Minimizacija vremena kod problema inverznog klatna

Đorđe Mirosavić

May 16, 2022

1 Uvod

Imamo na raspolaganju klatno koje se nalazi na kolicima koja se pokreću, i ono što želimo je da od pozicije A do pozicije B dođe za minimum vremena.

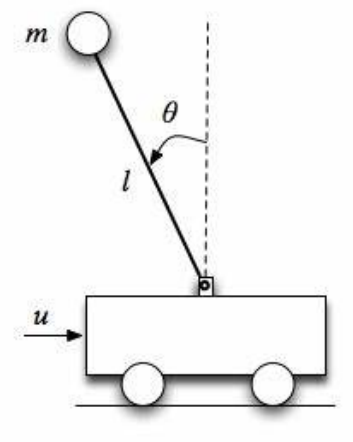


Figure 1: Model sistema

2 Matematički model

Sledeće jednačine opisuju dati sistem:

$$\dot{x_1} = x_2$$

$$\dot{x_2} = \frac{1}{-I(M+m)+Mm^2}[(I+m^2)b * x_2 + m^2g^2 * x_3 + I + m^2 * u]$$

$$\dot{x_3} = x_4$$

$$\dot{x_4} = \frac{1}{-I(M+m)+Mm^2}[-mb * x_2 + mg(M+m) * x_3 + m * u]$$

Gde x_1 predstavlja pomeraj kolica po x osi, x_3 je ugao koji klatno zaklapa sa y-osom, x_2 i x_4 su brzine promena veličina x_1 i x_3, a u je upravljanje.
I - moment inercije
M - masa kolica
m - masa kuglice
g - gravitaciono ubrzanje
b - koeficijent trenja
l - dužina štapa
Uzimimo sledeće parametre za naš model

$$M = 0.5kg$$
$$m = 0.2kg$$
$$b = 0.1m$$
$$I = 0.006\frac{kg}{m^2}$$
$$g = 9.8\frac{m}{s^2}$$
$$l = 0.3m$$

Uređeni model glasi ovako:

$$\dot{x_1} = x_2$$
$$\dot{x_2} = -0.1818x_2 + 2.673x_3 + 1.818u$$
$$\dot{x_3} = x_4$$
$$\dot{x_4} = -0.4545x_2 + 31.18x_3 + 4.545u$$

3 LQR

Da bismo mogli da upravljamo modelom u povratnoj sprezi, potrebno je da ga pre svega stabilizujemo oko radne tačke, odnosno dobijemo upravljanje koje će dato klatno stabilizovati. Kod nas će se ona nalaziti na poziciji kada je ugao koji klatno zaklapa jednak nuli.
Stoga, iskoristićemo LQR metodu.
Koristićemo MATLAB-ovu funkciju za to.

```
R = 2       % definiše težinu parametra za upravljanje
Q = C'* C   % definiše težinu parametra za model
% Q[0][0]   % definiše težinu parametra pozicije, a Q[2][2] definiše težinu parametra ugla teta
Q(1,1) = 10;
Q(3,3) = 50;
K = lqr(A, B, Q, R);
```

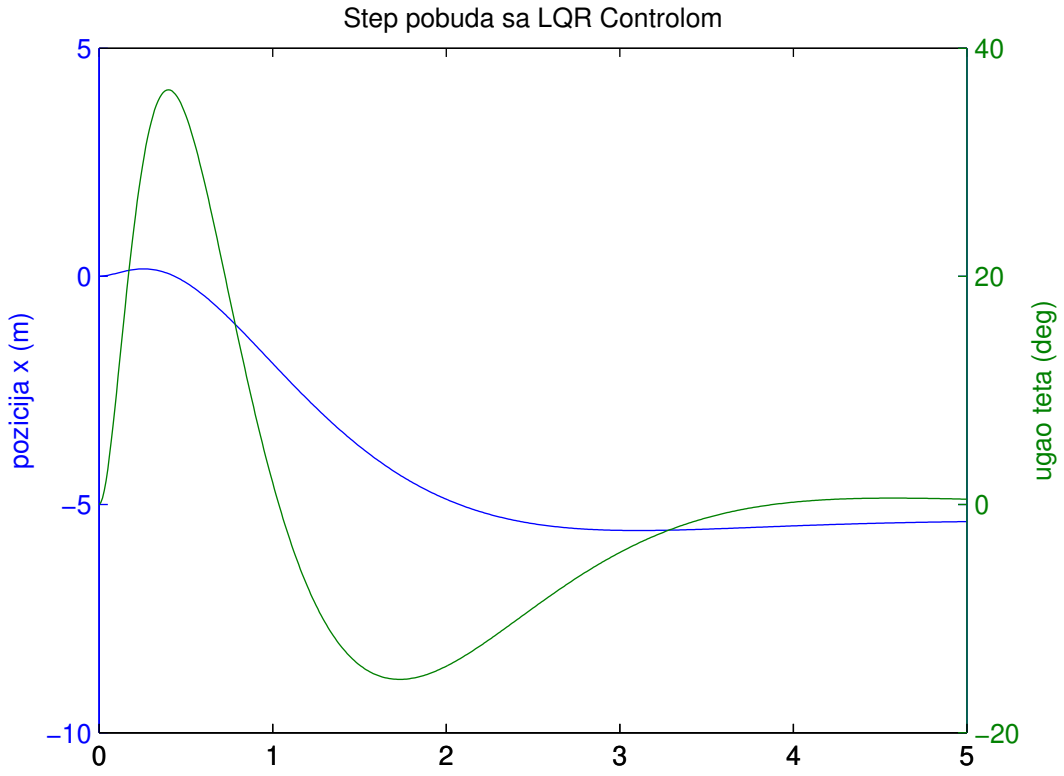
Da bismo se uverili da je upravljanje uspešno, to možemo proveriti tako što simuliramo sistem na step pobudu. Ali pre toga, formirajmo novi model.

```
A_new = [(A - B*K)];
B_new = [B];
C_new = [C];
D_new = [D];

states = {'x' 'x_dot' 'theta' 'theta_dot'};
inputs = {'u'};
outputs = {'x'; 'theta'};
sys_ss_new = ss(A_new,B_new,C_new,D_new,'statename',states,'inputname',inputs,'outputname',outputs)

t = 0:0.01:5;
step = 12*ones(size(t));

[y, t, x] = lsim(sys_ss_new, step, t);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2).*(180/pi),'plot');
set(get(AX(1),'Ylabel'),'String','pozicija x (m)')
set(get(AX(2),'Ylabel'), 'String','ugao teta (deg)')
title('Step pobuda sa LQR Control-om')
```



4 Minimizacija vremena i optimalno upravljanje

4.1 Hamiltonijan

Sada kada smo stabilizovali naš model, sada možemo da vršimo optimizaciju. Pošto vršimo minimazaciju vremena, kriterijum optimalnosti će nam glasiti ovako:

$$\int_0^T \alpha + \frac{1}{2}u^2 \, \mathrm{d}x.$$

Gde je α promenljiv parametar.

$$H = \alpha + \frac{1}{2}u^2 + \dot{x_1}p_1 + \dot{x_2}p_2 + \dot{x_3}p_3 + \dot{x_4}p_4$$

$$H = \alpha + \frac{1}{2}u^2 + x_2p_1 + (-0.1818x_2 + 2.673x_3 + 1.818u)p_2 + x_4p_3 + (-0.4545x_2 + 31.18x_3 + 4.545u)p_4$$

$$p_i = -(\frac{\partial H}{\partial x_i})$$

Zbog kasnije simulacije u MATLAB-u, p-ove ćemo obeležavati takođe sa x, samo što će p_1 biti x_5, p_2 će biti x_6... Ovo radimo zbog kasnije simulacije, odnosno MATLAB zahteva da imamo niz diferencijalnih jednačina iskazane samo preko jedne promenljive, odnosno x.

$$(\frac{\partial H}{\partial u}) = 0$$

Iz ove jednačine dobijamo da je $u = -(1.818x_6 + 4.545x_8)$

4.2 Sistem

Definišimo sada jednačine koje je potrebno rešiti da bismo dobili rešenje za poziciju, ugao, i upravljanje.

$$\dot{x_1} = x_2$$
$$\dot{x_2} = -0.1818x_2 + 2.673x_3 - 1.818(1.818x_6 + 4.545x_8)$$
$$\dot{x_3} = x_4$$
$$\dot{x_4} = -0.4545x_2 + 31.18x_3 - 4.545(1.818x_6 + 4.545x_8)$$
$$\dot{x_5} = -(4.0656x_6 + 10.1639x_8)$$
$$\dot{x_6} = -(x_5 + 4.9155x_6 + 12.2888x_8)$$
$$\dot{x_7} = -(-38.8818x_6 - 72.7044x_8)$$
$$\dot{x_8} = -(-7.6185x_6 + x_7 - 19.0462x_8)$$

Odnosno, dati sistem bi se u MATLAB-u predstavio kao:

```
sistem = @(t, x, T) T*[x(2)
                4.0656*x(1)+4.9155*x(2)-38.8818*x(3)-7.6185*x(4)-1.8182*(1.8182*x(6)+4.5455*x(8))
                x(4)
                10.1639*x(1)+12.2888*x(2)-72.7044*x(3)-19.0462*x(4)-4.5455*(1.8182*x(6)+4.5455*x(8))
                -(4.0656*x(6)+10.1639*x(8))
                -(1.0000*x(5)+4.9155*x(6)+12.2888*x(8))
                -(-38.8818*x(6)-72.7044*x(8))
                -(-7.6185*x(6)+1.0000*x(7)-19.0462*x(8))];
```

Dati izraz se množi sa nekom promenljivom T jer da bismo našil optimalno vreme, potrebno je da uvedemo još jednu promenljivu koja bi imitirala vreme kao još jedan parametar koji se optimizuje.

4.3 Granični uslovi

Sada je potrebno da vidimo kako bismo želeli da se naš sistem ponaša u početnom i krajnjem trenutku. Pošto ćemo imati neke uslove na početku a neke na kraju, ovo se još naziva i Two point boundary problem.
Želimo da nam sistem krene od početnog trenutka, odnosno da mu pozicija u početku bude jednaka nuli. Odnosno: $x_1(0) = 0$. Želimo i da se kolica na kraju pomere ko krajnje pozicije - $x_1(T) = krajnjaPozicija$. Uzmimo da želimo da se pomerimo na rastojanje 6m od početka. $x_1(T) = 6$. Brzina je $0\frac{m}{s}$ u obe pozicije - $x_2(0) = 0$, $x_2(T) = 0$
Od interesa nam je i da klatno u početku stoji pravo i na početku i na kraju, odnosno da je $x_3(0) = 0$, $x_3(T) = 0$. Što se tiče brzine ω ugla, uzmimo da je klatno u početku mirno bez početne brzine, a da u krajnjem trenutku može da ima proizvoljnu brzinu, jer ukoliko i ovde zahtevamo da i promena ugla bude 0 $x_4(0) = 0$, onda ćemo dobiti upravljanje koje ne možemo fizički da ostvarimo, a ukoliko nema to ograničenje, dobićemo da sama promena ugla i nije toliko velika u krajnjem trenutku. Tj. u krajnjem trenutku imamo uslov da je $p_4(T) = 0$ odnosno po našoj notaciji $x_8(T) = 0$. I na kraju, imamo i uslov da je $H(T) = 0$.
Pošto MATLAB sve ovo posmatra kao ograničenja u formi $g(x) = 0$, potrebno je to i da ispoštujemo kada budemo pisali funkciju koja sadrži granične uslove.

```
alpha = 6;

granicni_uslovi = @(x_0,x_T, T) [
x_0(1)          % klatno krece od nule
x_0(2)          % bez pocetne brzine
x_0(3)          % uzmimo da klatno stoji pravo na pocetku.
                % da je pod uglom od 0.1rad pisali bismo x_0(3)-0.1
x_0(4)          % i da nema pocetno ugaono ubrzanje
x_T(1)-6        % uzmimo da je krajnja pozicija 6
```

```
x_T(2)          % i da nece imati brzinu u tom trenutku
x_T(3)          % bitno nam je da klatno uspravno na kraju
x_T(8)          % ... ali ne i koju ce ugaonu brzinu imati

0.5*(1.818*x_T(6)+4.5455*x_T(8))^2+alpha
+x_T(5)*x_T(2)
+x_T(6)*(4.0656*x_T(1)+4.9155*x_T(2)-38.8818*x_T(3)-7.6185*x_T(4)
-1.8182*(1.8182*x_T(6)+4.5455*x_T(8)))
+x_T(7)*x_T(4)
+x_T(8)*(10.1639*x_T(1)+12.2888*x_T(2)-72.7044*x_T(3)-19.0462*x_T(4)
-4.5455*(1.8182*x_T(6)+4.5455*x_T(8)))];
```

4.4 Simulacija

Funkcija koji ćemo koristiti je `bvp4c`. Ona kao parametre prima sistem, granične uslove kao i početna pogađanja za rešenja diferencijalnih jednačina, kao i početno pogađanje za malopređašnji parametar T koji nam treba za optimizaciju vremena. Tj. funkcija `bvpinit` vrši poslednje 2 stavke, a `bvp4c` ih koristi kao parametar. Takođe, potrebno je da odredimo u koliko tačaka želimo da nađemo rešenje (neka to bude 1500), kao i na kom intervalu da traži rešenje. Pošto je ovo slučaj optimizacije vremena, onda je potrebno da interval bude od 0 do 1 jer tako funkcija zahteva, a onda ćemo dobijena rešenja skalirati, i dobiti stvarna vremena tj. `time`.

```
tacke = linspace(0,1,1250);
pocetno_pogadjanje = [0;0.1;0.2;0.3;0.4;0.8;0.9;1];
pocetno_pogadjanje_T = 3;

solinit = bvpinit(tacke, pocetno_pogadjanje, pocetno_pogadjanje_T);       % time is not fixed
sol = bvp4c(sistem, granicni_uslovi, solinit);

opt_upravljanje = - (1.818.*sol.y(6,:) + 4.545.*sol.y(8,:));
time = sol.parameters*sol.x;
```

Moguće i da dobijemo upravljanje tako da od A do B dodje za neko naše definisano vreme. To bismo uradili tako što ne bismo imali promenljivu T u sistemu i graničim uslovima.

```
sistem = @(t, x) [x(2)
                  4.0656*x(1)+...]
granicni_uslovi = @(x_0,x_T) [x_0(1)
                              x_0(2)...]
```

Simulacija bi glasila na sličan način, s razlikom da nemamo pogađanje promenljive T, kao i to da ne treba da skaliramo konačno vreme tj. `time`

```
tacke = linspace(0,željeno_vreme,1250);
pocetno_pogadjanje = [0;0.1;0.2;0.3;0.4;0.8;0.9;1];

solinit = bvpinit(tacke, pocetno_pogadjanje);       % time is fixed
sol = bvp4c(sistem, granicni_uslovi, solinit);

opt_upravljanje = - (1.818.*sol.y(6,:) + 4.545.*sol.y(8,:));
time = sol.x;
```

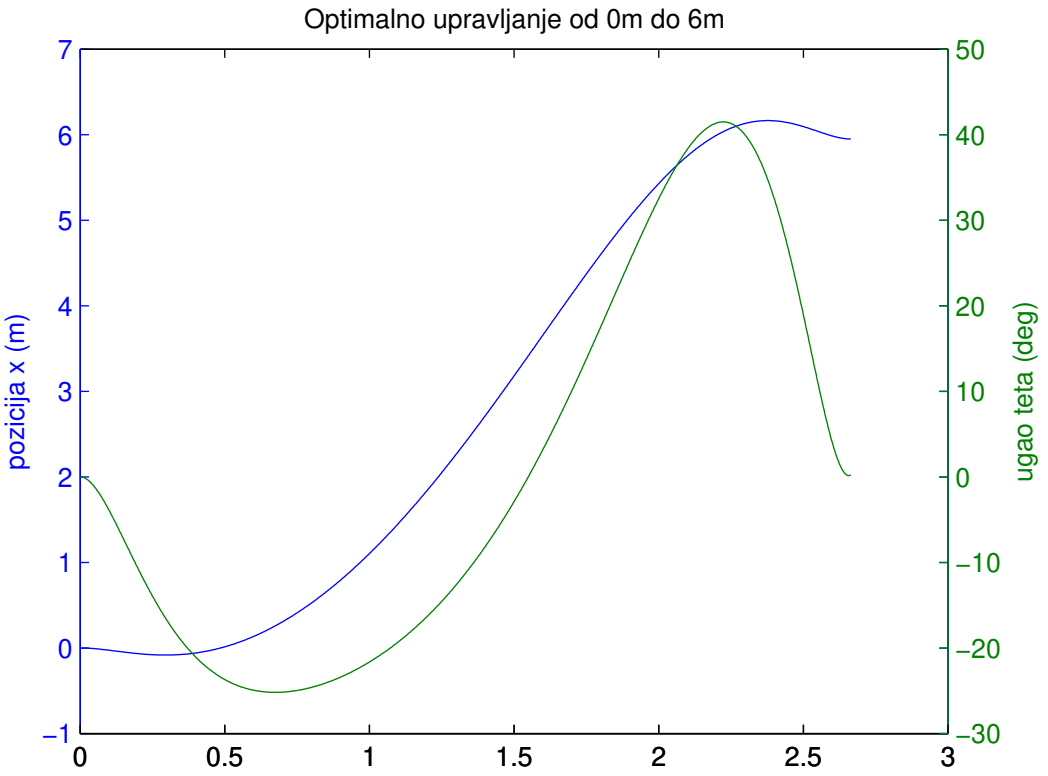
Ali mi ćemo se držati prvog načina, gde imamo promenljivu T.
Radi testiranja mžemo da iscrtamo i kako izgleda promena ugla, pozicije u upravljanje.

```
figure;
plot(time, sol.y(1,:))          % promena pozicije
title('x');
figure;
```

```
plot(time, sol.y(3,:).*(180/pi))% promena ugla (deg)
title('teta_{deg}')
figure;
plot(time, opt_upravljanje);
title('u');
```

Za kraj ćemo korišćenjem funkcije `lsim` simulirati sistem sa novim upravljanjem i videti kakav će odziv sistem dati.

```
[y, t, x] = lsim(sys_ss_new, opt_upravljanje, time);
[AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2).*(180/pi),'plot');
set(get(AX(1),'Ylabel'),'String','pozicija x (m)')
set(get(AX(2),'Ylabel'), 'String','ugao teta (deg)')
title('Optimalno upravljanje od 0m do 6m')
```



Vidimo da sistem u početku treba malo da se vrati unazad ne bi li krenuo unapred. A pri kraju je malo premašio cilj, pa se vratio da bi se klatno ispravilo. Unosom komande `>> time(end)` dobijamo da je potrebno 2.6639s da dođemo od 0m do 6m.