

Seminář 3

Snížení barevné hloubky

Náhodné rozptylování

Náhodné rozptylování zpracování prvek po prvku

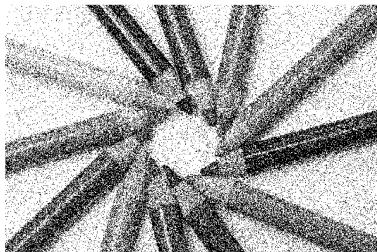
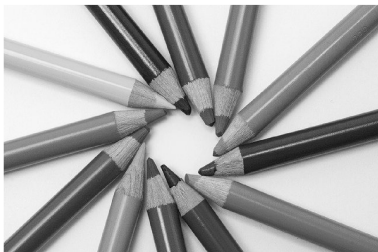
```
f = rgb2gray(imread('pastelky.png'));
f2 = imread("skala.png");

f_max = max(max(f));

g = uint8(zeros(size(f,1),size(f,2)));

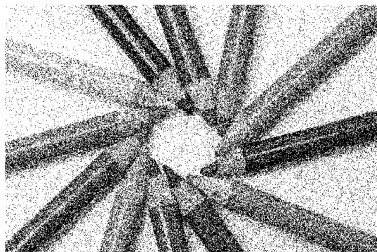
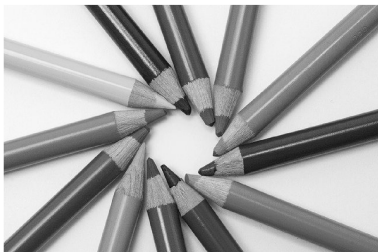
for i = 1 : size(f,1)
    for j = 1 : size(f,2)
        % randi vrati nahodne cislo
        r = randi(f_max);
        if f(i,j) > r
            g(i,j) = g(i,j) +1;
        end
    end
end

figure
subplot(1,2,1)
imshow(f,[])
subplot(1,2,2)
imshow(g,[]);
```



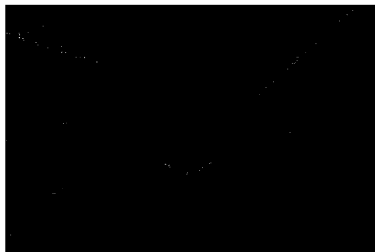
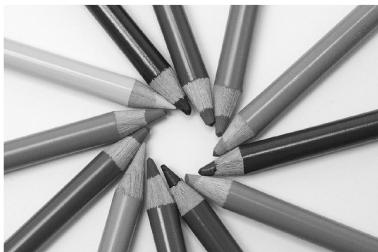
Zpracování pomocí maticových operací

```
[m,n] = size(f);  
f_max = max(max(f));  
g = uint8(zeros(size(f,1),size(f,2)));  
  
g = g + uint8(f >= randi(f_max,[m,n]));  
figure  
subplot(1,2,1)  
imshow(f,[]);  
subplot(1,2,2)  
imshow(g,[]);
```



Špatný přístup - porovnávají se všechny pixely se stejnou hodnotou.

```
[m,n] = size(f);  
f_max = max(max(f));  
g = uint8(zeros(size(f,1),size(f,2)));  
  
g = (f >= randi(f_max));  
  
figure  
subplot(1,2,1)  
imshow(f,[]);  
subplot(1,2,2)  
imshow(g,[]);
```



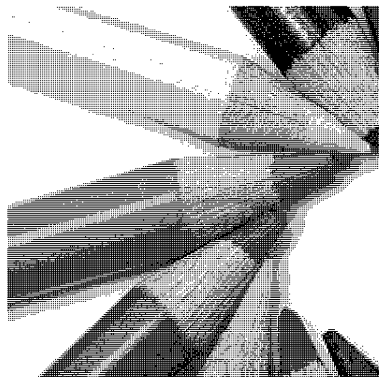
Maticové rozptylování

Zvětšení velikosti obrazu

```
[ g ] = matrix_dithering( f );
```

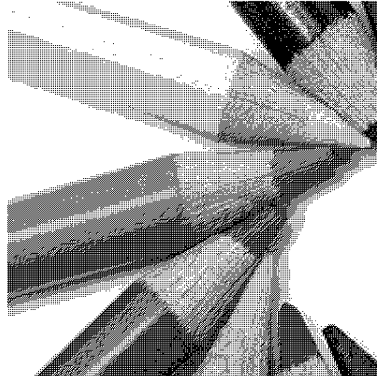
```
patern = 2x2  
204    153  
102     51
```

```
figure  
subplot(1,2,1)  
imshow(f(100:300,100:300),[]);  
subplot(1,2,2)  
imshow(g(200:600,200:600),[]);
```

Nevhodně zvolené matice

```
[ g ] = matrix_dithering2( f );  
  
figure  
subplot(1,2,1)  
imshow(f(100:300,100:300),[]);  
subplot(1,2,2)  
imshow(g(200:600,200:600),[]);
```

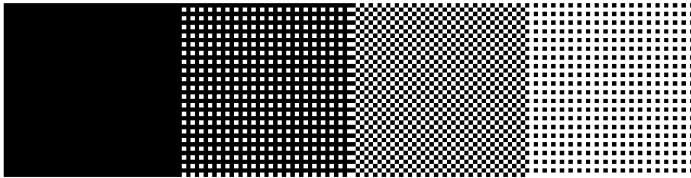
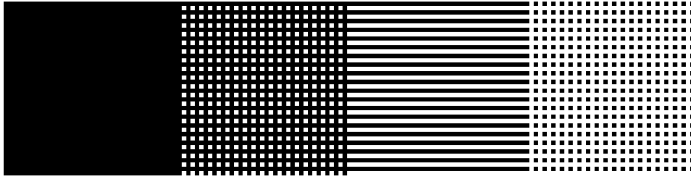


```
[ g2a ] = matrix_dithering( f2 );
```

```
patern = 2x2  
    204    153  
    102     51
```

```
[ g2b ] = matrix_dithering2( f2 );
```

```
figure,  
subplot(2,1,1), imshow(g2a);  
subplot(2,1,2), imshow(g2b);
```

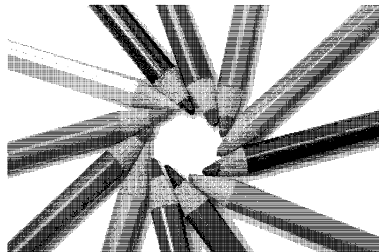
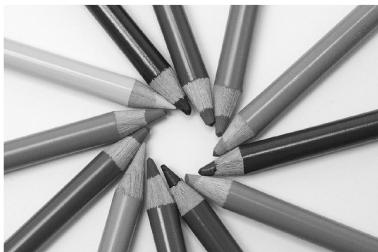


Maticové rozptylování se zachováním velikosti

```
[ g ] = matrix_dithering3( f );
```

```
patern = 2x2  
204    153  
102     51
```

```
figure  
subplot(1,2,1)  
imshow(f,[]);  
subplot(1,2,2)  
imshow(g,[]);
```

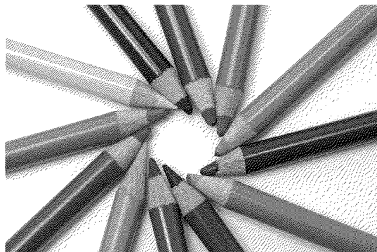
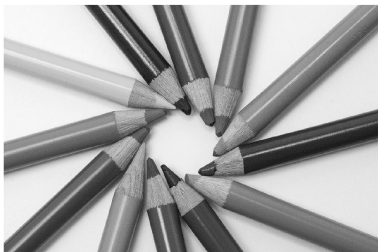


Rozptylování s distribucí chyby

Floyd Steinberg

```
g = floyd_steinberg( f );
```

```
figure  
subplot(1,2,1)  
imshow(f,[]);  
subplot(1,2,2)  
imshow(g,[]);
```



Barevné obrázky

Náhodné rozptylování

```
f = imread('pastelky.png');
[m,n,o] = size(f);
f_red = f(:,:,1);
f_green = f(:,:,2);
f_blue = f(:,:,3);

% červená složka
f_max_r = max(max(f_red));
g_red = uint8(zeros(m,n));
g_red = g_red + uint8(f_red >= randi(f_max_r,[m,n]));

% zelená složka
f_max_g = max(max(f_green));
g_green = uint8(zeros(m,n));
g_green = g_green + uint8(f_green >= randi(f_max_g,[m,n]));

% modrá složka
f_max_b = max(max(f_blue));
g_blue = uint8(zeros(m,n));
g_blue = g_blue + uint8(f_blue >= randi(f_max_b,[m,n]));

g=[];
```

```
g(:,:,1) = g_red;
g(:,:,2) = g_green;
g(:,:,3) = g_blue;
```

```
figure
subplot(1,2,1)
imshow(f,[]);
subplot(1,2,2)
imshow(g,[]);
```



Maticové rozptylování

```
f_red = f(:,:,1);
f_green = f(:,:,2);
f_blue = f(:,:,3);

g_red = matrix_dithering3( f_red );
```

```
patern = 2x2
    204    153
    102     51
```

```
g_green = matrix_dithering3( f_green );
```

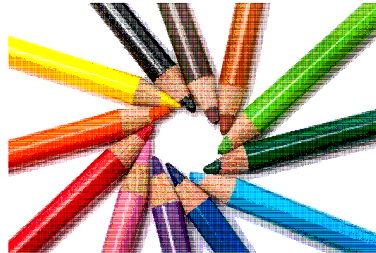
```
patern = 2x2
    204    153
    102     51
```

```
g_blue = matrix_dithering3( f_blue );
```

```
patern = 2x2  
204 153  
102 51
```

```
g=[];  
g(:,:,1) = g_red;  
g(:,:,2) = g_green;  
g(:,:,3) = g_blue;
```

```
figure  
subplot(1,2,1)  
imshow(f,[]);  
subplot(1,2,2)  
imshow(g,[]);
```



S distribucí chyby

```
f_red = f(:,:,1);  
f_green = f(:,:,2);  
f_blue = f(:,:,3);
```

```
g_red = floyd_steinberg( f_red );  
g_green = floyd_steinberg( f_green );  
g_blue = floyd_steinberg( f_blue );
```

```
g=[];
g(:,:,1) = g_red;
g(:,:,2) = g_green;
g(:,:,3) = g_blue;
```

```
figure
subplot(1,2,1)
imshow(f,[]);
subplot(1,2,2)
imshow(g,[]);
```



Barevné palety

Porovnání univerzální a adaptivní palety.

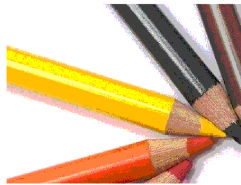
rgb2ind()

```
f = imread('pastelky2.png');
tol = 1/4;
[x1,map1] = rgb2ind(f, tol, 'nodither');
[x2,map2] = rgb2ind(f, 54, 'nodither');

subplot(1,3,1), imshow(f);
```




```
subplot(1,3,2), imshow(x1,map1);  
subplot(1,3,3), imshow(x2,map2);
```



```
subplot(2,1,1), imshow((cat(3, map1(:,1)',map1(:,2)', map1(:,3)')));
title("Univerzalni");
subplot(2,1,2), imshow((cat(3, map2(:,1)',map2(:,2)', map2(:,3)')));
title("Adaptivni");
```



Všechny barvy

```
f = imread('allcolor.png');
tol = 1/3;
[x1,map3] = rgb2ind(f, tol, 'nodither');
[x2,map4] = rgb2ind(f, 64, 'nodither');

subplot(1,3,1), imshow(f);
subplot(1,3,2), imshow(x1,map3);
subplot(1,3,3), imshow(x2,map4);
```



```
subplot(2,1,1), imshow((cat(3, map3(:,1)',map3(:,2)', map3(:,3)')));  
title("Univerzalni");  
subplot(2,1,2), imshow((cat(3, map4(:,1)',map4(:,2)', map4(:,3)')));  
title("Adaptivni");
```

Univerzalni

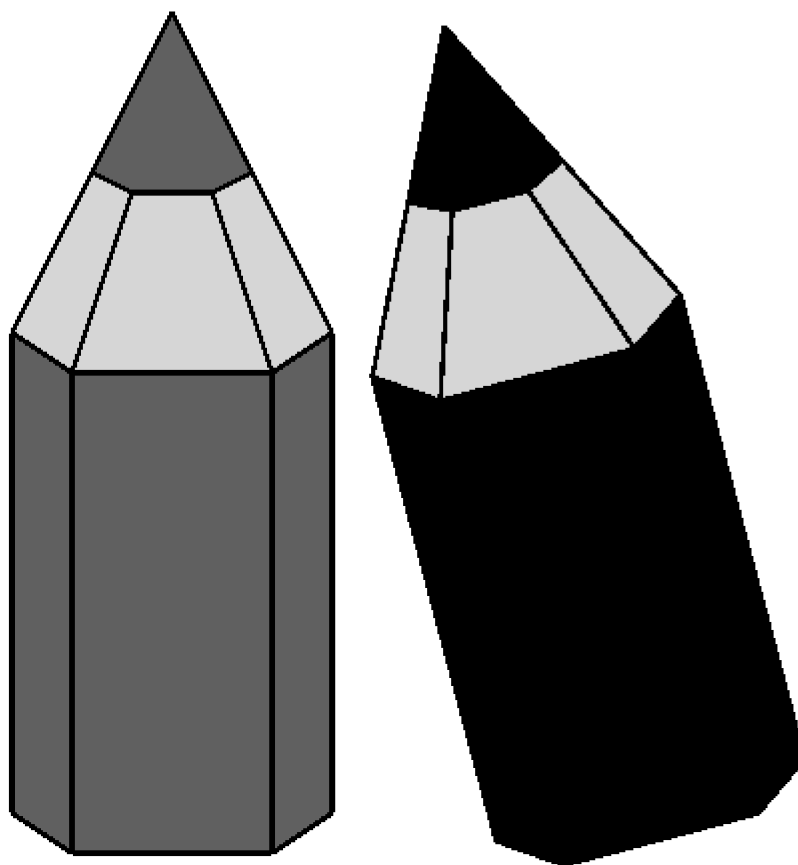


Adaptivni



Redundance - kódování

```
I1 = rgb2gray(imread('red_kodovani.png'));  
figure, imshow(I1);
```



```
[M,N] = size(I1);
```

Počet barev v obraze

```
barvy = unique(I1)
```

```
barvy = 4x1 uint8 column vector
    0
   96
  214
  255
```

Pravděpodobnost každé barvy (histogram)

```
n = imhist(I1);
p = n/(M*N);
pravdepodobnost = p(double(barvy)+1)
```

```
pravdepodobnost = 4x1
    0.2555
    0.2029
    0.1098
```

Kódování - 8 kód

počet bitů použitých ke kódování

```
l1 = 8*ones(256,1);

l1_avg = sum(l1.*p);
display(l1_avg);
```

```
l1_avg = 8
```

Kódování nestejně dlouhými kódy

- 0 : 01
- 96 : 000
- 214 : 010
- 255 : 1

```
l2 = zeros(256,1);
l2(1)=2;
l2(97) = 3;
l2(215) = 3;
l2(256) = 1;

l2_avg = sum(l2.*p);
display(l2_avg);
```

```
l2_avg = 1.8810
```

Komprese a relativní redundance

```
b1 = M*N*l1_avg;
b2 = M*N*l2_avg;

% komprese
C = b1/b2;
display(C);
```

```
C = 4.2530
```

```
% relativni redundance
R = 1-(1/C);
display(R);
```

```
R = 0.7649
```

Huffmanovo kódování

Huffmanovo kódování na obrázku

- 0 : a
- 96 : b
- 214 : c
- 255 : d

```
I = imread('red_kodovani.png');
[m, n] = size(I);
vstup_pom = I(:);
vstup(vstup_pom==0) = 'a';
vstup(vstup_pom==96) = 'b';
vstup(vstup_pom==214) = 'c';
vstup(vstup_pom==255) = 'd';

A = 'abcd';
Acell = cellstr(A)';
p_A = arrayfun(@(x)sum(vstup==x), A)/numel(vstup);
n = size(A,2)
```

n = 4

```
% kodova abeceda
B = {'0' '1'}
```

```
B = 1x2 cell
'0'      '1'
```

```
m = size(B,2)
```

m = 2

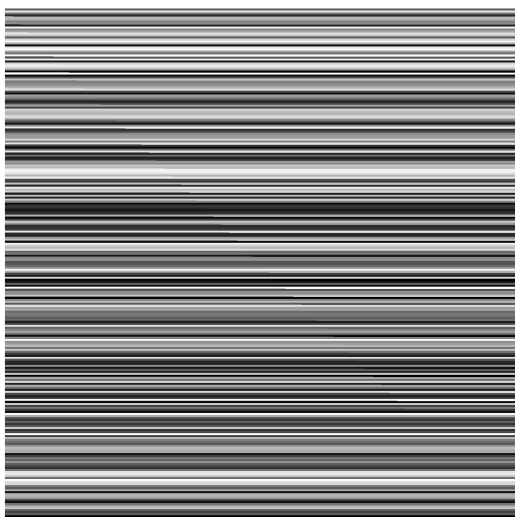
```
kody = Huffman(Acell, B, p_A, true);
table(A',kody,'VariableNames',{'Znak', 'Kod'})
```

```
ans = 4x2 table
```

	Znak	Kod
1	a	'11'
2	b	'100'
3	c	'101'
4	d	1x1 cell

Prostorová redundance

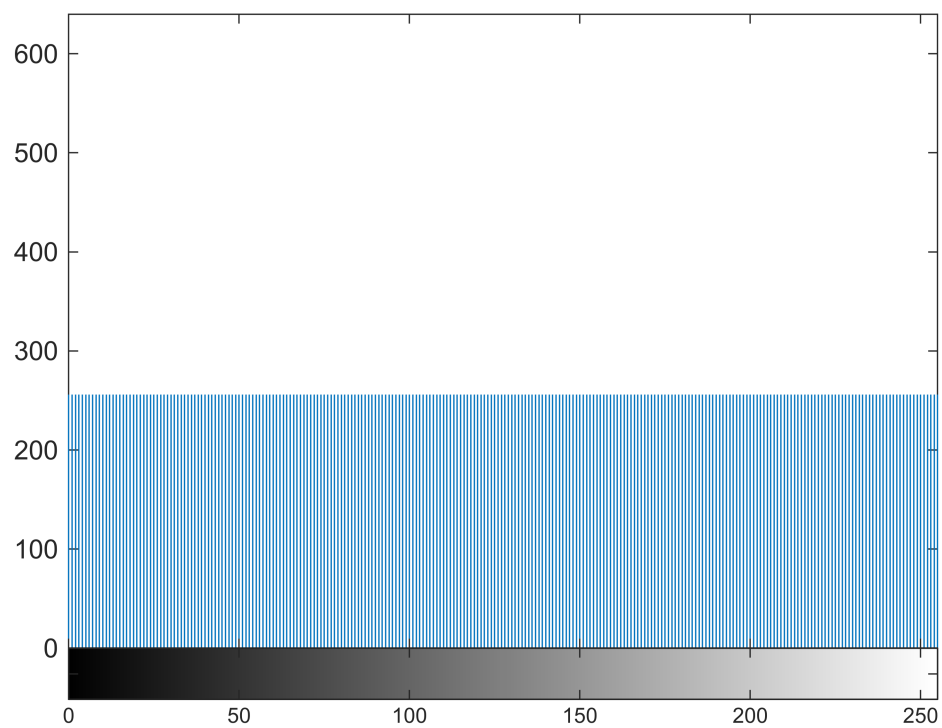
```
I2 = imread('red_prostor.png');
figure, imshow(I2);
```



```
[M,N] = size(I2);
```

Histogram

```
figure, imhist(I2);
```



Pravděpodobnost každé barvy

```
n = imhist(I2);  
p = n/(M*N)
```

```
p = 256×1  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
0.0039  
⋮
```

RLE

Podívejte se na funkce RLEenc a RLEdec na konci.

```
vstup = 'aaaabbcccaabb'
```

```
vstup =  
'aaaabbcccaabb'
```

```
% kodovani  
enc = RLEenc(vstup)
```

```
enc =  
'4a2b3c2a2b'
```

```
dec = RLEdec(enc)
```

```
dec =  
'aaaabbcccaabb'
```

Je momožné, aby bylo toto kódování neefektivní?

RLE aplikované na obrázek

```
I = imread('prikladLena.png');  
[m, n] = size(I);  
vstup_pom = I(:);  
vstup(vstup_pom) = 'w';  
vstup(vstup_pom==0) = 'b';
```

Délka vstupu. Jedná se o binární obrázek, každý pixel je možné kódovat 1 bitem.

```
delka_vstup = m*n
```

```
delka_vstup = 262144
```

Délka kódu. Každé číslo je kódováno jedním bytem a znak jedním bitem.

```
[enc delka_kod] = RLEenc(vstup);  
delka_kod
```

```
delka_kod = 80370
```

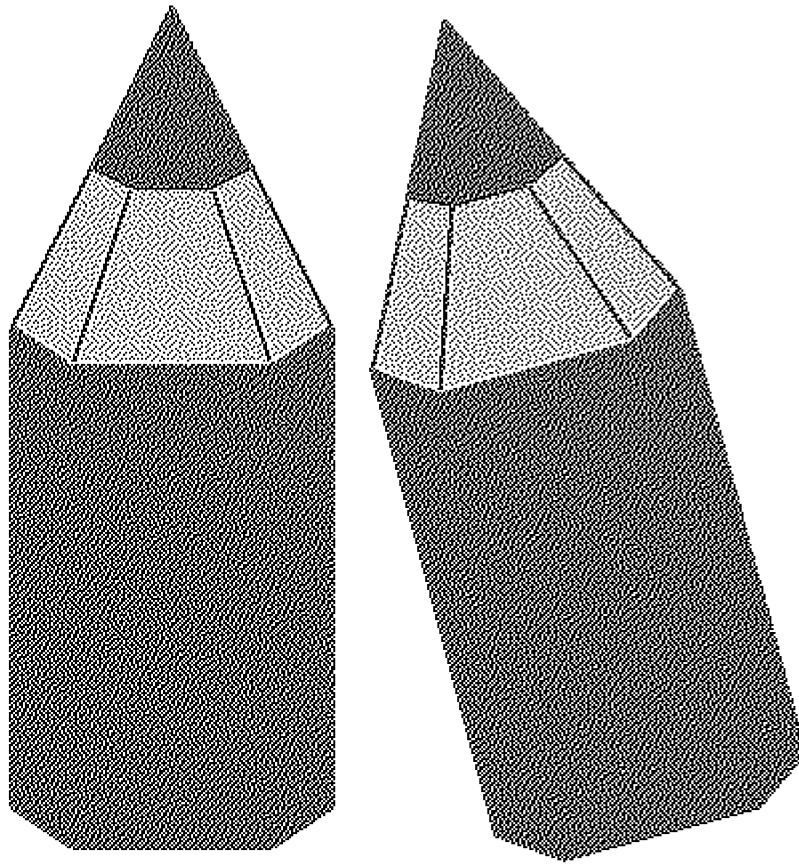
Nerelevantní informace

```
I3 = imread('red_info.png');  
  
figure, imshow(I3);
```



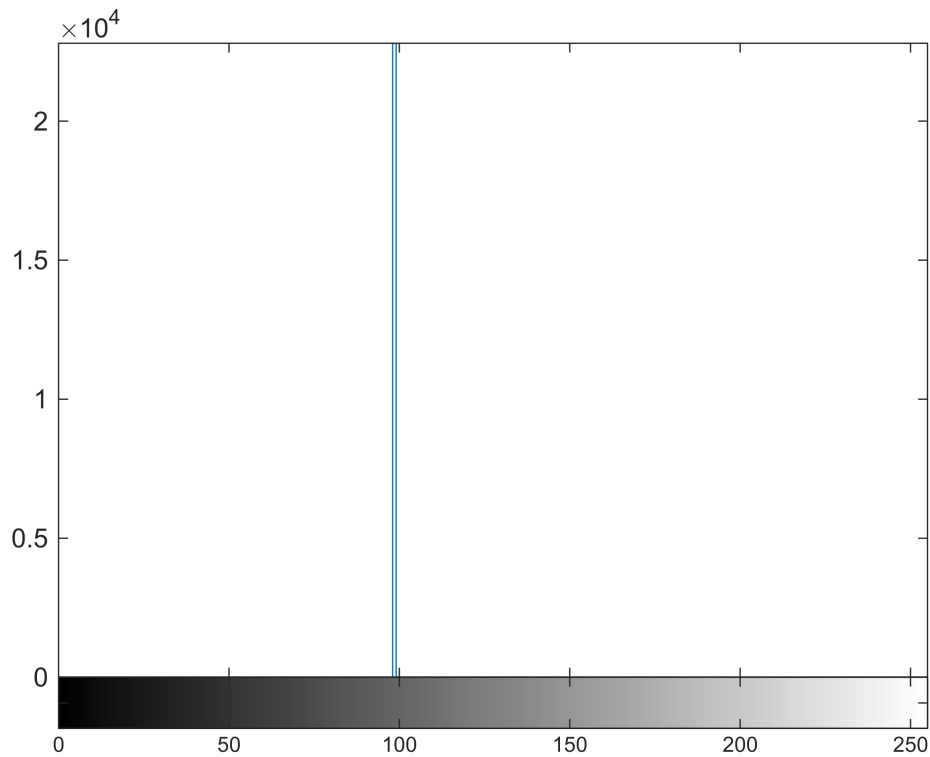
Obrázek s roztaženým kontrastem

```
figure, imshow(I3,[]);
```



Histogram

```
figure, imhist(I3,256);
```



JPEG komprese

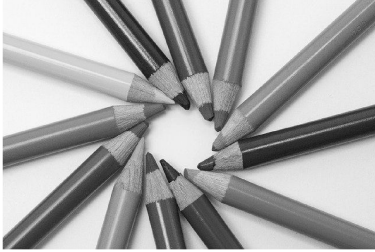
```
I = rgb2gray(imread('pastelky.png'));

imwrite(I,'p100.jpg','Quality', 100);
imwrite(I,'p80.jpg','Quality', 80);
imwrite(I,'p50.jpg','Quality', 50);
imwrite(I,'p10.jpg','Quality', 10);
imwrite(I,'p5.jpg','Quality', 5);

I1 = imread('p100.jpg');
I2 = imread('p80.jpg');
I3 = imread('p50.jpg');
I4 = imread('p10.jpg');
I5 = imread('p5.jpg');

figure,
subplot(2,2,1), imshow(I2);
title("80");
subplot(2,2,2), imshow(I3);
title("50");
subplot(2,2,3), imshow(I4);
title("10");
subplot(2,2,4), imshow(I5);
title("5");
```

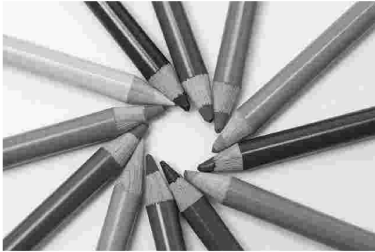
80



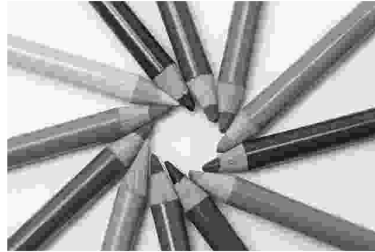
50



10



5



```
p100 = imfinfo('p100.jpg');
fprintf("Obrazek p100 ma velikost %i byte", p100.FileSize);
```

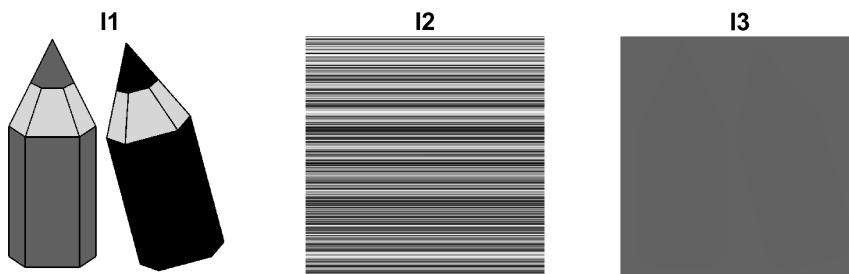
Obrazek p100 ma velikost 136317 byte

```
p5 = imfinfo('p5.jpg');
fprintf("Obrazek p5 ma velikost %i byte", p5.FileSize);
```

Obrazek p5 ma velikost 6123 byte

Entropie

```
I1 = imread('red_kodovani.png');
figure,
subplot(1,3,1), imshow(I1);
title("I1");
I2 = imread('red_prostor.png');
subplot(1,3,2), imshow(I2);
title("I2");
I3 = imread('red_info.png');
subplot(1,3,3), imshow(I3);
title("I3");
```



```
J1 = entropy(I1);
J2 = entropy(I2);
J3 = entropy(I3);
```

```
display(J1);
```

```
J1 = 1.8431
```

```
display(J2);
```

```
J2 = 8
```

```
display(J3);
```

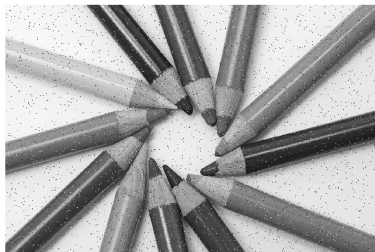
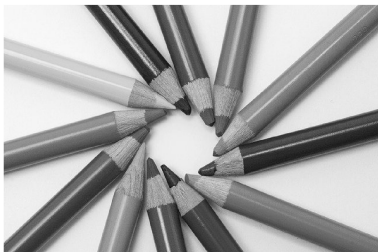
```
J3 = 0.8989
```

Měření kvality komprese

Mean-squared error

```
I = rgb2gray(imread('pastelky.png'));
I_noise = imnoise(I, 'salt & pepper', 0.02);
```

```
figure,
subplot(1,2,1), imshow(I);
subplot(1,2,2), imshow(I_noise);
```



```
% immse = Mean-Squared Error.
```

```
mse = immse(I, I_noise);
```

```
display(mse);
```

```
mse = 438.4469
```

```
% root mean-squared error
```

```
rmse = sqrt(immse(I, I_noise));
```

```
display(rmse);
```

```
rmse = 20.9391
```

Subjektivní hodnocení

```
I1 = rgb2gray(imread('kriteria1.png'));
```

```
I2 = rgb2gray(imread('kriteria2.png'));
```

```
Warning: PNG library warning:  
iCCP: cHRM chunk does not match sRGB
```

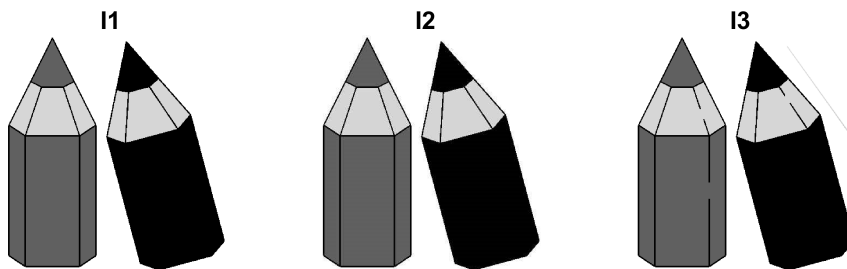
```
I3 = rgb2gray(imread('kriteria3.png'));
```

```
figure,  
subplot(1,3,1), imshow(I1);
```

```

title('I1');
subplot(1,3,2), imshow(I2);
title('I2');
subplot(1,3,3), imshow(I3);
title('I3');

```



```

mse1 = immse(I1, I2);
fprintf("Obrazek I2 ma mse rovnu %i", mse1);

```

Obrazek I2 ma mse rovnu 3.180263e+01

```

mse2 = immse(I1, I3);
fprintf("Obrazek I3 ma mse rovnu %i", mse2);

```

Obrazek I3 ma mse rovnu 2.721793e+01

Přímky (úsečky)

Rasterizace přímky - DDA algoritmus

DDA algoritmus

1. Z koncových bodů $[x_1, y_1]$ a $[x_2, y_2]$ urči směrnici m .
2. Inicializuj bod $[x, y]$ hodnotou $[x_1, y_1]$.
3. Dokud je $x \leq x_2$ opakuj:

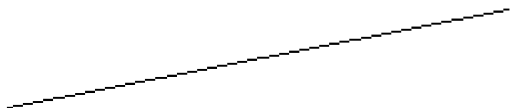
- Vykresli bod $[x, \text{zaokrouhlené}(y)]$

- $x = x + 1$

- $y = y + m$

Tento algoritmus je jen pro $0 \leq m \leq 1$ pro ostatní m je potřeba upravit. pro $|m| > 1$ se úsečka přimyká k y a tak se y zvětšuje o 1 a x o $1/m$

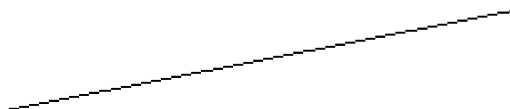
```
usecka = DDA([-100,1], [150,50],[60,300]);  
figure, imshow(usecka);
```



Bresenhamův algoritmus

bresenhamuv_algoritmus() -- pro názornost jen pro přímky přichylující se k ose x .

```
usecka = bresenhamuv_algoritmus([-100,1], [150,50],[60,300]);  
figure, imshow(usecka);
```



Bresenhamův algoritmus - přerušovaná čára

ba_prerusovana() -- pro názornost jen pro přímky přichylující se k ose x .

úseky jsou vždy stejné délky ve smyslu počtu pixelů

```
usecka = ba_prerusovana([-100,1], [100,1],[300,300], 10); % poslední argument =  
delka useku  
usecka2 = ba_prerusovana([-100,-100], [100,99],[300,300], 10); % poslední  
argument = delka useku  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```

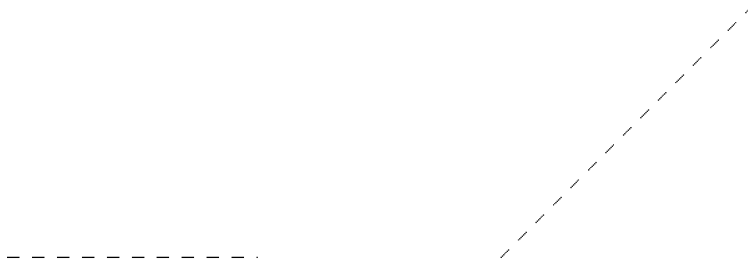


Bresenhamův algoritmus - přerušovaná čára

`ba_prerusovana2()` -- pro názornost jen pro přímky přichylující se k ose x.

úseky jsou vždy stejné délky

```
usecka = ba_prerusovana2([-100,1], [100,1],[300,300], 10); % posledni argument =  
delka useku  
usecka2 = ba_prerusovana2([-100,-100], [100,99],[300,300], 10); % posledni  
argument = delka useku  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```



Bresenhamův algoritmus - silná čára

`ba_silna()` -- pro názornost jen pro přímky přichylující se k ose x.

tloušťka čáry je dána počtem pixelů

```
usecka = ba_silna([-100,1], [100,1],[300,300], 10); % posledni argument = tloustka
usecka2 = ba_silna([-100,-100], [100,99],[300,300], 10); % posledni argument =
tloustka

figure,
subplot(1,2,1), imshow(usecka);
subplot(1,2,2), imshow(usecka2);
```



Bresenhamův algoritmus - silná čára

`ba_silna2()` -- pro názornost jen pro přímky přichylující se k ose x.

tloušťka čáry je dána šířkou

```
usecka = ba_silna2([-100,1], [100,1],[300,300], 10); % posledni argument =  
tloustka  
usecka2 = ba_silna2([-100,-100], [100,99],[300,300], 10); % posledni argument =  
tloustka  
  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```



DDA antialias

vyhlazení úsečky

```
usecka = DDA([-100,1], [100,40],[60,205]);  
usecka2 = DDAalias([-100,1], [100,40],[60,205]);  
figure,  
subplot(1,2,1), imshow(usecka);  
subplot(1,2,2), imshow(usecka2);
```

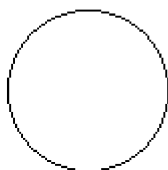


Kružnice

Bresenhamův algoritmus

`ba_kruznice()`

```
kruznice = ba_kruznice(40, [50,50], [100,100]);  
figure, imshow(kruznice);
```



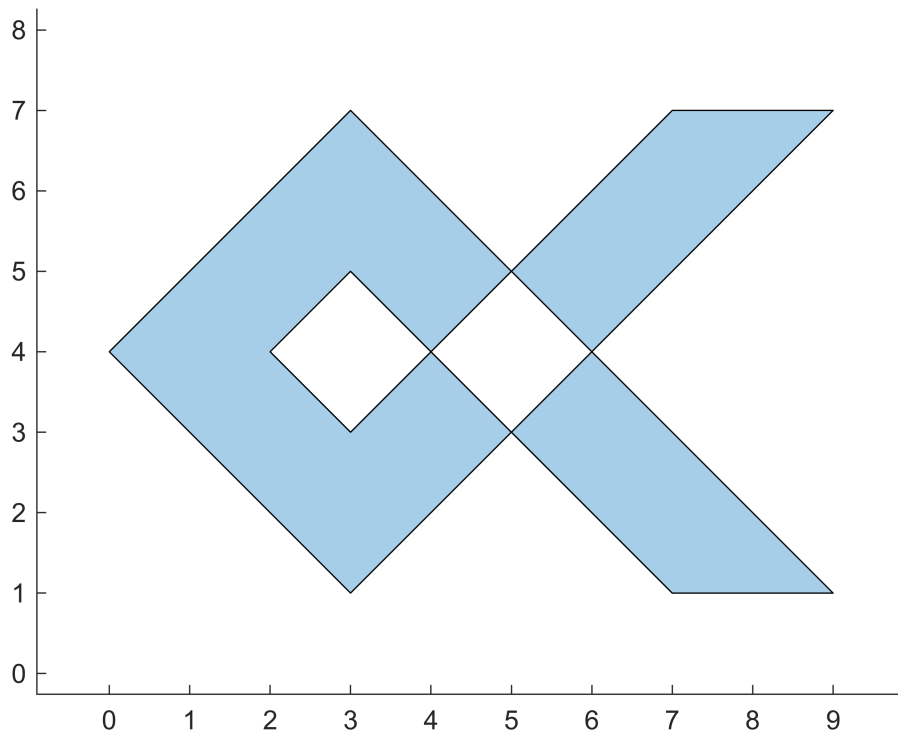
Geometricky zadaná hranice

`polyshape()`

```

X = [0 3 9 7 3 2 3 7 9 3 0];
Y = [4 1 7 7 3 4 5 1 1 7 4];
pgon = polyshape(X, Y, 'Simplify', false);
figure, plot(pgon)
axis equal

```



Polygon tvořený více hranicemi

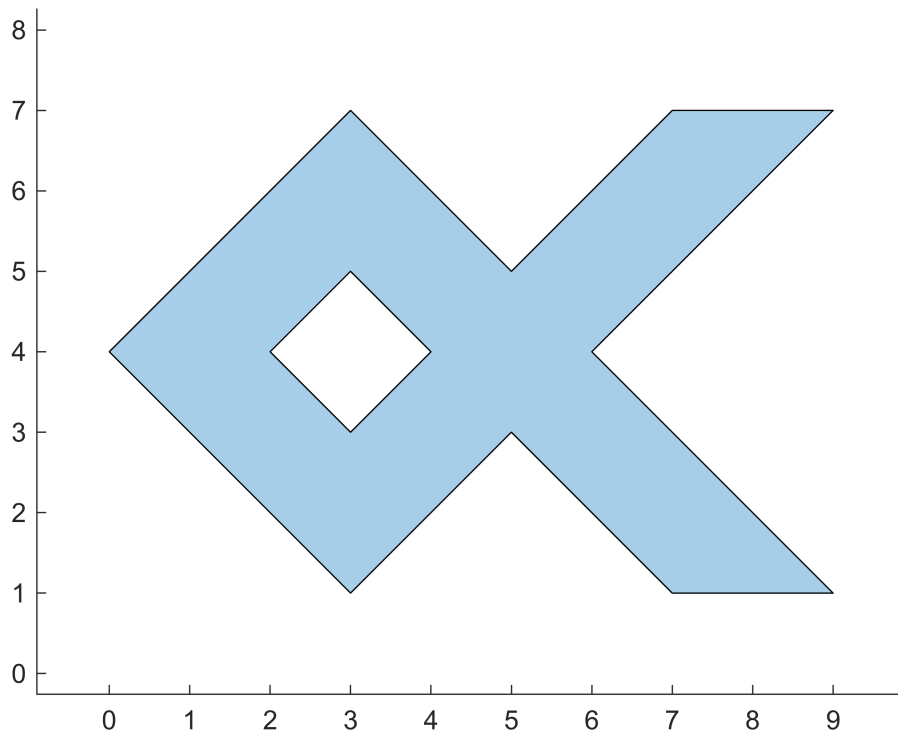
```

% vnější hranice
X1 = [0 3 5 7 9 6 9 7 5 3 0];
Y1 = [4 1 3 1 1 4 7 7 5 7 4];

% díra
X2 = [2 3 4 3 2];
Y2 = [4 3 4 5 4];

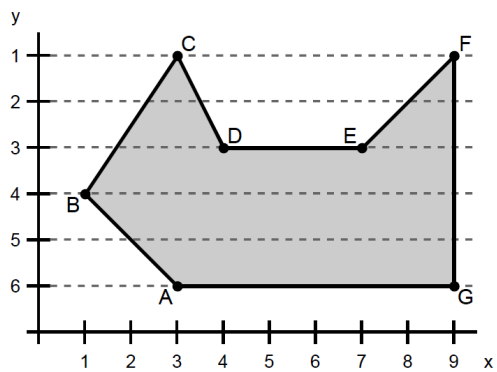
pgon = polyshape({X1 X2}, {Y1, Y2});
figure, plot(pgon)
axis equal

```



Řádkové vyplňování

Příklad ze slide 3



radkove_vyplnovani()

```
img = zeros(70,100);
body = [60,30; 40,10; 10,40; 30,40; 30,70; 10,90; 60,90;60,30];

radkove_vyplnovani(body,img);
```

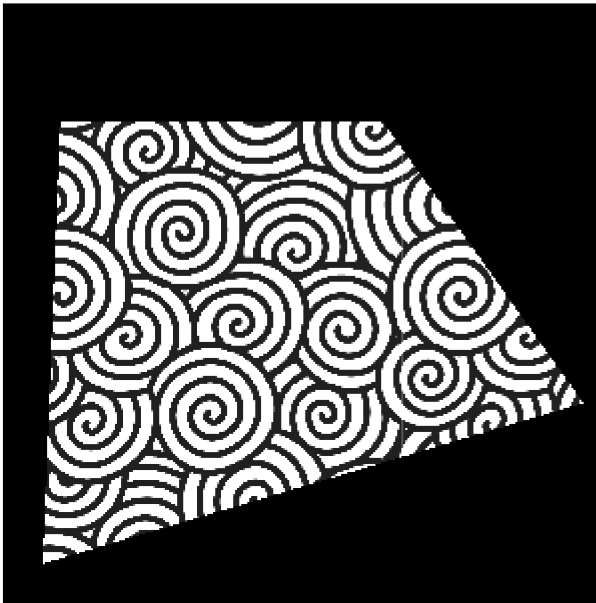



Vyplnění oblasti obrázkem

radkove_vyplnovani_vzor()

```
vzor = rgb2gray(imread("pattern_small.png"));
img = uint8(zeros(300,300));
body = [50,30; 280,21; 200,290; 60,190];

img = radkove_vyplnovani_vzor(body,img, vzor);
figure, imshow(img);
```



Šrafování

vodorovné šrafy - vynecháváme některé řádky

rv_srafy1()

```
img = zeros(70,100);
body = [60,30; 20,50; 20,90; 60,90];
```

```
%body = [60,30; 40,10; 10,40; 30,40; 30,70; 10,90; 60,90;60,30];
```

```
img = rv_srafy1(body,img);  
figure, imshow(img);
```



Šrafování

svislé šrafy - několik pixelů kresíme čáru, několik ne

rv_srafy2()

```
img = zeros(70,100);  
body = [60,30; 20,50; 20,90; 60,90];
```

```
img = rv_srafy2(body,img);  
imshow(img);
```



Šrafování

svislé šrafy správně

rv_srafy3()

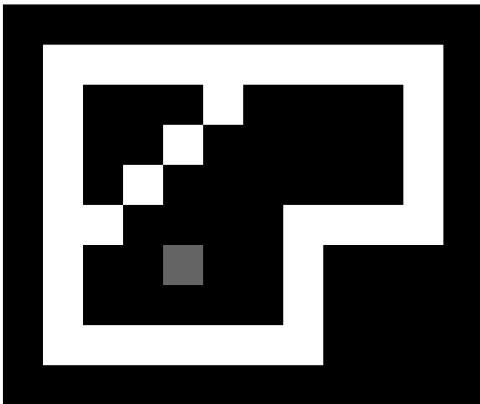
```
img = zeros(70,100);  
body = [60,30; 20,50; 20,90; 60,90];  
%body = [60,30; 40,10; 10,40; 30,40; 30,70; 10,90; 60,90;60,30];
```

```
img = rv_srafy3(body,img);  
figure, imshow(img);
```



Hranice v rastru

```
% I = logical(imread('oblst_ukol.png'));  
% seed = [60,60];  
I = imread('oblastslide.png');  
seed = [7,5];  
I2 = I;  
I2(seed(1), seed(2)) = 100;  
  
imshow(imresize(I2,20, 'nearest'));
```

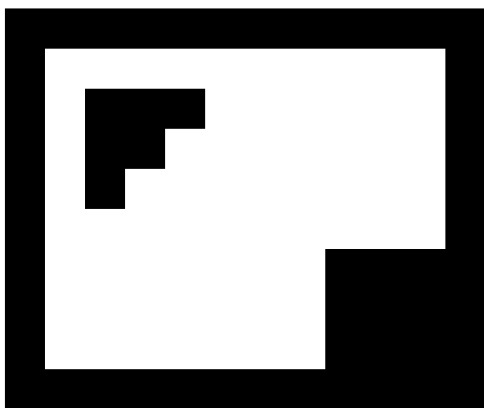


Semínkové vyplňování

funkce `imfill()`

4-sousedná oblast

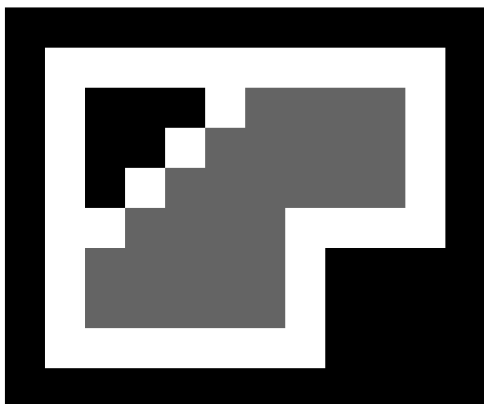
```
BW2= imfill(logical(I),seed,4);  
figure, imshow(imresize(BW2,20, 'nearest'));
```



Vlastní funkce

seminkove_vyplnovani()

```
figure,
J = seminkove_vyplnovani(I, seed);
figure, imshow(imresize(J,20, 'nearest'));
```



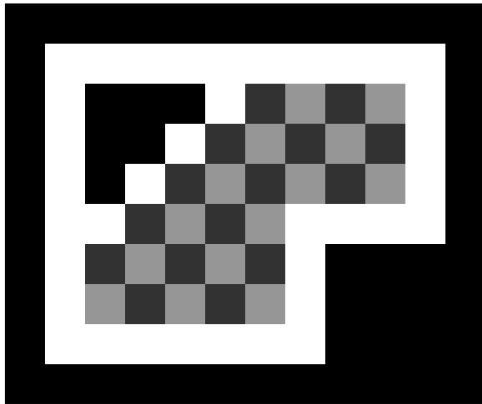
Vyplnění vzorem

seminkove_vyplnovani2()

```
vzor = [50 150 50 150;
        150 50 150 50];

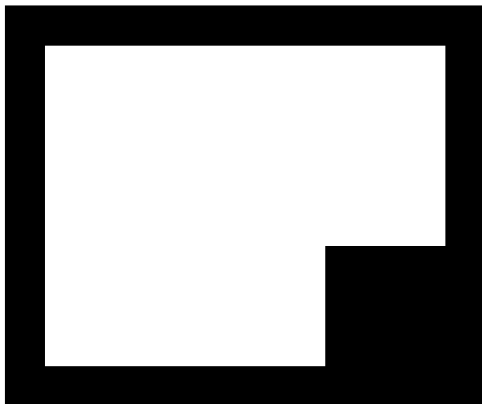
J = seminkove_vyplnovani2(I, seed, vzor);
```

```
figure, imshow(imresize(J,20, 'nearest'));
```



8-sousedná oblast

```
BW2= imfill(logical(I),seed,8);  
figure, imshow(imresize(BW2,20, 'nearest'));
```



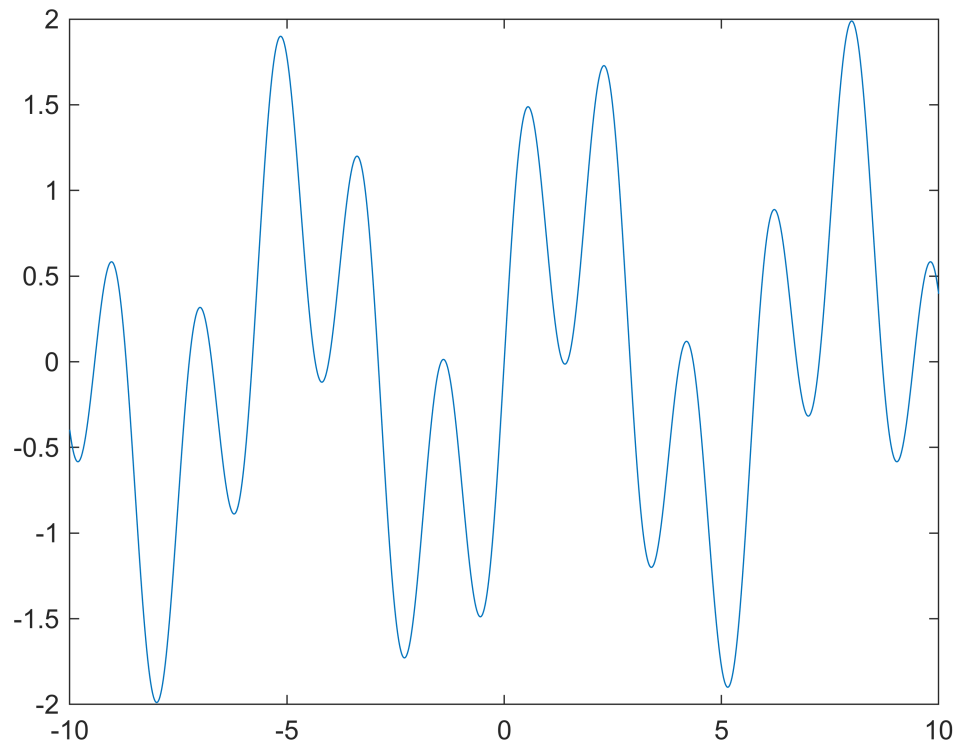
Zadání a vykreslení křivek

Explicitní zadání

$y = f(x)$

```
y = @(x) sin(x) + sin((10.0 / 3.0) .* x);  
figure,
```

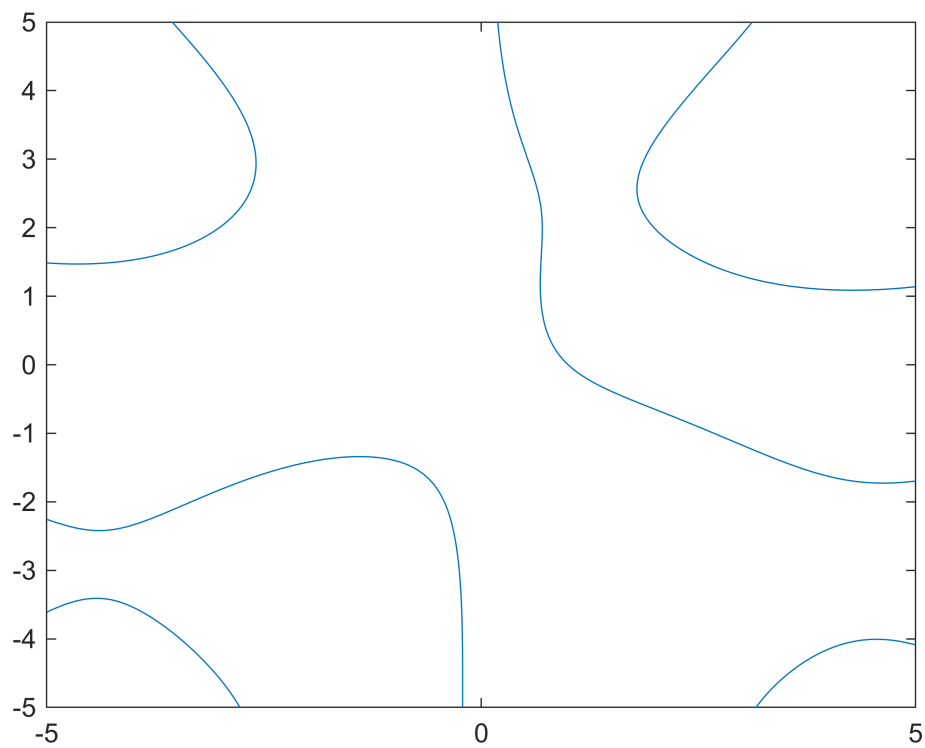
```
fplot(y,[-10,10]);
```



Implicitní zadání

$f(x,y) = 0$

```
figure,  
fp = fimplicit(@(x,y) y.*sin(x) + x.*cos(y) - 1);
```

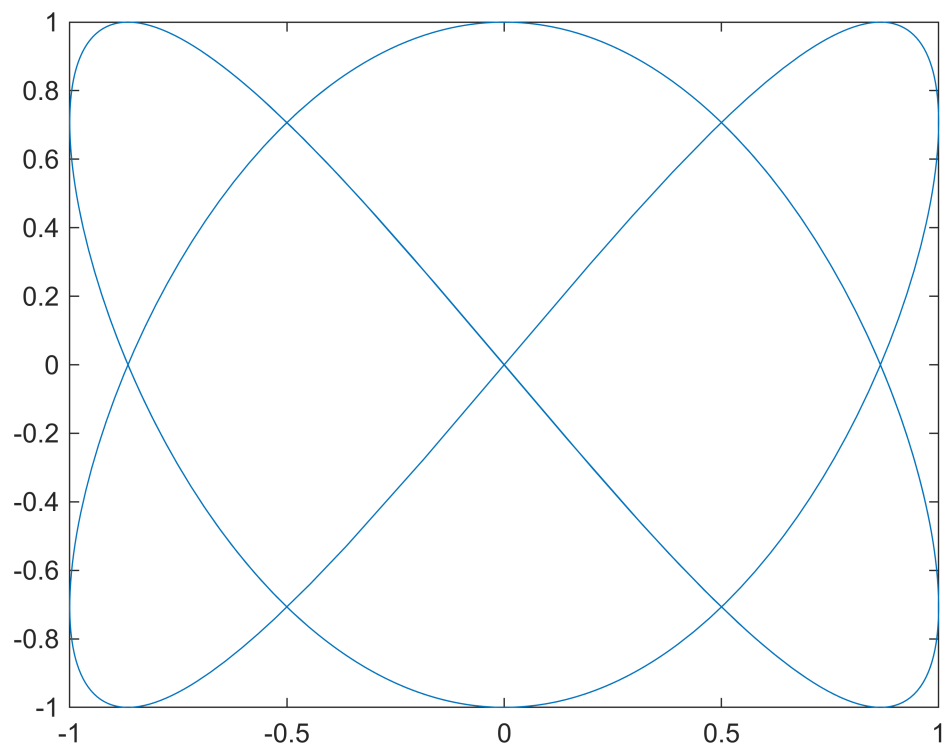


Parametrické zadání

$x = x(t)$

$y = y(t)$

```
xt = @(t) sin(2*t);  
yt = @(t) sin(3*t);  
  
figure, fplot(xt,yt,[-5,5]);
```



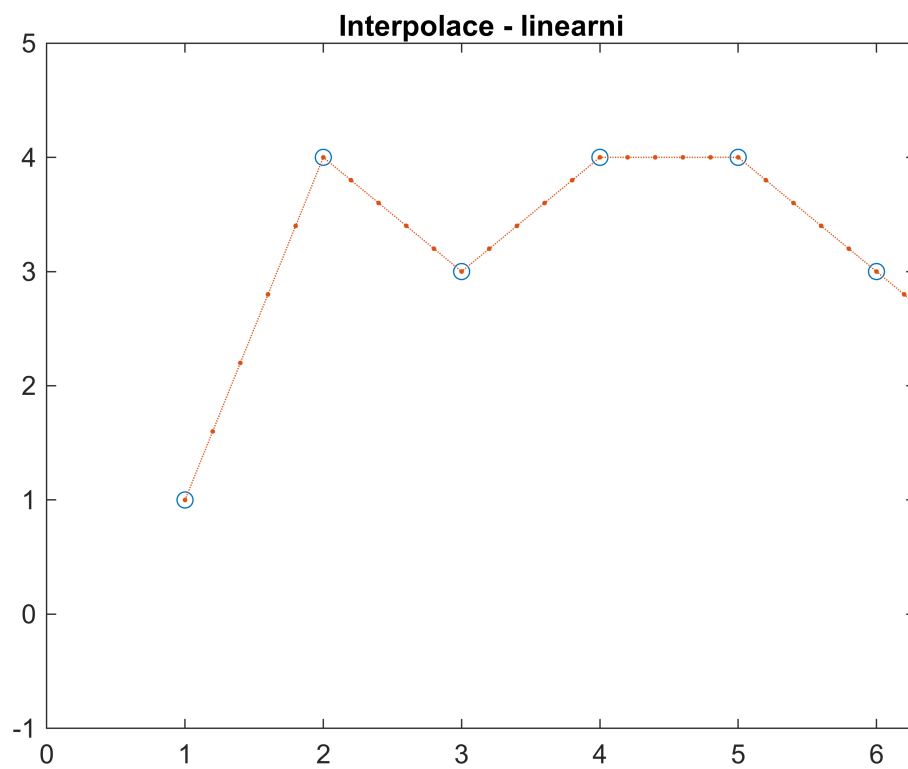
Interpolační křivky

```
vystup_y = interp1(vstup_x, vstup_y, vystup_x, metoda);
```

```
vstup_x = [1 2 3 4 5 6 7 8];
vstup_y = [1 4 3 4 4 3 2 1];
vystup_x = 0:.2:10;
```

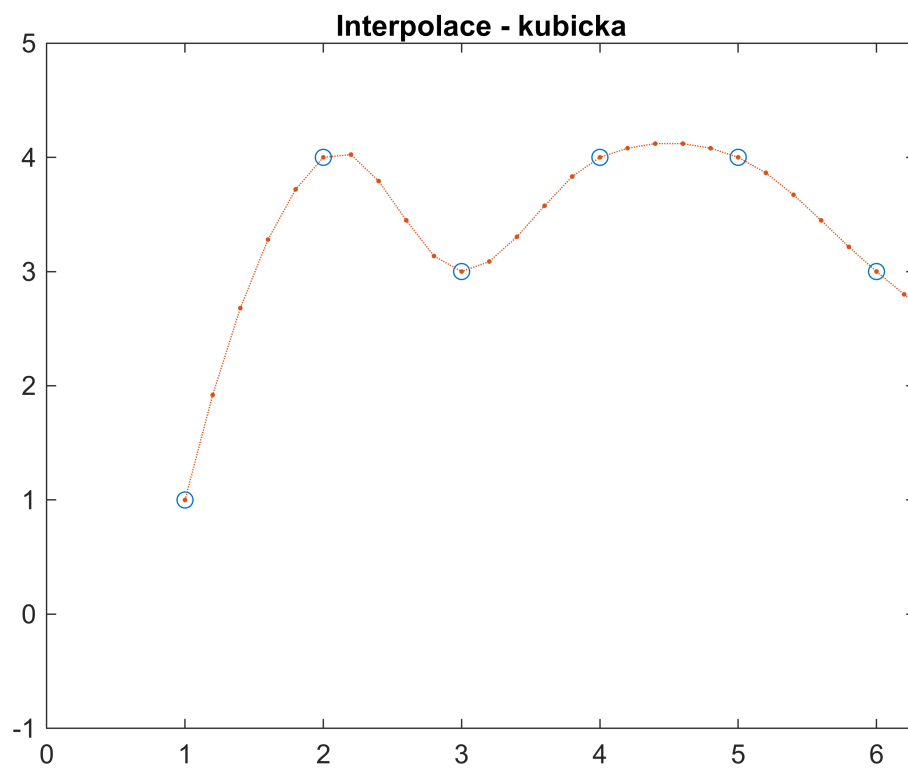
Lineární interpolace

```
vystup_y = interp1(vstup_x,vstup_y,vystup_x,'linear');
figure, plot(vstup_x,vstup_y,'o',vystup_x,vystup_y,':');
xlim([0 2*pi]);
ylim([-1 5])
title('Interpolace - lineární');
```

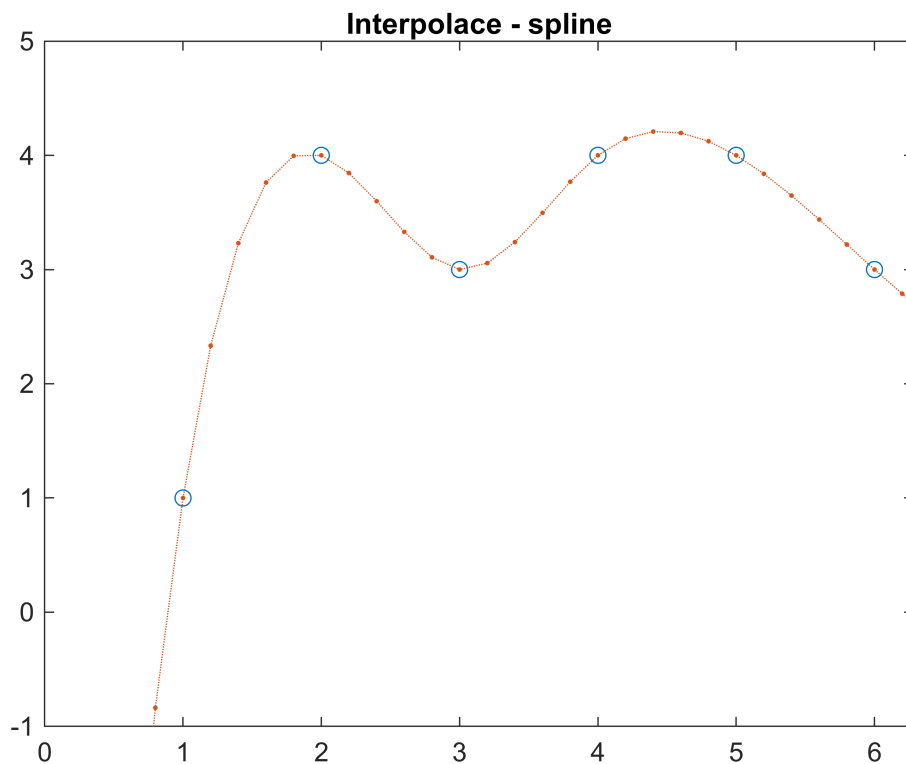
Kubická interpolace

```
vystup_y = interp1(vstup_x,vstup_y,vystup_x,'cubic');  
figure, plot(vstup_x,vstup_y,'o',vystup_x,vystup_y,':');  
xlim([0 2*pi]);  
ylim([-1 5])  
title('Interpolace - kubicka');
```



Spline interpolace

```
vystup_y = interp1(vstup_x,vstup_y,vystup_x,'spline');  
figure, plot(vstup_x,vstup_y,'o',vystup_x,vystup_y,':');  
xlim([0 2*pi]);  
ylim([-1 5])  
title('Interpolace - spline');
```



Hermitovské kubiky

$$P(t) = (2t^3 - 3t^2 + 1)P_0 + (t^3 - 2t^2 + t)\vec{p}_0' + (-2t^3 + 3t^2)P_1 + (t^3 - t^2)\vec{p}_1'$$

```
t = linspace(0,1,100);
% bod P0 = [x1, y1]
x1 = 0; y1 = 0;
P0 = [x1 y1];
% bod P1 = [x2, y2]
x2 = 1; y2 = 5;
P1 = [x2, y2];

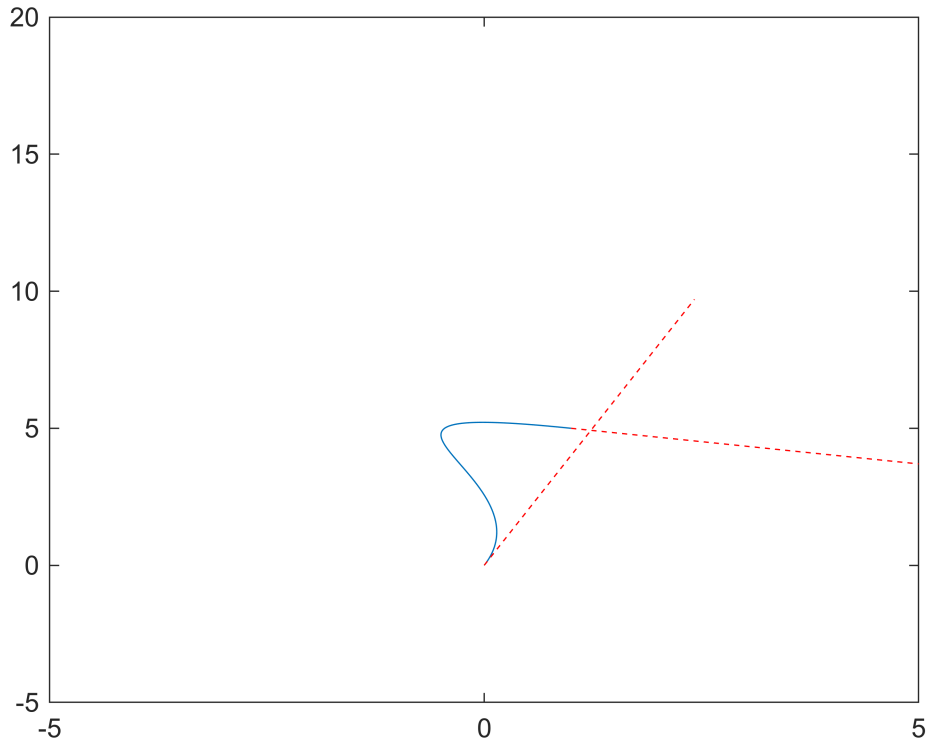
uhel1 = 76;
uhel2 = -18;
velikost1 = 10;
velikost2 = 10;
p0 = velikost1 * [cosd(uhel1) sind(uhel1)];
p1 = velikost2 * [cosd(uhel2) sind(uhel2)];

pp0 = P0 + p0;
pp1 = P1 + p1;

X = (2*t.^3 - 3*t.^2 + 1)*P0(1) + (t.^3 - 2*t.^2 + t)*pp0(1) + (-2*t.^3 + 3*t.^2) *
P1(1) + (t.^3 - t.^2)*pp1(1);
```

```
Y = (2*t.^3 - 3*t.^2 + 1)*P0(2) + (t.^3 - 2*t.^2 + t)*p0(2) + (-2*t.^3 + 3*t.^2) *  
P1(2) + (t.^3 - t.^2)*p1(2);
```

```
figure, plot(X,Y,'-')  
hold on  
plot([x1 pp0(1)],[y1 pp0(2)],'r--')  
plot([x2 pp1(1)],[y2 pp1(2)],'r--')  
xlim([-5 5]);  
ylim([-5 20]);  
hold off
```



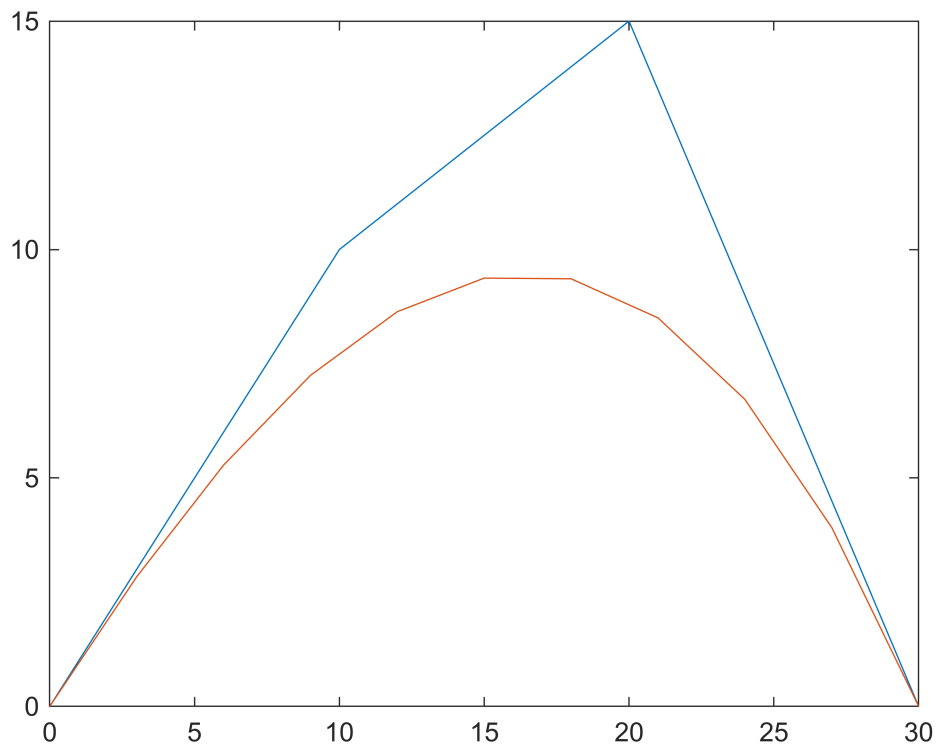
Aproximační křivky

Beziérovky křivky

Naivní algoritmus.

```
b = [];  
P = [0 10 20 30;  
     0 10 15 0];  
  
for i = 0 : 0.1 : 1  
    b = [b, bezier( i, P )];  
end  
  
figure,  
plot(P(1,:), P(2,:));
```

```
hold on;
plot(b(1,:), b(2,:));
hold off
```



Rekurzivní algoritmus de Casteljau

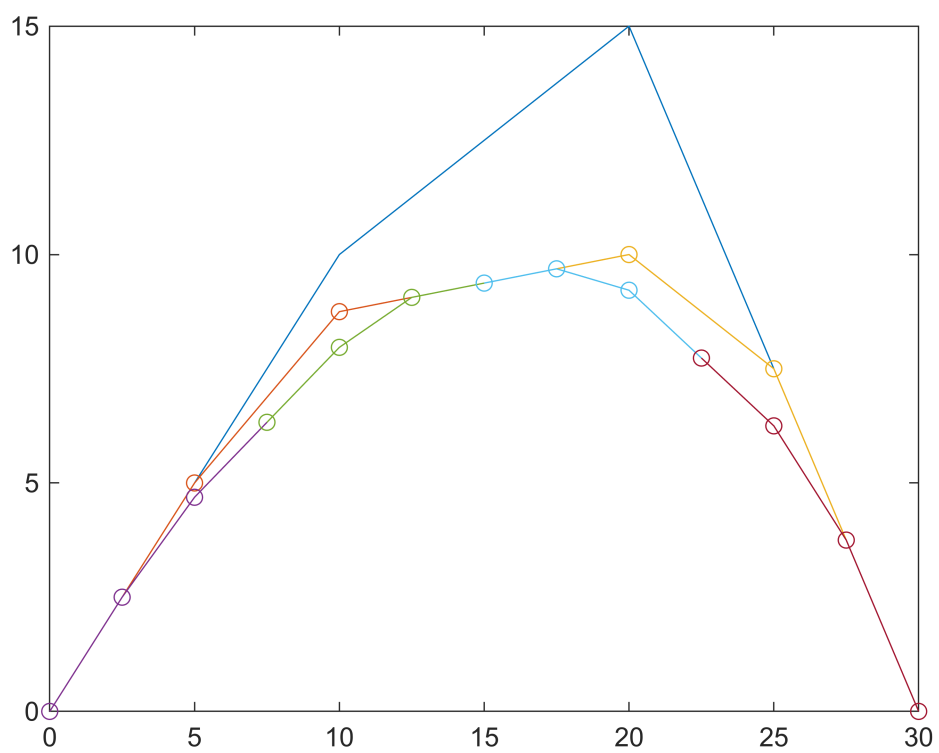
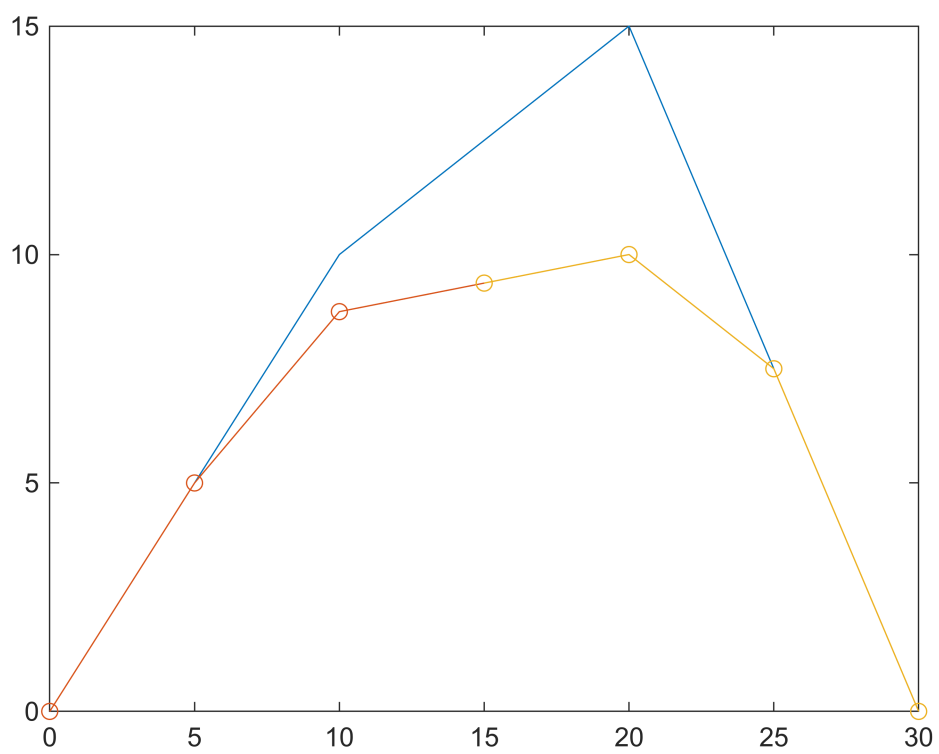
```
P = [0 10 20 30;
      0 10 15 0];
t = 1/2;

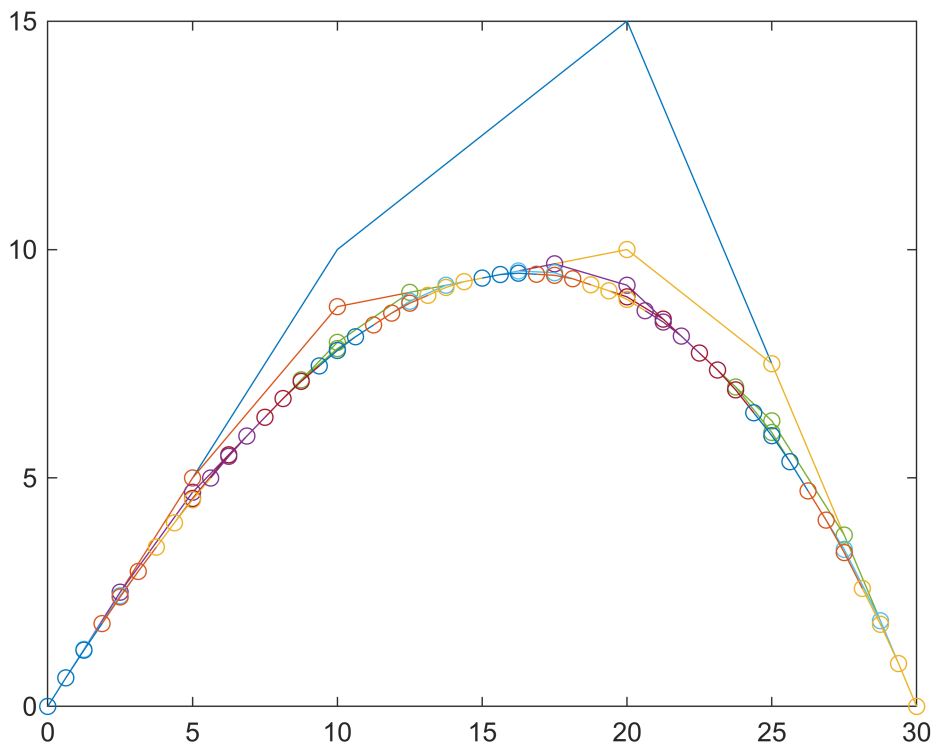
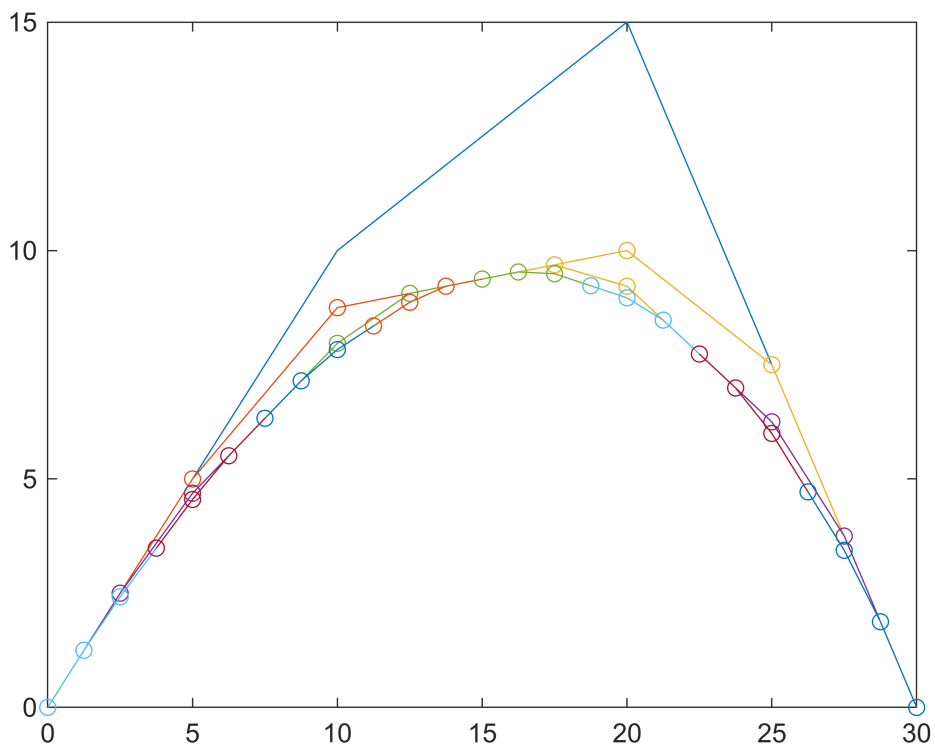
for n = 1 : 4 % pocet iteraci

    figure, plot(P(1,:), P(2,:)); % vykreslení kontrolních bodů

    hold on;
    B = bezierCastlejau(P, n, t);
    hold off;

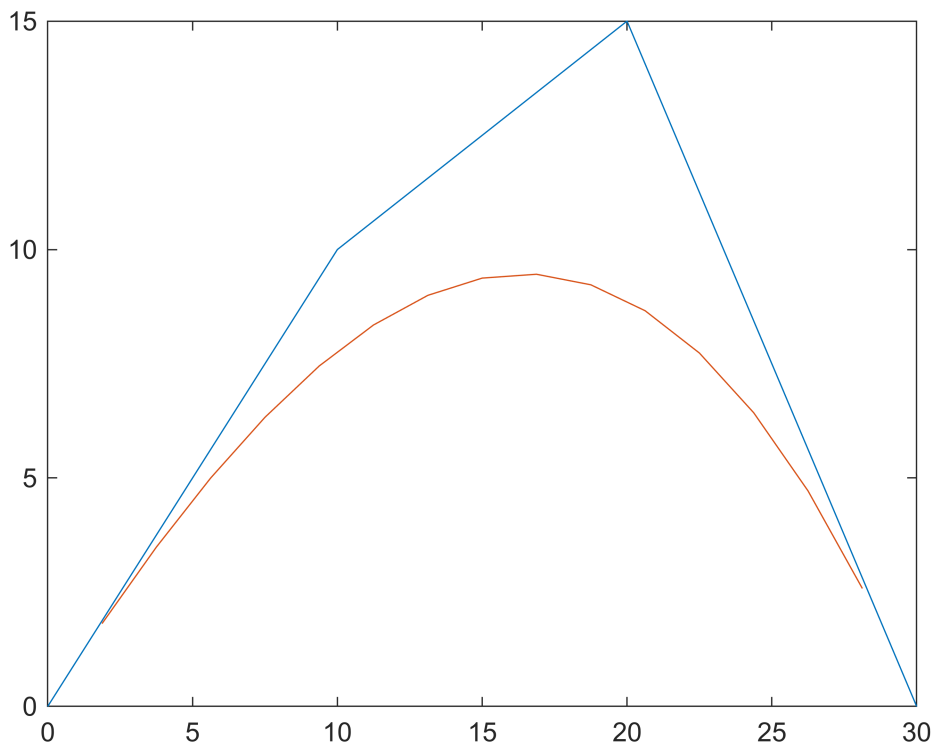
end
```





```
figure,
plot(P(1,:), P(2,:));
```

```
hold on;
plot(B(1,:), B(2,:));
hold off
```



Pomocné funkce

```
function [enc, delka] = RLEenc(vstup)
% kodovani
enc = '';
r = 0;
delka = 0;
for i = 1 : size(vstup,2)
    a = vstup(i);
    if r == 0
        x = a;
        r = 1;
    else
        if a == x
            r = r + 1;
        else
            enc = [enc num2str(r) x];
            x = a;
            r = 1;
            delka = delka + 9; %pocet je 1 byte, znak 1 bite
        end
    end
end
```



```

        end
    end
end
enc = [enc num2str(r) a];
delka = delka + 9;
end

function dec = RLEdec(enc)
    % dekodovani
    r = 0;
    dec = '';
    for i = 1 : 2 : size(enc,2)
        n = str2num(enc(i));
        znak = enc(i+1);
        dec = [dec repmat(znak,[1 n])];
    end
end

function kody = Huffman(A, B, p_A, prvni)
    n = size(A,2);
    m = size(B,2);
    kody = cell(1,n);
    if n <= m
        for i = 1 : n
            kody{i} = B(i);
        end
    else
        if prvni
            m2 = mod(n-2, (m-1)) +2;
        else
            m2 = m;
        end
        [p_A_sort,indexy] = sort(p_A,'descend');
        A2 = A(indexy);
        A3 = A2(1 : n - m2);
        p_A3 = p_A_sort(1 : n - m2);
        p_A3 = [p_A3 sum(p_A_sort(n - m2+1:end))];
        A3 = [A3 [A2{n - m2+1:end}]];
        kody2 = Huffman(A3, B, p_A3, false);

        for i = 1 : n - m2
            kody{indexy(i)} = kody2{i};
        end
        for i = 1 : m2
            kody{indexy(n - m2 + i)} = [char(kody2{end}) B{i}];
        end
    end
end

function vystup = Huffman_enc(vstup, znaky,kody)

```

```

vystup = [];
for i = 1 : size(vstup,2)
    index = find(znaky==vstup(i),1,'first');
    vystup = [vystup,kody{index}];
end
endfunction out = DDA(A,B,velikost)
out = ones(velikost);
% nastaveni offsetu pro primky se souradnicema <=0

if (A(1) <=0 || B(1) <= 0)
    offset_x = - min(A(1),B(1)) + 1;
else
    offset_x = 0;
end

if (A(2) <=0 || B(2) <= 0)
    offset_y = - min(A(2),B(2)) + 1;
else
    offset_y = 0;
end

% z koncovych bodu se urci smernice
dx = B(1)-A(1);
dy = B(2)-A(2);

m = dy/dx;

%inicializace [x,y]
x = A(1);
y = A(2);

while(x <= B(1))
    out(round(y)+offset_y, x+offset_x) = 0;
    x = x +1;
    y = y + m;
end

out = flipud(out); % otoceni obrazku
end

```

Bresengamův algoritmus

```

function out = bresenhamuv_algoritmus(A,B,velikost)
out = ones(velikost);

% nastaveni offsetu pro primky se souradnicema <=0
if (A(1) <=0 || B(1) <= 0)
    offset_x = - min(A(1),B(1)) + 1;
else
    offset_x = 0;

```

```

end

if (A(2) <=0 || B(2) <= 0)
    offset_y = - min(A(2),B(2)) + 1;
else
    offset_y = 0;
end

% z koncovych bodu se urci konstanty
dx = B(1)-A(1);
dy = B(2)-A(2);

k1 = 2*dy;
k2 = 2*(dy-dx);

% rozhodovaci clen
p = 2*(dy-dx);

%inicializace [x,y]
x = A(1);
y = A(2);

while(x <= B(1))
    x = x +1;
    if (p >0)    % p je kladne
        y = y +1;
        p = p + k2;
    else        % p neni kladne
        p = p + k1;
    end
    out(y+offset_y, x+offset_x) = 0; %vykresleni bodu    (v obrazcich je prvni
souradnice sloupec a druhy radek)
end

out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - přerušovaná čára (naivní)

```

function out = ba_prerusovana(A,B,velikost, delka)
    aktualni_delka = delka;
    plny = 1;

    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0

    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else

```

```

        offset_x = 0;
    end

    if (A(2) <= 0 || B(2) <= 0)
        offset_y = - min(A(2), B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    % inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        x = x + 1;
        if (p > 0) % p je kladne
            y = y + 1;
            p = p + k2;
        else % p neni kladne
            p = p + k1;
        end

        % vykresleni jen, pokud je v plnem useku
        if(plny == 1)
            out(y+offset_y, x+offset_x) = 0; % vykresleni bodu (v obrazcich je
            % prvni souradnice sloupec a druhy radek)
        end

        aktualni_delka = aktualni_delka - 1;
        if(aktualni_delka == 0)
            if(plny == 1)
                plny = 0;
            else
                plny = 1;
            end
            aktualni_delka = delka;
        end
    end
end

```

```

    out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - přerušovaná čára

```

function out = ba_prerusovana2(A,B,velikost, delka)
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    delka = round((abs(dx) / sqrt(dx^2 + dy^2) )*delka);
    aktualni_delka = delka;
    plny = 1;

    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    %inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        x = x +1;
        if (p >0) % p je kladne
            y = y +1;
            p = p + k2;
        else % p neni kladne
            p = p + k1;
        end
    end

```

```

    %vykresleni jen, pokud je v plnem useku
    if(plny == 1)
        out(y+offset_y, x+offset_x) = 0; %vykresleni bodu (v obrazcich je
        prvni souradnice sloupec a druhy radek)
    end

    aktualni_delka = aktualni_delka -1;
    if(aktualni_delka ==0)
        if(plny == 1)
            plny = 0;
        else
            plny = 1;
        end
        aktualni_delka = delka;
    end
end
out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - silná čára (naivní)

```

function out = ba_silna(A,B,velikost, tloustka)
    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    %inicializace [x,y]
    x = A(1);

```

```

y = A(2);

while(x <= B(1))
    x = x +1;
    if (p >0)    % p je kladne
        y = y +1;
        p = p + k2;
    else        % p není kladne
        p = p + k1;
    end
    out(y+offset_y: y+offset_y + tloustka-1, x+offset_x) = 0; %vykresleni
    tloustka bodu

end
out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - silná čára

```

function out = ba_silna2(A,B,velikost, tloustka)
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    tloustka = round((sqrt(dx^2 + dy^2) / abs(dx))*tloustka);

    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci konstanty
    k1 = 2*dy;
    k2 = 2*(dy-dx);

    % rozhodovaci clen
    p = 2*(dy-dx);

    %inicializace [x,y]
    x = A(1);

```

```

y = A(2);

while(x <= B(1))
    x = x +1;
    if (p >0)    % p je kladne
        y = y +1;
        p = p + k2;
    else        % p neni kladne
        p = p + k1;
    end

    out(y+offset_y: y+offset_y + tloustka-1, x+offset_x) = 0; %vykresleni
    tloustka bodu

end

out = flipud(out); % otoceni obrazku
end

```

DDA antialias

```

function out = DDAalias(A,B,velikost)
    out = ones(velikost);

    % nastaveni offsetu pro primky se souradnicema <=0
    if (A(1) <=0 || B(1) <= 0)
        offset_x = - min(A(1),B(1)) + 1;
    else
        offset_x = 0;
    end

    if (A(2) <=0 || B(2) <= 0)
        offset_y = - min(A(2),B(2)) + 1;
    else
        offset_y = 0;
    end

    % z koncovych bodu se urci smernice
    dx = B(1)-A(1);
    dy = B(2)-A(2);

    m = dy/dx;

    %inicializace [x,y]
    x = A(1);
    y = A(2);

    while(x <= B(1))
        out(floor(y)+offset_y, x+offset_x) = y - floor(y);
        out(ceil(y)+offset_y, x+offset_x) = ceil(y) - y;

```



```

        x = x + 1;
        y = y + m;
    end

    out = flipud(out); % otoceni obrazku
end

```

Bresenhamův algoritmus - kružnice

```

function out = ba_kruznice(r, stred, velikost)
    out = ones(velikost);

    % offset, aby byla kruznice uprostred. Pripadne by zde
    offset_x = stred(1);
    offset_y = stred(2);

    devx = 3;
    devy = 2*r - 2;

    % rozhodovaci clen
    p = 1 - r;

    x = 0;
    y = r;

    while(x <= y)
        out(x+offset_x, y+offset_y) = 0;
        out(-x+offset_x, y+offset_y) = 0;
        out(x+offset_x, -y+offset_y) = 0;
        out(-x+offset_x, -y+offset_y) = 0;
        out(y+offset_y, x+offset_x) = 0;
        out(-y+offset_y, x+offset_x) = 0;
        out(y+offset_y, -x+offset_x) = 0;
        out(-y+offset_y, -x+offset_x) = 0;
        if (p >= 0)
            p = p - devy;
            devy = devy - 2;
            y = y - 1;
        end
        p = p + devx;
        devx = devx + 2;
        x = x+1;
    end
end

```

Řádkové vyplňování

```

function radkove_vyplnovani(body,img)
    pocet_bodu = size(body,1)-1;

```

```

% smernice jednotlivych usecek
% (y2 - y1) / (x2-x1)

smernice = zeros(1,pocet_bodu);
for i = 1: pocet_bodu
    smernice(i) = (body(i+1,2)-body(i,2))/(body(i+1,1)-body(i,1));
end

% Hrany, ktere se pocitaji
hrany = [];
for i = 1: pocet_bodu
    if(smernice(i) < Inf && smernice(i) > -Inf)
        hrany = [hrany; body(i,:), body(i+1,:)];
    end
end

pocet_hran = size(hrany,1);
%display(hrany);

%zde by melo dojít ke zkraceni vseh usecek

figure,

% nalezeni vseh hranic na y
for j = min(body(:,1)) : max(body(:,1))
    pruseciky_x = [];
    for i = 1 : pocet_hran
        t = (j - hrany(i,1))/(hrany(i,3)-hrany(i,1));

        if (t>=0 && t <=1) %kontrola zda lezi na usecce
            x = hrany(i,2) + t*(hrany(i,4)-hrany(i,2));
            pruseciky_x = [pruseciky_x, x];
        end
    end

    %display(pruseciky_x);
    pruseciky_x = sort(pruseciky_x);
    for k = size(pruseciky_x,2) : -2 : 2
        for l = round(pruseciky_x(k - 1)) : round(pruseciky_x(k))
            img(j,l) = 1;
        end
    end
    imshow(img);
    pause(0.1)
end
end

```

Řádkové vyplňování vzorem

```
function img = radkove_vyplnovani_vzor(body,img, vzor)
```

```

[mv, nv] = size(vzor);
pocet_bodu = size(body,1)-1;

smernice = zeros(1,pocet_bodu);
for i = 1: pocet_bodu
    smernice(i) = (body(i+1,2)-body(i,2))/(body(i+1,1)-body(i,1));
end

hrany = [];
for i = 1: pocet_bodu
    if(smernice(i) < Inf && smernice(i) > -Inf)
        hrany = [hrany; body(i,:), body(i+1,:)];
    end
end

pocet_hran = size(hrany,1);

for j = min(body(:,1)) : max(body(:,1))
    pruseciky_x = [];
    for i = 1 : pocet_hran
        t = (j - hrany(i,1))/(hrany(i,3)-hrany(i,1));

        if (t>=0 && t <=1)
            x = hrany(i,2) + t*(hrany(i,4)-hrany(i,2));
            pruseciky_x = [pruseciky_x, x];
        end
    end

    pruseciky_x = sort(pruseciky_x);
    for k = 1 : 2 : size(pruseciky_x,2)-1
        for l = round(pruseciky_x(k)) : round(pruseciky_x(k+1))
            img(j,l) = vzor(mod(j,mv)+1,mod(l,nv)+1);
        end
    end
end
end
end

```

Řádkové vyplňování šrafování vodorovné

```

function img = rv_srafy1(body,img)
    pocet_bodu = size(body,1)-1;

    % smernice jednotlivych usecek
    % (y2 - y1) / (x2-x1)
    smernice = zeros(1,pocet_bodu);
    for i = 1: pocet_bodu
        smernice(i) = (body(i+1,2)-body(i,2))/(body(i+1,1)-body(i,1));
    end

    % Hrany, ktere se pocitaji

```

```

hrany = [];
for i = 1: pocet_bodu
    if(smernice(i) < Inf && smernice(i) > -Inf)
        hrany = [hrany; body(i,:), body(i+1,:)];
    end
end

pocet_hran = size(hrany,1);
%display(hrany);

%zde by melo dojít ke zkrácení všech usecek

% nalezení všech hranic na y
for j = min(body(:,1)) : 3 : max(body(:,1))
    pruseciky_x = [];
    for i = 1 : pocet_hran
        t = (j - hrany(i,1))/(hrany(i,3)-hrany(i,1));

        if (t>=0 && t <=1) %kontrola zda lezi na usecce
            x = hrany(i,2) + t*(hrany(i,4)-hrany(i,2));
            pruseciky_x = [pruseciky_x, x];
        end
    end

    %display(pruseciky_x);
    pruseciky_x = sort(pruseciky_x);
    for k = size(pruseciky_x,2) : -2 : 2
        %for k = 1 : 2 : size(pruseciky_x,2)
        for l = round(pruseciky_x(k - 1)) : round(pruseciky_x(k))
            img(j,l) = 1;
        end
    end
end
end
end

```

Řádkové vyplňování šrafování svislé naivní

```

function img = rv_srafy2(body,img)
    pocet_bodu = size(body,1)-1;

    smernice = zeros(1,pocet_bodu);
    for i = 1: pocet_bodu
        smernice(i) = (body(i+1,2)-body(i,2))/(body(i+1,1)-body(i,1));
    end

    hrany = [];
    for i = 1: pocet_bodu
        if(smernice(i) < Inf && smernice(i) > -Inf)
            hrany = [hrany; body(i,:), body(i+1,:)];
        end
    end
end

```

```

end

pocet_hran = size(hrany,1);

delka_useku = 5;

for j = min(body(:,1)) : max(body(:,1))
    pruseciky_x = [];
    for i = 1 : pocet_hran
        t = (j - hrany(i,1))/(hrany(i,3)-hrany(i,1));

        if (t>=0 && t <=1)
            x = hrany(i,2) + t*(hrany(i,4)-hrany(i,2));
            pruseciky_x = [pruseciky_x, x];
        end
    end

    pruseciky_x = sort(pruseciky_x);
    for k = 1 : 2 : size(pruseciky_x,2)-1
        plny = 1;
        aktualni_delka = delka_useku;
        for l = round(pruseciky_x(k)) : round(pruseciky_x(k+1))
            if(plny==1)
                img(j,l) = 1;
            end
            aktualni_delka = aktualni_delka -1;
            if(aktualni_delka ==0)
                if(plny == 1)
                    plny = 0;
                else
                    plny = 1;
                end
                aktualni_delka = delka_useku;
            end
        end
    end
end
end
end
end

```

Řádkové vyplňování šrafování svislé

```

function img = rv_srafy3(body,img)
    % srafy vztazene k x = 1
    x = 1;
    % x = min(body(:,2));

    n = size(img,2);
    delka_useku = 5;
    aktualni_delka = delka_useku;

```

```

plny = 1;
srafy = zeros(1,n);
for i = x : n
    if(plny == 1)
        srafy(i) = 1;
    end
    aktualni_delka = aktualni_delka -1;
    if(aktualni_delka ==0)
        if(plny == 1)
            plny = 0;
        else
            plny = 1;
        end
        aktualni_delka = delka_useku;
    end
end

pocet_bodu = size(body,1)-1;

smernice = zeros(1,pocet_bodu);
for i = 1: pocet_bodu
    smernice(i) = (body(i+1,2)-body(i,2))/(body(i+1,1)-body(i,1));
end

hrany = [];
for i = 1: pocet_bodu
    if(smernice(i) < Inf && smernice(i) > -Inf)
        hrany = [hrany; body(i,:), body(i+1,:)];
    end
end

pocet_hran = size(hrany,1);

for j = min(body(:,1)) : max(body(:,1))
    pruseciky_x = [];
    for i = 1 : pocet_hran
        t = (j - hrany(i,1))/(hrany(i,3)-hrany(i,1));

        if (t>=0 && t <=1)
            x = hrany(i,2) + t*(hrany(i,4)-hrany(i,2));
            pruseciky_x = [pruseciky_x, x];
        end
    end

    pruseciky_x = sort(pruseciky_x);
    for k = 1 : 2 : size(pruseciky_x,2)-1
        for l = round(pruseciky_x(k)) : round(pruseciky_x(k+1))
            img(j,l) = srafy(1);
        end
    end
end

```

```
end
end
```

Semínkové vyplňování

```
function I = seminkove_vyplnovani(I, seminko)
[m,n] = size(I);
if(I(seminko(1),seminko(2))== 0)
    I(seminko(1),seminko(2)) = 100;
    if(seminko(1) - 1 > 0)
        I = seminkove_vyplnovani(I, [seminko(1) - 1,seminko(2)]);
    end
    if(seminko(1) + 1 <= m)
        I = seminkove_vyplnovani(I, [seminko(1) + 1,seminko(2)]);
    end
    if(seminko(2) - 1 > 0)
        I = seminkove_vyplnovani(I, [seminko(1),seminko(2) - 1]);
    end
    if(seminko(2) + 1 <=n)
        I = seminkove_vyplnovani(I, [seminko(1),seminko(2) + 1]);
    end
end
end
```

Semínkové vyplňování vzorem

```
function I = seminkove_vyplnovani2(I, seminko, vzor)
[m,n] = size(I);
[mv, nv] = size(vzor);
if(I(seminko(1),seminko(2))== 0)
    I(seminko(1),seminko(2)) = vzor(mod(seminko(1),mv)+1,mod(seminko(2),nv)+1);
    if(seminko(1) - 1 > 0)
        I = seminkove_vyplnovani2(I, [seminko(1) - 1,seminko(2)], vzor);
    end
    if(seminko(1) + 1 <= m)
        I = seminkove_vyplnovani2(I, [seminko(1) + 1,seminko(2)], vzor);
    end
    if(seminko(2) - 1 > 0)
        I = seminkove_vyplnovani2(I, [seminko(1),seminko(2) - 1], vzor);
    end
    if(seminko(2) + 1 <=n)
        I = seminkove_vyplnovani2(I, [seminko(1),seminko(2) + 1], vzor);
    end
end
end

function B = bezierCastlejau(P, n, t)

% pokud je n = 1 spocitame pouze stredni bod a dale nedelime
P01 = (1 - t) * P(:, 1) + t * P(:, 2);
P12 = (1 - t) * P(:, 2) + t * P(:, 3);
```

```

P23 = (1 - t) * P(:, 3) + t * P(:, 4);

P012 = (1 - t) * P01 + t * P12;
P123 = (1 - t) * P12 + t * P23;

P0123 = (1 - t) * P012 + t * P123;

B1 = [P(:, 1) P01 P012 P0123];
B2 = [P0123 P123 P23 P(:,4)];
plot(B1(1, :), B1(2, :), 'o-');
plot(B2(1, :), B2(2, :), 'o-');

if(n == 1)
    % bod na krivce
    B = P0123;
else
    B = [bezierCastlejau([P(:, 1) P01 P012 P0123], n-1, t) P0123
bezierCastlejau([P0123 P123 P23 P(:,4)],n-1, t)];
end
end

function B = bezier( t, P )
    B = [0, 0]';
    n = size(P, 2);
    for i = 1:n
        B = B + bpol(t, i - 1, n - 1) * P(:, i);
    end
end

function value = bpol(t, i, n)
    value = nchoosek(n, i) * t^i * (1 - t)^(n - i);
end

```