# Iskanje presecišč dveh parametrično podanih krivulj

Matematično modeliranje, 2. domača naloga

Tilen Ožbot

Maj 2022

# 1 Opis problema

Imamo krivulji K in L v ravnini $\mathbb{R}^2$. Naloga je napisati funkcijo v Octave, ki poišče vse točke, v katerih se krivulji sekata. Podani imamo pripadajoči parametrizaciji $\mathbf{p}$(t) in $\mathbf{q}$(t), pripadajoča intervala I = [a, b] in J = [c, d].

Postopek za iskanje presečišč razdelimo na 3 dele:

1. Intervala I in J razdelimo na primerno majhne podintervale dolžine h, ki je prav tako parameter funkcije.
2. Krivulji K in L aproksimiramo z lomljenkama K' in L'. Zaporedne točke na lomljenkah dobimo tako, da izračunamo vrednost parametrizacije na razdeljenih intervalih I in J.
3. Iz presečišč lomljenk dobimo parametre pri katerih se lomljenke sekajo. Ti parametri služijo kot približek za vrednosti parametrov pri katerih se krivulji K in L sekata. S pomočjo Newtonove iteracije dobimo bolj natančne koordinate presečišč.

# 2 Resitev

## 2.1 Postopek reševanja

Definiramo funkcijo "presekKrivulj", ki kot parametre sprejme parametrizaciji obeh krivulj (**p** in **q**), njihove odvode (**pdot** in **qdot**), intervala I in J (**intp** in **intq**) ter korak **h**. Parametri **p**, **q**, **pdot** in **qdot** so kazalci na funkcije.

Funkcija vrne matriki **P** (približek presečišč krivulj K in L) in **Q** (seznam presečišč lomljenk K' in L').

```
function [P,Q] = presekKrivulj(p,pdot,intp,q,qdot,intq,h)
```

Najprej razdelimo intervala I in J na dele dolžine h.

```
interval_p = [intp(1):h:intp(2)];
interval_q = [intq(1):h:intq(2)];
```

Točke za lomljenki K' in L' pridobimo tako, da v funkciji p in q kot argument pošljemo izračunane intervale.

```
K_lomljenka_tocke = p(interval_p);
L_lomljenka_tocke = q(interval_q);
```

Za izracun presečišč lomljenk si pomagamo s funkcijo "presecisca", ki kot argumente sprejme dve matriki točk, korak ter začetek intervalov I in J, ter vrne matriko **Q**, ki vsebuje presečišča ter matriko T, ki vsebuje parametre, ki ustrezajo presečiščam.

```
[Q,T] = presecisca(K_lomljenka_tocke,L_lomljenka_tocke,h,intp(1),intq(1));
```

## 2.2 Newtonova metoda

Za točnejse točke presečišč uporabimo Newtonovo metodo. Newtonova metoda je numerična metoda za iskanje ničel funkcije. Ideja metode je naslednja:

- $x_0$ je približek za ničlo funkcije $f(x)$,
- naslednji približek $x_1$ za ničlo je ničla tangente na graf funkcije v točki $x_0$,
- postopek se nadaljuje in vrednosti (v večini primerov) konvergirajo k ničli funkcije $f(x)$.

Metodo lahko povzamemo v enačbi:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Če metodo posplošimo na $k$ spremenljivk in $k$ enačb:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_f(x_n)^{-1}\mathbf{F}(\mathbf{x}_n),$$

kjer je $\mathbf{J}_f(x_n)$ Jacobijeva matrika, $\mathbf{x}$ pa vektor dimenzije k.

Namesto računanja $\mathbf{J}_f(x_n)^{-1}$ in potem množenja s $\mathbf{F}(\mathbf{x}_n)$ raje rešimo linearni sistem in s tem pridobimo na hitrosti:

$$\mathbf{J}_f(x_n)\vec{y} = \mathbf{F}(\mathbf{x}_n).$$

Rešitev linearne sistema je enaka $\mathbf{J}_f(x_n)^{-1}\mathbf{F}(\mathbf{x}_n)$.

## 2.3 Newtonova iteracija v Octave

Za natančnejse točke presečišč potrebujemo natančnejse parametre. Te pridobimo z Newtonovo iteracijo in sicer kot ničle določene funkcije F, ki jo pridobimo tako, da odštejemo podani funkciji.

Naprimer: t $\rightarrow \begin{bmatrix} t \\ cos(t) \end{bmatrix}$, u $\rightarrow \begin{bmatrix} u \\ sin(u) \end{bmatrix}$, F $\rightarrow \begin{bmatrix} t-u \\ cos(t)-sin(u) \end{bmatrix}$

V Octave:

```
F = @(x) [p(x(1))(1) - q(x(2))(1); p(x(2))(2) - q(x(1))(2)];
```

Jacobijeva matrika ima obliko:

$$\mathbf{J_f} = \begin{bmatrix} \dfrac{\partial F_1}{\partial x_1} & \dfrac{\partial F_1}{\partial x_2} \\ \dfrac{\partial F_2}{\partial x_1} & \dfrac{\partial F_2}{\partial x_2} \end{bmatrix}.$$

Za zgorjni primer (odvode imamo podane):

$$\mathbf{J_f} = \begin{bmatrix} 1 & -1 \\ -\sin t & -\cos u \end{bmatrix}.$$

V Octave:

```
JF = @(x) [pdot(x(1))(1),-qdot(x(2))(1);pdot(x(1))(2),-qdot(x(2))(2)];
```

Za vsak parameter pokličemo Newtonovo metodo in dobljeni natančnejši parameter podamo v eno od funkcij p ali q ter rezultat (koordinate točke) shranimo v matriko P.

```
st_parametrov = length(T);
P = zeros(2, st_parametrov);
for i=1:st_parametrov
  P(:, i) = p(newton(F, JF, T(:, i))(1));
endfor
```

Vse uporabljene funkcije najdemo v 3. poglavju poročila.

## 3 Primeri

### 3.1 Primer 1

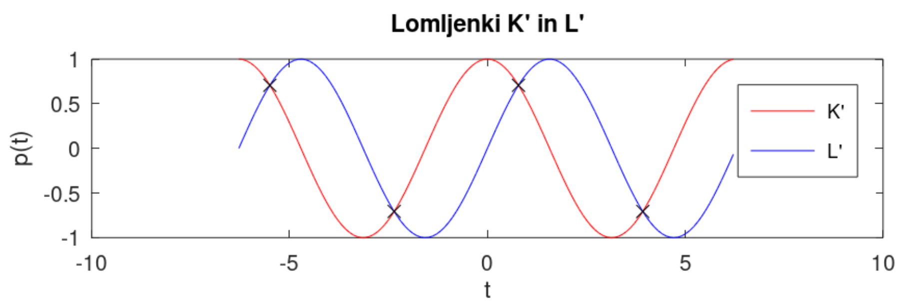
p(t) = $\begin{bmatrix} t \\ \cos t \end{bmatrix}$, q(u) = $\begin{bmatrix} u \\ \sin u \end{bmatrix}$,

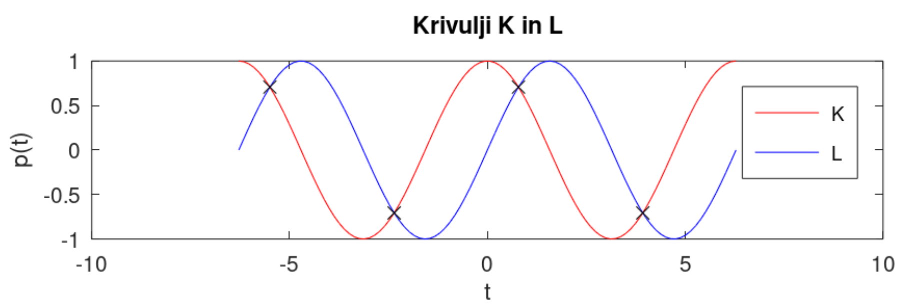
p'(t) = $\begin{bmatrix} 1 \\ -\sin p \end{bmatrix}$, q'(u) $\to \begin{bmatrix} 1 \\ \cos u \end{bmatrix}$,

I = $[-2\pi, 2\pi]$, J = $[-2\pi, 2\pi]$, h = 0.1

F = $\begin{bmatrix} t - u \\ \cos t - \sin u \end{bmatrix}$, $J_f = \begin{bmatrix} 1 & -1 \\ -\sin t & -\cos u \end{bmatrix}$

P = $\begin{bmatrix} -5.4978 & -2.3562 & 0.7854 & 3.9270 \\ 0.7071 & -0.7071 & 0.7071 & -0.7071 \end{bmatrix}$, Q = $\begin{bmatrix} -5.4978 & -2.3562 & 0.7854 & 3.9270 \\ 0.7067 & -0.7064 & 0.7063 & -0.7068 \end{bmatrix}$




### 3.2 Primer 2

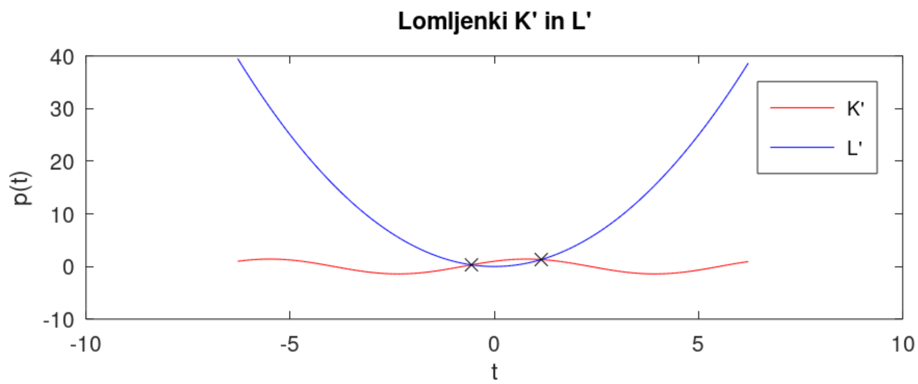
p(t) = $\begin{bmatrix} t \\ \cos t + \sin t \end{bmatrix}$, q(u) = $\begin{bmatrix} u \\ u^2 \end{bmatrix}$,

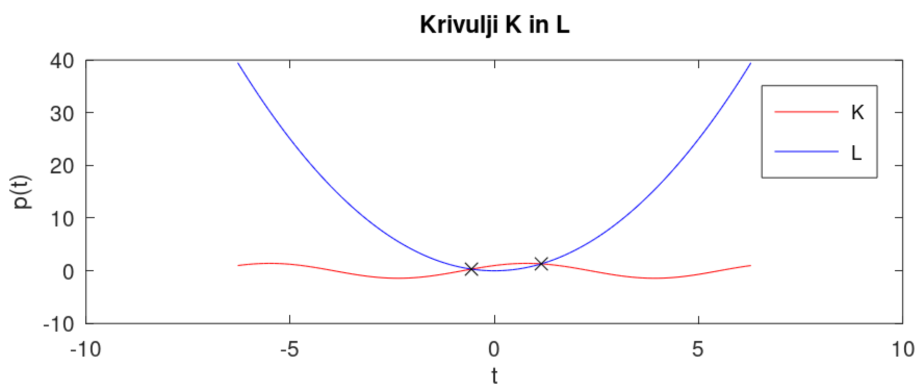
p'(t) = $\begin{bmatrix} 1 \\ \cos t - \sin t \end{bmatrix}$, q'(u) $\to \begin{bmatrix} 1 \\ 2u \end{bmatrix}$,

I = $[-2\pi, 2\pi]$, J = $[-2\pi, 2\pi]$, h = 0.1

F = $\begin{bmatrix} t - u \\ \cos t + \sin t - u^2 \end{bmatrix}$, $J_f = \begin{bmatrix} 1 & -1 \\ \cos u - \sin t & -2u \end{bmatrix}$

P = $\begin{bmatrix} -0.5610 & 1.1496 \\ 0.3147 & 1.3215 \end{bmatrix}$, Q = $\begin{bmatrix} -0.5602 & 1.1483 \\ 0.3155 & 1.3207 \end{bmatrix}$




### 3.3 Primer 3

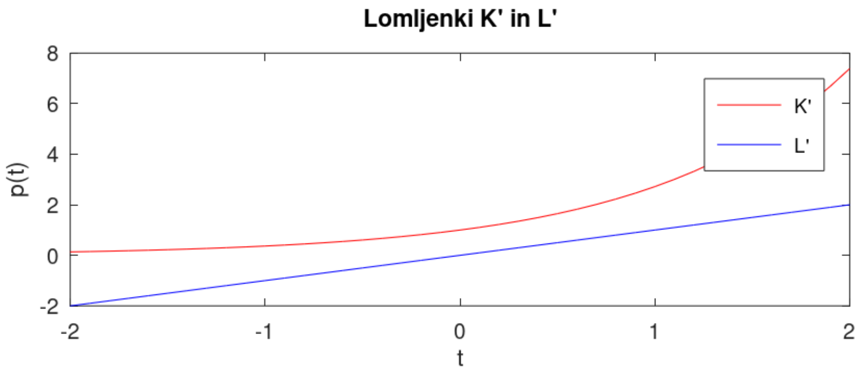
p(t) = $\begin{bmatrix} t \\ e^t \end{bmatrix}$, q(u) = $\begin{bmatrix} u \\ u \end{bmatrix}$,

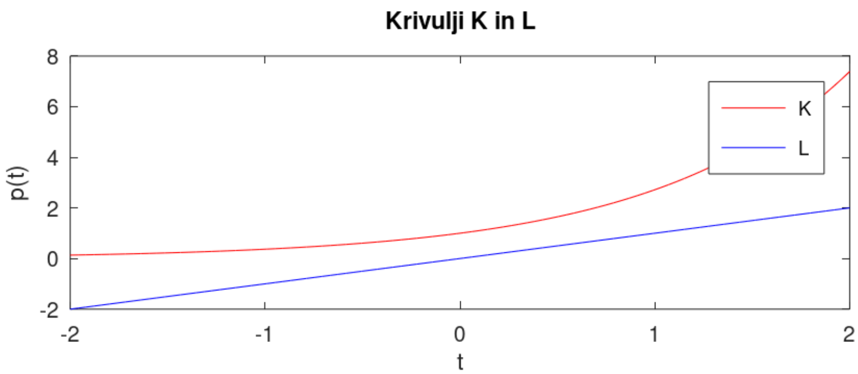
p'(t) = $\begin{bmatrix} 1 \\ e^t \end{bmatrix}$, q'(u) $\rightarrow$ $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$,

I = $[-2, 2]$, J = $[-2, 2]$, h = 0.1

F = $\begin{bmatrix} t - u \\ e^t - u \end{bmatrix}$, $J_f$ = $\begin{bmatrix} 1 & -1 \\ e^t & -1 \end{bmatrix}$

P = [   ], Q = [   ], se ne sekata




# 4 Octave funkcije

Funkcije presecisca, presecisce ter newton so gradivo pridobljeno iz spletne ucilnice.

## 4.1 presekKrivulj

```
function [P, Q] = presekKrivulj(p, pdot, intp, q, qdot, intq, h)
% avtor: Tilen Ožbot
% function [P, Q] = presekKrivulj(p, pdot, intp, q, qdot, intq, h)
% PARAMETRI:
% p in q sta kazalec na funkciji, ki opisujeta dve ravninsko krivulji,
% pdot in qdot sta kazalca na odvode funkcij p in q
% intp in intq sta intervala, na katerem sta parametrizirani krivulju
% h je dolzina podintervalov, na katere sta razdeljena intp in intq
% VRNJENE VREDNOSTI:
% P je seznam presecisc K in L
% Q je seznam presecisc lomljenk K' in L'
```

```
% Razdelimo interval intp na podintervale dolzine h
interval_p = [intp(1):h:intp(2)];
interval_q = [intq(1):h:intq(2)];

% Izracunamo tocke za lomljenki
K_lomljenka_tocke = p(interval_p);
L_lomljenka_tocke = q(interval_q);

% Izracunamo tocke za krivulji K in L na podanem intervalu
K = p(linspace(intp(1), intp(2)));
L = q(linspace(intq(1), intq(2)));

% Q so presecisca lomljenk, T so parametri pri katerih se lomljenke sekajo
[Q, T] = presecisca(K_lomljenka_tocke, L_lomljenka_tocke, h, intp(1), intq(1));

% Narisemo graf lomljenke K' in L'
subplot(2, 1, 1)
plot(K_lomljenka_tocke(1, :), K_lomljenka_tocke(2, :), 'r')
hold on
plot(L_lomljenka_tocke(1, :), L_lomljenka_tocke(2, :), 'b')
hold on
plot(Q(1, :), Q(2, :), 'x', 'color', 'k')
xlabel("t")
ylabel("p(t)")
legend("K'", "L'")
title("Lomljenki␣K'␣in␣L'")
%T = T(1, :)

% pripravimo matrike za Newtonovo metodo
F = @(x) [p(x(1))(1) - q(x(2))(1); p(x(2))(2) - q(x(1))(2)];

% pripravimo Jacobijevo matriko
JF = @(x) [pdot(x(1))(1) , -qdot(x(2))(1); pdot(x(1))(2), -qdot(x(2))(2)];

st_parametrov = length(T);
P = zeros(2, st_parametrov);

% Vsak parameter v matriki T uporabimo kakor zacetni priblizek za
% Newtonovo iteracijo in rezultat vnesemo v eno od funkcij (p ali q) ter
% dobljeno tocko shranimo v matriko P
for i=1:st_parametrov
  P(:, i) = p(newton(F, JF, T(:, i))(1));
endfor

% Narisemo graf za krivulji K in L
hold on
subplot(2, 1, 2)
plot(K(1, :), K(2, :), 'r')
hold on
```

```
plot(L(1, :), L(2, :), 'b')
hold on
plot(P(1, :), P(2, :), 'x', 'color', 'k')
xlabel("t")
ylabel("p(t)")
legend("K", "L")
```

### 4.2 presecisca

```
function [P, T] = presecisca(A, B, h = 1, a = 0, c = 0)
%[P, T] = presecisca(A, B) poisce presecisca
%P = [P1, P2,... , Pm] lomljenk
%A = [A1, A2, ..., Ak] in B = [B1, B2, ..., Bl].
%(Privzetek: Daljice so v genericni legi - presecisca so transverzalna.)
%Opcijski parametri so korak h ter zacetni tocki intervalov a in c
%za funkcijo presekKrivulj.

k = length(A);
l = length(B);
P = zeros(2, 0);
T = zeros(2, 0);

%poiscemo presecisca vseh parov daljic
for i = 1:(k - 1) % zanka po daljicah iz prve lomljenke (A)
        for j = 1:(l - 1) % zanka po daljicah iz druge lomljenka (B)
                [Q, U] = presecisce(A(:,[i i+1]), B(:, [j j+1]));
                %Ce se daljici sekata, presecisce dodamo v nabor presecisc P ...
                P = [P, Q];
                %... in dodamo se vrednost parametra, pri katerem se sekata, v T.
                if(length(U) != 0)
                        T = [T, U*h + [i*h + a; j*h + c]];
                end
        end
end
```

### 4.3 presecisce

```
function [P, T] = presecisce(A, B)
%[P, T] = presecisce(A, B) vrne presecisce daljic (oz. prazen stolpec,
%ce se daljici ne sekata) A1A2 in B1B2 v ravnini.
%A = [A1, A2], A1 in A2 sta krajevna vektorja tock na prvi daljici.
%B = [B1, B2], B1 in B2 sta krajevna vektorja tock na drugi daljici.
%P... krajevni vektor (stolpec) presecisca daljic.
%T = [t; u]... razmerji med razdaljo presecisca
%do zacetne tocke daljice in celotno dolzino daljice.

%Resimo izpeljani sistem enacb za parametra presecisca obeh premic nosilk.
T = [A(:, 2) - A(:, 1), -(B(:, 2) - B(:, 1))]\(B(:, 1) - A(:, 1));
%Preverimo, ce sta parametra znotraj intervala [0, 1] in ...
if(T(1) <= 1 && T(2) <= 1 && T(1) >= 0 && T(2) >= 0)
```

```
        %... vrnemo presecisce, ce sta, oziroma ...
        P = A(:, 1) + T(1)*(A(:, 2) - A(:, 1));
else
        %... vrnemo prazen stolpec, ce nista.
        P = zeros(2, 0);
        T = zeros(2, 0);
end
```

### 4.4 newton

```
function [X, n] = newton(F, JF, X0, tol = 1e-10, maxit = 100)
%X = newton(F, JF, X0, tol, maxit) solves the (nonlinear)
%system F(X) = 0 using the Newton's iteration with initial
%guess X0. (JF is the Jacobi matrix of F.)

for n = 1:maxit
        %Execute one step of Newton's iteration...
        X = X0 - feval(JF, X0)\feval(F, X0);
        %... and check if the new approximation is within prescribed tolerance.
        if(norm(X - X0) < tol)
                break;
        end
        X0 = X;
end

%A warning in case the last approximation is not within specified tolerance.
if(n == maxit)
        warning("no␣convergence␣after␣maxit␣iterations")
end
```