Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Geometrijsko zvezna gibanja togih teles

Matic Oskar Hajšen in Eva Zmazek

Ljubljana, 2019

Kazalo

Listings

1 Uvod

Z najino seminarsko bova prikazala, kako se da znanje, pridobljeno pri tem predmetu, uporabiti pri upodobitvi gibanja togih teles, ki se uporabljajo pri računalniških animacijah in v robotiki. Za opis teh gibanj bomo uporabljali kvaternione in bezierjeve krivulje na kvaternionih.

2 Teoretično ozadje

2.1 Homogene in kartezične koordinate

Imejmo vektor p v 3-dimenzionalnem prostoru s homogenimi koordinatami $p = (p_0, p_1, p_2, p_3)^T \in \mathbb{R}^4/\{(0,0,0,0)^T\}$. Če je prva komponenta p_0 neničelna, lahko za točko p definiramo prirejene kartezične koordinate $\underline{p} = (\underline{p_1}, \underline{p_2}, \underline{p_3})^T \in \mathbb{R}^3$, pri čemer velja $\underline{p_i} = \frac{p_i}{p_0}$ za $i = 1, 2, 3$. Na tak način vektorja p in λp opisujeta isto točko $\underline{p}$ za poljubno neničelno realno število λ. Vektorjem z ničelno prvo komponento priredimo točke v neskončnosti.

2.2 Zveza med koordinatami točk v fiksnem koordinatnem sistemu in koordinatami točk v gibajočem se koordinatnem sistemu

Definirajmo dva koordinatna sistema v $\mathbb{R}^3$:

- fiksen koordinatni sistem E^3 (običajen koordinatni sistem)
- gibajoč se koordinatni sistem $\hat{E}^3$

Točke lahko predstavimo v enem ali drugem.

Označimo s $\underline{p}$ točko glede na fiksen koordinatni sistem E^3, s $\underline{\hat{p}}$ pa glede na $\hat{E}^3$. Potrebujemo koordinatno transformacijo

$$\hat{E}^3 \to E^3$$

$$\underline{\hat{p}} \mapsto \underline{p}$$

Z uporabo homogenih koordinat, lahko transformacijo zapišemo s pomočjo matrike

$$M = \left[\begin{array}{c|ccc} m_{0,0} & 0 & 0 & 0 \\ \hline m_{1,0} & m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,0} & m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,0} & m_{3,1} & m_{3,2} & m_{3,3} \end{array}\right],$$

kjer velja $m_{0,0} \neq 0$. Preslikavo v homogenih koordinatah lahko torej zapišemo kot:

$$\underline{\hat{p}} \mapsto \underline{p} = M\underline{\hat{p}}$$

Vektorju $c = M(1,0,0,0)^T = (m_{0,0}, m_{1,0}, m_{2,3}, m_{3,0})^T$ zapisanemu v homogenih koordinatah pripada vektor $\underline{c} = (\frac{m_{1,0}}{m_{0,0}}, \frac{m_{2,0}}{m_{0,0}}, \frac{m_{3,0}}{m_{0,0}})^T$, zapisan v kartezičnih koordinatah.

Ta vektor opisuje položaj koordinatnega izhodišča gibajočega se koordinatnega sistema $\hat{E}^3$ glede na koordinatni sistem E^3.
3×3 matrika

$$\underline{R} = \frac{1}{m_{0,0}} \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix}$$

opisuje orientacijo gibajočega se koordinatnega sistema $\hat{E}^3$. Pravimo ji **rotacijska matrika**.
Oglejmo si, kaj naredi matrika M z vektorjem $[1, b_M, c_M, d_M]$:

$$M \cdot \begin{bmatrix} 1 \\ b_M \\ c_M \\ d_M \end{bmatrix} = \begin{bmatrix} m_{0,0} \\ m_{1,0} \\ m_{2,0} \\ m_{3,0} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ m_{1,1}b_M & m_{1,2}c_M & m_{1,3}d_M \\ m_{2,1}b_M & m_{2,2}c_M & m_{2,3}d_M \\ m_{3,1}b_M & m_{3,2}c_M & m_{3,3}d_M \end{bmatrix}$$

Dobimo vektor v homogeni obliki, ki ima na prvi komponenti vrednost $m_{0,0}$, preostale tri komponente pa predstavlja vektor

$$\begin{bmatrix} m_{1,0} \\ m_{2,0} \\ m_{3,0} \end{bmatrix} + \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \cdot \begin{bmatrix} b_M \\ c_M \\ d_M \end{bmatrix}$$

Ker je to vektor v homogeni obliki in ker je prva komponenta neničelna ($m_{0,0} \neq 0$), je njemu prirejen vektor v kartezični obliki enak

$$\frac{1}{m_{0,0}} \begin{bmatrix} m_{1,0} \\ m_{2,0} \\ m_{3,0} \end{bmatrix} + \frac{1}{m_{0,0}} \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{bmatrix} \cdot \begin{bmatrix} b_M \\ c_M \\ d_M \end{bmatrix},$$

ki pa je enak vsoti $\underline{c} + R \cdot \underline{\hat{p}}$
Transformacijo $\underline{\hat{p}} \mapsto \underline{p}$ v kartezičnih koordinatah zapišemo kot:

$$\underline{p} = \underline{c} + R\underline{\hat{p}}$$

2.3 Gibanje točk v času

Kadar je $\underline{c} = \underline{c}(t)$ in $R = R(t)$, govorimo o gibanju togega telesa:

$$\hat{E}^3 \times I \to E^3$$

$$(\underline{\hat{p}}, t) \mapsto \underline{c}(t) + R(t)\underline{\hat{p}} =: \underline{p}(t)$$

Krivulji $\underline{p}(t)$ pravimo **trajektorija** točke $\underline{\hat{p}}$

Če je $\underline{c}(t) = (0, 0, 0)$, potem trajektorija poljubne točke $\underline{\hat{p}}$ leži na sferi z radijem $||\underline{\hat{p}}||$ in središčem v koordinatnem izhodišču fiksnega koordinatnega sistema E^3. Rotacijski del gibanja $R(t)$ opisuje gibanje po enotski sferi, zato se imenuje tudi **sferični del gibanja togega telesa**. Problem je konstrukcija matrike R, ki mora biti ortogonalna. ($RR^T = R^T R = I$, $\det R = 1$).

2.4 Opis rotacij s kvaternioni

Pri opisovanju rotacij si lahko pomagamo s **kvaternioni**. Prostor kvaternionov $\mathbb{H}$ je 4-dimenzionalni vektorski prostor s standardno bazo

$$\underline{1} = (1, (0, 0, 0)^T)$$

$$\underline{i} = (0, (1, 0, 0)^T)$$
$$\underline{j} = (0, (0, 1, 0)^T)$$
$$\underline{k} = (0, (0, 0, 1)^T)$$

Vsak kvaternion $\mathcal{A}$ lahko zapišemo kot:

$$\mathcal{A} = (a_0, \underline{a}),\ a_0 \in \mathbb{R} \text{ skalarni del },\ \underline{a} = (a_1, a_2, a_3)^T \text{ vektorski del}$$

Na kvaternionih sta definirana seštevanje in množenje kot:

$$\mathcal{A} + \mathcal{B} = (a_0, \underline{a}) + (b_0, \underline{b}) = (a_0 + b_0, \underline{a} + \underline{b})$$

$$\mathcal{A} \cdot \mathcal{B} = (a_0 \cdot b_0 - \underline{a} \cdot \underline{b}, a_0\underline{b} + b_0\underline{a} + \underline{a} \times \underline{b})$$

Konjugirana vrednost kvaretniona $\mathcal{A} = (a_0, \underline{a})$ je definirana kot $\overline{\mathcal{A}} = (a_0, -\underline{a})$.

Definicija 2.1. Preslikava $\chi : \mathbb{H}\backslash\{0\} \to SO_3$ oblike

$$Q \mapsto \frac{1}{q_0^2 + q_1^2 + q_2^2 + q_3^2} \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_2) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

$$Q = (q_0, (q_1, q_2, q_3)^T)$$

se imenuje **kinematična preslikava**.

Matika $\chi(Q)$ je rotacijska matrika. Velja pa tudi obratno. Vsako rotacijsko matriko R lahko zapišemo v zgornji obliki, to je, lahko jo preslikamo v dva **antipodna kvaterniona** oblike

$$\pm Q = \pm(q_0, (q_1, q_2, q_3)^T),$$
$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$$

Kinematična preslikava poda korespondenco med 3D rotacijami in parom antipodnih točk na 4D enotski sferi $S^3 \subseteq R^4$.

Ker velja $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$, so vrednosti $|q_i|;\ i = 0, 1, 2, 3$ na zaprtem interalu med 0 in 1. Vrednost q_0 in vektor $(q_1, q_2, q_3)^T$ lahko zato zapišemo v obliki:

$$q_0 = \cos(\frac{\phi}{2})$$

in

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \sin(\frac{\phi}{2}) \cdot \vec{r};\ \ \vec{r} \text{ enotski vektor}$$

Če kvaternion Q zapišemo v tej obliki, ima rotacija, prirejena temu kvaternionu lepo geometrijsko interpretacijo. Predstavlja namreč rotacijo za kot ϕ okrog osi $\vec{r}$.

Ker lahko vsako rotacijo zapišemo v tej obliki, lahko tako zapišemo tudi rotacijo iz poglavja 2.2. Če imamo podano preslikavo M, poiščimo, kako za to preslikavo definiramo kvaternion Q. Priemerjajmo matriki $\mathbb{R}$ in poglavja 2.2 in $\mathbb{R}$, zapisanega s kvaternioni.

$$\begin{aligned} m_{0,0} + m_{1,1} + m_{1,2} + m_{3,3} &= 4q_0^2 \\ m_{3,2} - m_{2,3} = 2 \cdot (q_2 q_3 - q_0 q_1) - 2 \cdot (q_2 q_3 + q_0 q_1) &= 4q_0 q_1 \\ m_{1,3} - m_{3,1} = 2 \cdot (q_1 q_3 + q_0 q_2) - 2 \cdot (q_1 q_3 + q_0 q_2) &= 4q_0 q_2 \\ m_{2,1} - m_{1,2} = 2 \cdot (q_1 q_2 + q_0 q_4) - 2 \cdot (q_1 q_2 + q_0 q_4) &= 4q_0 q_3 \end{aligned}$$

$$\begin{aligned} m_{3,2} - m_{2,3} &= 4q_0 q_1 \\ m_{0,0} + m_{1,1} - m_{1,2} - m_{3,3} &= 4q_1^2 \\ m_{2,1} + m_{1,2} &= 4q_1 q_2 \\ m_{1,3} + m_{3,1} &= 4q_1 q_3 \end{aligned}$$

$$\begin{aligned} m_{1,3} - m_{3,1} &= 4q_0 q_2 \\ m_{2,1} + m_{1,2} &= 4q_1 q_2 \\ m_{0,0} - m_{1,1} + m_{1,2} - m_{3,3} &= 4q_2^2 \\ m_{3,2} + m_{2,3} &= 4q_2 q_3 \end{aligned}$$

$$\begin{aligned} m_{2,1} - m_{1,2} &= 4q_0 q_3 \\ m_{1,3} + m_{3,1} &= 4q_1 q_3 \\ m_{3,2} + m_{2,3} &= 4q_2 q_3 \\ m_{0,0} - m_{1,1} - m_{1,2} + m_{3,3} &= 4q_0^2 \end{aligned}$$

Opazimo, da v vsakem sklopu razmerja med vrednostmi enaka

$$q_0 : q_1 : q_2 : q_3$$

Ker q_0, q_1, q_2, q_3 niso hkrati enaki 0, bo vsaj eno izmed zgornjih razmerij različno od $0 : 0 : 0 : 0$. Tisto razmerje nato uporabimo kot razmerje $q_0 : q_1 : q_2 : q_3$. Skupaj z enakostjo $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$ nato izračunamo kvaternion $Q = (q_0, q_1, q_2, q_3)^T$ (bolj natančno sta v množici rešitev dva antipodna kvaterniona).

2.5 Bezierjeve krivulje

Z uporabo kinematične preslikave lahko za konstrukcijo sferičnih gibanj uporabimo Bezierjeve krivulje. Izberemo kontrolne kvaternione $Q_0, Q_1, \ldots, Q_n$.

$$Q(t) = \sum_{i=0}^{n} Q_i B_i^n(t)$$

Bezierjeva krivulja $Q(t)$ v času t opiše kvaternion, ki mu priredimo rotacijo $R(t)$:

$$\chi(Q(t)) = R(t)$$

Rotacija, ki je določena z Bezierjevo krivuljo $Q(t) = \sum\limits_{i=0}^{n} Q_i B_i^n(t)$ stopnje n, je sferično razionalno gibanje stopnje $2n$.

Gibanje koordinatnega izhodišča zapišemo v obliki

$$\underline{c}(t) = \frac{w(t)}{||Q(t)||^2};\ w(t) := (w_1(t), w_2(t), w_3(t)).$$

3 Implementacija

3.1 Kvaternioni

Ker si pri opisovanju rotacij pomagamo s kvaternioni, sva na kvaternionih definirala naslednje funkcije:

Listing 1: array2quat

```
function Q = array2quat(a, b, c, d)
%ARRAY2QUAT prejme 4 parametre, ki jih pretvori v kvaternion (
    vektorsko obliko)
%input:
%a, b, c, d      komponente
%output:
%Q               kvaternion

Q = [a, b, c, d];
end
```

Listing 2: quat_vec

```
function v = quat_vec(Q)
%QAUD_VEC poda vektorski del kvaterniona Q
%input:
%Q               kvaternion (q0, q1, q2, q3)
%output:
%v               vektorski komponenta Q—ja, (q1, q2, q3)

v = [Q(2) Q(3) Q(4)];
end
```

Listing 3: quatmultiply

```
function c = quatmultiply(a,b)
%QUATMULTIPLY izracuna produkt dveh kvaternionov, kot je opisano v
    clanku
%
% Input:
% a ... prvi kvaternion
% b ... drugi kvaternion
%
% Output:
% c ... produkt (nov kvaternion)

c = zeros(1,4);
a_s = a(1);
b_s = b(1);

```

```
a_v = quat_vec(a);
b_v = quat_vec(b);

c(1) = a_s * b_s − a_v * b_v';
c_v = a_s * b_v + b_s * a_v + cross(a_v,b_v);

c(2) = c_v(1);
c(3) = c_v(2);
c(4) = c_v(3);
end
```

Listing 4: conj_quat

```
function Q = conj_quat(q)
%CONJ_QUAT vrne konjugirano vrednost podanega kvaterniona q
%input:
%q                kvaternion
%output:
%Q                konjugiran qvaternion

Q = [q(1), −q(2:4)];
end
```

Listing 5: quat_exp

```
function e = quat_exp(q, t)
%QUAT_EXP vrne potenco z osnovo q (kvaternion) in eksponentom t
%input:
% q               kvaternion [a, b, c, d]
% t               eksponent
%output:
% e               rezultat q^t

if t==−1
    e = conj_quat(q)/norm(q);
else
    a = q(1);
    v = quat_vec(q);
    theta = acos(a/norm(q));
    n = v/norm(v);

    e = norm(q)^t*[cos(t*theta), n*sin(t*theta)];
end
```

Listing 6: quat_rot_mat

```
function H = quat_rot_mat(Q)
%QUAT_ROT_MAT oblikuje rotacijsko matriko, prirejeno kvaternionu Q
```

```
%input:
%Q          kvaternion
%
%output:
%H          rotacijska matrika za sfericno gibanje

H = zeros(3,3);
h = sum(Q.^2);

if h == 0
    H = eye(3,3);
else
    H(1,1) = Q(1)^2+Q(2)^2 - Q(3)^2 - Q(4)^2;
    H(1,2) = 2*(Q(2)*Q(3) - Q(1)*Q(4));
    H(1,3) = 2*(Q(2)*Q(4) + Q(1)*Q(3));

    H(2,1) = 2*(Q(2)*Q(3) + Q(1)*Q(4));
    H(2,2) = Q(1)^2 - Q(2)^2 + Q(3)^2 - Q(4)^2;
    H(2,3) = 2*(Q(3)*Q(4) - Q(1)*Q(2));

    H(3,1) = 2*(Q(2)*Q(4) - Q(1)*Q(3));
    H(3,2) = 2*(Q(3)*Q(4) + Q(1)*Q(2));
    H(3,3) = Q(1)^2 - Q(2)^2 - Q(3)^2 + Q(4)^2;

    H = 1/h.*H;
end

end
```

3.2 Kocka

Listing 7: kocka

```
function oglisca = kocka(T0, T1)
%Vrne kocko, definirano z diagonalo (T0,T1)
a = [T1(1) - T0(1) 0 0];
b = [0 T1(2) - T0(2) 0];
c = [0 0 T1(3) - T0(3)];

oglisca = [T0; T0+b; T0+a+b; T0+a; T0+c; T0+c+b; T0+a+b+c; T0+a+c];
%ploskve = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
end
```

Listing 8: kocka_vek

```
function oglisca = kocka_vek(X, Y, Z, T0)
%input:
```

```
% x,y,z          vektorji stranic x,y,z stranic
% T0             izhodisce
%output:
% oglisca        vsa oglisca kocke, ki se zacne v T0

oglisca = [T0; T0+Y; T0+X+Y; T0+X; T0+Z; T0+Z+Y; T0+X+Y+Z; T0+X+Z];
%ploskve = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
end
```

Listing 9: kot_v_kvat

```
function r = kot_v_kvat(fi, e)
%input:
% fi             kot
% e              vektor osi
%output:
% Q              kvaternion
%
% sprejme kot fi in os e ter vrne kvaternion, ki predstavlja rotacijo
% za kot fi okoli osi e

e = e/norm(e);
r = [cos(fi/2) sin(fi/2)*e(1) sin(fi/2)*e(2) sin(fi/2)*e(3)];
end
```

Listing 10: risi_kocko

```
function risi_kocko(K, barva)
% sprejme koordinate oglisc K (dobljene iz funkcije kocka) in barvo
% ter jo narise v figuro

ploskve = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
patch('Vertices', K, 'Faces', ploskve, 'FaceColor', barva);

end
```

```
function K1 = rotiraj_kocko(K0, kot_x, kot_y, kot_z)
%zarotira kocko K0 za podane kote

K1 = zeros(size(K0));
for i = 1:8
    K1(i,:) = quat_vec(rot_vek_za_kot(K0(i,:), kot_x, kot_y, kot_z));
end
end
```

Listing 11: rotirana_kocka

```
function [K0, x0, y0, z0] = rotirana_kocka(x,y,z,Q)
```

```

H = quat_rot_mat(Q);
x0 = (H*x')';
y0 = (H*y')';
z0 = (H*z')';
K0 = kocka_vek(x0,y0,z0, [0 0 0]);

end
```

3.3 Razvrsti

```
function b = bezier (B,t)
% Opis :
% bezier vrne tocke na Bezierjevi krivulji pri danih
% parametrih
%
% Definicija :
% b = bezier (B,t)
%
% Vhodna podatka :
% B     matrika velikosti n+1 x d, ki predstavlja kontrolne
%       tocke Bezierjeve krivulje stopnje n v
%       d- dimenzionalnem prostoru ,
% t     seznam parametrov dolzine k, pri katerih racunamo
%       vrednost Bezierjeve krivulje
%
% Izhodni podatek :
% b     matrika velikosti k x d, kjer i-ta vrstica
%       predstavlja tocko na Bezierjevi krivulji pri
%       parametru iz t na i- tem mestu

[n,d] = size(B);
k = length(t);
b = zeros(k,d);

for i=1:k
    for j=1:d
        D = decasteljau(B(:,j)',t(i));
        b(i,j) = D(1,n);
    end
end
```

```
function D = decasteljau (b,t)
% Opis :
% decasteljau vrne shemo de Casteljaujevega postopka za dan
```

```
% seznam koordinat b pri danem parametru t
%
% Definicija :
% D = decasteljau (b,t)
%
% Vhodna podatka :
% b      seznam koordinat kontrolnih tock Bezierjeve krivulje
%        stopnje n,
% t      parameter , pri katerem racunamo koordinato
%        Bezierjeve krivulje
%
% Izhodni podatek :
% D      tabela velikosti n+1 x n+1, ki predstavlja de
%        Casteljaujevo shemo za koordinate b pri parametru t
%        ( element na mestu (1,n +1) je koordinata Bezierjeve
%        krivulje pri parametru t, elementi na mestih (i,j)
%        za i > n−j+2 so NaN )

n = length(b);
D = [b', NaN(n,n−1)];

for r=1:n
    for i=0:n−r−1
        D(i+1,r+1) = (1−t)∗D(i+1,r) + t∗D(i+2,r);
    end
end
```

```
function  plot_kontrolne_kocke(x0,y0,z0,T0, B,c, zac_barva,
    vmes_barva, kon_barva, pavza)
%input:
% x,y,z               vektorji, ki dolocajo zacetno kocko
% T0                  izhodidce kocke
% B                   matrika n x 4, v kateri so kontrolni kvaternioni
    kot
%                     vrstice
% c                   translacijska funkcija
% barve               kake barve naj bodo kontrolni kvadri
% pavza               ali naj pavzira vmes
%output:
% narise kontroln
    kvadre − to so kvadri, ki bi jih dobili s premikanjem
% osnovnega z hi(Q_i), kjer je Q_i kontrolni kvaternion


for i = 1:size(B,1)
    if pavza
        pause
```

```
    end
    switch i
        case 1
            barva = zac_barva;
        case size(B,1)
            barva = kon_barva;
        otherwise
            barva = vmes_barva;
    end
    Q = B(i,:);
    H = quat_rot_mat(Q);
    x = (H*x0')';
    y = (H*y0')';
    z = (H*z0')';
    T = T0+c(i,:);
    risi_kocko(kocka_vek(x, y, z, T),barva);
end
```

```
function plot_tirnice(P, c)
%input:
% P                 premaknjeni vektorji kvadra
% c                 translacijska funkcija
%ouput:
% narise tirnice, po katerih se premikajo oglisca kvadra

n = size(c, 1);

%en vektor
V_x = zeros(n,3);
V_y = zeros(n,3);
V_z = zeros(n,3);
V_xy = zeros(n,3);
V_zy = zeros(n,3);
V_xz = zeros(n,3);
V = zeros(n,3);


for i = 1:n
    Pi = P{i};
    V_x(i,:) = Pi(1,:);
    V_y(i,:) = Pi(2,:);
    V_z(i,:) = Pi(3,:);
end

% dva vektorja
V_xy = V_x + V_y + c;
```

```
V_zy = V_z + V_y + c;
V_xz = V_x + V_z + c;

% vsi
V = V_x + V_y + V_z + c;

V_x = V_x + c;
V_y = V_y + c;
V_z = V_z + c;


plot3(V_x(:,1), V_x(:,2), V_x(:,3),...
    V_y(:,1), V_y(:,2), V_y(:,3),...
    V_z(:,1), V_z(:,2), V_z(:,3),...
    V_xy(:,1), V_xy(:,2), V_xy(:,3),...
    V_zy(:,1), V_zy(:,2), V_zy(:,3),...
    V_xz(:,1), V_xz(:,2), V_xz(:,3),...
    V(:,1), V(:,2), V(:,3),...
    c(:,1), c(:,2), c(:,3),...
    'LineWidth',1.5)


end
```

Listing 12: polepsaj_sbezier

```
function mat_Q = polepsaj_sbezier(Q)
%spremeni celico Q v matriko mat_Q

n = length(Q);
mat_Q = zeros(n,4);
for i = 1:n
    mat_Q(i,:) = Q{i}{1};
end

end
```

Listing 13: rot_vek_za_kot

```
function v = rot_vek_za_kot(vec, kot_x,kot_y,kot_z)
%input:
% vec                   3D vektor
% kot_x,y,z             koti v x,y,z osi, za katere rotiramo vektor v
%output:
% v                     rotirani vektor

q = angle2quat(kot_x, kot_y, kot_z);
v = quatmultiply(q, quatmultiply([0 vec], conj_quat(q)));
```

```
end
```

```
```

```
```

```
% definicija kocke

% STRANICE KOCKE
x0 = [1 0 0];
y0 = [0 1.5 0];
z0 = [0 0 2];
T0 = [0 0 0];
K = kocka_vek(x0,y0,z0, T0);


% KVATERNIONI ZA OBRACANJE KOCKE

% Q0 = angle2quat(0, 0, 0);
% Q1 = angle2quat(pi/2, 0, 0);
% Q2 = angle2quat(pi/2, pi/4, 0);
% Q3 = angle2quat(pi/2, pi/4, pi/3);

% KVATERNIONI ZA OBRACANJE KOCKE

Q0 = kot_v_kvat(0, [1,0,0]);
Q1 = kot_v_kvat(-pi/2, [1,0,0]);
Q2 = kot_v_kvat(-pi/2, [0,0,1]);
Q3 = kot_v_kvat(3*pi/4, [1,0,-1]);

Q = [Q0; Q1; Q2; Q3];


% PRIMER: pri temle naj bi sel po dolgi poti, vendar se mi zdi
%  da je brez popravka boljse (glej funkcijo slerp)
% Q0 = kot_v_kvat(0, [1,0,0]);
% Q1 = kot_v_kvat(pi/4, [1,0,0]);
% Q2 = kot_v_kvat(-pi/2, [0,0,1]);
% Q3 = kot_v_kvat(3*pi/4, [1,0,-1]);



% BEZIERJEVA KRIVULJA ZA TRANSLACIJO
b0 = [-9 -9 -9];
b1 = [6 2 5];
b2 = [-8 6 9];
b3 = [5 5 -5];
B = [b0; b1; b2; b3];
```

```

n = 15;
t = linspace(0,1,n);
os = 10;

[mat_Q, w, c] = izracunaj_vse(Q,B,t);
narisi_vse(x0,y0,z0,T0, mat_Q,c,t, os, 1, 1, 1)
%plot3(B(:,1),B(:,2),B(:,3))
%scatter3(B(:,1),B(:,2),B(:,3))

% ZLEPKI?
% to sta ze dva premika, ki se nadaljujeta
% samo C^0 zveznost

QQ = [Q3; kot_v_kvat(pi/4, [-1,0,1]); kot_v_kvat(pi/2, [1,0,0]); kot_v_kvat(pi/3, [1,0,1])];
BB = [b3; 2 3 4; 5 6 7; -5 -5 5];
[mat_QQ, ww, cc] = izracunaj_vse(QQ,BB,t);
narisi_vse(x0,y0,z0,T0, mat_QQ,cc,t, os, 1, 1, 1)
%plot3(BB(:,1),BB(:,2),BB(:,3))
%scatter3(BB(:,1),BB(:,2),BB(:,3))

QQQ = [QQ(end,:); kot_v_kvat(-pi/3, [1,0,1]); kot_v_kvat(pi/2, [0,0,1]); kot_v_kvat(-pi/2, [1,0,0])];
BBB = [BB(end,:); 0 9 0; 0 0 -9; 0 0 0];
[mat_QQQ, www, ccc] = izracunaj_vse(QQQ,BBB,t);
narisi_vse(x0,y0,z0,T0, mat_QQQ,ccc,t, os, 1, 1, 1)
%plot3(BBB(:,1),BBB(:,2),BBB(:,3))
%scatter3(BBB(:,1),BBB(:,2),BBB(:,3))

% ce zdruzimo kontrolne poligone, dobimo precej drugacne premik
% N = 5*n;
% skupaj_Q = [Q; QQ(2:end-1,:); QQQ];
% skupaj_B = [B; BB(2:end-1,:); BBB];
% [mat_skuQ, skuw, skuc] = izracunaj_vse(skupaj_Q,skupaj_B,linspace(0,1,N));
% narisi_vse(x0,y0,z0,T0, mat_skuQ,skuc,linspace(0,1,N), os, 1, 1, 1)
```

Ker sva rotacijo želela izraziti s kotom ϕ in osjo $\vec{r}$, sva definirala funkcijo, ki kot ϕ in enotski vektor e, ki leži na osi r, pretvori v ustrezen kvaternion Q.

Listing 14: For educational purposes

```
function r = kot_v_kvat(fi, e)
%input:
% fi            kot
% e             vektor osi
%output:
```

```
% Q              kvaternion
%
% sprejme kot fi in os e ter vrne kvaternion, ki predstavlja rotacijo
% za kot fi okoli osi e

e = e/norm(e);
r = [cos(fi/2) sin(fi/2)*e(1) sin(fi/2)*e(2) sin(fi/2)*e(3)];
end
```

Listing 15: For educational purposes

```
% example of while loop using placeholders
while ⟨condition⟩
  if ⟨something-bad-happens⟩
    break
  else
    % do something useful
  end
  % do more things
end
```