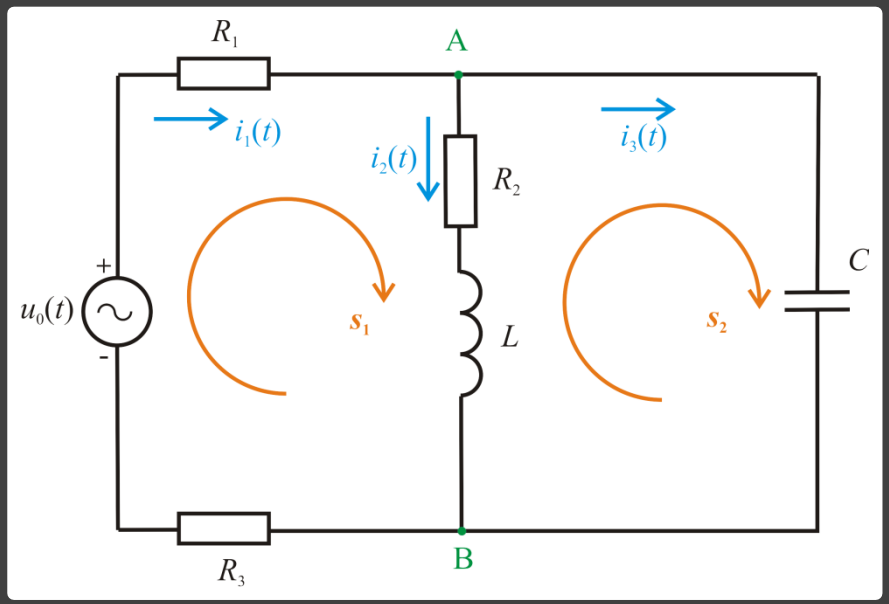
`angle(C).*180/pi` – ve stupních

```
ans =
    63.43   153.43  -126.87
   -36.87        0    90.00
```

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 $\underline{U_0} = U_m e^{j\varphi}$.

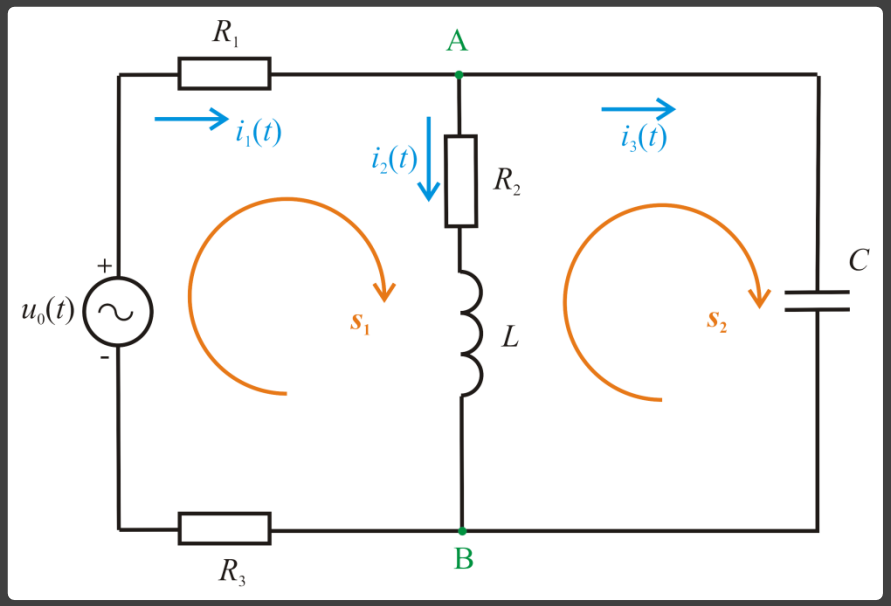
Dáno: $R_1 = 40\ \Omega$; $R_2 = 30\ \Omega$; $R_3 = 10\ \Omega$; $L = 0,1\text{mH}$; $C = 200\mu\text{F}$;
$u_0(t) = 10 \sin(\omega t + 30°)$ V, $\omega = 2\pi f$; $f = 50$ Hz; $\underline{U_0} = 10\, e^{j\,30°}$ V.



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:

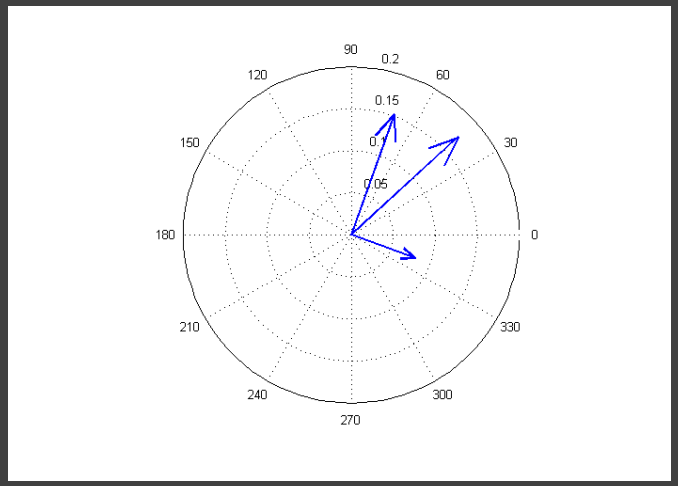
```
x = A \ b
```

```
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
```



Řešení soustavy lineárních rovnic

Pokračování příkladu:

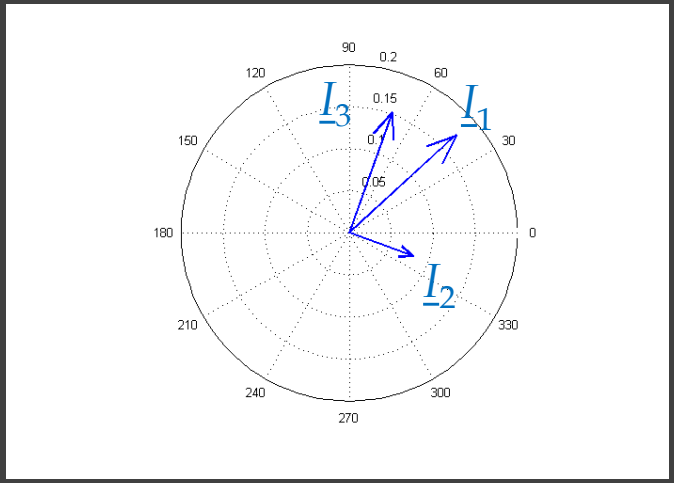
```
x = A \ b
```

```
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
```



Řešení - časové průběhy větvových proudů:

$i_1(t) = 0{,}17 \sin(\omega t + 42{,}4°)$ [A]
$i_2(t) = 0{,}08 \sin(\omega t - 19{,}7°)$ [A]
$i_3(t) = 0{,}15 \sin(\omega t + 70{,}3°)$ [A]

Polynomy

Polynom (mnohočlen) – výraz ze součtů (rozdílů), násobků a celočíselných mocnin proměnných ve tvaru:
$k(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n$

Koeficienty polynomu – čísla $a_0, a_1, a_2, \ldots, a_n$.

Stupeň polynomu $k(x)$ – nejvyšší exponent proměnné x s nenulovým koeficientem

Ve výpočetních systémech jsou polynomy představovány **řádkovými vektory**, které obsahují **koeficienty** seřazené sestupně podle mocniny proměnné:
$k(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n$ – koeficienty – vzestupně
$k(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x^1 + a_0 x^0$ – koeficienty – sestupně
`polynom =` $[a_n, a_{n-1}, \ldots, a_2, a_1, a_0]$

Jak výpočetní systém ví, zda to jsou vektory nebo polynomy?
Nijak. Jsou to vektory a čistě záleží, jak je použijeme.

Polynomy

Polynomy – jako vektory:

$k_1(x) = x^2 + 5x^1 + 6$

$k_1(x) = 1x^2 + 5x^1 + 6x^0$

```
k1 = [1, 5, 6];
```

Stupeň polynomu: 2

Počet prvků ve vektoru: 3

```
length(k1)
ans =  3
```

$k_2(x) = x^3 + 5x + 6$

$k_2(x) = 1x^3 + 0x^2 + 5x^1 + 6x^0$

```
k2 = [1, 0, 5, 6];
```

Stupeň polynomu: 3

Počet prvků ve vektoru: 4

```
length(k2)
ans =  4
```

$k_3(x) = x^4 + 5x$

$k_3(x) = 1x^4 + 0x^3 + 0x^2 + 5x^1 + 0x^0$

```
k3 = [1, 0, 0, 5, 0];
```

Stupeň polynomu: 4

Počet prvků ve vektoru: 5

```
length(k3)
ans =  5
```

Funkce pro práci s polynomy

`polyval(polynom,x)` – vyčíslení `polynom`u (určení hodnoty) pro hodnotu (příp. hodnoty) x.

Příklad: Vyčíslení polynomů k, p, q pro hodnotu $x = 2$

$$k(x) = x^2 + 5x + 6$$
$$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$$
$$q(x) = -x^4 + x^3 - x$$

$k(x) = 1x^2 + 5x^1 + 6x^0$

```
k = [1, 5, 6];
```

Stupeň polynomu: 2

```
length(k)
ans =  3
```

$p(x) = 4x^5 + 0x^4 + 3.1x^3 + (-7)x^2 + 0x^1 + 11x^0$

```
p = [4, 0, 3.1, -7, 0, 11];
```

Stupeň polynomu: 5

```
length(p)
ans =  6
```

$q(x) = -1x^4 + 1x^3 + 0x^2 + (-1)x^1 + 0x^0$

```
q = [-1, 1, 0, -1, 0];
```

Stupeň polynomu: 4

```
length(q)
ans =  5
```

Funkce pro práci s polynomy

Pokračování příkladu:

$k(x) = x^2 + 5x + 6, \quad x = 2$

$k(2) = 2^2 + 5*2 + 6$

```
k = [1,5,6];
```

```
polyval(k,2)
ans =  20
```

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11, \quad x = 2$

```
p = [4,0,3.1,-7,0,11];
```

$p(2) = 4*2^5 + 3.1*2^3 - 7*2^2 + 11$

```
polyval(p,2)
ans =  135.80
```

$q(x) = -x^4 + x^3 - x, \quad x = 2$

```
q = [-1,1,0,-1,0];
```

$q(2) = -2^4 + 2^3 - 2$

```
polyval(q,2)
ans = -10
```

Funkce pro práci s polynomy

Pokračování příkladu:

$k(x) = x^2 + 5x + 6, \quad x = 2$

```
k = [1,5,6];
```

$k(2) = 2^2 + 5*2 + 6 = 20$

```
polyval(k,2)
ans =  20
```

```
2.^2 + 5.*2 + 6
ans =  20
```

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11, \quad x = 2$

```
p = [4,0,3.1,-7,0,11];
```

$p(2) = 4*2^5 + 3.1*2^3 - 7*2^2 + 11$

```
polyval(p,2)
ans =  135.80
```

```
4.*2.^5+3.1.*2.^3-7.*2.^2+11
      ans =  135.80
```

$q(x) = -x^4 + x^3 - x, \quad x = 2$

```
q = [-1,1,0,-1,0];
```

$q(2) = -2^4 + 2^3 - 2$

```
polyval(q,2)
ans = -10
```

```
-2.^4 + 2.^3 - 2
ans = -10
```

Funkce pro práci s polynomy

Příklad: Vyčíslení polynomů k, p, q pro hodnoty vektoru x z intervalu od -2 do 2 s krokem 1

$k(x) = x^2 + 5x + 6$

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$

$q(x) = -x^4 + x^3 - x$

```
x = [-2:2]
x =
    -2   -1    0    1    2
```

```
length(x)
ans =  5
```

```
k = [1, 5, 6];
polyval(k,x)
ans =
     0    2    6   12   20
```

$k(-2) = -2^2 + 5*(-2) + 6 = 0$

$k(-1) = -1^2 + 5*(-1) + 6 = 2$

$k(0) = 0^2 + 5*0 + 6 = 6$

$k(1) = 1^2 + 5*1 + 6 = 12$

$k(2) = 2^2 + 5*2 + 6 = 20$

```
x.^2 + 5.*x + 6
ans =
     0    2    6   12   20
```

Funkce pro práci s polynomy

Pokračování příkladu:

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$,

$q(x) = -x^4 + x^3 - x$, pro x z intervalu od -2 do 2 s krokem 1

```
x = [-2:2]
x =
     -2          -1           0           1           2
```

```
length(x)
ans =  5
```

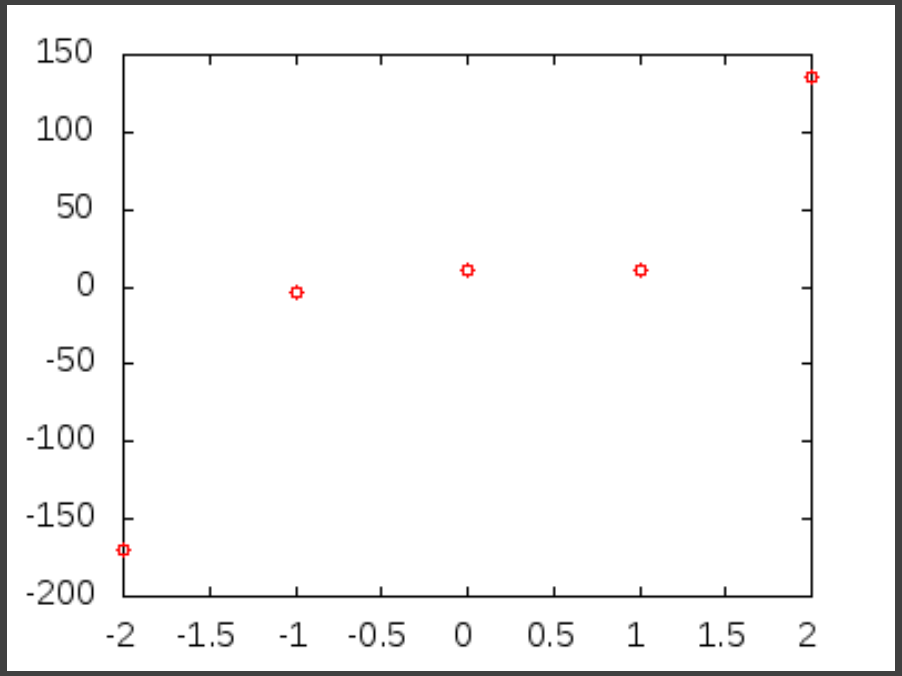
```
p = [4, 0, 3.1, -7, 0, 11];
polyval(p,x)
ans =
-169.80     -3.10     11.00     11.10     135.80
```

```
q = [-1, 1, 0, -1, 0];
polyval(q,x)
ans =
    -22          -1           0          -1         -10
```

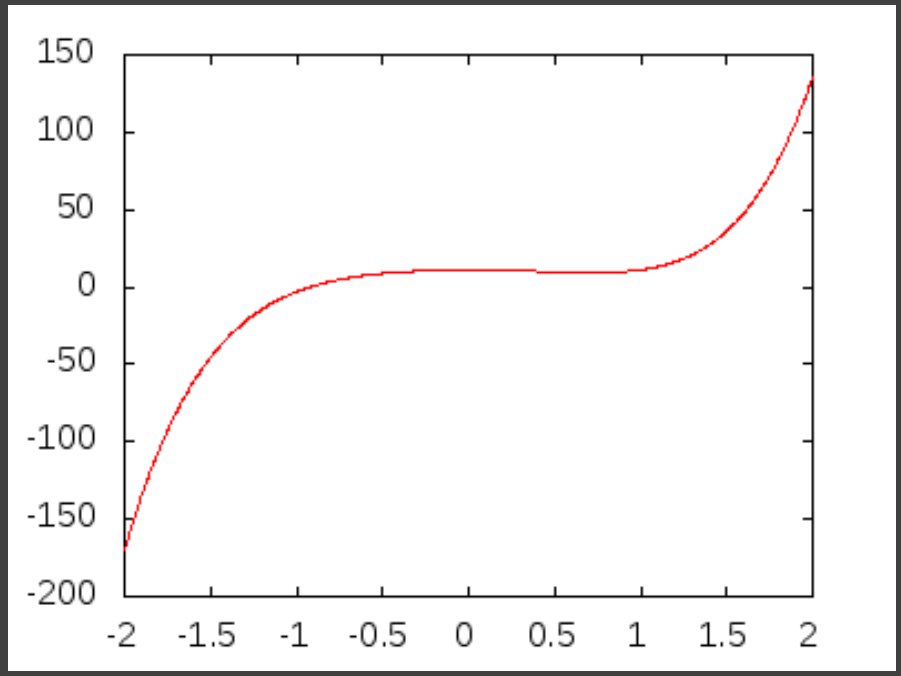
Funkce pro práci s polynomy

Příklad: Vykreslení hodnot polynomu $p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ pro hodnoty vektoru x z intervalu od **-2** do **2**

```
x1 = [-2:2];
p = [4,0,3.1,-7,0,11];
hP = polyval(p,x1)
plot(x1, hP, 'or')
```




```
x2 = linspace(-2,2,1000);
p = [4,0,3.1,-7,0,11];
hP = polyval(p,x2)
plot(x2, hP, '.r')
```

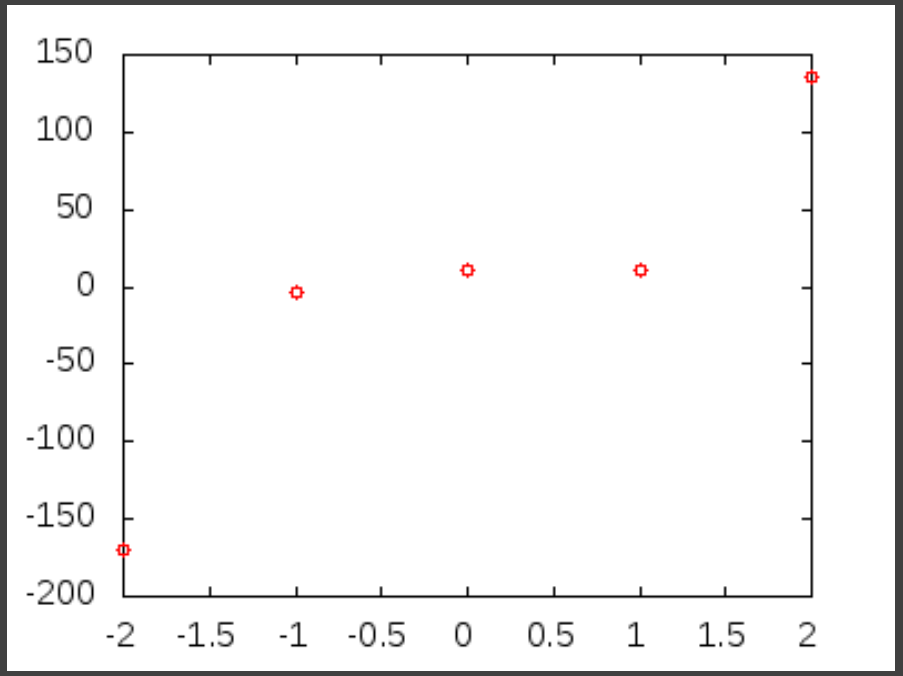


Funkce pro práci s polynomy

Příklad: Vykreslení hodnot polynomu $p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ pro hodnoty vektoru x z intervalu od **-2** do **2**

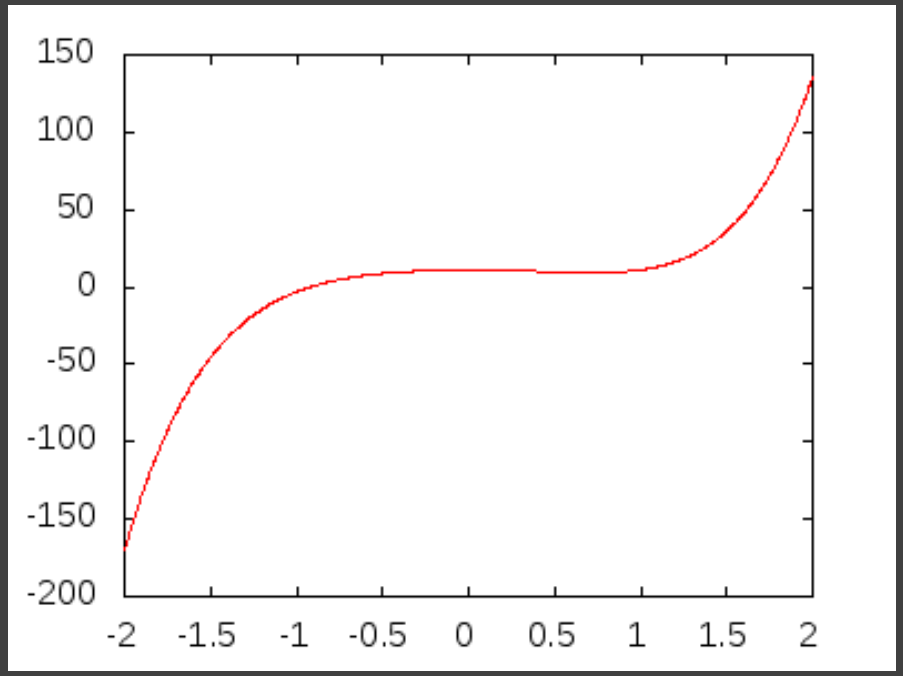
5 prvků na ose x

```
x1 = [-2:2];
p = [4,0,3.1,-7,0,11];
hP = polyval(p,x1)
plot(x1, hP, 'or')
```



1000 prvků na ose x

```
x2 = linspace(-2,2,1000);
p = [4,0,3.1,-7,0,11];
hP = polyval(p,x2)
plot(x2, hP, '.r')
```



Funkce pro práci s polynomy

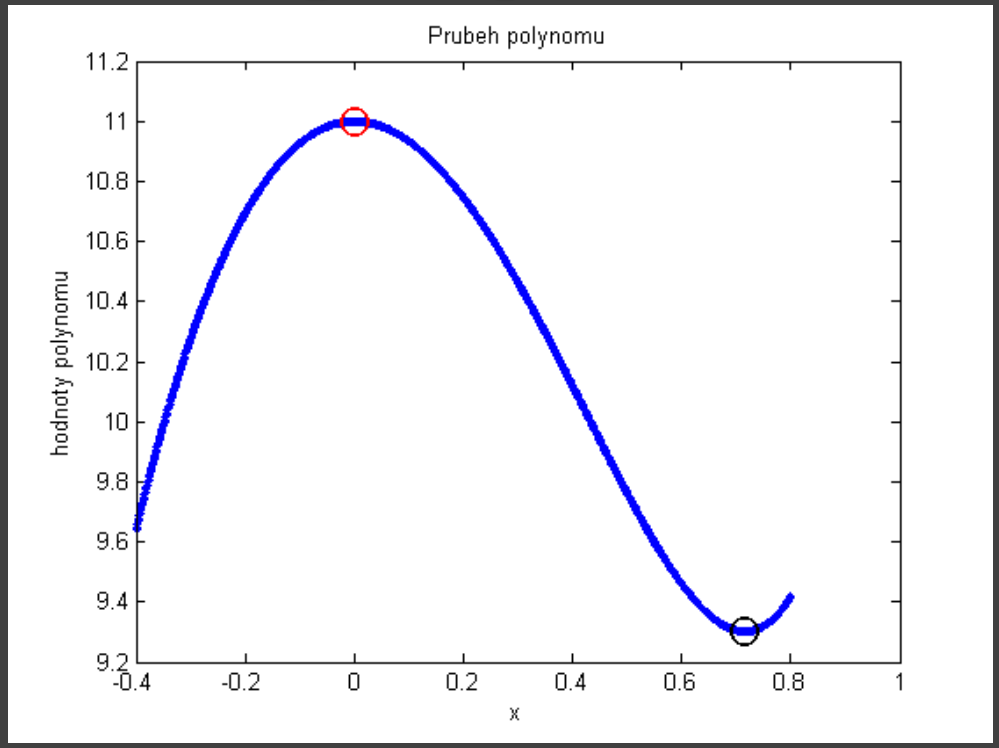
Příklad: Funkce, která vykreslí zadaný polynom p v zadaném rozsahu a najde a označí v grafu maximum a minimum v daném rozsahu.

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
hold off
end
```

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce s příslušnými vstupními daty:

```
p = [4,0,3.1,-7,0,11];
kresliPolynom(p,-0.4,0.8,1000)
```



`Max = 11` – maximum z vektoru `hodnotyPolynomu`
`indexMax = 334` – poloha (index) maxima ve vektoru
`Min = 9.3020` – minimum z vektoru `hodnotyPolynomu`
`indexMin = 930` – poloha (index) minima ve vektoru

Funkce pro práci s polynomy

conv(p,q) – **součin** polynomů p a q:

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ `p = [4,0,3.1,-7,0,11];`

$q(x) = -x^4 + x^3 - x$ `q = [-1,1,0,-1,0];`

```
s = conv(p,q)
s =
 Columns 1 through 7:
 -4.00    4.00   -3.10    6.10   -7.00  -14.10   18.00
Columns 8 through 10:
  0.00  -11.00    0.00
```

```
length(s)
ans =  10
```

Funkce pro práci s polynomy

conv(p,q) – **součin** polynomů p a q:

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ `p = [4,0,3.1,-7,0,11];`

$q(x) = -x^4 + x^3 - x$ `q = [-1,1,0,-1,0];`

```
s = conv(p,q)
s =
 Columns 1 through 7:
 -4.00    4.00   -3.10    6.10   -7.00  -14.10   18.00
Columns 8 through 10:
  0.00  -11.00    0.00
```

```
length(s)
ans =  10
```

Počet prvků ve vektoru: **10** => stupeň polynomu s: **9**

Tedy:

$s(x) = p(x)^* q(x) = (4x^5 + 3.1x^3 - 7x^2 + 11)^*(-x^4 + x^3 - x) =$

$= -4x^9 + 4x^8 - 3.1x^7 + 6.1x^6 - 7x^5 - 14.1x^4 + 18x^3 - 11x$

Funkce pro práci s polynomy

`deconv(p,q)` – vrací vektor koeficientů **podílu** polynomů p a q:

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$

$q(x) = -x^4 + x^3 - x$

```
p = [4,0,3.1,-7,0,11];
q = [-1,1,0,-1,0];
```

```
deconv(p,q)
ans =   -4.00   -4.00
```

$$\frac{p(x)}{q(x)} = \frac{4x^5 + 3.1x^3 - 7x^2 + 11}{-x^4 + x^3 - x}$$

Funkce pro práci s polynomy

deconv(p,q) – vrací vektor koeficientů podílu polynomů p a q:

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ `p = [4,0,3.1,-7,0,11];`

$q(x) = -x^4 + x^3 - x$ `q = [-1,1,0,-1,0];`

```
deconv(p,q)
ans =   -4.00   -4.00
```

$$\frac{p(x)}{q(x)} = \frac{4x^5 + 3.1x^3 - 7x^2 + 11}{-x^4 + x^3 - x}$$

[r,z] = deconv(p,q) – vrací koeficienty podílu polynomů p a q ve vektoru r a zbytek po dělení polynomů z

```
[r,z] = deconv(p,q)
r =
      -4.00   -4.00
z =
       0       0      7.10   -11.00   -4.00   11.00
```

```
length(r)
ans =   2
```

```
length(z)
ans =   6
```

Tedy:

$$r(x) = \frac{p(x)}{q(x)} = \frac{4x^5 + 3.1x^3 - 7x^2 + 11}{-x^4 + x^3 - x} = -4x^1 - 4x^0 = -4x - 4,$$

zbytek $z(x) = 0x^5 + 0x^4 + 7.1x^3 - 11x^2 - 4x + 11x^0 = 7.1x^3 - 11x^2 - 4x + 11$

Funkce pro práci s polynomy

$p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$
$q(x) = -x^4 + x^3 - x$

```
p = [4,0,3.1,-7,0,11];
q = [-1,1,0,-1,0];
```

```
[r,z] = deconv(p,q)
r =
    -4.00   -4.00
z =
     0       0       7.10   -11.00   -4.00   11.00
```

r – podíl polynomů
z – zbytek po dělení

Zkouška:

Je-li $\frac{p(x)}{q(x)} = r(x)$, zbytek $z(x)$,

potom $p(x) = q(x) * r(x) + z(x)$

```
u = conv(q,r) + z
u =
    4.00    0       3.10    -7.00    0       11.00
```

```
length(u)
ans =  6
```

Tedy: $u(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$, tj. polynom $p(x)$.

Funkce pro práci s polynomy

`roots(p)` – výpočet **kořenů** polynomu p

Příklad: výpočet kořenů polynomu $p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ (všechna x pro která je hodnota polynomu 0), tj. řešení rovnice:

$4x^5 + 3.1x^3 - 7x^2 + 11 = 0$

$4x^5 + 0x^4 + 3.1x^3 + (-7)x^2 + 0x^1 + 11x^0 = 0$

```
p = [4,0,3.1,-7,0,11];
x = roots(p)
x =
   0.97274 + 0.57470i
   0.97274 - 0.57470i
  -0.51238 + 1.44129i
  -0.51238 - 1.44129i
  -0.92071
```

Tj. řešení rovnice (kořeny polynomu p) jsou :

$x_1 = 0.97274 + 0.57470i$

$x_2 = 0.97274 - 0.57470i$

$x_3 = -0.51238 + 1.44129i$

$x_4 = -0.51238 - 1.44129i$

$x_5 = -0.92071$

Pozn. Stupněm polynomu $p(x)$ rozumíme nejvyšší exponent proměnné x s nenulovým koeficientem. Stupeň polynomu $p(x)$ je 5 => položíme-li polynom p rovný 0, má pak 5 kořenů)

Funkce pro práci s polynomy

roots(p) – výpočet **kořenů** polynomu p

Příklad: výpočet kořenů polynomu $p(x) = 4x^5 + 3.1x^3 - 7x^2 + 11$ (všechna x pro která je hodnota polynomu 0), tj. řešení rovnice:

$4x^5 + 3.1x^3 - 7x^2 + 11 = 0$

$4x^5 + 0x^4 + 3.1x^3 + (-7)x^2 + 0x^1 + 11x^0 = 0$

```
p = [4,0,3.1,-7,0,11];
x = roots(p)
x =
   0.97274 + 0.57470i
   0.97274 - 0.57470i
  -0.51238 + 1.44129i
  -0.51238 - 1.44129i
  -0.92071
```

```
polyval(p,x)  - zkouška
ans =
  1.0e-015 *
     0.1066 - 0.1954i
     0.1066 + 0.1954i
    -0.4263 - 0.2398i
    -0.4263 + 0.2398i
    -0.0533
```

Pozn. Stupněm polynomu $p(x)$ rozumíme nejvyšší exponent proměnné x s nenulovým koeficientem. Stupeň polynomu $p(x)$ je 5 => položíme-li polynom p rovný 0, má pak 5 kořenů)

Funkce pro práci s polynomy

poly(x) – zjištění **koeficientů polynomu** z **kořenů** (má-li vektor *x* *n* prvků, vypíše se vektor s *n*+1 prvky – koeficienty polynomu *n*-tého stupně), funkce je opakem **roots()**, první koeficient polynomu, tj. koeficient u nejvyššího exponentu proměnné *x* je vždy roven 1.

Příklad:

```
p = [4,0,3.1,-7,0,11];
x = roots(p)
x =
   0.9727 + 0.5747i
   0.9727 - 0.5747i
  -0.5124 + 1.4413i
  -0.5124 - 1.4413i
  -0.9207
```

```
length(p)
ans =  6
```

```
length(x)
ans =  5
```

```
a = poly(x)
a =
    1.00   0   0.775 -1.75   0   2.75
4.*a
ans =
    4.00   0   3.10  -7.00   0  11.00
```

```
length(a)
ans =  6
```

Funkce pro práci s polynomy

poly(x) – zjištění **koeficientů polynomu** z **kořenů** (má-li vektor x n prvků, vypíše se vektor s n+1 prvky – koeficienty polynomu n-tého stupně), funkce je opakem **roots()**, první koeficient polynomu, tj. koeficient u nejvyššího exponentu proměnné x je vždy roven 1.

Příklad:

```
p = [4,0,3.1,-7,0,11];
x = roots(p)
x =
   0.9727 + 0.5747i
   0.9727 - 0.5747i
  -0.5124 + 1.4413i
  -0.5124 - 1.4413i
  -0.9207
```

```
length(p)
ans =  6
```

```
length(x)
ans =  5
```

```
a = poly(x)
a =
   1.00   0   0.775  -1.75   0   2.75
```

```
length(a)
ans =  6
```

$x^5 + 0.775x^3 - 1.75x^2 + 11$

```
4.*a
ans =
   4.00   0   3.10  -7.00   0  11.00
```

$4x^5 + 3.1x^3 - 7x^2 + 11$

Řešení kvadratické rovnice

Kvadratická rovnice – kvadratický polynom, který je položen rovný nule. Stupeň kvadratického polynomu (např. $k(x) = x^2 + 5x + 6$) je 2. Kvadratická rovnice má tedy 2 kořeny.

Příklad: řešení rovnice $x^2 + 5x + 6 = 0$

$k(x) = x^2 + 5x + 6$

$k(x) = 1x^2 + 5x^1 + 6x^0$

```
k = [1, 5, 6];
x = roots(k);
x =
   -3.0000
   -2.0000
```

– tj. x_1,

– tj. x_2,

tato kvadratická rovnice má reálné kořeny.

Zkouška:

$(-3)^2 + 5*(-3) + 6 = 0$

$(-2)^2 + 5*(-2) + 6 = 0$, tj.

```
polyval(k,x(1))
ans =  1.3323e-15
polyval(k,x(2))
ans =  2.2204e-16
```

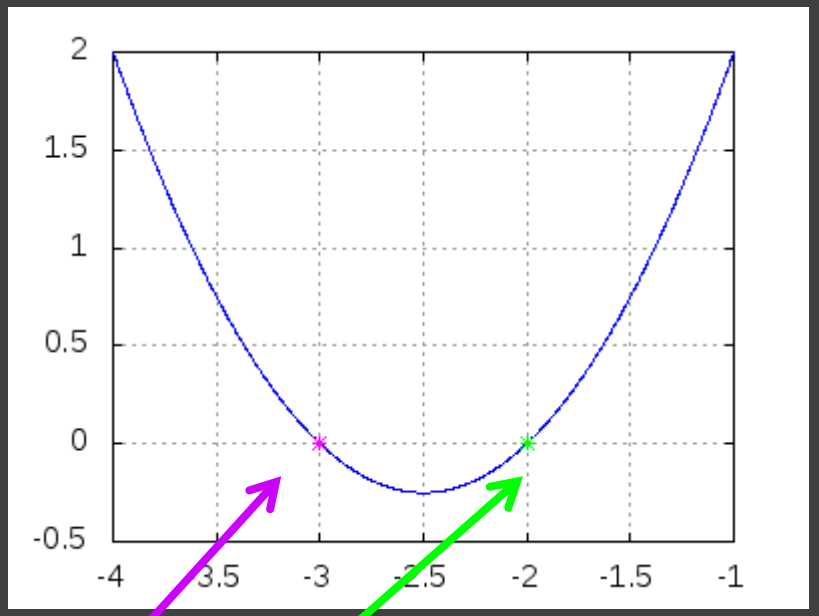
Zjištění koeficientů polynomu z kořenů

```
poly(x)
ans =
     1     5     6
```

Funkce pro práci s polynomy

Pokračování příkladu:
Spočítáme průběh polynomu
$k(x) = x^2 + 5x + 6$
a vykreslíme graf i s kořeny

```
k = [1, 5, 6];
x = roots(k);
xx = linspace(-4,1,1000);
hP = polyval(k,xx);
plot(xx,hP)
hold on
plot(x(1), polyval(k,x(1)),'*m')
plot(x(2), polyval(k,x(2)),'*g')
hold off
```



Řešení kvadratické rovnice

Příklad: řešení rovnice $2x^2 + 4x + 6 = 0$

$k_2(x) = 2x^2 + 4x + 6$

$k_2(x) = 2x^2 + 4x^1 + 6x^0$

```
k2 = [2,4,6];
x2 = roots(k2)
x2 =
   -1.0000 + 1.4142i  – tj. x1,
   -1.0000 - 1.4142i  – tj. x2,
```

tato kvadratická rovnice má komplexní kořeny.

Zkouška:

```
polyval(k2,x2(1))
ans =
      8.8818e-016
polyval(k2,x2(2))
ans =
      8.8818e-016
```

Zjištění koeficientů polynomu z kořenů

```
f = poly(x2)
f =
    1.00    2.00    3.00
2.*f
ans =
    2.00    4.00    6.00
```

$x^2 + 2x + 3$

$2x^2 + 4x + 6$, tj. polynom k_2

Řešení kvadratické rovnice

Pokračování příkladu:

Obě strany kvadratické rovnice můžeme vydělit dvěma:

$2x^2 + 4x + 6 = 0 \quad | : 2$

$1x^2 + 2x + 3 = 0,$

tj. toto jsou koeficienty získané pomocí `poly(x2)`

```
roots([1,2,3])
ans =
   -1.0000 + 1.4142i
   -1.0000 - 1.4142i
```

– tj. x_1,

– tj. x_2,

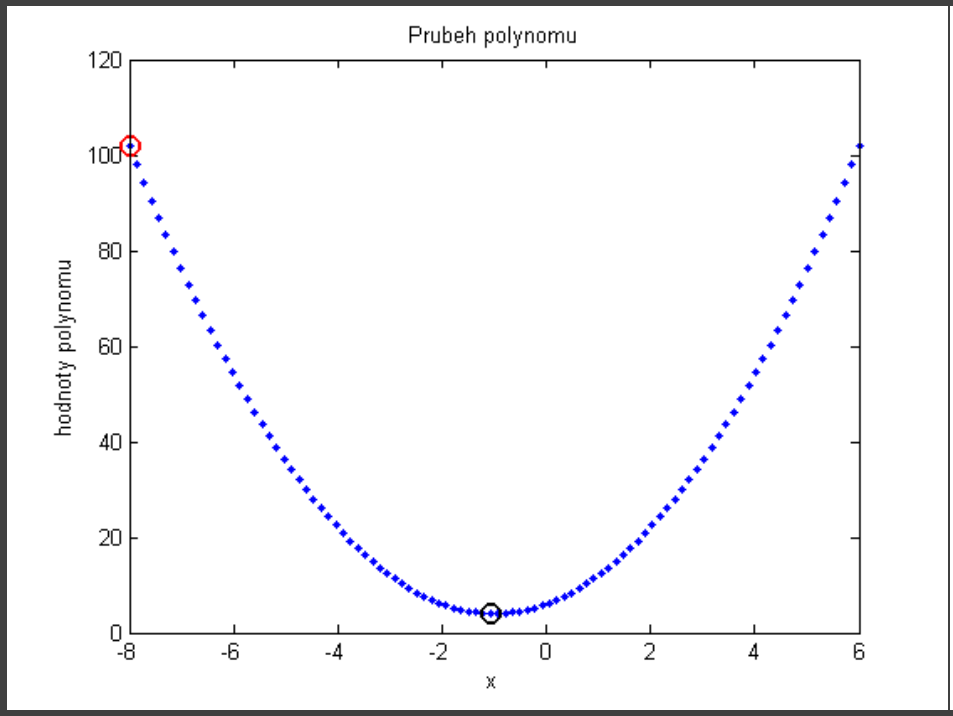
=> kvadratická rovnice $1x^2 + 2x + 3 = 0$ má totožné řešení jako kvadratická rovnice $2x^2 + 4x + 6 = 0$.

Řešení kvadratické rovnice

Pokračování příkladu s využitím naší uživatelské funkce **`kresliPolynom`** s parametry **`p,odkudX,kamX,prvkuX`** pro vykreslení polynomu:

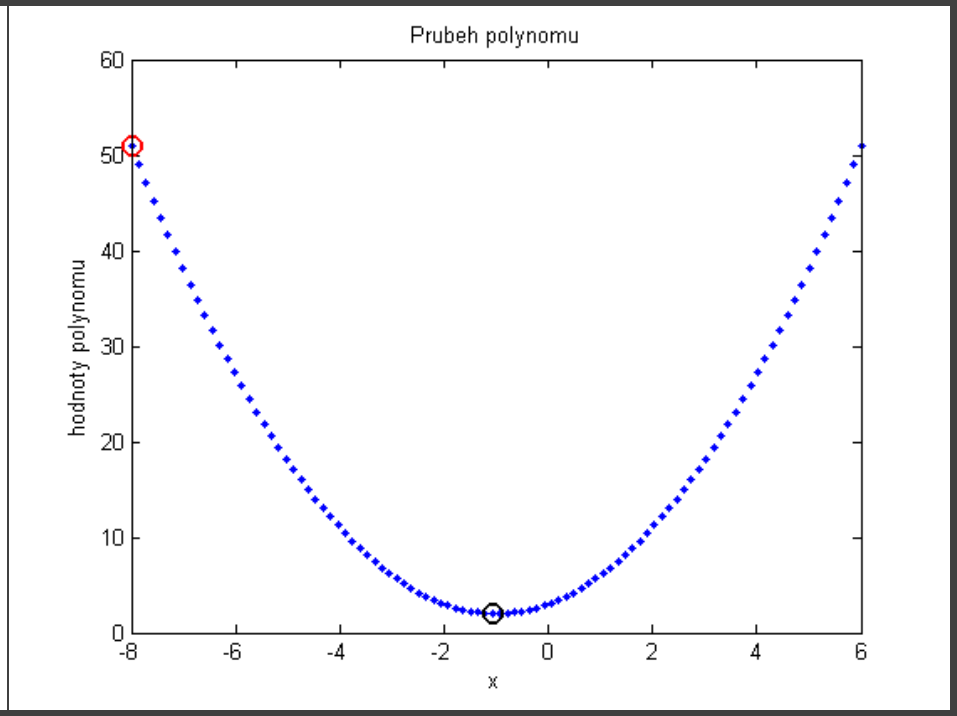
$2x^2 + 4x + 6$ v rozsahu od -8 do 6

```
kresliPolynom([2,4,6],-8,6,100)
```



$1x^2 + 2x + 3$ v rozsahu od -8 do 6

```
kresliPolynom([1,2,3],-8,6,100)
```



Pozn. Komplexní kořeny nelze zobrazit s polynomem, ale jen v komplexní rovině

Řešení kvadratické rovnice

Pokračování příkladu s využitím naší uživatelské funkce **`kresliPolynom`** s parametry **`p,odkudX,kamX,prvkuX`** pro vykreslení polynomu:

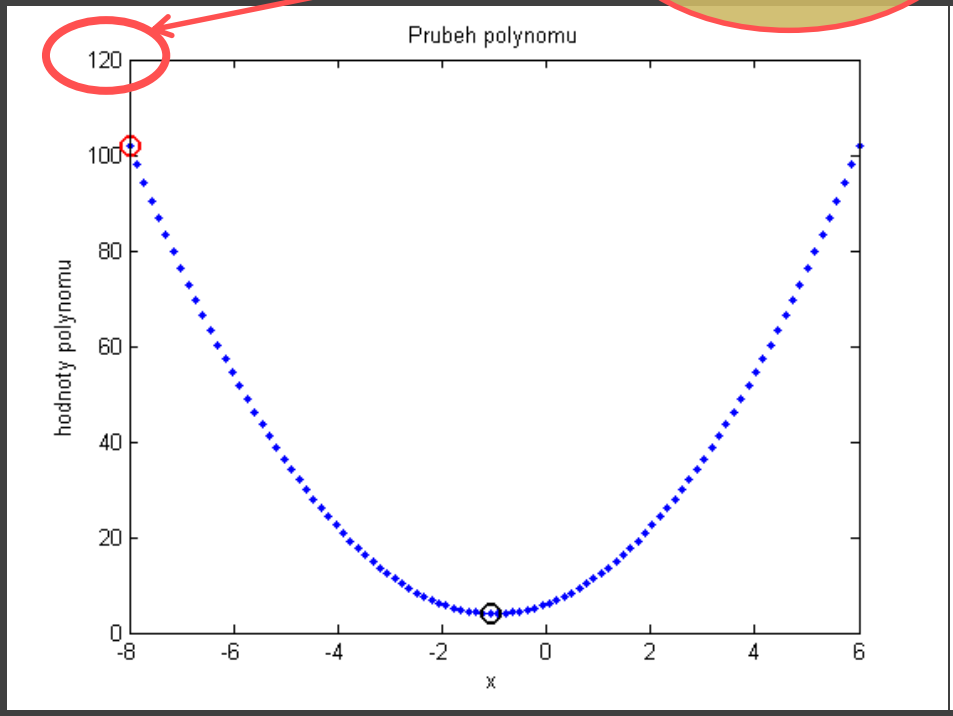
$2x^2 + 4x + 6$ v rozsahu od -8 do 6

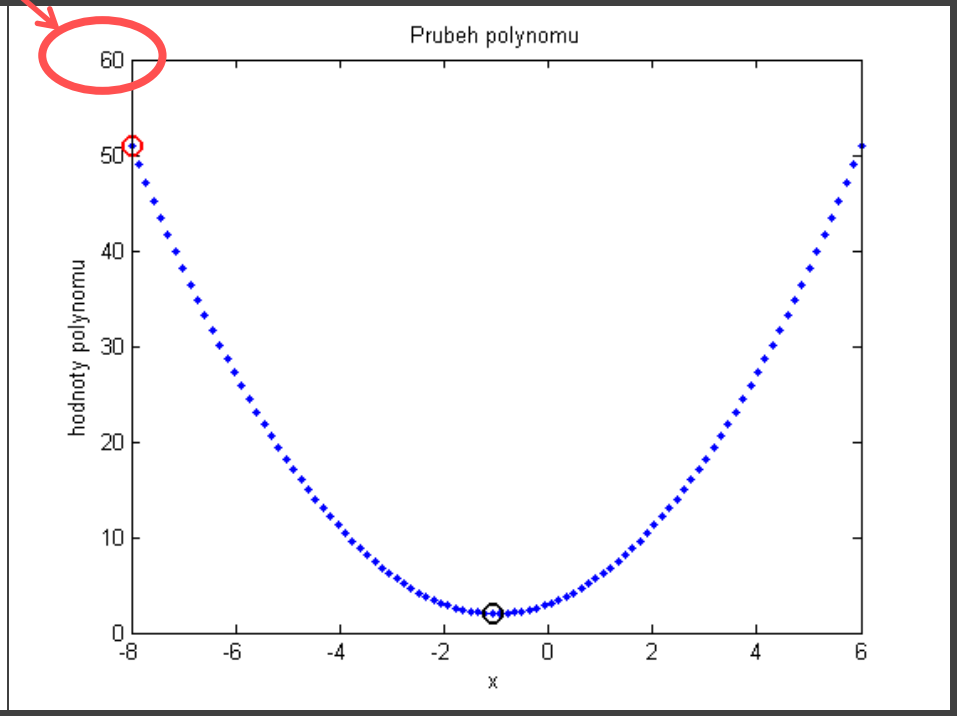
```
kresliPolynom([2,4,6],-8,6,100)
```

$1x^2 + 2x + 3$ v rozsahu od -8 do 6

```
kresliPolynom([1,2,3],-8,6,100)
```

Rozsah os *y* se liší





Pozn. Komplexní kořeny nelze zobrazit s polynomem, ale jen v komplexní rovině

Řešení kvadratické rovnice

Příklad:
řešení rovnice $2x^2 + 6 = 0$

$k_3(x) = 2x^2 + 0x^1 + 6x^0$

```
k3 = [2,0,6];
x3 = roots(k3)
x3 =
   0 + 1.7321i  – tj. x1,
   0 - 1.7321i  – tj. x2,
```

tato kvadratická rovnice má komplexní kořeny bez reálné části.

```
polyval(k3,x3)
ans =
  1.0e-015 *
    0.8882
    0.8882
```

```
poly(x3).*2
ans =
     2     0     6
```

Příklad:
řešení rovnice $x^2 - 2x + 1 = 0$

$k_4(x) = 1x^2 + (-2)x^1 + 1x^0$

```
k4 = [1,-2,1];
x4 = roots(k4)
x4 =
     1          – tj. x1,
     1          – tj. x2,
```

platí $x_1 = x_2$, tato kvadratická rovnice má dvojnásobný reálný kořen.

```
polyval(k4,x4)
ans =
     0
     0
```

```
poly(x4)
ans =
     1    -2     1
```

Funkce pro práci s polynomy

Příklad: řešení rovnice s polynomem vyššího stupně

$$x^6 - 14x^4 + 49x^2 = 36 \quad |- 36$$

$$x^6 - 14x^4 + 49x^2 - 36 = 0$$

Polynom $x^6 - 14x^4 + 49x^2 - 36$ položíme rovný 0,

$1x^6 + 0x^5 + (-14)x^4 + 0x^3 + 49x^2 + 0x^1 + (-36)x^0 = 0$

```
v = [1,0,-14,0,49,0,-36];
x = roots(v)
x =
   -3.0000
    3.0000
   -2.0000
   -1.0000
    2.0000
    1.0000
```

$x_1 = -3$

$x_2 = 3$

$x_3 = -2$

$x_4 = -1$

$x_5 = 2$

$x_6 = 1$

```
length(v)
ans =  7
```

počet prvků ve vektoru je vždy o 1 větší než stupeň polynomu

Pozn. Stupeň polynomu $v(x)$ je 6 => položíme-li polynom v rovný 0, má pak 6 kořenů, rovnice má tedy 6 řešení – počet řešení je stejný jako stupeň polynomu.

Funkce pro práci s polynomy

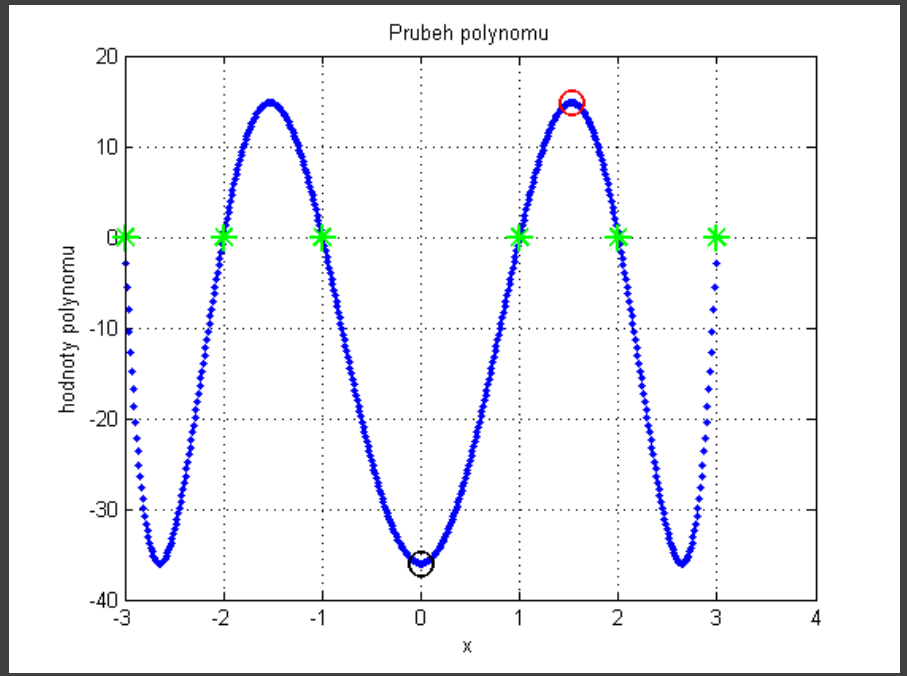
Pokračování příkladu: řešení rovnice

$$x^6 - 14x^4 + 49x^2 = 36$$

```
v = [1,0,-14,0,49,0,-36];
x = roots(v)
x =
   -3.0000
    3.0000
   -2.0000
   -1.0000
    2.0000
    1.0000
```

Volání funkce s příslušnými vstupními daty:

```
kresliPolynom(v,-3,3,500)
hold on
plot(roots(v),0,'g*')
hold off
```



Využijme funkci pro vykreslení polynomu: **kresliPolynom** pro *v* v rozsahu od **-3** do **3** a do grafu přikreslíme všechna řešení rovnice.

Funkce pro práci s polynomy

Pokračování příkladu: řešení rovnice

$$x^6 - 14x^4 + 49x^2 = 36$$

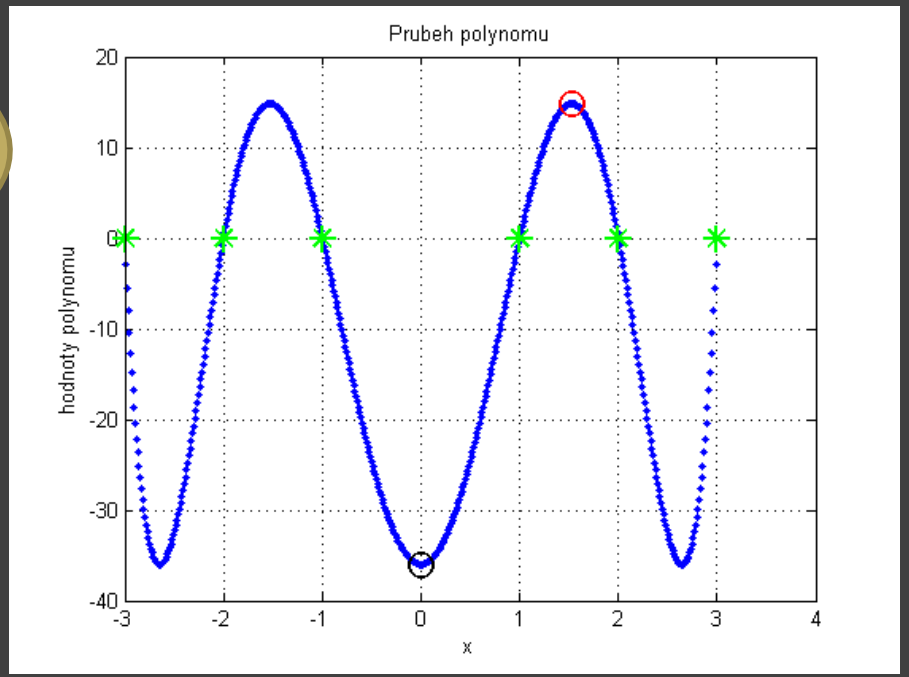
```
v = [1,0,-14,0,49,0,-36];
x = roots(v)
x =
   -3.0000
    3.0000
   -2.0000
   -1.0000
    2.0000
    1.0000
```

a to by mohlo být přímo součástí funkce `kresliPolynom()`

Využijme funkci pro vykreslení polynomu: `kresliPolynom` pro v v rozsahu od **-3** do **3** a do grafu přikreslíme všechna řešení rovnice.

Volání funkce s příslušnými vstupními daty:

```
kresliPolynom(v,-3,3,500)
hold on
plot(roots(v),0,'g*')
hold off
```



Funkce pro práci s polynomy

Příklad: Funkce, která vykreslí zadaný polynom p v zadaném rozsahu, najde a označí v grafu maximum a minimum v daném rozsahu a pokud jsou všechny kořeny reálné, vykreslí je také do grafu. Výstupním parametrem budou kořeny polynomu p.

Takže do uživatelské funkce `kresliPolynom()`, která byla uvedena dříve, přidáme výpočet a zobrazení kořenů (do grafu vykreslíme jen reálné kořeny)

Pozn. Komplexní kořeny lze zobrazit jen v komplexní rovině.

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);




hold off
end
```

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
```

výpočet kořenů

```
hold off
end
```

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')


end
hold off
end
```

výpočet kořenů

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')


end
hold off
end
```

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')
else
    disp('koreny nejsou realne')
end
hold off
end
```

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

```
function kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')
else
    disp('koreny nejsou realne')
end
hold off
end
```

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

hláška pro komplexní kořeny

```
function k = kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')
else
    disp('koreny nejsou realne')
end
hold off
end
```

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

hláška pro komplexní kořeny

```
function k = kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')
else
    disp('koreny nejsou realne')
end
hold off
end
```

Přidán výstupní parametr funkce

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

hláška pro komplexní kořeny

```
function k = kresliPolynom(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
[Max,indexMax] = max(hodnotyPolynomu)
plot(x(indexMax),Max,'or')
[Min,indexMin] = min(hodnotyPolynomu)
plot(x(indexMin),Min,'ok')
k = roots(p);
if (isreal(k))
    plot(k, polyval(p,k),'og')
else
    disp('koreny nejsou realne')
end
hold off
end
```

Přidán výstupní parametr funkce

výpočet kořenů

zobrazení kořenů (do grafu jen reálné kořeny) – zelená kolečka

hláška pro komplexní kořeny

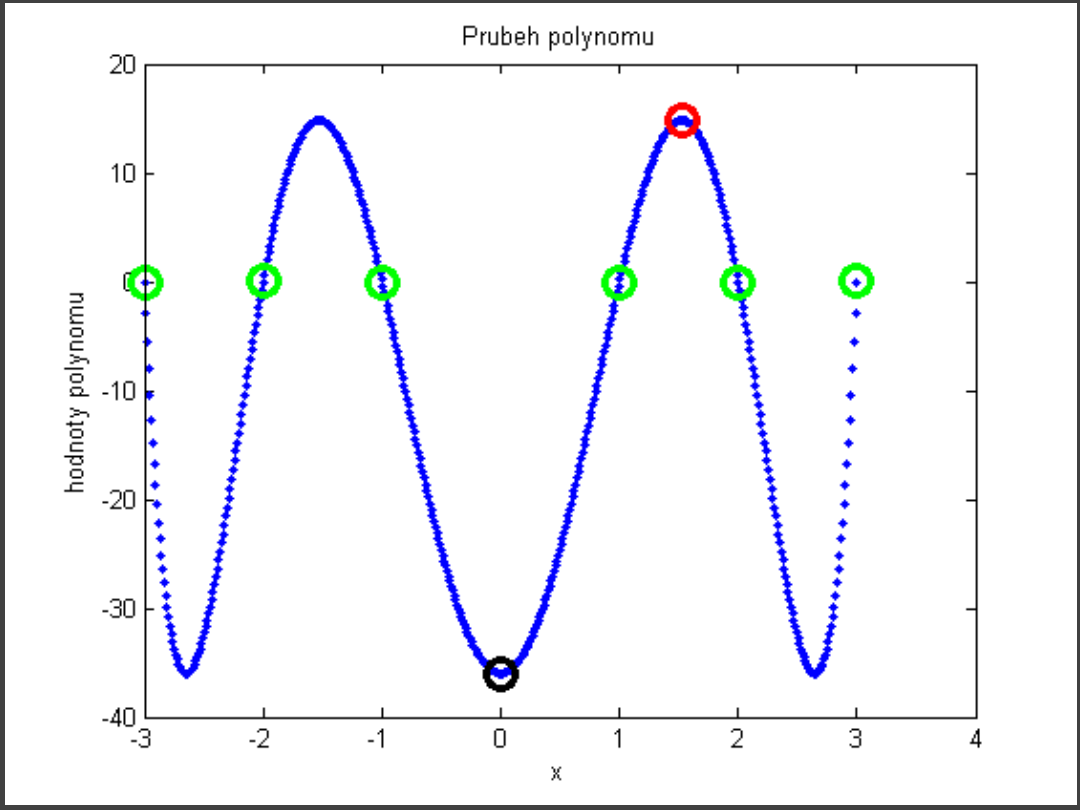
Funkce pro práci s polynomy

Pokračování příkladu: volání funkce pro polynom

$$v(x) = x^6 - 14x^4 + 49x^2 - 36$$

v rozsahu od **-3** do **3** s 500 prvky na ose x

```
v = [1,0,-14,0,49,0,-36];
x = kresliPolynom(v,-3,3,500)
```




```
Max =
    14.8128
indexMax =
    378
Min =
   -35.9982
indexMin =
    250
x =
   -3.0000
    3.0000
   -2.0000
   -1.0000
    2.0000
    1.0000
```

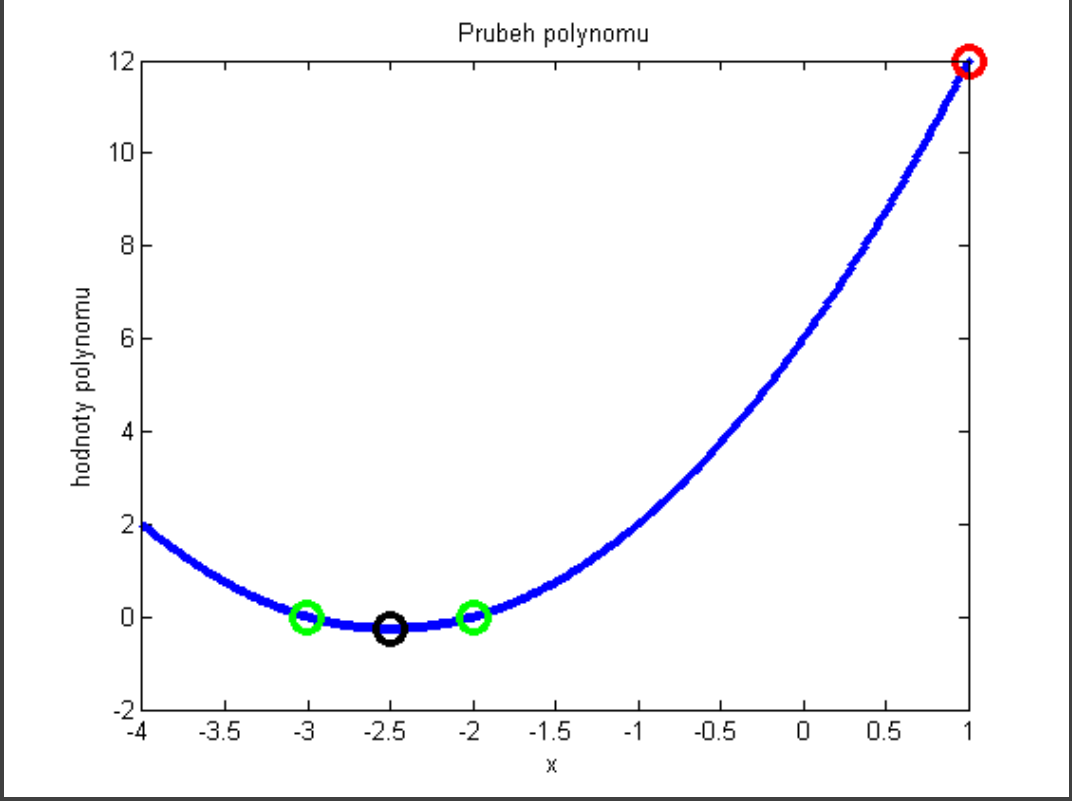
Funkce pro práci s polynomy

Pokračování příkladu: volání funkce pro polynom

$$k(x) = x^2 + 5x + 6$$

v rozsahu od **-4** do **1** s 500 prvky na ose x

```
k = [1, 5, 6];
x = kresliPolynom(k,-4,1,500)
```




```
Max =
    12
indexMax =
   500
Min =
   -0.2500
indexMin =
   151

x =
   -3.0000
   -2.0000
```

Funkce pro práci s polynomy

Příklad: řešení rovnice s polynomem vyššího stupně

$$7x^7 = -7x \quad | + 7x$$

$$7x^7 + 7x = 0$$

Polynom $7x^7 + 7x$ položíme rovný 0,

$$7x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 7x^1 + 0x^0 = 0$$

```
w = [7,0,0,0,0,0,7,0];
x = roots(w)
x =
   0                      – tj. x1 ,
  -0.8660 + 0.5000i       – tj. x2 ,
  -0.8660 - 0.5000i       – tj. x3 ,
   0.0000 + 1.0000i       – tj. x4 ,
   0.0000 - 1.0000i       – tj. x5,
   0.8660 + 0.5000i       – tj. x6 ,
   0.8660 - 0.5000i       – tj. x7,
```

```
length(w)
ans =  8
```

počet prvků ve vektoru je vždy o 1 větší než stupeň polynomu

Pozn. Stupeň polynomu $w(x)$ je 7 => položíme-li polynom w rovný 0, má pak 7 kořenů, rovnice má tedy 7 řešení – počet řešení je stejný jako stupeň polynomu.

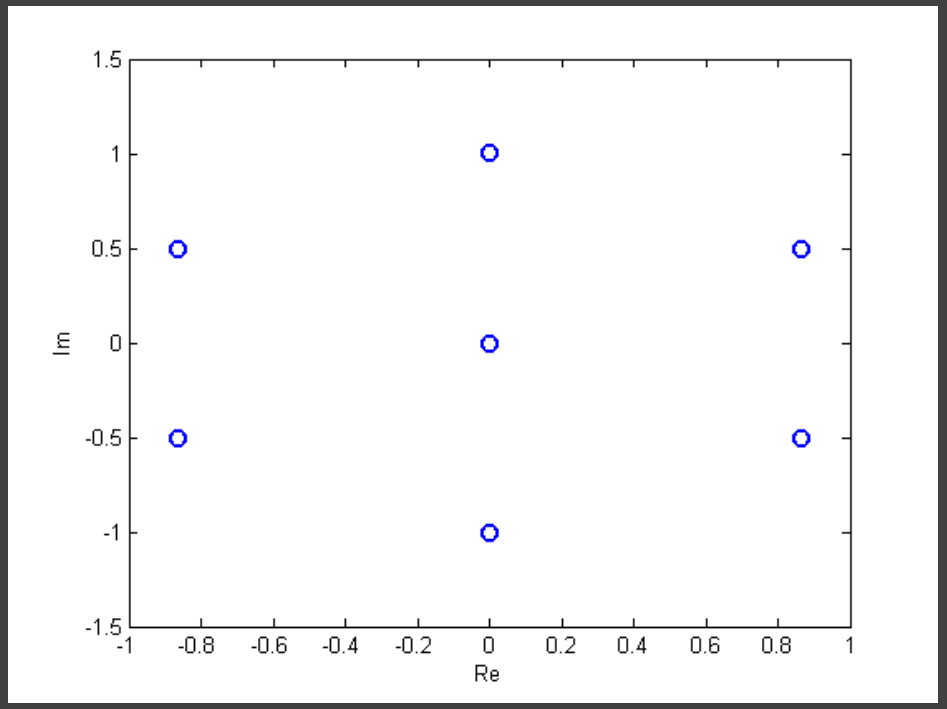
Funkce pro práci s polynomy

Pokračování příkladu: řešení rovnice

$$7x^7 = -7x$$

```
w = [7,0,0,0,0,0,7,0];
x = roots(w)
x =
     0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

```
plot(x,'o')
xlabel('Re')
ylabel('Im')
```



Pozn. Komplexní kořeny lze zobrazit jen v komplexní rovině (např. pomocí `plot()` na ose x reálná část, na ose y imaginární část).

Funkce pro práci s polynomy

Pokračování příkladu: řešení rovnice

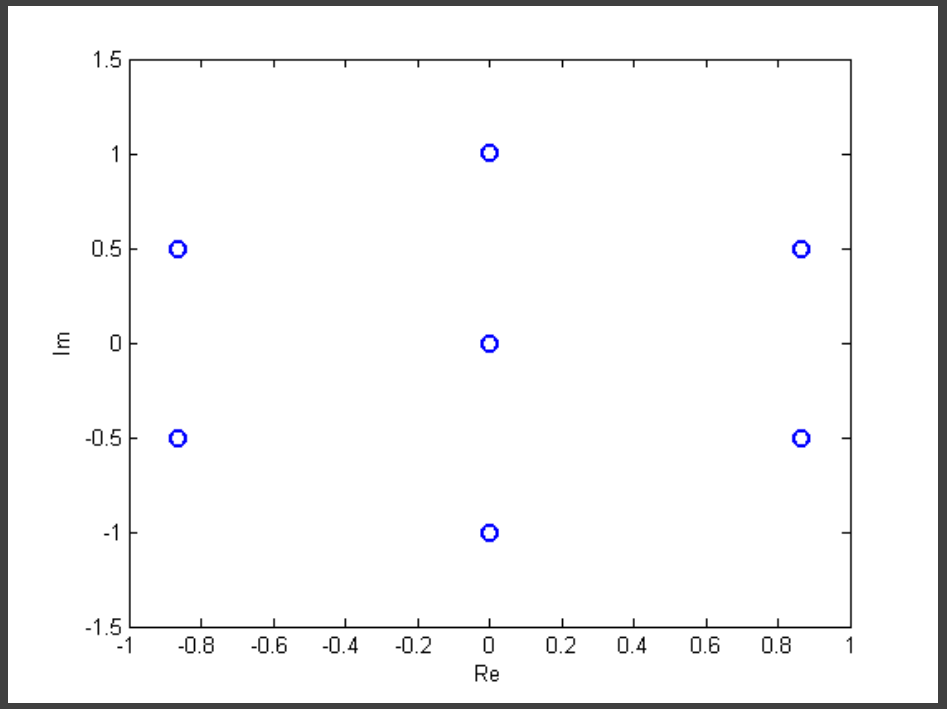
$$7x^7 = -7x$$

```
w = [7,0,0,0,0,0,7,0];
x = roots(w)
x =
```

reálný kořen

```
     0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

```
plot(x,'o')
xlabel('Re')
ylabel('Im')
```



Pozn. Komplexní kořeny lze zobrazit jen v komplexní rovině (např. pomocí `plot()` na ose x reálná část, na ose y imaginární část).

Funkce pro práci s polynomy

Pokračování příkladu: řešení rovnice

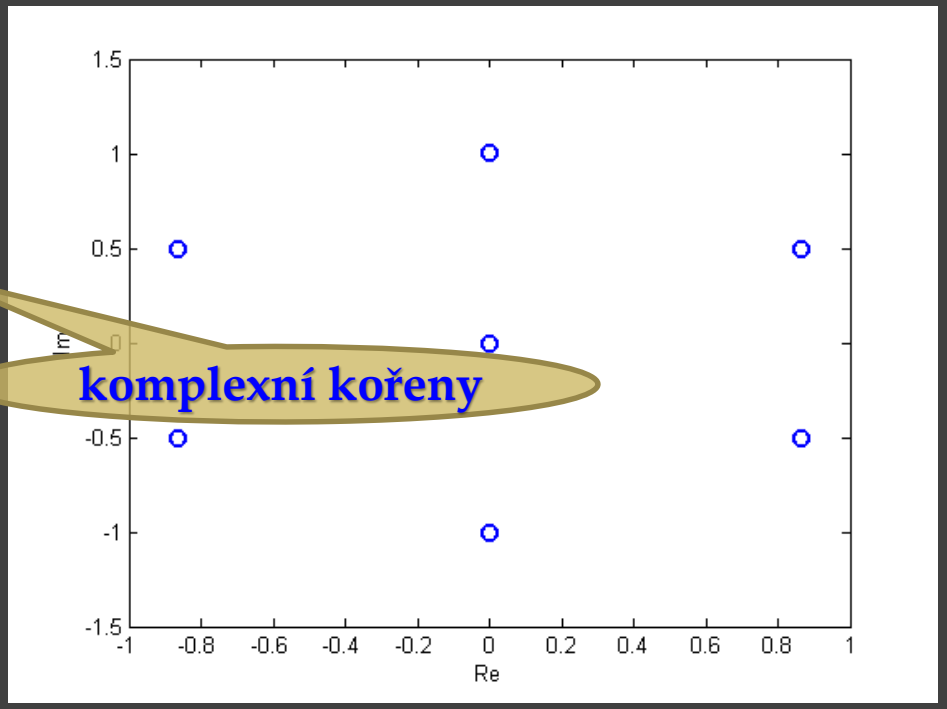
$$7x^7 = -7x$$

```
w = [7,0,0,0,0,0,7,0];
x = roots(w)
x =
     0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

reálný kořen

komplexní kořeny

```
plot(x,'o')
xlabel('Re')
ylabel('Im')
```

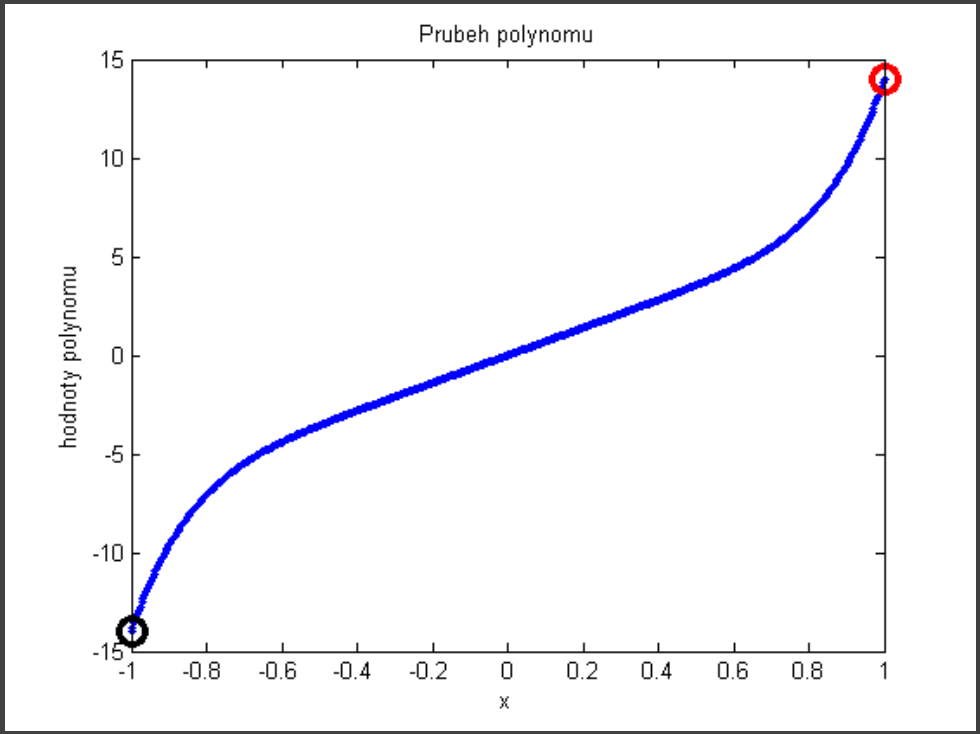


Pozn. Komplexní kořeny lze zobrazit jen v komplexní rovině (např. pomocí `plot()` na ose x reálná část, na ose y imaginární část).

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynom** pro polynom $w(x) = 7x^7 + 7x$ v rozsahu od **-1** do **1** s 500 prvky na ose x

```
w = [7,0,0,0,0,0,7,0];
x = kresliPolynom(w,-1,1,500)
```

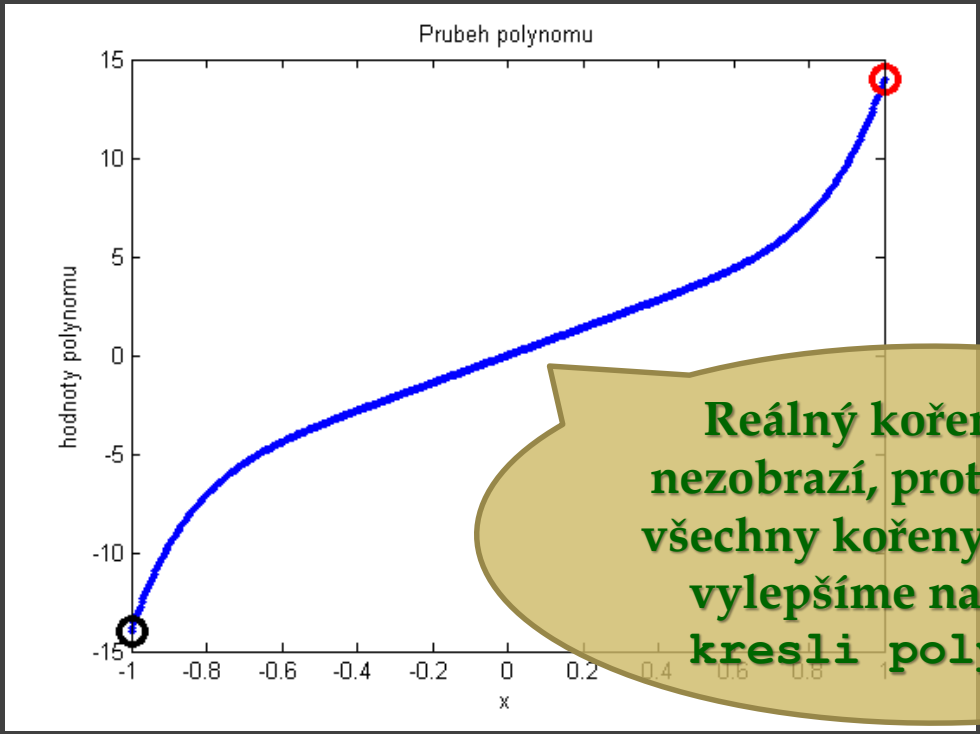



```
Max =
    14
indexMax =
   500
Min =
   -14
indexMin =
     1
koreny nejsou realne
x =
    0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **`kresliPolynom`** pro polynom $w(x) = 7x^7 + 7x$ v rozsahu od **-1** do **1** s 500 prvky na ose x

```
w = [7,0,0,0,0,0,7,0];
x = kresliPolynom(w,-1,1,500)
```




```
Max =
    14
indexMax =
   500
Min =
   -14
indexMin =
     1
koreny nejsou realne
x =
         0
   -0.8660 + 0.5000i
   -0.8660 - 0.5000i
    0.0000 + 1.0000i
    0.0000 - 1.0000i
    0.8660 + 0.5000i
    0.8660 - 0.5000i
```

Reálný kořen se také nezobrazí, protože nejsou všechny kořeny reálné, ale vylepšíme naši funkci **`kresli polynom()`**…

Funkce pro práci s polynomy

Příklad: Do již dříve vytvořené funkce `kresliPolynom()`, která vykreslí zadaný polynom p v zadaném rozsahu, najde a označí v grafu maximum a minimum v daném rozsahu, přidáme výpočet a zobrazení kořenů (do grafu lze vykreslit jen reálné kořeny).

Takže v uživatelské funkci `kresliPolynom()`, která byla uvedena dříve, nebudeme zjišťovat, zdali jsou všechny kořeny reálné, ale zjistíme to u každého zvlášť a do grafu vykreslíme právě reálné kořeny.

Funkce bude mít název `kresliPolynomSkoreny`.

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on




hold off
end
```

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on




hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, jež vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');




hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');




hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);




hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);
for cykl = 1:length(k)




end
hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);
for cykl = 1:length(k)
    if (isreal(k(cykl)))

    end
end
hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu');
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);
for cykl = 1:length(k)
    if (isreal(k(cykl)))

    end
end
hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací indexy prvků, které se rovnají maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

test, je-li *cykl*-tý kořen (*cykl*-tý prvek vektoru *k*) reálný

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu')
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);
for cykl = 1:length(k)
    if (isreal(k(cykl)))
        plot(k(cykl),0,'og');
    end
end
hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací index prvku, který se rovná maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

test, je-li *cykl*-tý kořen (*cykl*-tý prvek vektoru *k*) reálný

```
function k=kresliPolynomSkoreny(p,odkudX,kamX,prvkuX)
x = linspace(odkudX,kamX,prvkuX);
hodnotyPolynomu = polyval(p,x);
pMax = find(hodnotyPolynomu == max(hodnotyPolynomu));
pMin = find(hodnotyPolynomu == min(hodnotyPolynomu));
plot(x,hodnotyPolynomu,'.')
title('Prubeh polynomu');
xlabel('x')
ylabel('hodnoty polynomu')
hold on
plot(x(pMax),hodnotyPolynomu(pMax),'or');
plot(x(pMin),hodnotyPolynomu(pMin),'ok');
k = roots(p);
for cykl = 1:length(k)
    if (isreal(k(cykl)))
        plot(k(cykl),0,'og');
    end
end
hold off
end
```

Maximum a minimum zde nalezeno pomocí `find()`, která vrací index prvku, který se rovná maximu (příp. minimu) z vektoru `hodnotyPolynomu` (proměnné `pMax`, `pMin`).

přidání maxim a minim do grafu

výpočet kořenů

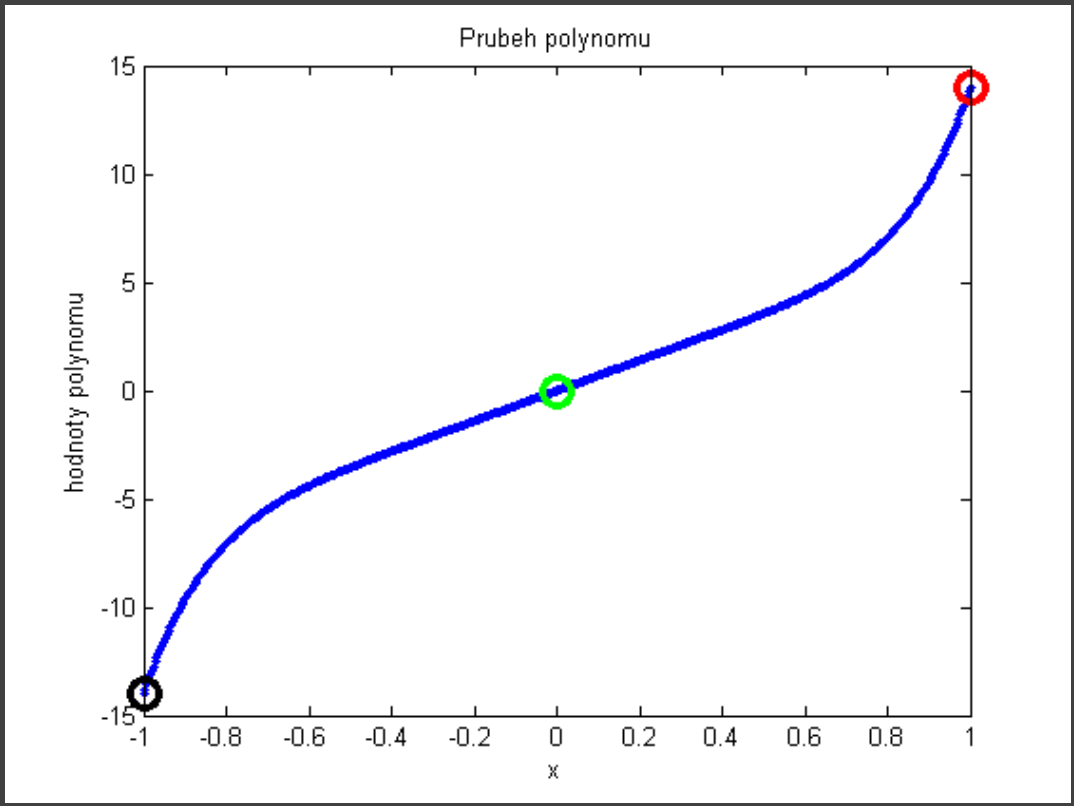
test, je-li *cykl*-tý kořen (*cykl*-tý prvek vektoru *k*) reálný

y-ová souřadnice kořenu polynomu je nula

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny**
pro polynom $w(x) = 7x^7 + 7x$
v rozsahu od **-1** do **1** s 500 prvky na ose x

```
w = [7,0,0,0,0,0,7,0];
x = kresliPolynomSkoreny (w,-1,1,500)
```

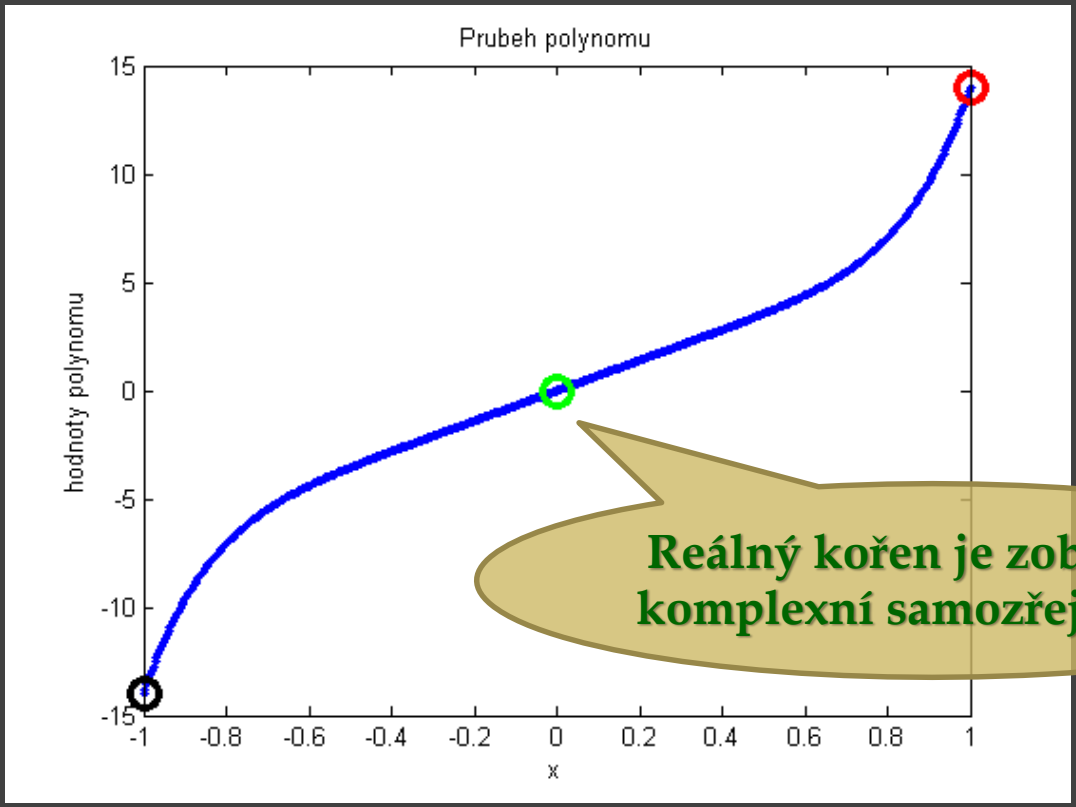



```
x =
   0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny**
pro polynom $w(x) = 7x^7 + 7x$
v rozsahu od **-1** do **1** s 500 prvky na ose x

```
w = [7,0,0,0,0,0,7,0];
x = kresliPolynomSkoreny (w,-1,1,500)
```



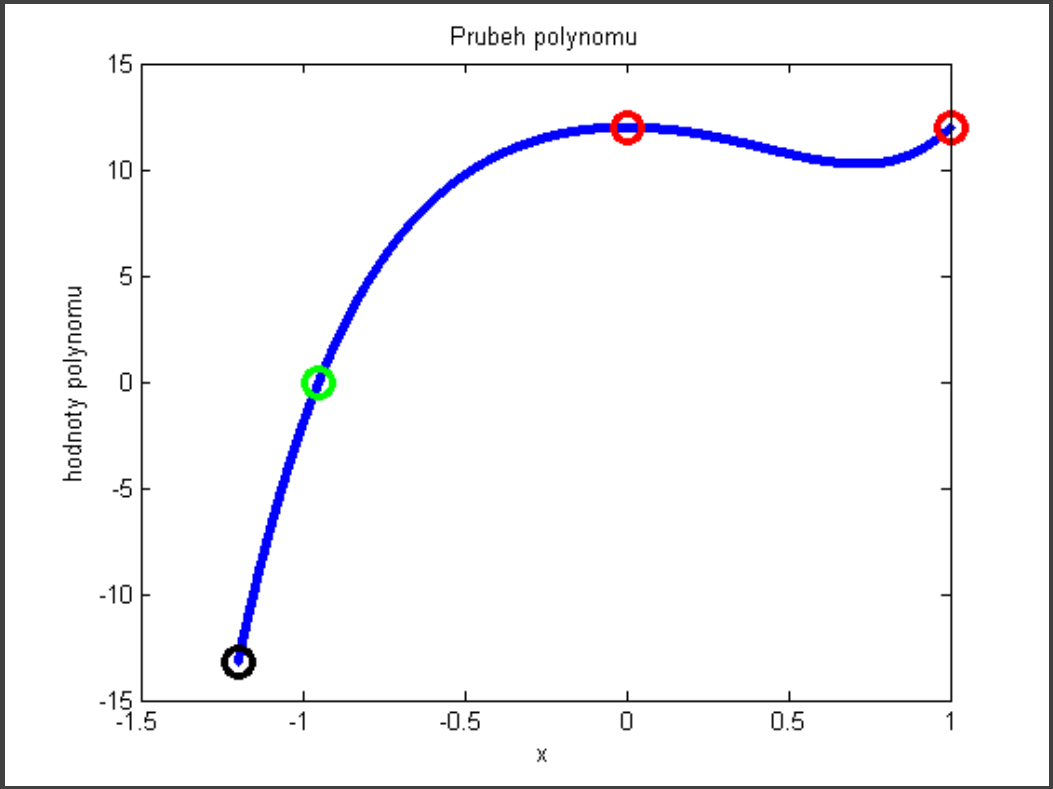

```
x =
   0
  -0.8660 + 0.5000i
  -0.8660 - 0.5000i
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.8660 + 0.5000i
   0.8660 - 0.5000i
```

Reálný kořen je zobrazen, komplexní samozřejmě ne.

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny**
pro polynom $m(x) = 4x^5 + 3x^3 - 7x^2 + 12$
v rozsahu od **-1,2** do **1** s 10000 prvky na ose x

```
m = [4,0,3,-7,0,12];
x = kresliPolynomSkoreny (m,-1.2,1,10000)
```

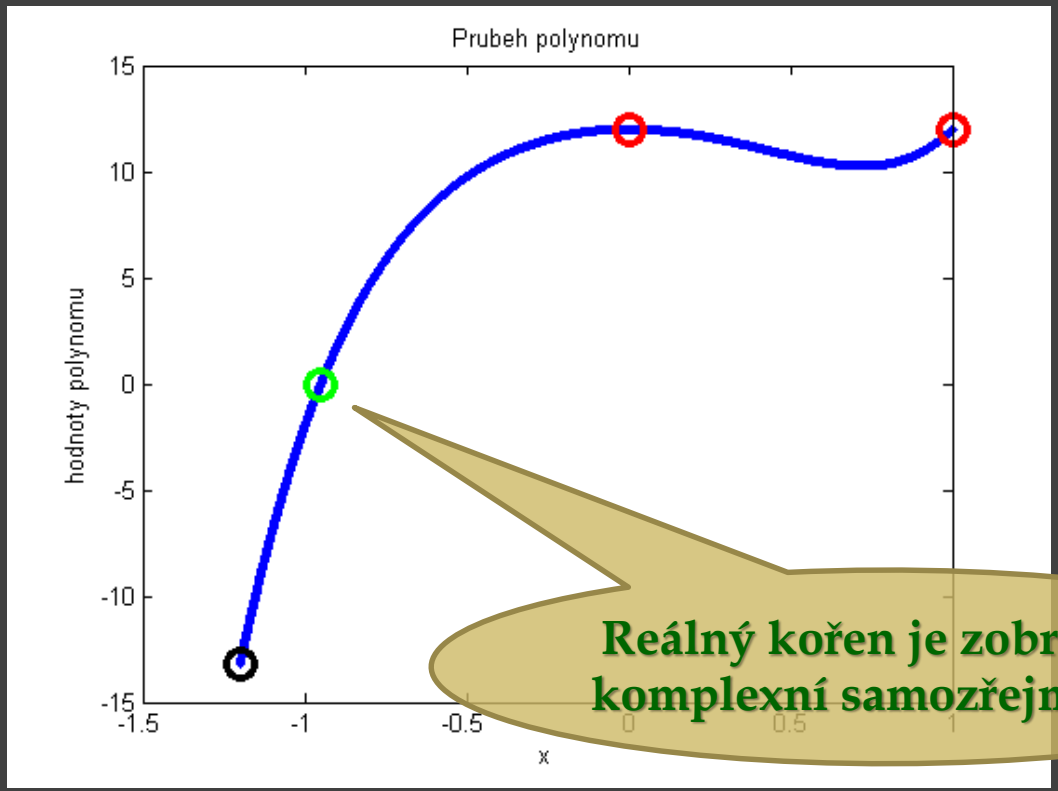



```
x =
   0.9916 + 0.5911i
   0.9916 - 0.5911i
  -0.5164 + 1.4499i
  -0.5164 - 1.4499i
  -0.9504
```

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny**
pro polynom $m(x) = 4x^5 + 3x^3 - 7x^2 + 12$
v rozsahu od **-1,2** do **1** s 10000 prvky na ose x

```
m = [4,0,3,-7,0,12];
x = kresliPolynomSkoreny (m,-1.2,1,10000)
```



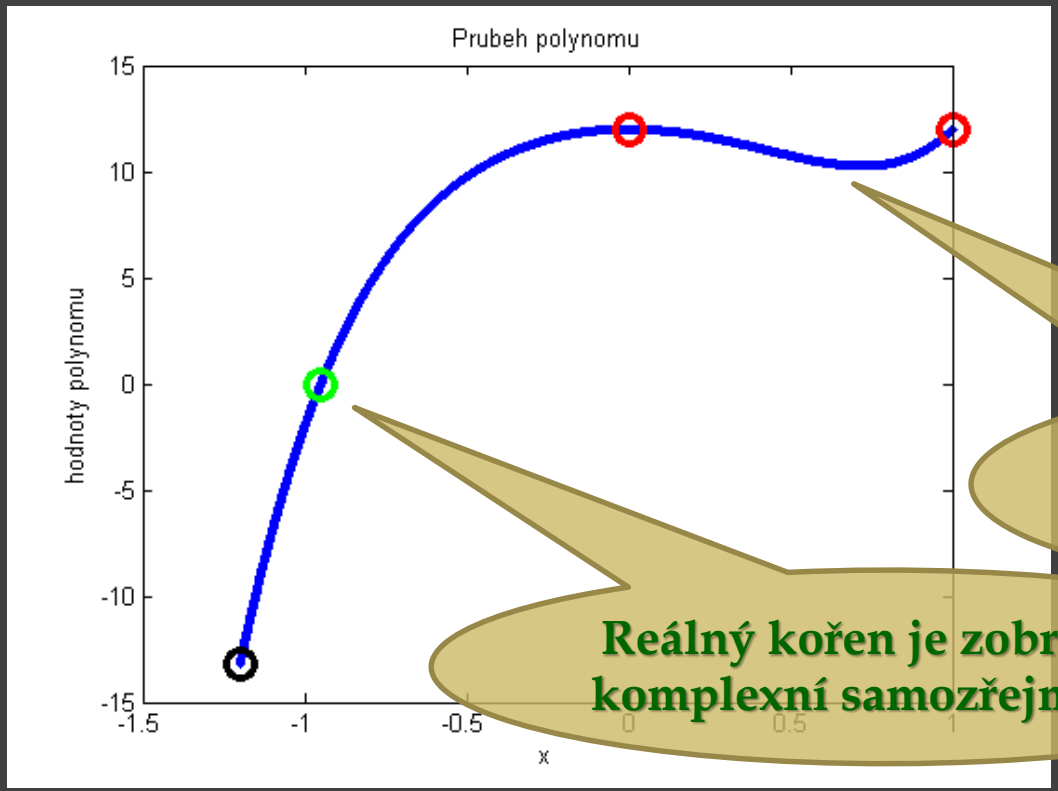

```
x =
   0.9916 + 0.5911i
   0.9916 - 0.5911i
  -0.5164 + 1.4499i
  -0.5164 - 1.4499i
  -0.9504
```

Reálný kořen je zobrazen, komplexní samozřejmě ne.

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **`kresliPolynomSkoreny`**
pro polynom $m(x) = 4x^5 + 3x^3 - 7x^2 + 12$
v rozsahu od **-1,2** do **1** s 10000 prvky na ose x

```
m = [4,0,3,-7,0,12];
x = kresliPolynomSkoreny (m,-1.2,1,10000)
```




```
x =
   0.9916 + 0.5911i
   0.9916 - 0.5911i
  -0.5164 + 1.4499i
  -0.5164 - 1.4499i
  -0.9504
```

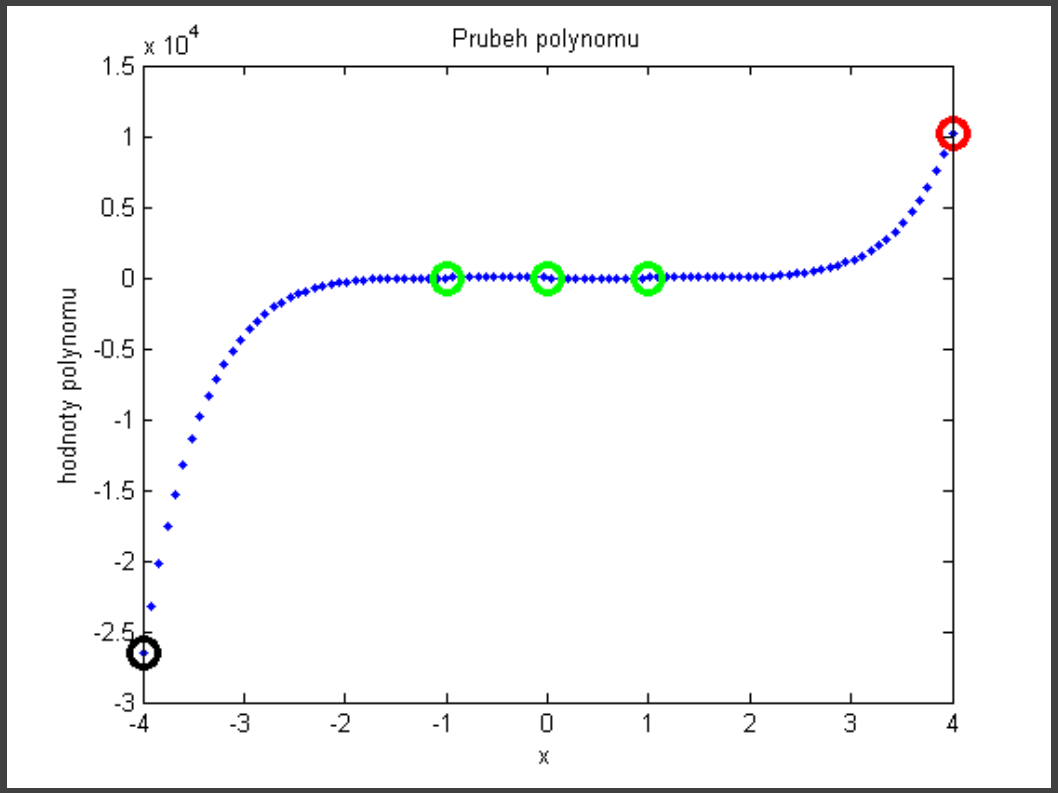
V daném rozsahu zobrazena dvě maxima nalezená pomocí `find()`.

Reálný kořen je zobrazen, komplexní samozřejmě ne.

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny** pro polynom $n(x) = x^7 - 2x^6 + 2x^5 - x^3 + 2x^2 - 2x$ v rozsahu od **-4** do **4** s 100 prvky na ose x

```
n = [1,-2,2,0,-1,2,-2,0];
x = kresliPolynomSkoreny(n,-4,4,100)
```

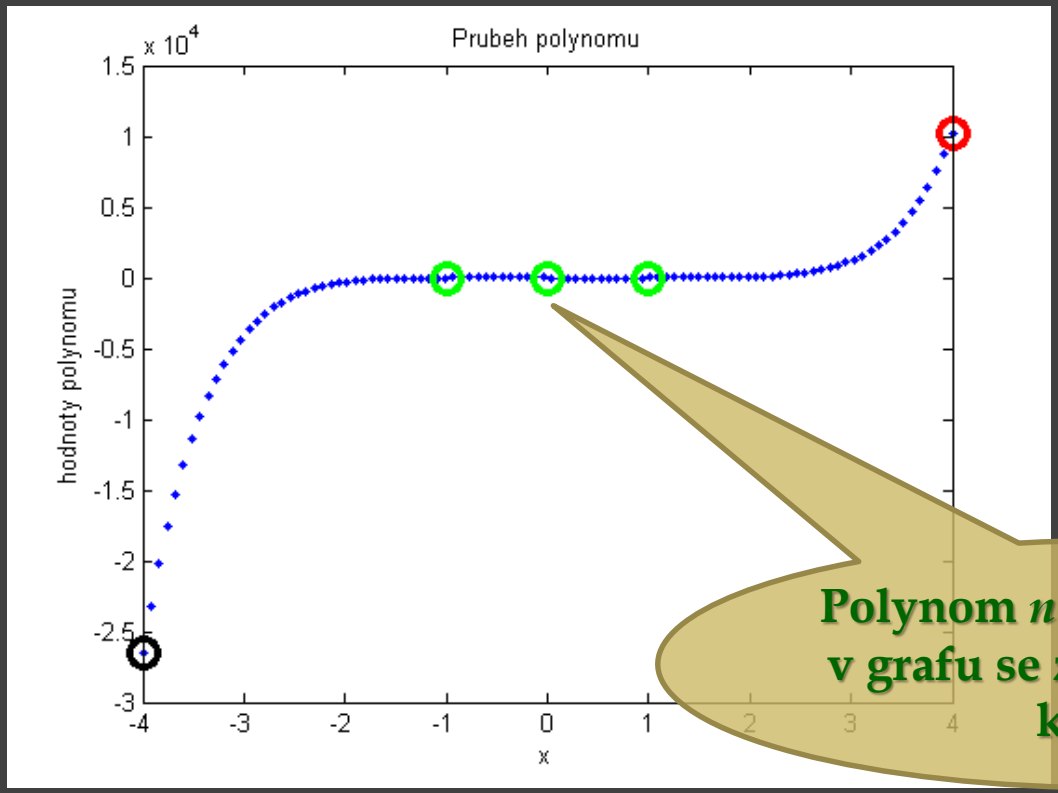



```
x =
     0
  -1.0000
   1.0000 + 1.0000i
   1.0000 - 1.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
   1.0000
```

Funkce pro práci s polynomy

Pokračování příkladu: volání funkce **kresliPolynomSkoreny** pro polynom $n(x) = x^7 - 2x^6 + 2x^5 - x^3 + 2x^2 - 2x$ v rozsahu od **-4** do **4** s 100 prvky na ose x

```
n = [1,-2,2,0,-1,2,-2,0];
x = kresliPolynomSkoreny(n,-4,4,100)
```




```
x =
   0
  -1.0000
   1.0000 + 1.0000i
   1.0000 - 1.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
   1.0000
```

Polynom $n(x)$ má 7 kořenů, v grafu se zobrazí 3 reálné kořeny

Polynomiální regrese

`polyfit(x, y, st)` – proloží množinu bodů o souřadnicích obsažených ve vektorech *x* a *y* polynomem stupně *st*. Výstupem jsou koeficienty polynomu zvoleného stupně *st*.

Polynomiální regrese

polyfit(x, y, st) – proloží množinu bodů o souřadnicích obsažených ve vektorech *x* a *y* polynomem stupně *st*. Výstupem jsou koeficienty polynomu zvoleného stupně *st*.

Příklad: Při měření byla získána data uvedená v tabulce. Provedeme polynomiální regresi - proložení (aproximaci) zadaných hodnot polynomem.

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];
plot(t,u,'o')
xlabel('t[s]')
ylabel('u[V]')
title('Data z mereni')
```

Body na příslušných souřadnicích zobrazeny jako kolečka.

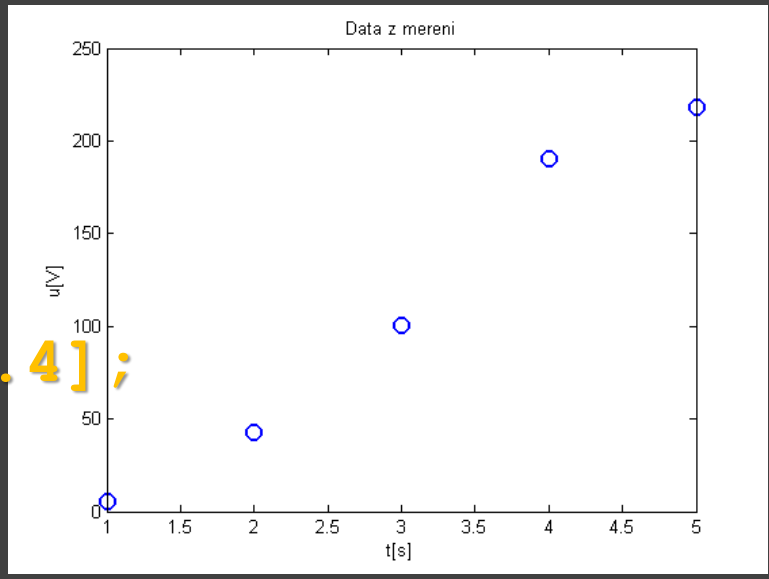
Polynomiální regrese

polyfit(x, y, st) – proloží množinu bodů o souřadnicích obsažených ve vektorech *x* a *y* polynomem stupně *st*. Výstupem jsou koeficienty polynomu zvoleného stupně *st*.

Příklad: Při měření byla získána data uvedená v tabulce. Provedeme polynomiální regresi - proložení (aproximaci) zadaných hodnot polynomem.

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];
plot(t,u,'o')
xlabel('t[s]')
ylabel('u[V]')
title('Data z mereni')
```



Body na příslušných souřadnicích zobrazeny jako kolečka.

Polynomiální regrese

Pokračování příkladu: Proložíme naměřené hodnoty polynomem 3. řádu a vykreslíme jeho průběh společně s naměřenými hodnotami do grafu. Nejprve vypočteme regresní polynom:

```
regr = polyfit(t,u,3)
regr =
   -6.8583   62.6964 -110.3452   61.5800
```

tj.:

$$\mathrm{regr}(x) = -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x^1 + 61.5800\,x^0$$
$$= -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x + 61.5800$$

Polynomiální regrese

Pokračování příkladu: Proložíme naměřené hodnoty **polynomem 3. řádu** a vykreslíme jeho průběh společně s naměřenými hodnotami do grafu. Nejprve vypočteme regresní polynom:

```
regr = polyfit(t,u,3)
regr =
   -6.8583   62.6964 -110.3452   61.5800
```

stupeň polynomu

naměřená data

koeficienty polynomu

tj.:

$$\text{regr}(x) = -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x^1 + 61.5800\,x^0$$
$$= -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x + 61.5800$$

Polynomiální regrese

Pokračování příkladu: Proložíme naměřené hodnoty **polynomem 3. řádu** a vykreslíme jeho průběh společně s naměřenými hodnotami do grafu. Nejprve vypočteme regresní polynom:

```
regr = polyfit(t,u,3)
regr =
   -6.8583   62.6964 -110.3452   61.5800
```

stupeň polynomu

naměřená data

koeficienty polynomu

tj.:

$\mathrm{regr}(x) = -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x^1 + 61.5800\,x^0$

$= -6.8583\,x^3 + 62.6964\,x^2 - 110.3452\,x + 61.5800$

Regresní křivku (zjištěný polynom) přidáme do grafu s daty. Pro zobrazení polynomu musíme zjemnit x-ovou osu (na ní zobrazeno `t = [1:5]` – krok 1), vytvoříme vektor **`rt`**:

```
rt = [0:0.05:6];
```

Vektor **`rt`** je z intervalu od **0** do **6** => dali jsme „kousek okolo **`t`** “.

Pak vypočteme (vyčíslíme) **h**odnoty **p**olynomu pro všechna **`rt`**:

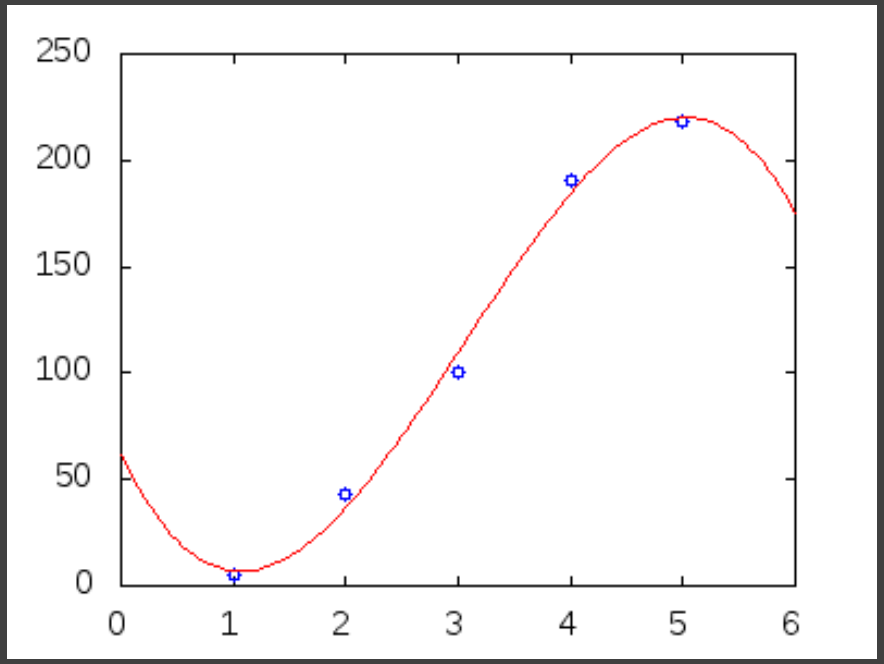
```
hp = polyval(regr,rt);
```

Polynomiální regrese

Pokračování příkladu:

```
plot(t, u,'o',rt, hp, 'r')
```

Prokládané **body** (zobrazené jako **kolečka**) i **křivka** (**polynom**) jsou v jednom grafu, polynom zobrazen červeně.



Křivka (**polynom 3. stupně**) není úplně v pořádku, zvláště na krajích intervalu.

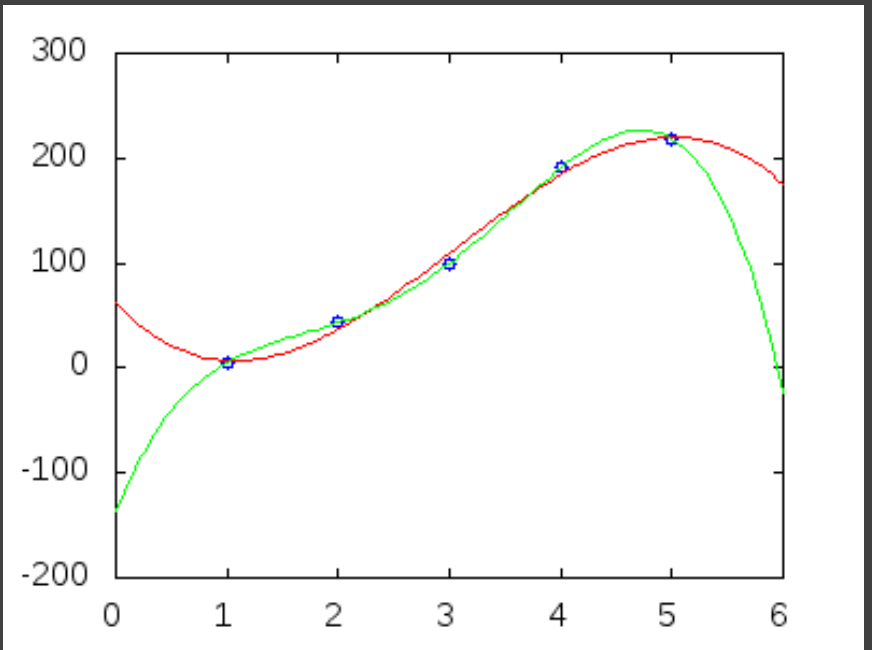
Zkusíme proložit naměřené hodnoty **polynomem 4. stupně** a vykreslíme jeho průběh společně s naměřenými hodnotami a polynomem 3. stupně do grafu.

Polynomiální regrese

Pokračování příkladu:

```
regr_moc = polyfit(t,u,4)
regr_moc =
  -4.5875   48.1917 -164.7125  263.2083 -136.6000
hp_moc = polyval(regr_moc,rt);
hold on
plot(rt, hp_moc, 'g')
hold off
```

$$\text{regr\_moc}(x) = -4.5875\,x^4 + 48.1917\,x^3 - 164.7125\,x^2 + 263.2083\,x - 136.6$$



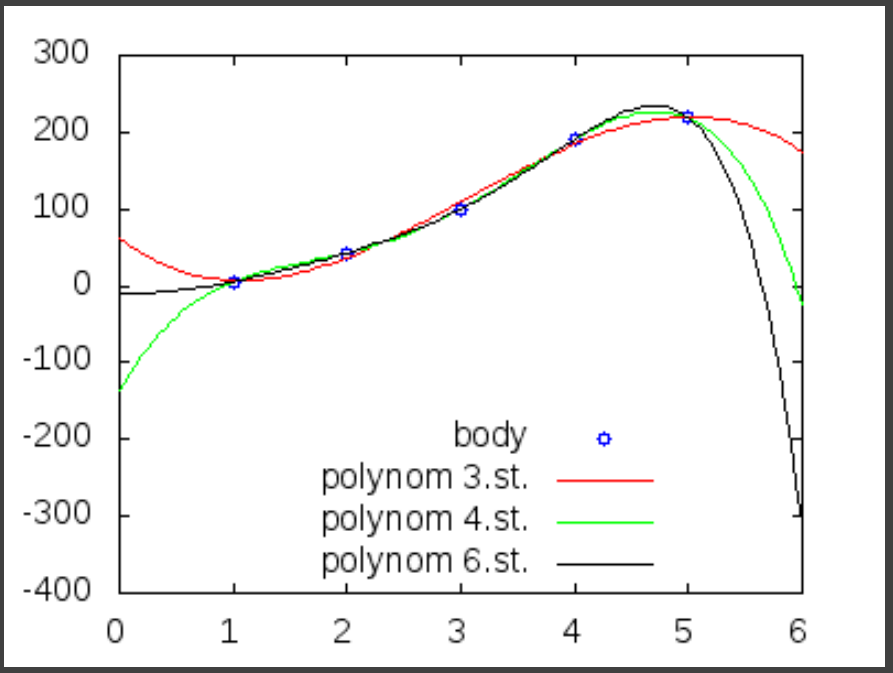
Prokládané **body** (**kolečka**) i křivky (polynomy) jsou v jednom grafu, **polynom 3. st.** zobrazen červeně, **polynom 4. st.** zobrazen zeleně.

Dostali jsme nevhodnou regresní **křivku** (zelená), nesmyslnou. Ani ta červená **křivka** není úplně v pořádku … Zkusíme tedy ještě polynom ještě vyššího stupně.

Polynomiální regrese

Pokračování příkladu:

```
regr_jeste_vic = polyfit(t,u,6);
hp_jeste_vic = polyval(regr_jeste_vic,rt);
plot(t,u,'o', rt,hp,'r', rt,hp_moc,'g',...
rt,hp_jeste_vic,'k')
legend('body', 'polynom 3.st.', 'polynom 4.st.',...
'polynom 6.st.', 'Location','South')
```

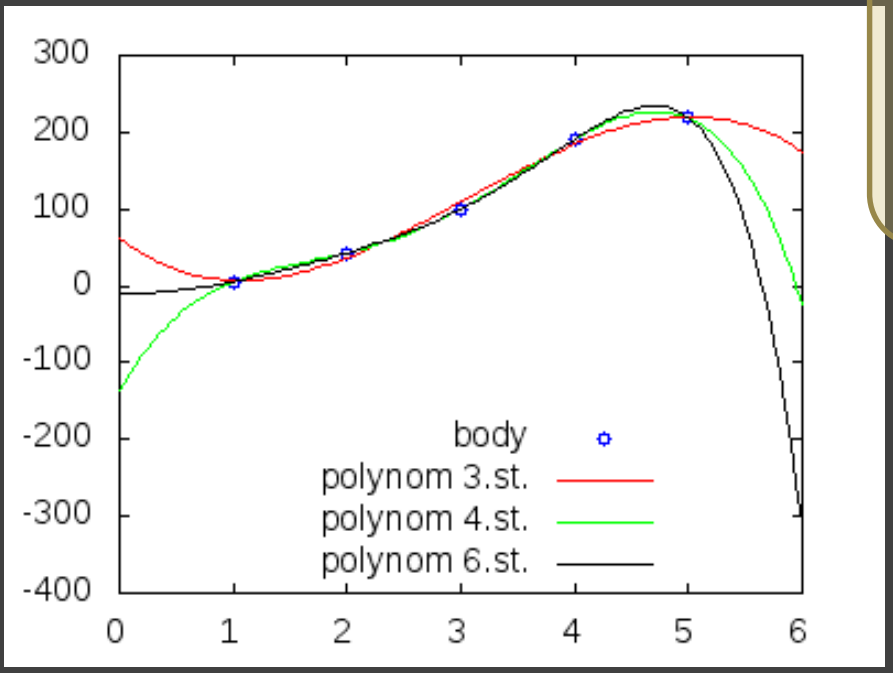


Polynom 6. stupně je zobrazen **černě**, je to ještě horší než proložení polynomy **3. st.** a **4. st.**, proto zkusíme proložit naměřené hodnoty polynomem **2. stupně**.

Polynomiální regrese

Pokračování příkladu:

```
regr_jeste_vic = polyfit(t,u,6);
hp_jeste_vic = polyval(regr_jeste_vic,rt);
plot(t,u,'o', rt,hp,'r', rt,hp_moc,'g',...
rt,hp_jeste_vic,'k')
legend('body', 'polynom 3.st.', 'polynom 4.st.',...
'polynom 6.st.', 'Location','South')
```



`legend(...,'Location','řetězec')` přidává legendu v určené poloze vzhledem k osám, `řetězec` může být např.:, `'North'`, `'NorthEast'` (defaultně) ,`'South'`, `'East'`,`'West'`,`'NorthWest'` apod.

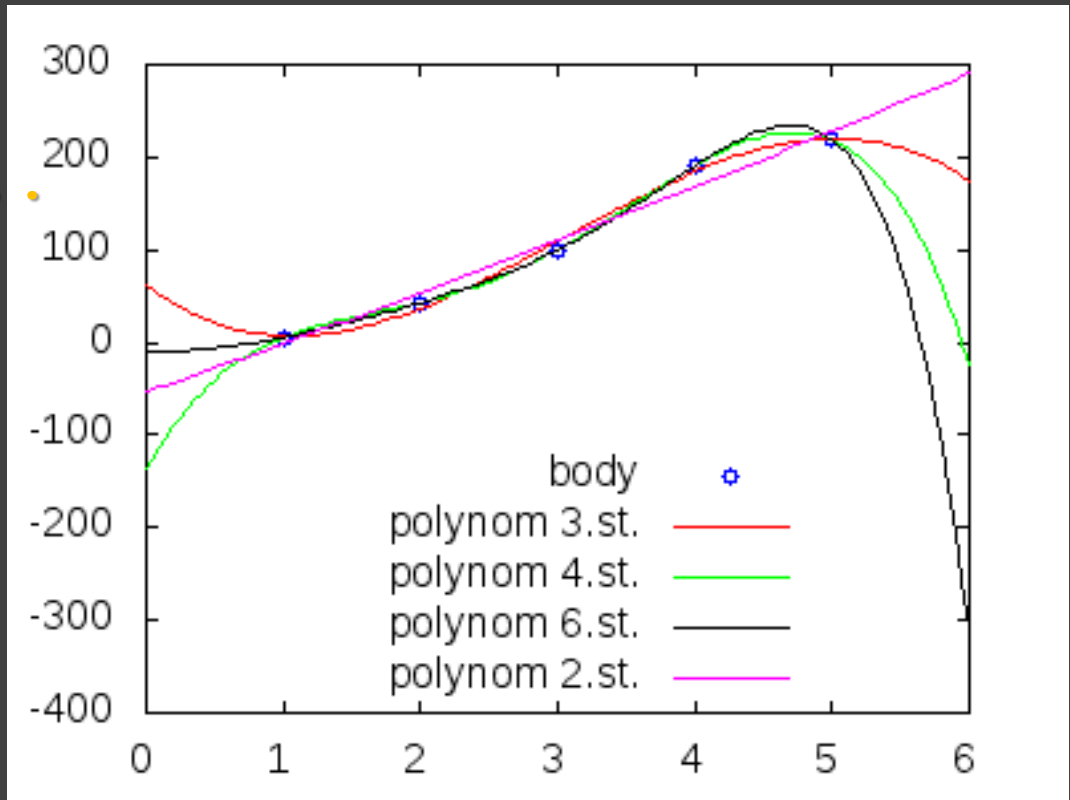
Polynom 6. stupně je zobrazen **černě**, je to ještě horší než proložení polynomy **3. st.** a **4. st.**, proto zkusíme proložit naměřené hodnoty polynomem **2. stupně**.

Polynomiální regrese

Pokračování příkladu:

```
regr_mene = polyfit(t,u,2)
regr_mene =
          0.9714    51.5114   -53.6400
hp_mene = polyval(regr_mene,rt);
plot(t, u,'o',...
rt,hp,'r',...
rt,hp_moc,'g',...
rt,hp_jeste_vic,'k',...
rt,hp_mene, 'm')
legend('body', ...
'polynom 3.st.', ...
'polynom 4.st.', ...
'polynom 6.st.', ...
'polynom 2.st.', ...
'Location','South')
```

$\text{regr\_mene}(x) = 0.9714\,x^2 + 51.5114\,x - 53.64$



Polynomiální regrese

Pokračování příkladu: Nejlépe by prokládaným datům vyhověl **polynom 2. stupně**. Zkusíme ještě přímku (**polynom 1. stupně**):

```
regr_min = polyfit(t,u,1)
regr_min =
          57.3400  -60.4400
hp_min = polyval(regr_min,rt);
plot(t,u,'o', rt,hp_mene,'m', rt,hp_min,'c')
legend('body','polynom 2.st.','polynom 1.st.',...
'Location','SouthEast')
```

$\text{regr\_min}(x) = 57.34\,x - 60.44$

Polynom **1. stupně** (přímka) a polynom **2. stupně** (parabola) na intervalu od **0** do **6** mají velmi podobný průběh a jsou **vhodné** pro proložení naměřených dat uvedených v tabulce.