Eva Zmazek

PROJEKT PRI PREDMETU MATEMATIČNO MODELIRANJE

Kazalo

1 Diskretna verižnica na lihem številu členov

Naj bo diskretna verižnica sestavljena iz $2p+1$ členkov dolžin

$$L_i,\ i = 1, 2, \ldots, 2p+1$$

in mas

$$M_i,\ i = 1, 2, \ldots, 2p+1.$$

Ker je verižnica simetrična, mora veljati $L_i = L_{2p+2-i}$ ter $M_i = M_{2p+2-i}$. Zaradi tega pogoja je dovolj podati le prvih $p+1$ vrednosti (v programu L in M). Poiskati moramo koordinate krajišč

$$(x_i, y_i),\ i = 1, 2, \ldots, 2p+2,$$

kjer sta točki (x_1, y_1) in (x_{2p+2}, y_{2p+2}) predpisani vnaprej. Zaradi ravnotežnega pogoja moramo minimizirati

$$F(x, y) = \sum_{i=1}^{2p+1} M_i \frac{y_i + y_{i+1}}{2}, \tag{1}$$

saj je težišče posamezne palice na razpolovišču.

Označimo s $x_s = \frac{x_1 + x_{2p+2}}{2}$ simetrijsko os simertične verižnice. Iskane koordinate morajo zadoščati pogojem

$$(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 = L_i^2, \qquad i = 1, 2, \ldots, 2p+1, \tag{2}$$

$$x_{2p+3-i} = 2 \cdot x_s - x_i, \qquad i = 1, 2, \ldots, p+1, \tag{3}$$

$$y_{2p+3-i} = y_i, \qquad i = 1, 2, \ldots, p+1. \tag{4}$$

Vezani ekstrem (1)-(2) prevedemo na nevezanega z uvedbo Lagrangeovih multiplikatorjev. tedaj iščemo nevezani ekstram funkcije

$$G(x,y,\lambda)=\sum_{i=1}^{2p+1}\{M_i\frac{y_i+y_{i+1}}{2}+\lambda_i[(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2-L_i^2]\}. \tag{5}$$

Ravnotežne enačbe dobimo kot enačbe

$$\begin{aligned}
\frac{1}{2}\frac{\partial G}{\partial x_i}&=0, && i=2,3,\ldots 2p+1,\\
\frac{1}{2}\frac{\partial G}{\partial y_i}&=0, && i=2,3,\ldots 2p+1,\\
\frac{\partial G}{\partial \lambda_i}&=0, && i=1,2,\ldots 2p+1.
\end{aligned}$$

ali

$$\begin{aligned}
\lambda_i\cdot(x_{i+1}-x_i)-\lambda_{i+1}\cdot(x_{i+2}-x_{i+1})&=0, && i=1,2,\ldots,2p\\
\lambda_i\cdot(y_{i+1}-y_i)-\lambda_{i+1}\cdot(y_{i+2}-y_{i+1})&=-\frac{M_{i+1}+M_i}{4}, && i=1,2,\ldots,2p\\
(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2&=L_i^2, && i=1,2,\ldots,2p+1.
\end{aligned}$$

Za lažje računanje uvedemo relativne koordinate

$$\begin{aligned}
\xi_i&=x_{i+1}-x_i, && i=1,2,\ldots,2p+1,\\
\eta_i&=y_{i+1}-y_i, && i=1,2,\ldots,2p+1,
\end{aligned}$$

od koder sledi

$$\begin{aligned}
x_i&=x_1+\sum_{j=1}^{i-1}\xi_i, && i=2,3,\ldots,2p+1,\\
y_i&=y_1+\sum_{j=1}^{i-1}\eta_i && i=2,3,\ldots,2p+1,
\end{aligned}$$

in zgornji sistem enačb preide v ekvivalentni sistem

$$\lambda_i\cdot\xi_i-\lambda_{i+1}\cdot\xi_{i+1}=0, \qquad i=1,2,\ldots,2p, \tag{6}$$

$$\lambda_i\cdot\eta_i-\lambda_{i+1}\cdot\eta_{i+1}=-\frac{1}{2}\mu_i, \qquad i=1,2,\ldots,2p, \tag{7}$$

$$\xi_i^2+\eta_i^2=L_i^2, \qquad i=1,2,\ldots,2p+1, \tag{8}$$

kjer je

$$\mu_i=\frac{M_{i+1}+M_i}{2}. \tag{9}$$

Iz enačbe (6) sedaj sledi

$$\lambda_i\cdot\xi_i=-\frac{1}{2u},\ i=1,2,\ldots,2p+1, \tag{10}$$

kjer je u konstanta, ki jo bomo še določili. Iz enačb (7) in (10) potem sledi

$$\frac{1}{2u}\frac{\eta_i}{\xi_i} - \frac{1}{2u}\frac{\eta_{i+1}}{\xi_{i+1}} = \frac{1}{2}\mu_i,\ i = 1, 2, \ldots, 2p,$$

ali

$$\frac{\eta_{i+1}}{\xi_{i+1}} = \frac{\eta_i}{\xi_i} - u \cdot \mu_i,\ i = 1, 2, \ldots, 2p,$$

ali

$$\frac{\eta_i}{\xi_i} = v - u \cdot \sum_{j=1}^{i-1} \mu_j,\ i = 1, 2, \ldots, 2p+1, \tag{11}$$

kjer bomo konstanto

$$v = \frac{\eta_1}{\xi_1}$$

določili v naslednjem koraku. Vstavimo za i vrednost $p+1$:

$$\eta_{p+1} = y_{p+2} - y_{p+1} = 0$$

$$\eta_{p+1} = \xi_{p+1} \cdot (v - u \cdot \sum_{j=1}^{p} \mu_j)$$

$$\xi_{p+1} \cdot (v - u \cdot \sum_{j=1}^{p} \mu_j) = 0$$

$$v - u \cdot \sum_{j=1}^{p} \mu_j = 0$$

$$v = u \cdot \sum_{j=1}^{p} \mu_j$$

Enačbe (8)-(11) nam sedaj povedo

$$1 + \left(u \cdot \sum_{j=1}^{p} \mu_j - u \cdot \sum_{j=1}^{i-1} \mu_j \right)^2 = \frac{L_i^2}{\xi_i^2},\ i = 1, 2, \ldots, 2p+1$$

od koder sledi

$$\xi_i = \frac{L_i}{\sqrt{1 + \left(u \cdot \sum_{j=1}^{p} \mu_j - u \cdot \sum_{j=1}^{i-1} \mu_j \right)^2}},\ i = 1, 2, \ldots, 2p+1 \tag{12}$$

Sedaj dobimo enačbo

$$U(u) = \sum_{i=1}^{2p+1} \xi_i - (x_{2p+2} - x_1) = 0. \tag{13}$$

S pomočjo funkcije *fzero* in začetnega približka $u0$ izračunamo ničlo te funkcije u.

1.1 Izračun koordinat krajišč simetrične diskretne verižnice na lihem številu členov z Matlabom

Računanja koordinat krajišč simetrične diskretne verižnice na lihem številu členov se lotimo s programom Matlab. Za ta izračun uporabimo dve funkciji: *dis_ver_l* in *enacba_u*

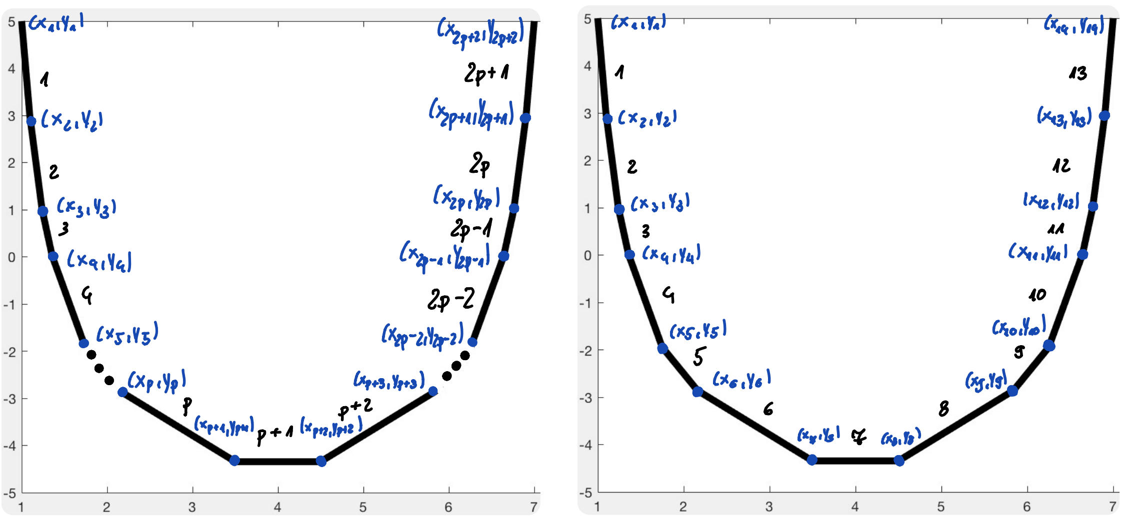
Število členkov $(n+1)$ je liho, zato lahko označimo $n+1=2p+1$ oz. $n=2p$ ali $p=\frac{n}{2}$. Vektor L podaja dolžine prvih $p+1$ palic, ostale dolžine so simetrične prvim p palicam. Prav tako velja za mase palic. Računanje koordinat si bomo ogledali skupaj s primerom verižnice, kjer so dolžine palic leve strani in sredinske palice enake

$$L=[2,2,1,2,1,2,1].$$

Mase teh palic so enake

$$M=[5,8,2,4,1,1,1].$$

V tem primeru je $p=6$.



Slika 1: Primer diskretne verižnice za $p \in \mathbb{N}$ (levo) in za $p=6$ (desno).

Vrednosti p in n lahko razberemo iz dolžine vektorja L (vrstica 3 in 4 v kodi):

```
1 p = length(L)-1;
2 n = 2*p;
```

```
1 p = 6
2 n = 12;
```

Vektorja L in M preuredimo tako, da bosta podajala dolžine in mase vseh palic:

```
1 L = [L(1:end-1), flip(L)];
2 M = [M(1:end-1), flip(M)];
```

```
1 L = [2,2,1,2,1,2,1,2,1,2,1,2,2];
2 M = [5,8,2,4,1,1,1,1,1,4,2,8,5];
```

Za izračun vrednosti u potrebujemo $\sum_{j=1}^{i-1} \mu_j$, ki si jih po vrsti za $i=1,2,\ldots,2p+1$ zapišemo v vektor *vsote_mi*:

```
mi = zeros(1,n);
mi = 1/2*(M(1:end-1)+M(2:end));
vsote_mi = [0 cumsum(mi)];
```

```
mi = [0,0,0,0,0,0,0,0,0,0,0,0];
mi = [6.5,5,3,2.5,1,1,1,1,2.5,3,5,6.5];
vsote_mi = [0,6.5,11.5,14.5,17,18,19,
            20,21,23.5,26.5,31.5,38];
```

Enačbo (13)

$$U(u) = \sum_{i=1}^{2p+1} \xi_i - (x_{2p+2} - x_1) = 0.$$

zapišemo v funkciji *enacba_u*. Tej funkciji podamo že preurejen vektor L, torej vektor z vsemi dolžinami palic. Teh palic je ravno $2p+1$, torej je dolžina vektorja L enaka $n+1$:

```
n = length(L)-1;
p = n/2;
```

```
n = 12;
p = 6;
```

Vektor *xi* podaja vrednosti ξ_i za $i = 1, 2, \ldots, 2p+1$ (vrstica 10). Izračunamo ga s pomočjo enačbe (12):

$$\xi_i = \frac{L_i}{\sqrt{1 + \left(u \cdot \sum\limits_{j=1}^{p} \mu_j - u \cdot \sum\limits_{j=1}^{i-1} \mu_j\right)^2}}, \; i = 1, 2, \ldots, 2p+1$$

Izraz v oklepaju označimo s *pom*. Ta izraz zdaj lahko izračunamo:

```
pom =u*(vsote_mi(p+1) - vsote_mi);
```

Vektor *xi* je potem enak:

```
xi = L./(1+pom.^2).^(1/2);
```

Enačbo U zdaj dobimo tako, da seštejemo elemente vektorja *xi* ter tej vsoti odšejemo x koordinato zadnjega krajišča ter ji prištejemo x koordinato prvega krajišča:

```
R = sum(xi)-(xn2-x1);
```

R je zdaj naša enačba U. Znotraj funkcije *dis_ver_l* jo poimenujemo kar *enacba*. S funkcijo *fzero* in začetnim približkom $u0$ poiščemo ničlo te funkcije u:

```
enacba = @(u) enacba_u(u, zac, L, vsote_mi);
u = fzero(enacba, u0);
```

```
u = -1.109843351708608;
```

Izraz v oklepaju enačbe (12) zdaj zapišemo z izračunano vrednostjo u ter izračunamo vrednosti vektorja *xi*:

```
pom =u*(vsote_mi(p+1) - vsote_mi);
xi = L./(1+pom.^2).^(1/2);
```

Vektorja *xi* in *eta* sta povezana z enačbo (11):

```
eta = xi.*pom;
```

Rešitev te funkcije je matrika velikosti $2 \times (2p+2)$, oblike

$$\begin{pmatrix} x_1 & x_2 & x_3 & \dots & x_{2p+1} & x_{2p+2} \\ y_1 & y_2 & y_3 & \dots & y_{2p+1} & y_{2p+2} \end{pmatrix}$$

Z vektorjem *zac* so podane koordinate:

$$\begin{aligned} x_1 &= zac(1) & \qquad x_{2p+2} &= zac(2) \\ y_1 &= zac(3) & \qquad y_{2p+2} &= zac(3) \end{aligned}$$

Ostale koordinate izračunamo s pomočjo enačb:

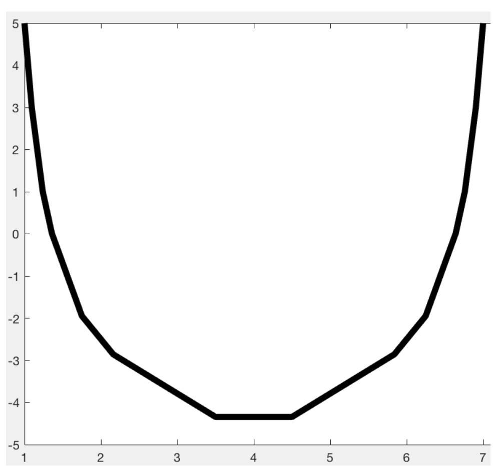
$$\begin{aligned} x_i &= x_1 + \sum_{j=1}^{i-1} \xi_i, & \qquad i &= 2, 3, \dots, 2p+1 \\ y_i &= y_1 + \sum_{j=1}^{i-1} \eta_i & \qquad i &= 2, 3, \dots, 2p+1 \end{aligned}$$

```
X = [zac(1), zac(1) + cumsum(xi);zac(3), zac(3) + cumsum(eta)];
```

$$X = \begin{pmatrix} 1.00 & 1.09 & 1.24 & 1.36 & 1.75 & 2.16 & 3.50 & 4.50 & \dots & 7.00 \\ 5.00 & 3.00 & 1.01 & 0.01 & -1.95 & -2.86 & -4.34 & -4.34 & \dots & 5.00 \end{pmatrix}$$

To diskretno verižnico narišemo z ukazom:

```
plot(X(1,:),X(2,:),'color','k','LineWidth',5,'MarkerSize',2,);
```



Slika 2: Diskretna verižnica za zgornji primer.

1.1.1 dis_ver_l

```
function X = dis_ver_l(u0, zac, L, M)

p = length(M)-1;
n = 2*p;

L = [L(1:end-1), flip(L)];
M = [M(1:end-1), flip(M)];

mi = zeros(1,n);
mi = 1/2*(M(1:end-1)+M(2:end));
vsote_mi = [0 cumsum(mi)];

enacba = @(u) enacba_u(u, zac, L, vsote_mi);
u = fzero(enacba, u0);

pom =u*(vsote_mi(p+1) - vsote_mi);
xi = L./(1+pom.^2).^(1/2);
eta = xi.*pom;

X = [zac(1), zac(1) + cumsum(xi);zac(3), zac(3) + cumsum(eta)];
end
```

1.1.2 enacba_u

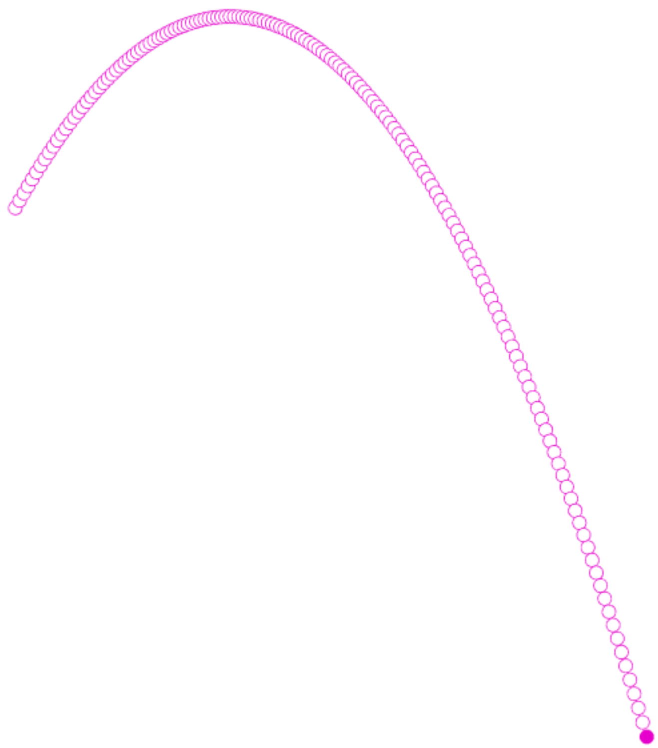
```
function R = enacba_u(u, zac, L, vsote_mi)

n = length(L)-1;
p = n/2;

x1 = zac(1);
xn2 = zac(2);

pom =u*(vsote_mi(p+1) - vsote_mi);
xi = L./(1+pom.^2).^(1/2);

R = sum(xi)-(xn2-x1);
end
```

2 Odbijanje lahke žogice

Lahka žogica z začetno točko $(x0(1), x0(2))$ in začetno hitrostjo $(v0(1), vo(2))$ se giblje po krivulji, podani s parametrom t in enačbama:

$$x(t) = x0(1) + v0(1) \cdot t \tag{14}$$

$$y(t) = x0(2) + v0(2) \cdot t - g \cdot \frac{t^2}{2} \tag{15}$$



2.1 Odbijanje žogice v Matlabu

Želimo vedeti na katerih točkah in s kakšno hitrostjo se žogica odbija od simetrične diskretn verižnice, katere parametri so podani s seznamom *opcije_veriznica*. Ta seznam je sestavljen iz parametrov $u0$ (začetni približek za parameter u), *zac* (položaj začetnih koordinat verižnice), L (dolžine členov verižnice), M (mase členov verižnice). S seznamom *opcije_zogica* sta podana parametra $x0$ (začetni položaj točke) ter $v0$ (začetna hitrost točke). Našo rešitev potovanja žogice želimo tudi vizualizirati.

Funkcija *zogica_odboji* nam glede na podatke za verižnico in začetne podatke za žogico za n odbojev izračuna točke odbojev ter začetne hitrosti takoj po odboju. Točke odbojev poda kot matriko velikosti

$2 \times (n+1)$, kjer i-ti stolpec predstavlja točko $(i-1)$-ega odboja, prvi stolpec pa začetni položaj žogice. Podobno začetne hitrosti poda kot matriko $2 \times (n+1)$, kjer i-ti stolpec predstavlja začetno hitrost po $(i-1)$-em odboji, prvi stolpec pa začetno hitrost žogice. Če žogica zleti iz verižnice po i-tem odboju, zadnji stolpec predstavlja zadnjo točko odboja (ali hitrost odboja v V) od verižnice ali pa začetno točko (začetno hitrost v V), če se žogica nikoli ne odbije od verižnice. Kot argument funkcije *zogica_odboji* hkrati podamo parametre *odmik*, *debelina* in *narisi*. To so argumenti za risanje potovanja žogice. S parametrom *odmik* podamo razdaljo, za katero naj se poveča okno risanja v vse 4 smeri (prej se je okno prilagodilo risanju verižnice). S parametrom *debelina* podajamo debelino žogice (približno 16). Parameter *narisi* ima lahko vrednosti 0 ali 1 in pove, ali se naj izriše potovanje žogice, ali ne. Zadnje tri parametre ne rabimo podati, saj jih funkcija v primeru, da niso podani, izbere sama.

Najprej izračunamo koordinate krajišč diskretne verižnice:

```
X = dis_ver_l(u0, zac, L, M);
```

V naslednjem koraku za n odbojev naredimo naslednji postopek:

- Shranimo začetni položaj in začetno hitrost žogice pred i-tim odbojem; $i = 1, 2, \ldots, n$.

```
    N(:,i) = x0;
    V(:,i) = v0;
```

- S funkcijo *odboj* izračunamo, v kateri točki se žogica naslednjič odbije od verižnice (p), kakšno začetno hitrost bo imela po odboju (*v_out*) ter koliko časa potrebuje, da pride do naslednje odbojne točke ($tv1$).

```
    [p, v_out, tv1] = odboj(v0, x0, X, odmik);
```

- Če je parameter $narisi = 1$, s funkcijo *graf* narišemo potovanje žogice od začetne točke do točke odboja (oz. do roba okna, če točka zleti iz verižnice). Ta funkcija prejme tudi parameter *zadnji*, ki pove, ali je odboj zadnji med odboji. Če je, se zadnji položaj žogice ne zamegli. Hkrati ima vsako potovanje od ene odbojne točke do druge, drugo barvo potovanja, podano z lestvico RGB (v našem primeru $[i/n, 0, 1-i/n]$).

```
if narisi
        if i==n
                zadnji = 1;
        else
                zadnji = 0;
        end
        graf(x0, v0, tv1, X, g, [i/n,0,1-i/n], odmik, debelina, zadnji)
end
```

- Preverimo, ali je žogica zletela iz verižnice. V primeru, da je, prekinemo for zanko.

```
    if v_out == [Inf, Inf]
        break
    end
```

- Za novo začetno hitrost vzamemo hitrost *v_out*, za novi začetni položaj pa odbojno točko p.

```
    v0 = v_out;
    x0 = p;
```

Na zadnji stolpec N shranimo še zadnjo odbojno točko ter na zadnji stolpec V začetno hitrost po zadnjem odboju.

V funkciji *zogica_odboji* smo klicali funkcijo *odboj*, ki izračuna odbojno točko p, začetno hitrost po odboju *v_out* ter čas potovanja od začetne točke do odbojne točke $tv1$. Za argumente dobi ta funkcija parametre: začetni položaj žogice $x0$, začetno hitrost žogice $v0$, matriko krajišč verižnice X ter vrednost, za katero razširimo širino okna *odmik*. Znotraj te funkcije računamo tudi hitrost $v1$. To je hitrost, s katero žogica prileti v odbojno točko.

Vsak člen verižnice predstavlja del premice, ki je podana s predpisom $y_i(x) = k_i \cdot (x - X(1,i)) + X(1,i)$; $k_i = \frac{X(1,i+1)-X(1,i)}{X(2,i+1)-X(2,i)}$. Pripravimo si vektor koeficientov k_i, ki pripadajo členom verižnice:

```
koeficienti = (X(2,2:end)-X(2, 1:end-1))./(X(1,2:end)-X(1,1:end-1));
```

V naslednjem koraku ločimo dva primera

1. v0(1) = 0: V primeru, da x koordinata začetnega položaja $x0$ leži med x koordinatama prvega in zadnjega krajišča verižnice, je ta koordinata enaka x-koordinati točke odboja. Če x koordinata začetnega položaja $x0$ ne leži med x koordinatama prvega in zadnjega krajiščema verižnice, pa to pomeni, da se točka ne odbije od verižnice (označimo s $p = [Inf, Inf]$). y-koordinata točke odboja je enaka vrednosti funkcije premice v točki $x = x0(1)$ za člen verižnice, ki seka navpičnico $x = x0(1)$.

   ```
       p = [x0(1); koeficienti(odsek)*(x0(1)-X(1,odsek)) + X(2,odsek)];
   ```

   Za izračun časa, ki ga žogica porabi, da pripotuje od začetne točke do točke odboja, se spomnimo naslednjih dveh enačb, ki predstavljata potovanje lahke žogice:

   $$x(t) = x0(1) + v0(1) \cdot t$$
   $$y(t) = x0(2) + v0(2) \cdot t - g \cdot \frac{t^2}{2}$$

   Vemo, da je $v0(1) = 0$, zato se x koordinata ohrani. Opazujemo le drugo enačbo ob času $tv1$:

   $$y(tv1) = p(2) = koeficienti(odsek) \cdot (x0(1) - X(1, odsek)) + X(2, odsek) = x0(2) + v0(2) \cdot tv1 - g \cdot \frac{tv1^2}{2}$$

   To je kvadratna enačba s pozitivno rešitvijo (negativni čas nas ne zanima):

   $$tv1 = (v0(2) + sqrt(v0(2)^2 - 2 \cdot g \cdot (p(2) - x0(2))))/g$$

   ```
           tv1 = (v0(2) + sqrt(v0(2)^2-2*g*(p(2)-x0(2))))/g;
   ```

   Hitrost žogice v y smeri se s časom spreminja kot:

   $$v1_y(t) = v0(2) - g \cdot t$$

   Žogica ima torej ob času $tv1$ hitrost enako:

   $$v1 = [0, v0(2) - g \cdot tv1]$$

```
    v1 = [v0(1); v0(2)-g*(p(1)-x0(1))/v0(1)];
```

2. $v0(1) \neq 0$: V tem primeru lahko krivuljo potovanja žogice izrazimo kot funkcijo $y(x)$:

```
    y = @(x) x0(2) + (x-x0(1)).*(v0(2) - g*((x-x0(1))/v0(1))/2)/v0(1);
```

S funkcijo *presecisca* poiščemo presečišča parabole potovanja žogice z verižnico. Presečišče iščemo najprej kot presečišče parabole z vsako premico, ki predstavlja člen verižnice, posebaj,

```
    a = g/(2*v0(1)^2);
    b = koeficienti(i)-v0(2)/v0(1);
    c = koeficienti(i)*(x0(1)-X(1,i)) + (X(2,i)-x0(2));
    D = b^2-4*a*c;
    pre1 = x0(1) + (-b-sqrt(D))/(2*a);
    pre2 = x0(1) + (-b+sqrt(D))/(2*a);
```

nato pa preverimo, če to presečišče leši na delu premice (v tem primeru ga dodamo v seznam *presecisce*), ki predstavlja verižnico:

```
    if X(1,i)<=pre1 && pre1<=X(1,i+1)
        presecisce = [presecisce, pre1];
    end
    if X(1,i)<=pre2 && pre2<=X(1,i+1) && pre2~=pre1
        presecisce = [presecisce, pre2];
    end
```

Ta presečišca imamo zdaj shranjena pod spremenljivko *presecisce*. V naslednjem koraku s funkcijo *izberi_presecisce* pogledamo, ali je katero od teh presečišč naše iskano. Če smo našli dve presečišči, je naše iskano presečišče na desni strani, če je $v0(1) > 0$ in na levi strani, če je $v0(1) < 0$:

```
if length(presecisce) >= 2
    if v0(1)>0
        p = [presecisce(end); y(presecisce(end))];
    else
        p = [presecisce(1); y(presecisce(1))];
    end
```

Če smo našli samo eno presečišče, in je to presečišče različno od začetne točke, je to naše iskano presečišče, sicer žogica skoči iz verižnice:

```
elseif length(presecisce) == 1
    if abs(presecisce(1) - x0(1)) < eps
        p = [Inf; Inf];
    else
        p = [presecisce(1); y(presecisce(1))];
    end
```

Prav tako skoči iz verižnice, če nismo našli nobenega presečišča:

```
else
    p = [Inf; Inf];
end
```

Čas potovanja lahko izračunamo iz enačbe:

$$x(t) = x0(1) + v0(1) \cdot t$$

Za $tv1$ je ta enačba enaka:

$$p(1) = x0(1) + v0(1) \cdot tv1$$

oz.

$$tv1 = \frac{p(1) - x0(1)}{v0(1)}$$

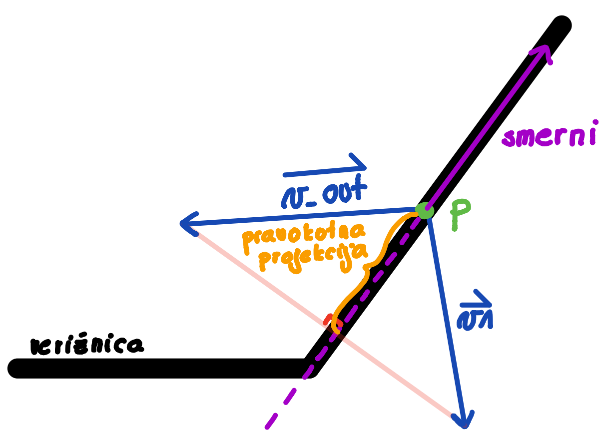
```
    tv1=(p(1)-x0(1))/v0(1);
```

Hitrost ob času $tv1$ v x smeri se ohrani, v y smeri pa je enaka:

$$v1_y(tv1) = v0(2) - g \cdot tv1$$

```
    v1 = [v0(1); v0(2)-g*(p(1)-x0(1))/v0(1)];
```

V tem koraku imamo izračunane parametre p, $tv1$ ter $v1$. To so vsi parametri, ki jih potrebujemo, da izračunamo za odboj hitrosti v točki odboja:




```
    smerni = [1;koeficienti(odsek)];
    pravokotna_projekcija = (v1'*smerni)/norm(smerni);
    v_out = -v1 + 2*pravokotna_projekcija*smerni/norm(smerni);
```

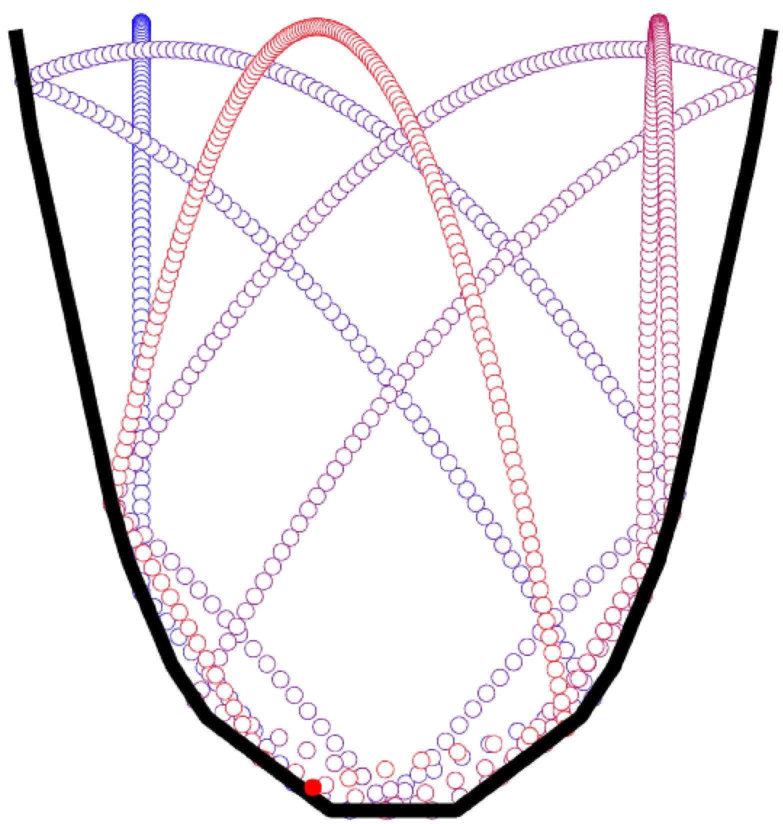
Če žogica zleti iz verižnice, s funkcijo *presecisce_z_robom_ekrana* izračunamo čas $tv1$, ki ga žogica porabi, da pripotuje do roba ekrana, hitrost *v_out* pa nastavimo na $[Inf; Inf]$, da for zanka iz funkcije *zogica_odboji* zazna, da je žogica zletela iz verižnice.

S funkcijo *graf* potovanje žogice rišemo tako, da se žogica nariše za njen premik v času $0.02s$.

Na naslednji sliki je prikazano potovanje žogice za primer:

```
opcije_veriznica.u0 = 0;
opcije_veriznica.zac = [1,7,5];
opcije_veriznica.L = [2,7,1,2,1,2,1];
opcije_veriznica.M = [5,8,2,6,1,5,1];
opcije_zogica.x0 = [2;1];
opcije_zogica.v0 = [0;9];

[N3, V3] = zogica_odboji(20, opcije_zogica, opcije_veriznica);
```

Točka, na kateri se žogica vstavi, je točka $(3.283799105442183, -9.248930217470967)$, hitrost, s katero bo v naslednjem trenutko potovala dalje, pa je enaka $(14.881839596366820, 7.772379461300024)$.



2.1.1 zogica_odboji

```
function [N,V] = zogica_odboji(n, opcije_zogica, opcije_veriznica, odmik,
    debelina, narisi)

if nargin < 6
    narisi = 1;
end
if nargin < 5
    debelina = 18;
end
if nargin < 4
    odmik = 2;
end
v0 = opcije_zogica.v0;
x0 = opcije_zogica.x0;
u0 = opcije_veriznica.u0;
zac = opcije_veriznica.zac;
L = opcije_veriznica.L;
M = opcije_veriznica.M;

N = zeros(2,n+1);
V = zeros(2,n+1);
T = zeros(1,n);

g = 9.8;

X = dis_ver_l(u0, zac, L, M);

for i=1:n
    N(:,i) = x0;
    V(:,i) = v0;
    [p, v_out, tv1] = odboj(v0, x0, X, odmik);
    T(i) = tv1;
    if narisi
        if i==n
            zadnji = 1;
        else
            zadnji = 0;
        end
        graf(x0, v0, tv1, X, g, [i/n,0,1-i/n], odmik, debelina, zadnji)
    end
    if v_out == [Inf, Inf]
        break
    end
    v0 = v_out;
    x0 = p;
```

```
end
N(:,n+1) = x0;
V(:,n+1) = v0;

end
```

2.1.2 odboj

```
function [p, v_out, tv1] = odboj(v0, x0, X, odmik)

p = zeros(2);
eps = 1.0e-15;
n = length(X) - 1;
g = 9.8;
koeficienti = (X(2,2:end)-X(2, 1:end-1))./(X(1,2:end)-X(1,1:end-1));

if v0(1) == 0
    if x0(1)>X(1,1) && x0(1)<X(1,end)
        odsek = find(sort([X(1,:), x0(1)])==x0(1))-1;
        p = [x0(1); koeficienti(odsek)*(x0(1)-X(1,odsek)) + X(2,odsek)];
        tv1 = (v0(2) + sqrt(v0(2)^2-2*g*(p(2)-x0(2))))/g;
        v1 = [0; v0(2)-g*tv1];
    else
        p = [Inf; Inf];
    end
else
    y = @(x) x0(2) + (x-x0(1)).*(v0(2) - g*((x-x0(1))/v0(1))/2)/v0(1);
    presecisce = presecisca(n, x0, v0, g, X, koeficienti);
    p = izberi_presecisce(x0, v0, presecisce, y, eps);
    tv1=(p(1)-x0(1))/v0(1);
    v1 = [v0(1); v0(2)-g*(p(1)-x0(1))/v0(1)];
end

if p ~= [Inf; Inf]
    odsek = find(sort([X(1,:), p(1)])==p(1))-1;
    smerni = [1;koeficienti(odsek)];
    pravokotna_projekcija = (v1'*smerni)/norm(smerni);
    v_out = -v1 + 2*pravokotna_projekcija*smerni/norm(smerni);
else
    tv1 = presecisce_z_robom_ekrana(x0, v0, X, odmik);
    v_out = [Inf, Inf];
end
```

2.1.3 presecisca

```
function presecisce = presecisca(n, x0, v0, g, X, koeficienti)

presecisce=[];
for i = 1:n
    a = g/(2*v0(1)^2);
    b = koeficienti(i)-v0(2)/v0(1);
    c = koeficienti(i)*(x0(1)-X(1,i)) + (X(2,i)-x0(2));
    D = b^2-4*a*c;
    pre1 = x0(1) + (-b-sqrt(D))/(2*a);
    pre2 = x0(1) + (-b+sqrt(D))/(2*a);
    if X(1,i)<=pre1 && pre1<=X(1,i+1)
        presecisce = [presecisce, pre1];
    end
    if X(1,i)<=pre2 && pre2<=X(1,i+1) && pre2~=pre1
        presecisce = [presecisce, pre2];
    end
end
end
```

2.1.4 izberi_presecisce

```
function p = izberi_presecisce(x0, v0, presecisce, y, eps)

if length(presecisce) >= 2
    if v0(1)>0
        p = [presecisce(end); y(presecisce(end))];
    else
        p = [presecisce(1); y(presecisce(1))];
    end
elseif length(presecisce) == 1
    if abs(presecisce(1) - x0(1)) < eps
        p = [Inf; Inf];
    else
        p = [presecisce(1); y(presecisce(1))];
    end
else
    p = [Inf; Inf];
end
end
```

2.1.5 presecisce_z_robom_ekrana

```
function tv1 = presecisce_z_robom_ekrana(x0, v0, X, odmik)

n = length(X);
p = n/2;

g = 9.8;

xmin = X(1,1)-odmik;
xmax = X(1,end)+odmik;
ymin = X(2,p)-odmik;
ymax = X(2,1)+odmik;

if v0(1)>0 && v0(2)>0
    tx = (xmax-x0(1))/v0(1);
    tv1 = tx;
elseif v0(1)>0 && v0(2)<=0
    tx = (xmax-x0(1))/v0(1);
    ty = (v0(2)-sqrt(v0(2)^2-4*(ymin-x0(2))*(g/2)))/(2*(ymin-x0(2)));
    tv1 = min(tx, ty);
elseif v0(1)<0 && v0(2)>0
    tx = (xmin-x0(1))/v0(1);
    tv1 = tx;
elseif v0(1)<0 && v0(2)<0
    tx = (xmin-x0(1))/v0(1);
    ty = (v0(2)-sqrt(v0(2)^2-4*(ymin-x0(2))*(g/2)))/(2*(ymin-x0(2)));
    tv1 = min(tx, ty);
end
```

2.1.6 graf

```
function graf(x0, v0, tv1, X, g, barva, odmik, debelina, zadnji)

hold on;
axis([X(1,1)-odmik, X(1,end)+odmik, min(X(2,:))-odmik, X(2,end)+odmik]);

x = @(t) x0(1) + v0(1)*t;
y = @(t) x0(2) + v0(2)*t - (g*t.^2)/2;

korak = 0.02;
T = 0:korak:tv1;

for t = T
    axis([X(1,1)-odmik, X(1,end)+odmik, min(X(2,:))-odmik, X(2,end)+odmik])
```

```
    plot(X(1,:),X(2,:),'color','k','LineWidth',5,'MarkerSize',2,'MarkerFaceColor',[0,0,1]);
    if t~=T(1)
        plot(x(t-korak), y(t-korak), '.', 'color', 'w', 'markersize', debelina-2);
    end
    plot(x(t), y(t), '.', 'color', barva, 'markersize', debelina);
    pause(0.000001);
end
if ~zadnji
    plot(x(T(end)), y(T(end)), '.', 'color', 'w', 'markersize', debelina-2);
end
hold off;
```

3 Zaključek

Program je napisan tako, da žogica začne potovati na katerikoli začetni točki in ne nujno pade samo na zadnji členek verižnice (kot je napisano v navodilu). Nahajanje po n odbojih pa je opisano s točkami odbojev.