

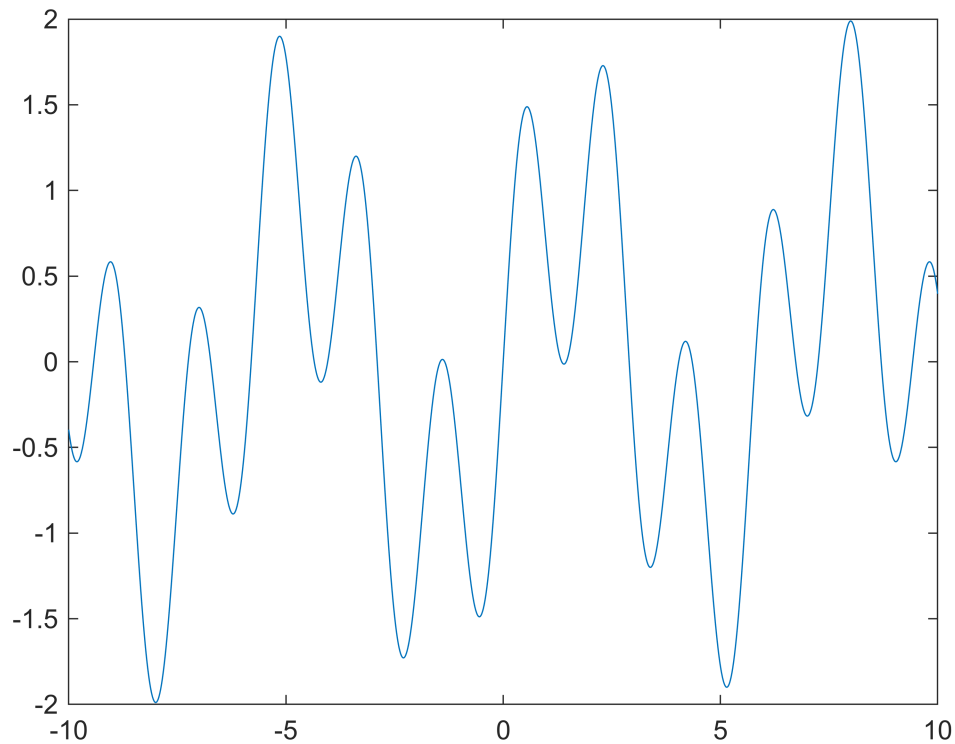
Seminář 11

Zadání a vykreslení křivek

Explicitní zadání

$$y = f(x)$$

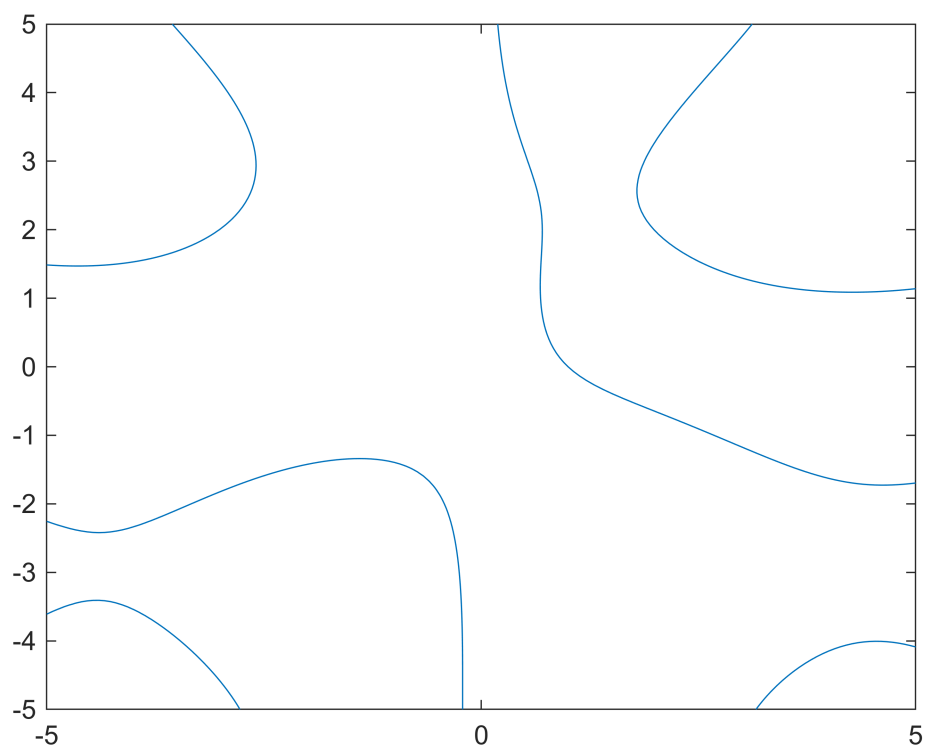
```
y = @(x) sin(x) + sin((10.0 / 3.0) .* x);  
figure,  
fplot(y,[-10,10]);
```



Implicitní zadání

$$f(x,y) = 0$$

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

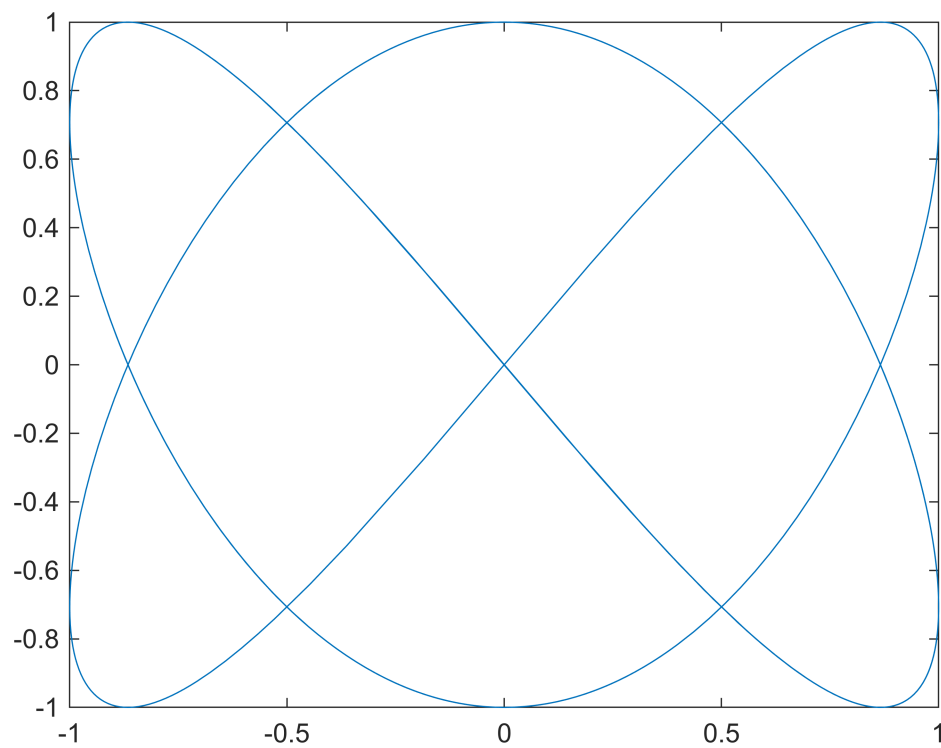


Parametrické zadání

$x = x(t)$

$y = y(t)$

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



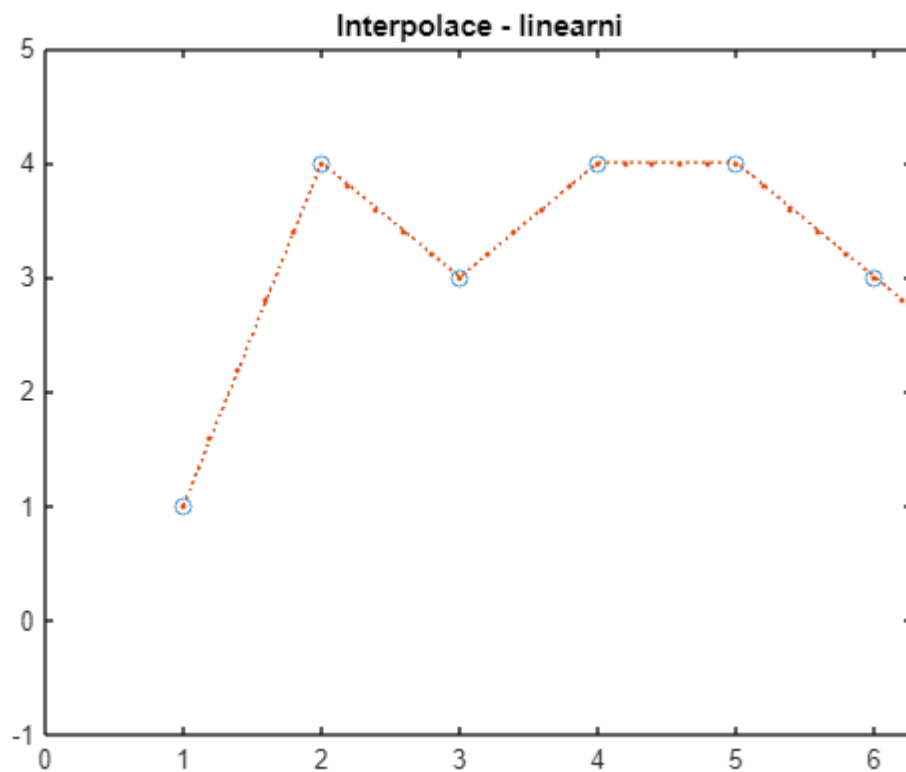
Interpolační křivky

```
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;
```

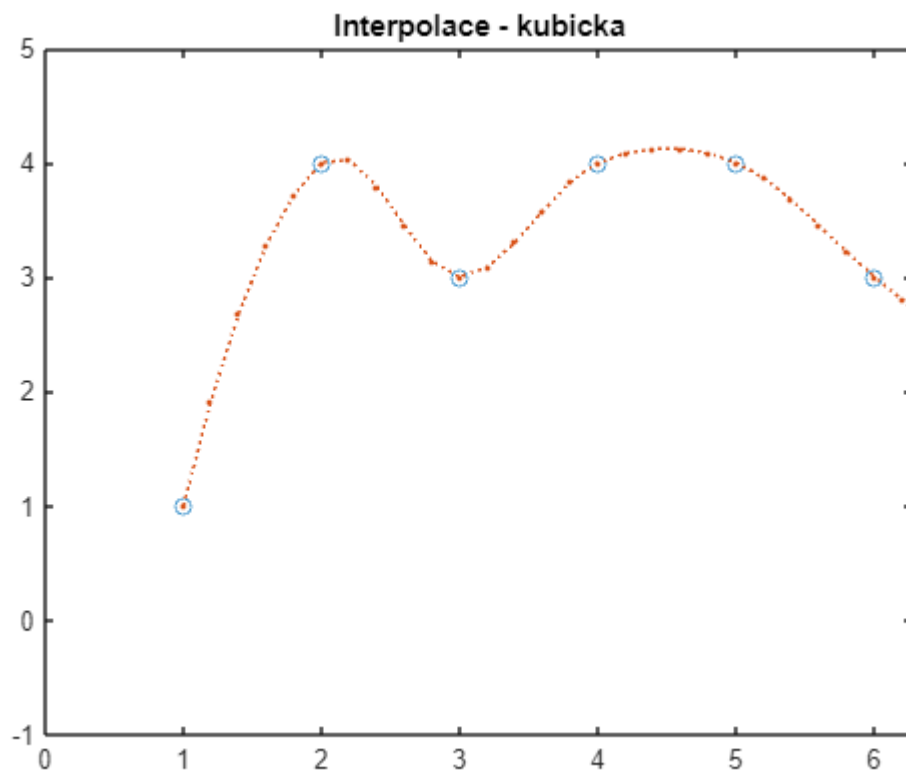
Lineární interpolace

```
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');
```



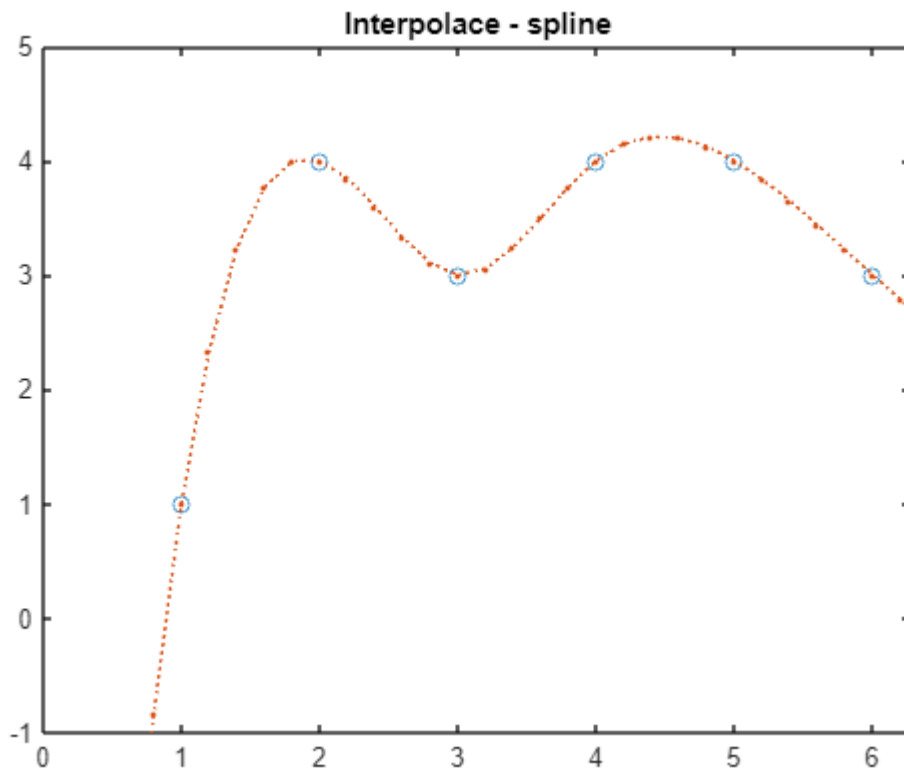
Kubická interpolace

```
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];

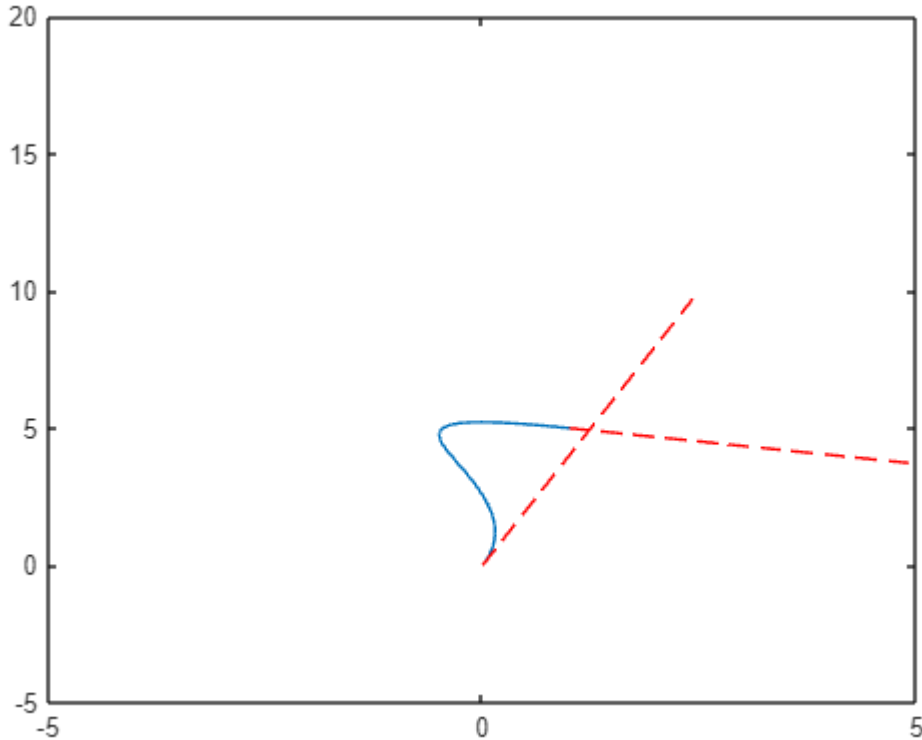
uhel1 = 76;
uhel2 = -18;
velikost1 = 10;
velikost2 = 10;
p0 = velikost1 * [cosd(uhel1) sind(uhel1)];
p1 = velikost2 * [cosd(uhel2) sind(uhel2)];

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

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



Úkol 1

Jak určíme návaznost Hermitovských kubik? Své tvrzení vyzkoušejte.

Aproximační křivky

Beziérovky křivky

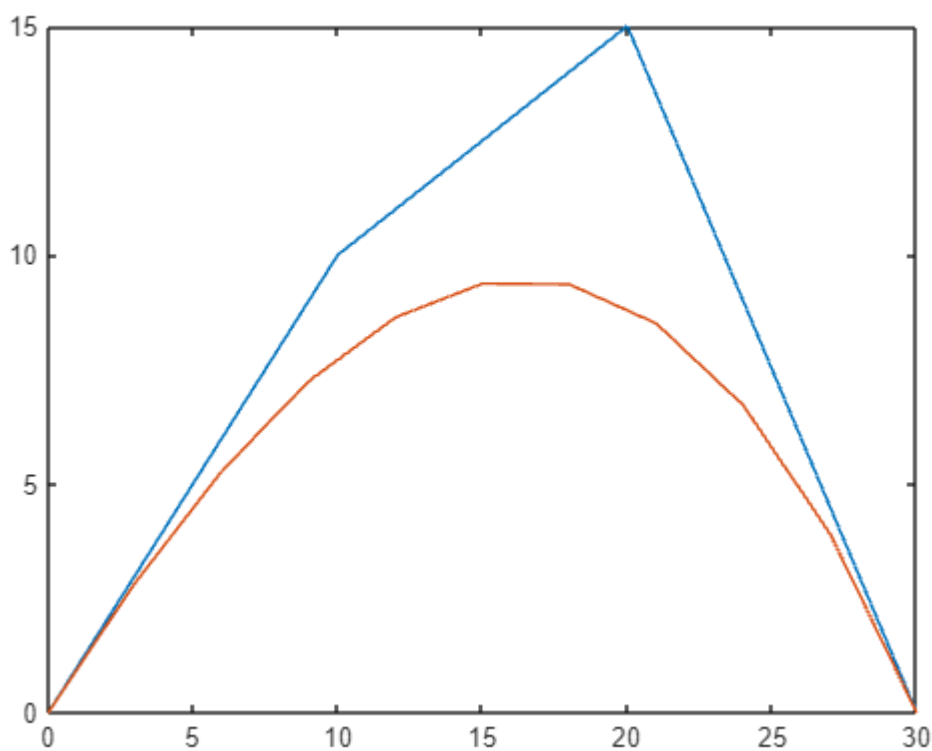
Naivní algoritmus.

Úkol 2

Naprogramujte funkci `B = bezier(t,P)` implementující naivní algoritmus pro výpočet aproximační Beziérovky kubiky. Funkce pro zadané `t` a vektor 2×4 kontrolních bodů (`P`) vrátí bod `B`. Pro výpočet kombinačního čísla můžete použít funkci `nchoosek()`.

Pro následující kód byste měli dostat následující výsledek.

```
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
```

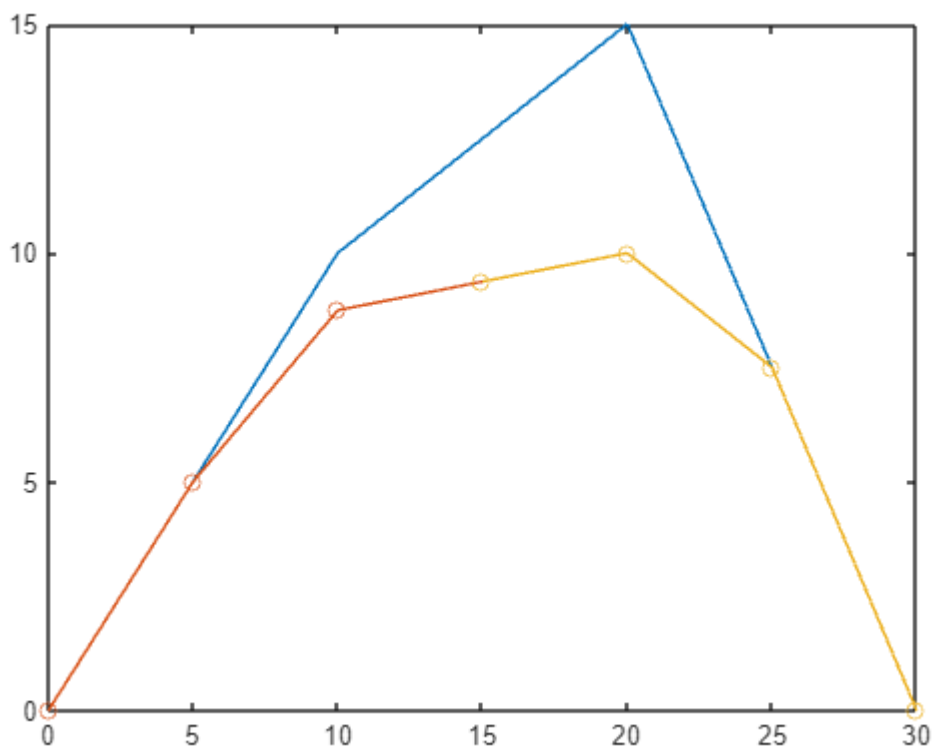
```
figure, plot(P(1,:), P(2,:)); % vykreslení kontrolních bodů
```

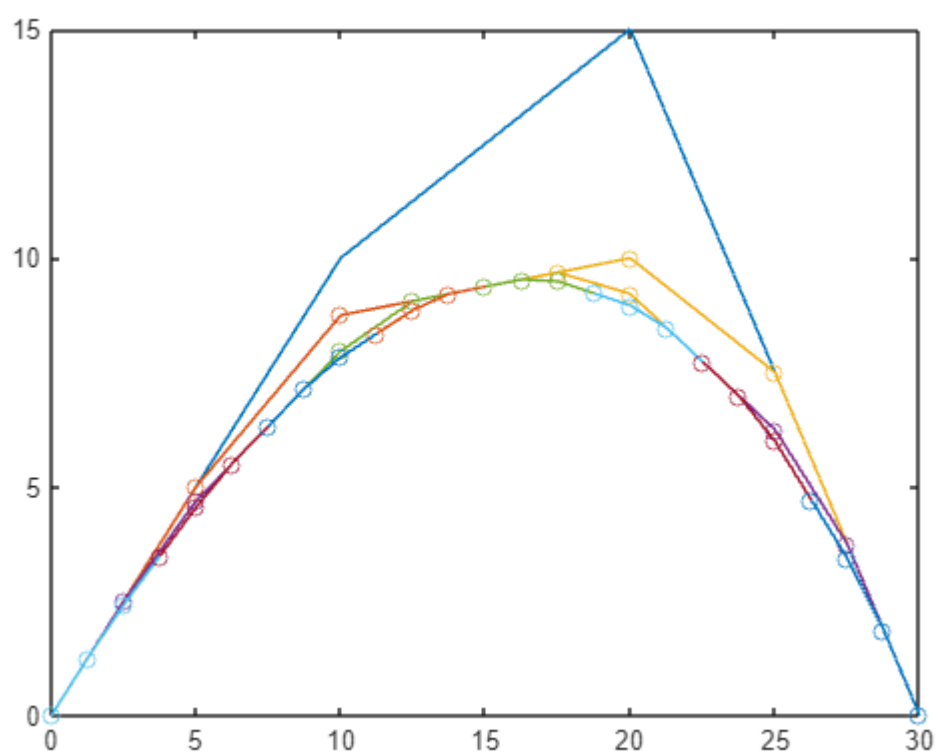
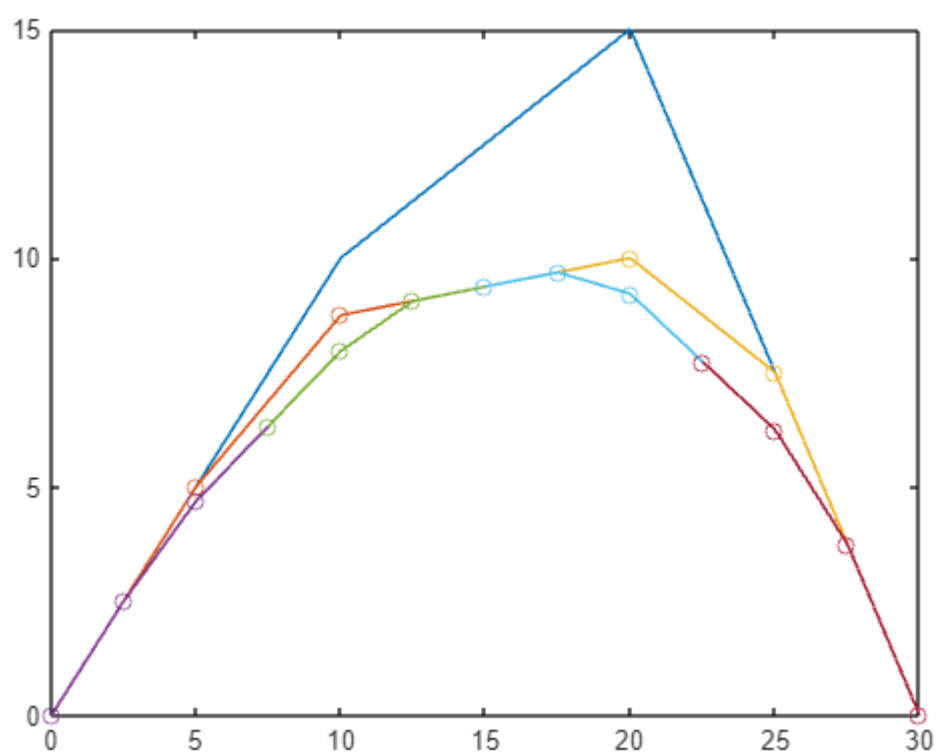
```
hold on;
```

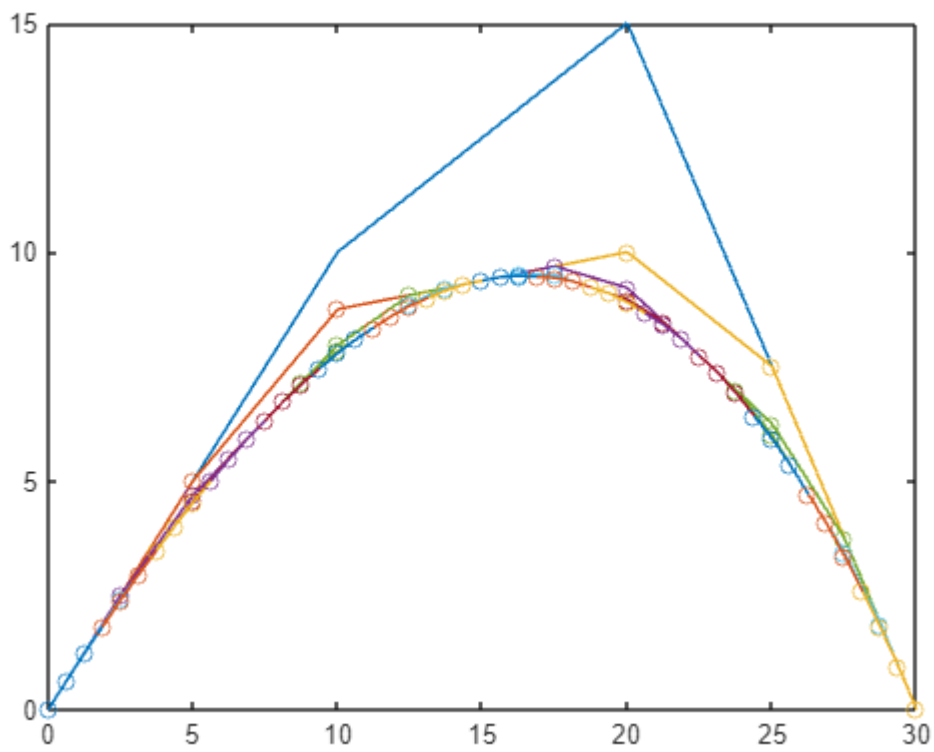
```
B = bezierCasteljau(P, n, t);
```

```
hold off;
```

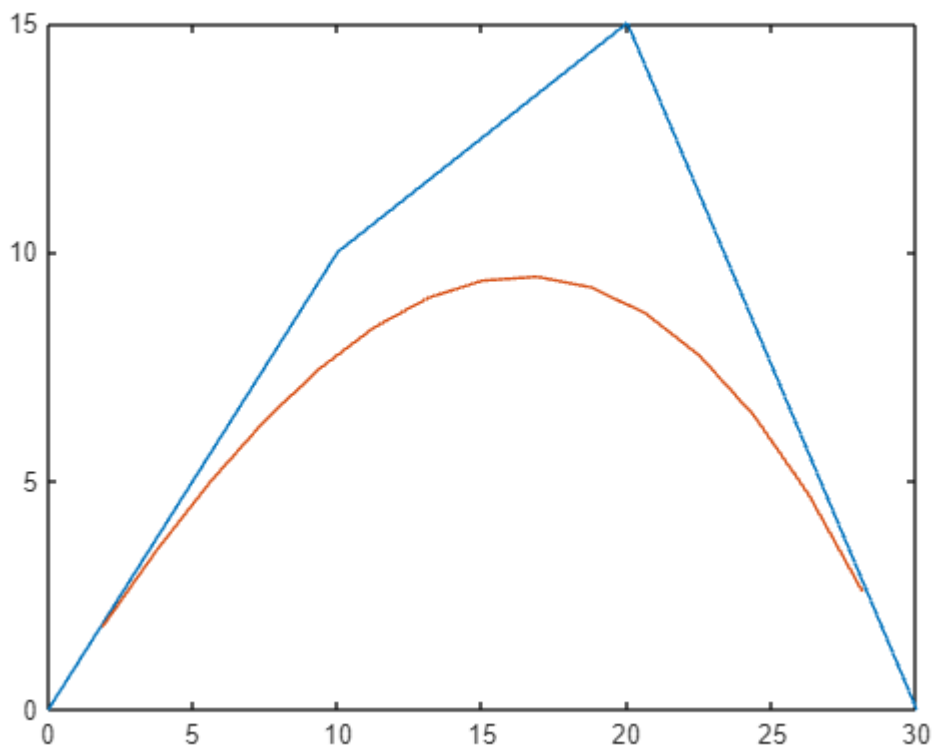
```
end
```







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



Úkol 3

Vyzkoušejte různé návaznosti Beziérových kubik.

Pomocné funkce

```
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

```