Optymalizacja w systemach sterowania-materiały pomocnicze do laboratorium

dr hab. inż. Piotr Bania, pba@agh.edu.pl, B1, p. 303

18 maja 2021

Rozwiązanie równania stanu

Układ równań

$$\dot{x} = f(t, x, u), x(0) = x_0, \tag{1}$$

gdzie $x(t) \in \mathbb{R}^n$, oraz u jest ustalonym wektorem, rozwiązujemy metodą Rungego-Kutty czwartego rzędu. Algorytm metody ma postać

$$x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{2}$$

$$k_1 = f(t_k, x_k, u), \tag{3}$$
$$k_2 = f(t_k + \tfrac{1}{2}h, x_k + \tfrac{1}{2}hk_1, u), \tag{4}$$
$$k_3 = f(t_k + \tfrac{1}{2}h, x_k + \tfrac{1}{2}hk_2, u), \tag{5}$$
$$k_4 = f(t_k + h, x_k + hk_3, u). \tag{6}$$

Implementacja RK4 w Matlab

```
function [t,x]=rk4(x0,u,tf,MDNS)
x0=x0(:);u=u(:);n=length(x0);
nt=1+floor(tf*MDNS);h=tf/nt;
x=zeros(nt+1,n);t=zeros(nt+1,1);
x(1,:)=x0';
tmp=zeros(n,1);xtmp=x0;tt=0;
dx1=zeros(n,1);dx2=zeros(n,1);
dx3=zeros(n,1);dx4=zeros(n,1);
h_2=h/2;h_6=h/6;h_26=2*h_6;
for i=1:nt
  dx1=rhs(tt,xtmp,u);tmp=xtmp+h_2*dx1;tt=tt+h_2;
  dx2=rhs(tt,tmp,u);tmp=xtmp+h_2*dx2;
  dx3=rhs(tt,tmp,u);tmp=xtmp+h*dx3;tt=tt+h_2;
  dx4=rhs(tt,tmp,u);
  xtmp=xtmp+h_6*(dx1+dx4)+h_26*(dx2+dx3);
  x(i+1,:)=xtmp';t(i+1)=tt;
end
```

Right hand side example

Wahadło matematyczne z silnikiem DC, można opisać równaniami

$$\ddot{x} + 2\xi\omega_0\dot{x} + \omega_0^2 \sin x = kz, \dot{z} = \tau^{-1}(u - z). \tag{7}$$

Są one równoważne układowi równań ($x_1 = x, x_2 = \dot{x}, x_3 = z$),

$$\dot{x}_1 = x_2, \tag{8}$$

$$\dot{x}_2 = -\omega_0^2 \sin x_1 - 2\xi\omega_0 x_2 + kx_3, \tag{9}$$

$$\dot{x}_3 = \frac{u - x_3}{\tau}. \tag{10}$$

Przykładową implementację prawej strony równań (8-10), pokazano poniżej.

```
function dx=rhs(t,x,u)
dx=zeros(3,1);w0=9;ksi=0.01;tau=0.04;k=250;
dx(1)=x(2);
dx(2)=-w0^2*sin(x(1))-2*ksi*w0*x(2)+k*x(3);
dx(3)=(u-x(3))/tau;
```

Całkowanie równań stanu na przedziałach stałości sterowania

Dany jest wektor czasów przełączeń sterowania

$$\tau = [0, \tau_1, \tau_2, ..., \tau_m], \tag{11}$$

oraz wektor lub macierz,

$$u = [u_1, ..., u_m], \tag{12}$$

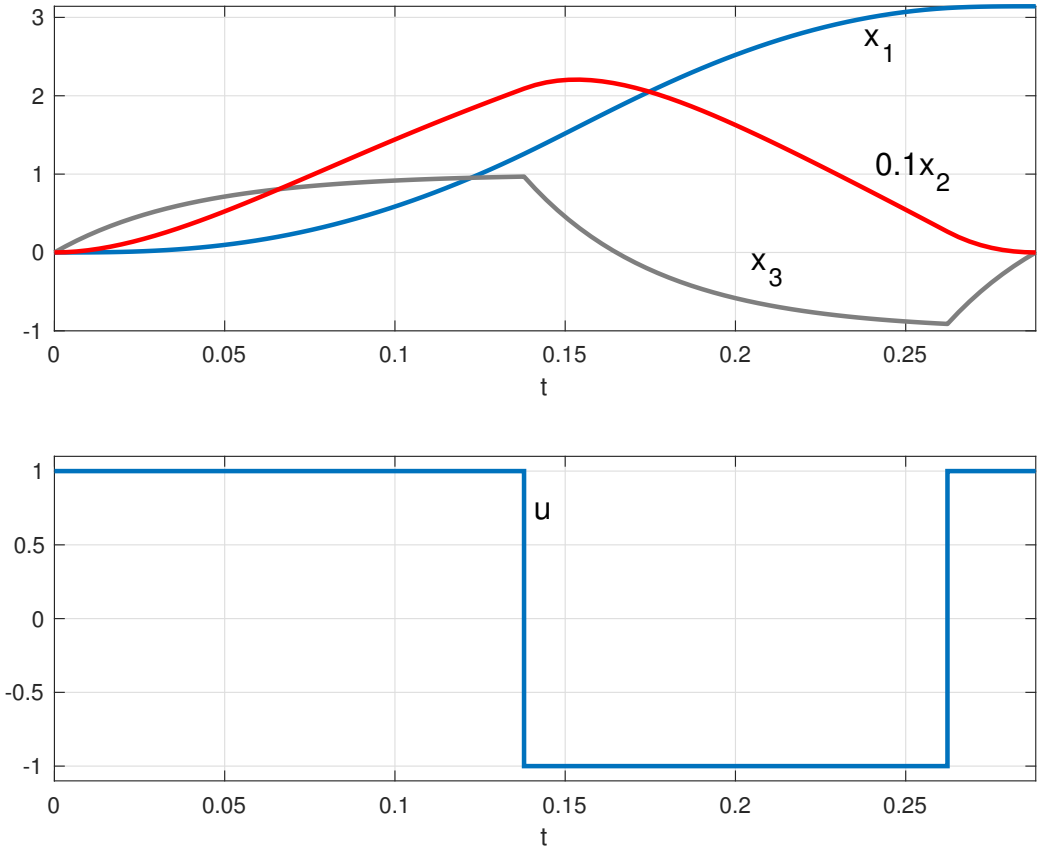
wartości przyjmowanych przez sterowanie w przedziale czasu $[\tau_{k-1}, \tau_k), k = 1, ..., m$, tj. $u(t) = u_k$, gdy $t \in [\tau_{k-1}, \tau_k)$. Zakładamy, że sterowanie u jest funkcją prawostronnie ciągłą dla $t < \tau_m$ oraz lewostronnie ciągłą w punkcie $t = \tau_m$. Należy napisać procedurę całkowania równań (1).

Procedura całkowania na przedziałach stałości sterowania.

```
function [t,x,uk,nseg]=get_tx(tau,u,x0,MDNS)
[ntau,nu]=size(u);n=length(x0);tau=[0;tau];
nt=1+sum(1+floor(MDNS*diff(tau)));
x=zeros(nt,n);
uk=zeros(nt,nu);
t=zeros(nt,1);nseg=zeros(ntau,1);kk=1;
for k=1:ntau
    [ttmp,xtmp]=rk4(x0,u(k,:)',tau(k+1)-tau(k),MDNS);% RK4
    x0=xtmp(end,:)';%warunek poczatkowy do nastepnego
    ni=1+floor(MDNS*(tau(k+1)-tau(k)));%liczba krokow
    x(kk:kk+ni-1,:)=xtmp(1:end-1,:);%zapis x
    t(kk:kk+ni-1)=tau(k)+ttmp(1:end-1,:);%zapis t
    uk(kk:kk+ni-1,:)=kron(u(k,:),ones(ni,1));%zapis uk
    kk=kk+ni;nseg(k)=kk;
end
x(end,:)=xtmp(end,:);% dodaje stan koncowy
uk(end,:)=u(end,:);% dodaje sterowanie na koncu
t(end)=tau(end);%dodaje czas koncowy
% t-wektor czasow [0, t1, ..., tn],
```

Powyższa procedura zwraca wektor czasów $t = [t_0, ..., t_{n_t}] \in \mathbb{R}^{n_t}$, macierz x wymiaru $n_t \times n$, macierz uk, wymiaru $n_t \times n_u$, zawierającą wartości sterowania w chwilach t_k oraz wektor $nseg \in \mathbb{R}^m$, zawierający indeksy czasów przełączeń sterowania tj. $t(nseg(j)) = \tau_j$.

Przykład całkowania równań (8-10)



Rysunek: Rozwiązanie równań (8-10), uzyskane za pomocą powyższego algorytmu oraz sterowanie (u dołu). Dla lepszej czytelności, x_2 podzielono przez 10.

Funkcja celu dla zadań sterowania docelowego i czasooptymalnego

Dla zadań sterowania docelowego lub czasooptymalnego, funkcja celu ma postać

$$J(u,T) = T + \tfrac{1}{2}(x(T) - x_f)^T W (x(T) - x_f), \tag{13}$$

gdzie x_f, jest pożądanym stanem docelowym. Macierz $W = W^T \geqslant 0$, jest macierzą wag w postaci

$$W = \rho \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}, \tag{14}$$

gdzie $\rho > 0$, jest dostatecznie dużym współczynnikiem kary. Jeżeli zmienna x_i, nie jest brana pod uwagę, to przyjmujemy $w_i = 0$.

Implementacja funkcji celu

```
function [q,g]=cost_fun(tau,u,x0,xf,W,MDNS)
% xf - stan docelowy, W=W^T>0 - macierz wag
%calkowanie rownania stanu
[~,x]=get_tx(tau,u,x0,MDNS);
% roznica pomiedzy stanem koncowym i docelowym
dxend=x(end,:)'-xf;
% funkcja celu
q=tau(end)+0.5*dxend'*W*dxend;
if nargout>1
    %opcjonalne obliczenie gradientu
    g=get_grad(tau,u,x0,xf,W,MDNS);
end
```

Równania sprzężone

Dla systemu

$$\dot{x} = f(t, x, u), t \in [0, T], x(0) = x_0, \tag{15}$$

oraz kosztu $J(u, T) = q(T, x(T))$, równania sprzężone mają postać

$$\dot{\psi} = -A(t, x(t), u(t))^T \psi, \psi(T) = -\nabla_x q(T, x(T)), \tag{16}$$

gdzie elementy macierzy A, dane są równością

$$A_{i,j}(t, x(t), u(t)) = \frac{\partial f_i(t, x(t), u(t)}{\partial x_j}. \tag{17}$$

Równanie sprzężone należy rozwiązywać wstecz w czasie. Rozwiązywanie równania różniczkowego w czasie odwróconym, oznacza rozwiązywanie w przód równania z prawą stroną pomnożoną przez -1, tj.

$$\dot{\xi} = A(T - t, x(T - t), u(T - t))^T \xi, \xi(0) = -\nabla_x q(T, x(T)). \tag{18}$$

Prawa strona równań sprzężonych

Dla równań (8-10), mamy

$$\dot{\psi}_1 = \omega_0^2 \cos(x_1)\psi_2, \tag{19}$$

$$\dot{\psi}_2 = -\psi_1 + 2\xi\omega_0\psi_2, \tag{20}$$

$$\dot{\psi}_3 = -k\psi_2 + \frac{1}{\tau}\psi_3. \tag{21}$$

Jeżeli funkcja kosztu jest dana wzorem (13), to warunek końcowy ma postać

$$\psi(T) = -W(x(T) - x_f). \tag{22}$$

Prawa strona równań sprzężonych-implementacja

```
function dp=prhs(t,x,u,p)
%Adjoint equations
dp=zeros(3,1);
w0=9;ksi=0.01;tau=0.04;k=250;
dp(1)=w0^2*cos(x(1))*p(2);
dp(2)=-p(1)+2*ksi*w0*p(2);
dp(3)=-k*p(2)+p(3)/tau;
```

Do całkowania równań sprzężonych wykorzystujemy odpowiednio zmodyfikowaną metodę RK4, przy czym wartości rozwiązania równania stanu w punktach $t_k + \frac{1}{2}h$, obliczamy ze wzoru

$$x(t_k + \tfrac{1}{2}h) \approx \tfrac{1}{2}(x_k + x_{k+1}). \tag{23}$$

Wartości $x(t_k + \frac{1}{2}h)$, oznaczono w poniższym kodzie przez $x05$. Kod wykorzystuje, obliczone wcześniej przez funkcję get_tx, czasy t_k, wartości sterowań $u(t_k)$ oraz trajektorię $x(t_k)$.

Całkowanie równań sprzężonych - implementacja

```
function p=rk4p(pf,t,x,uk)
nt=length(t);np=length(pf);x=flipud(x);uk=flipud(uk);
t=t(end)-t;t=flipud(t);p=zeros(nt,np);p(1,:)=pf';
tmp=zeros(np,1);ptmp=pf;tt=0;
dp1=zeros(np,1);dp2=zeros(np,1);
dp3=zeros(np,1);dp4=zeros(np,1);
for k=1:nt-1
  h=t(k+1)-t(k);h_2=h/2;h_6=h/6;h_26=2*h_6;
  dp1=-prhs(tt,x(k,:)',uk(k+1,:)',ptmp);
  tmp=ptmp+h_2*dp1;tt=tt+h_2;
  x05=0.5*(x(k,:)+x(k+1,:))';
  dp2=-prhs(tt,x05,uk(k+1,:)',tmp);tmp=ptmp+h_2*dp2;
  dp3=-prhs(tt,x05,uk(k+1,:)',tmp);tmp=ptmp+h*dp3;tt=tt+h_2;
  dp4=-prhs(tt,x(k+1,:)',uk(k+1,:)',tmp);
  ptmp=ptmp+h_6*(dp1+dp4)+h_26*(dp2+dp3);
  p(k+1,:)=ptmp';
end
p=flipud(p);
```

Pochodne funkcji kosztu

Przyrost funkcji kosztu, spowodowany zmianą sterowania u, o Δu, wynosi

$$\Delta J = -\int_0^T \phi(t)\Delta u(t)dt, \tag{24}$$

gdzie funkcja

$$\phi(t) = \nabla_u H(t, \psi(t), x(t), u(t)), \tag{25}$$

jest nazwana funkcją przełączającą oraz

$$H(t, \psi, x, u) = \psi^T f(t, x, u), \tag{26}$$

jest Hamiltonianem. Sterowanie optymalne maksymalizuje, w każdej chwili czasu, funkcję H, tj.

$$u^*(t) = \mathrm{argmax}_{u \in [u_{min}, u_{max}]} H(t, \psi(t), x(t), u).$$

Gradient funkcji kosztu

Jeżeli u jest funkcją przedziałami stałą, scharakteryzowaną przez czasy przełączeń (11) oraz wartości na przedziałach stałości (12), to ze wzorów (24-26) oraz (13-18) wynika, że

$$\frac{\partial J}{\partial \tau_i} = \phi(\tau_i)(u(\tau_i^+) - u(\tau_i^-)), i = 1, 2, ..., m-1, \tag{27}$$

$$\frac{\partial J}{\partial u_i} = -\int_{\tau_{i-1}}^{\tau_i} \phi(t)dt, i = 1, 2, ..., m, \tag{28}$$

$$\frac{\partial J}{\partial T} = 1 - H(T, \psi(T), x(T), u(T^-)), \tag{29}$$

przy czym $u(\tau_i^-), u(\tau_i^+)$, oznaczają lewą i prawą granicę sterowania w punkcie $t = \tau_i$.

Gradient funkcji kosztu-implementacja

Jeżeli optymalizacja dotyczy tylko czasów przełączeń, to gradient dany wzorami (27) i (29) można obliczyć za pomocą poniższej funkcji.

```
function g=get_grad(tau,u,x0,xf,W,MDNS)
%solve state equation, forward
[t,x,uk,nseg]=get_tx(tau,u,x0,MDNS);
%final cond. psi
pf=-W*(x(end,:)'-xf);
%solve adjoint equation, backward
p=rk4p(pf,t,x,uk);
ng=length(tau);g=zeros(ng,1);
for k=1:ng-1
    hu=get_hu(tau(k),p(nseg(k),:)',x(nseg(k),:)',u(k+1,:)');
    %derrivative w.r.t. switching time
    g(k)=hu*(u(k+1)-u(k));
end
%derrivative w.r.t final time
g(end)=1-pf'*rhs(tau(end),x(end,:)',u(end,:)');
```

Funkcja przełączająca-implementacja

Dla równań (8-10), mamy

$$H(\psi, x, u) = \psi_1 x_2 + \psi_2(-\omega_0^2 \sin(x_1) - 2\xi\omega_0 x_2 + k x_3) + \psi_3 \tau^{-1}(u - x_3)$$

oraz

$$\phi = H_u = \frac{\psi_3}{\tau}.$$

Ponieważ H zależy liniowo od sterowania, to maksimum H względem u, osiągane jest na ograniczeniach. A zatem sterowanie jest typu *bang-bang* i w każdej chwili czasu przyjmuje jedną z wartości u_{min}, albo u_{max}.

```
function hu=get_hu(t,p,x,u)
%funkcja przelaczajaca,
%pochodna hamiltonianu wzgledem sterowania
tau=0.04;hu=p(3)/tau;
```

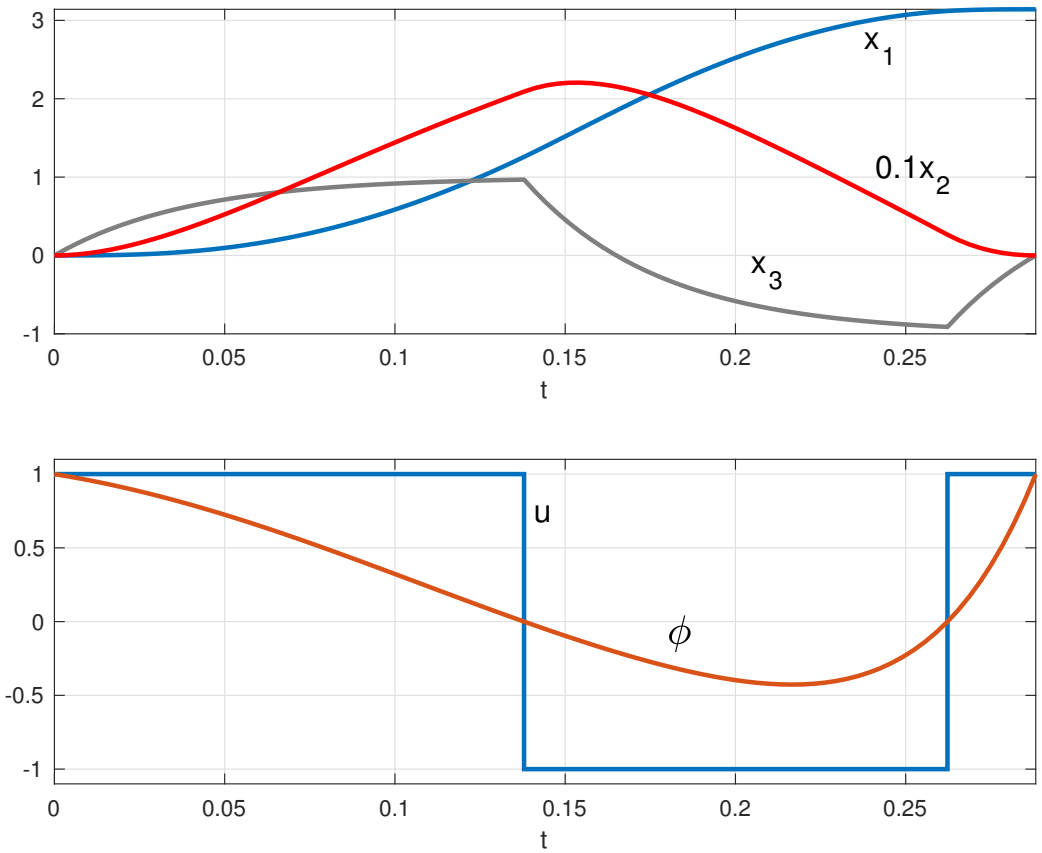
Przykład

Celem sterowania jest przeprowadzenie systemu (8-10), ze stanu początkowego $x_0 = [0, 0, 0]^T$, do stanu docelowego $x_f = [\pi, 0, 0]^T$. Zadanie rozwiązujemy poprzez minimalizację kosztu (13). Do minimalizacji funkcji kosztu wykorzystano funkcję *fmincon* Matlaba. Przy poszukiwaniu minimum należy uwzględnić ograniczenia $0 \leqslant \tau_1 \leqslant \tau_2, ..., \leqslant \tau_m = T$. Ograniczenia te są liniowe i można je zapisać w postaci $A\tau \leqslant b$. Poniższy kod umożliwia, zwykle po wielu nieudanych próbach, znalezienie rozwiązania optymalnego w klasie sterowań *bang-bang*.

Przykład

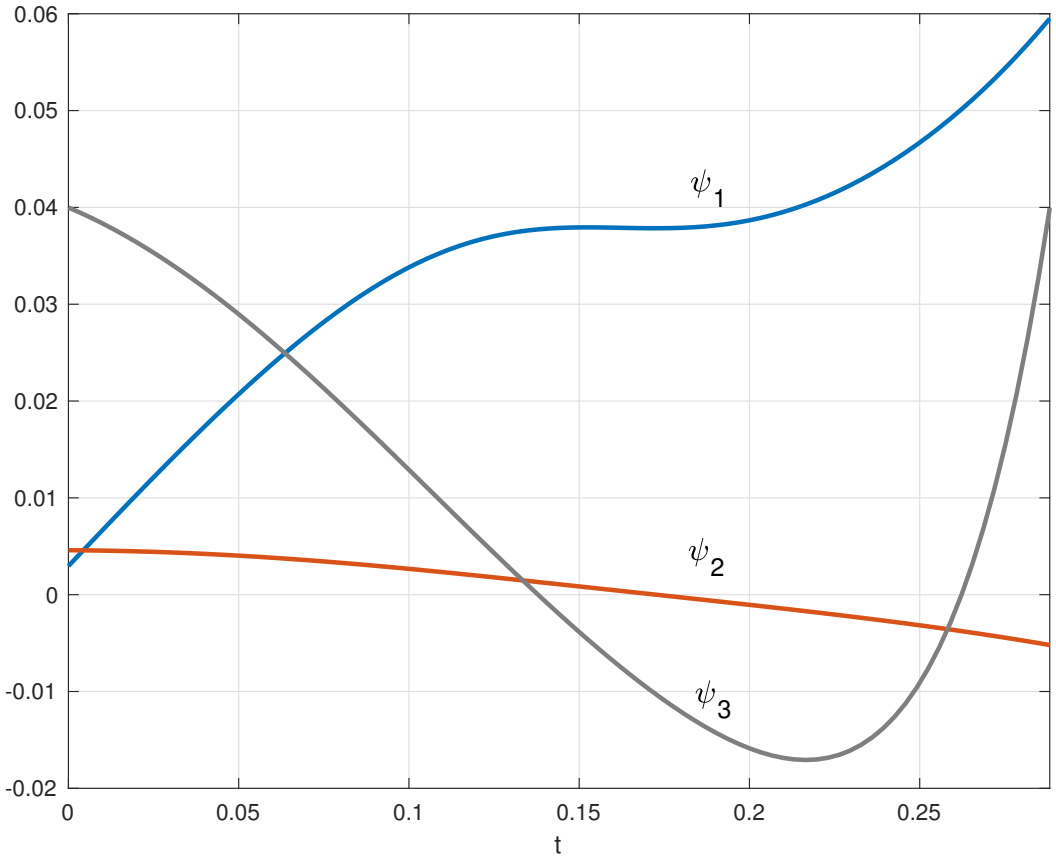
```
clear;close all;tauopt=[.15 .30 .40]';u=1.0*[1;-1;1];
x0=[0;0;0];xf=[pi;0;0];W=1000*eye(3);MDNS=300;
qh=@(tauopt) cost_fun(tauopt,u,x0,xf,W,MDNS);
nb=length(tauopt);b=0.0*ones(nb,1);A=-eye(nb);
for k=2:nb, A(k,k-1)=1;end %ograniczenia
options=optimoptions('fmincon');
options.SpecifyObjectiveGradient=true;
options.Display='iter';options.Algorithm='interior-point';
tauopt=fmincon(qh,tauopt,A,b,[],[],[],[],[],options);
[t,x,uk,nseg]=get_tx(tauopt,u,x0,MDNS);pf=-W*(x(end,:)'-xf);
p=rk4p(pf,t,x,uk);nt=length(t);hu=zeros(nt,1);
for k=1:nt
    hu(k)=get_hu(t(k),p(k,:)',x(k,:)',uk(k,:)');
end
hu=hu/max(abs(hu));
subplot(211);h=plot(t,x);set(h,'linewidth',2);
subplot(212);h=stairs(t,uk/max(uk));set(h,'linewidth',2);
hold on;h=plot(t,hu);grid;set(h,'linewidth',2);
axis([0 t(end) -1.1 1.1]);hold off
```

Rozwiązanie optymalne



Rysunek: Trajektoria (u góry), oraz sterowanie i funkcja przełączająca (u dołu), uzyskane za pomocą powyższego algorytmu. Dla lepszej czytelności, x_2 podzielono przez 10. Ponieważ funkcja ϕ, jest dodatnia wszędzie tam, gdzie sterowanie jest maksymalne i ujemna dla minimalnych wartości sterowania, to sterowanie u, jest lokalnie optymalne.

Zmienne sprzężone



Rysunek: Zmienne sprzężone odpowiadające rozwiązaniu optymalnemu.

Sprawdzenie poprawności obliczenia gradientu

Poprawność obliczenia gradientu można sprawdzić, porównując gradient obliczony za pomocą wzorów (27) i (29), z gradientem obliczonym metodą różnic skończonych. Przypomnijmy, że symetryczną aproksymację pochodnej funkcji $g(x_1, ..., x_n)$, obliczamy ze wzoru

$$\frac{\partial g}{\partial x_i} \approx \frac{g(x_1, ..., x_i + \epsilon, ..., x_n) - g(x_1, ..., x_i - \epsilon, ..., x_n)}{2\epsilon}, \epsilon > 0.$$

Dokładność wzoru jest rzędu $o(\epsilon^2)$. Stosując tę metodę do funkcji kosztu (13), otrzymamy prosty test poprawności obliczenia gradientu.

Sprawdzenie poprawności obliczenia gradientu względem czasów przełączeń i czasu końcowego - implementacja

```
clear;close all
tau=[.1;.2;0.3];u=1.0*[-1;1;-1];x0=[0;0;0];
xf=[pi;0;0];W=10*eye(3);MDNS=2000;
ep=1e-7;gnum=zeros(length(tau),1);
[~,g]=cost_fun(tau,u,x0,xf,W,MDNS);
for k=1:length(tau)
    ei=zeros(length(tau),1);ei(k)=1;
    taup=tau+ei*ep;taum=tau-ei*ep;
    [fp,~]=cost_fun(taup,u,x0,xf,W,MDNS);
    [fm,~]=cost_fun(taum,u,x0,xf,W,MDNS);
    gnum(k)=(fp-fm)/(2*ep);
end
%porownanie g - r. sprz., gnum - rozn.
[g';gnum']
```

Schodkowa aproksymacja sterowania

Załóżmy, że funkcja kosztu ma postać

$$J(u,T) = \int_0^T L(x,u)dt + q(x(T)), \tag{30}$$

gdzie horyzont T, jest ustalony oraz funkcje $L, q \in C^1$. Zakładamy, że sterowanie jest przedziałami stałe

$$u(t) = u_k, t \in [\tau_{k-1}, \tau_k), \tau_0 = 0, k = 1, .., m, \tag{31}$$

przy czym chwile czasu τ_k, są ustalone oraz $T = \tau_m$. Zmiennymi decyzyjnymi są liczby u_k.

Hamiltonian i równania sprzężone mają postać

$$H(t, \psi, x, u) = \psi^T f(t, x, u) - L(x, u), \tag{32}$$

$$\dot{\psi} = -A(t, x(t), u(t))^T \psi + L_x(x, u), \psi(T) = -q_x(x(T)). \tag{33}$$

Aby obliczyć całkę, występującą we wzorze (30), definiujemy nową zmienną stanu

$$\dot{x}_{n+1} = L(x, u), x_{n+1}(0) = 0 \tag{34}$$

i dołączamy ją do równań stanu. Aby obliczyć całki, we wzorze (28), dołączamy do równań sprzężonych nową zmienną stanu

$$\dot{\psi}_{n+1} = H_u, \psi_{n+1}(T) = 0. \tag{35}$$

Po rozwiązaniu rozszerzonych równań stanu (w przód) i równań sprzężonych (w tył), wartość funkcji kosztu i pochodne we wzorze (28), obliczamy ze wzorów

$$J(u_1, ..., u_m) = x_{n+1}(T) + q(x(T)), \tag{36}$$

$$\frac{\partial J(u_1, ..., u_m)}{\partial u_i} = \psi_{n+1}(\tau_{i-1}) - \psi_{n+1}(\tau_i), i = 1, ..., m. \tag{37}$$

Zadania nadążania i stabilizacji

W zadaniach nadążania lub stabilizacji, często rozpatruje się kwadratowe funkcje kosztu

$$J(u) = \frac{1}{2}\int_0^T (x(t) - x_{ref}(t))^T Q(x(t) - x_{ref}(t)) +$$

$$+ (u(t) - u_{ref}(t))^T R(u(t) - u_{ref}(t))dt + \frac{1}{2}(x(T) - x_f)^T W(x(T) - x_f), \tag{38}$$

gdzie x_{ref}, jest trajektorią pożądaną, u_{ref} jest sterowaniem odpowiadającym trajektorii x_{ref} oraz x_f, jest pożądanym stanem docelowym. Macierze Q, R, W, są symetryczne i dodatnio pół-określone. Dla takich zadań, równania sprzężone i funkcja przełączająca mają postać

$$\dot{\psi} = -A^T\psi + Q(x - x_{ref}), \psi(T) = -W(x(T) - x_f). \tag{39}$$

$$\phi = H_u = \nabla_u(\psi^T f(t, x, u)) + R(u - u_{ref}). \tag{40}$$

Hamiltonian maximizer

Jeżeli u jest rozwiązaniem optymalnym, to z warunku maksimum hamiltonianu wynika, że albo

$$\nabla_u(\psi^T f(t, x(t), u(t))) + R(u(t) - u_{ref}(t)) = 0, \qquad (41)$$

albo $u(t) = u_{min}$, albo $u(t) = u_{max}$. A zatem, sterowanie optymalne albo spełnia równanie (41), albo leży na górnym bądź dolnym ograniczeniu.

Przykład

Dla równań (8-10), przyjmujemy funkcję kosztu

$$J(u) = \frac{1}{2}\int_0^T (x_1(t) - \pi)^2 + u(t)^2 dt, \tag{42}$$

warunek początkowy $x_0 = (0, 0, 0)^T$ oraz $T = 1$. Przedział $[0, T]$, dzielimy na $m = 100$, części i szukamy aproksymacji schodkowej $u_1, ..., u_m$. Hamiltonian

$$\begin{aligned} H(\psi, x, u) &= \psi_1 x_2 + \psi_2(-\omega_0^2 \sin(x_1) - 2\xi\omega_0 x_2 + k x_3) + \\ &+ \psi_3 \tau^{-1}(u - x_3) - \frac{1}{2}(x_1(t) - \pi)^2 - \frac{1}{2}u(t)^2. \end{aligned} \tag{43}$$

Funkcja przełączająca

$$\phi = \tau^{-1}\psi_3 - u. \tag{44}$$

Rozwiązanie równania (41), *hamiltonian maximizer*, jest dane wzorem

$$u = \tau^{-1}\psi_3. \tag{45}$$

Równania sprzężone, wraz z dodatkowa zmienną sprzężoną, mają następującą postać

$$\dot{\psi}_1 = \omega_0^2 \cos(x_1)\psi_2 + x_1(t) - \pi, \tag{46}$$

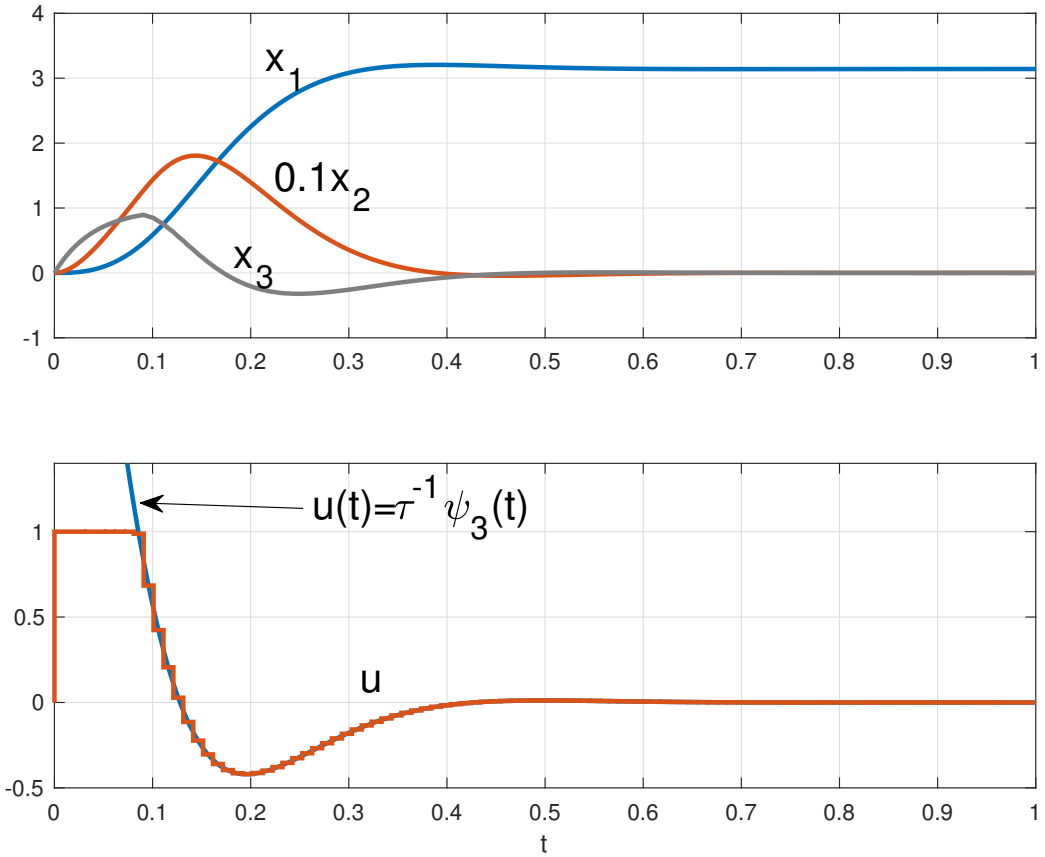
$$\dot{\psi}_2 = -\psi_1 + 2\xi\omega_0\psi_2, \tag{47}$$

$$\dot{\psi}_3 = -k\psi_2 + \frac{1}{\tau}\psi_3, \tag{48}$$

$$\dot{\psi}_4 = \tau^{-1}\psi_3 - u. \tag{49}$$

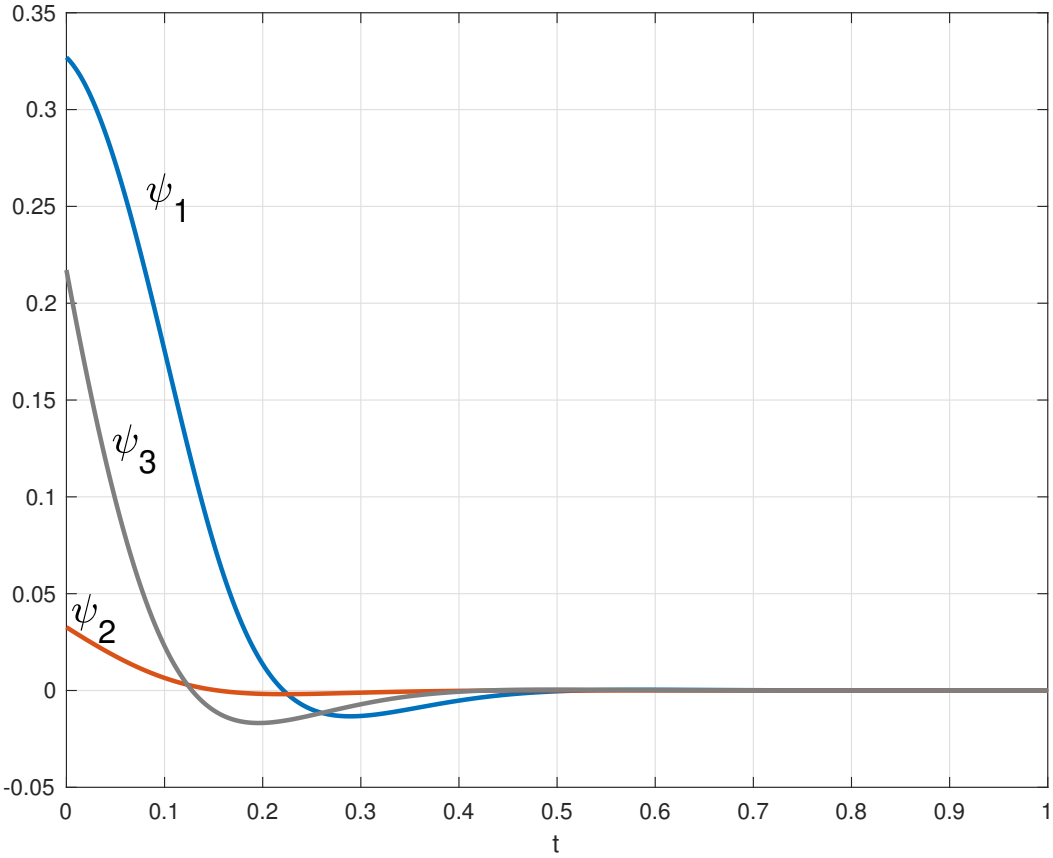
Obliczoną aproksymację schodkową sterowania optymalnego oraz trajektorię stanu i rozwiązanie równań sprzężonych, pokazano na rysunkach poniżej. Kod, za pomocą którego zrealizowano obliczenia, znajduje się poniżej.

Aproksymacja schodkowa rozwiązania optymalnego



Rysunek: Trajektoria (u góry), oraz sterowanie (u dołu). Dla lepszej czytelności, x_2 podzielono przez 10. Na dolnym wykresie pokazano też sterowanie wynikające z równania (45).

Zmienne sprzężone.



Rysunek: Zmienne sprzężone odpowiadające rozwiązaniu z rys. powyżej.

Kod dla aproksymacji schodkowej, prawa strona równań stanu

```
function dx=rhs_s(t,x,u)
dx=zeros(4,1);w0=9;ksi=0.01;tau=0.04;k=250;
dx(1)=x(2);
dx(2)=-w0^2*sin(x(1))-2*ksi*w0*x(2)+k*x(3);
dx(3)=(u-x(3))/tau;
% Additional variable x'=L(x,u).
dx(4)=0.5*((x(1)-pi)^2+u^2);
```

Dodatkowa zmienna stanu do obliczenia funkcji kosztu (30)

$$x_4(t) = \int_0^t L(x(s), u(s))ds, J(u) = x_4(T) + q(x(T)). \qquad (50)$$

Kod dla aproksymacji schodkowej, prawa strona równań sprzężonych

```
function dp=prhs_s(t,x,u,p)
% Adjoint equations
dp=zeros(4,1);
w0=9;ksi=0.01;tau=0.04;k=250;
dp(1)=w0^2*cos(x(1))*p(2)+(x(1)-pi);
dp(2)=-p(1)+2*ksi*w0*p(2);
dp(3)=-k*p(2)+p(3)/tau;
% Additional variable p'=H_u(t)
dp(4)=p(3)/tau-u;
```

Dodatkowa zmienna stanu do obliczenia całek (28)

$$\psi_4(t) = \int_0^t \phi(s)ds, \frac{\partial J}{\partial u_i} = \psi_4(\tau_{i-1}) - \psi_4(\tau_i), i = 1, ..., m. \quad (51)$$

Kod dla aproksymacji schodkowej, funkcja kosztu

```
function [q,g]=cost_fun_s(u,tau,x0,xf,W,MDNS)
% xf - stan docelowy, W=W^T>0 - macierz wag
%calkowanie rownania stanu
[~,x]=get_tx_s(tau,u,x0,MDNS);
% roznica pomiedzy stanem koncowym i docelowym
dxend=x(end,1:end-1)'-xf;
% funkcja celu
q=x(end,end)+0.5*dxend'*W*dxend;
if nargout>1
    %opcjonalne obliczenie gradientu
    g=get_grad_s(tau,u,x0,xf,W,MDNS);
end
```

Kod dla aproksymacji schodkowej, rozwiązanie równania (41), *hamiltonian maximizer*

```
function u=get_hmax(t,p,x,u)
%Hamiltonian maximizer
%the solution of H_u(t, p, x,u)=0
tau=0.04;u=p(:,3)/tau;
```

Kod dla aproksymacji schodkowej, RK4

```
function [t,x]=rk4_s(x0,u,tf,MDNS)
x0=x0(:);u=u(:);n=length(x0);
nt=1+floor(tf*MDNS);h=tf/nt;
x=zeros(nt+1,n);t=zeros(nt+1,1);
x(1,:)=x0';
tmp=zeros(n,1);xtmp=x0;tt=0;
dx1=zeros(n,1);dx2=zeros(n,1);
dx3=zeros(n,1);dx4=zeros(n,1);
h_2=h/2;h_6=h/6;h_26=2*h_6;
for i=1:nt
  dx1=rhs_s(tt,xtmp,u);tmp=xtmp+h_2*dx1;tt=tt+h_2;
  dx2=rhs_s(tt,tmp,u);tmp=xtmp+h_2*dx2;
  dx3=rhs_s(tt,tmp,u);tmp=xtmp+h*dx3;tt=tt+h_2;
  dx4=rhs_s(tt,tmp,u);
  xtmp=xtmp+h_6*(dx1+dx4)+h_26*(dx2+dx3);
  x(i+1,:)=xtmp';t(i+1)=tt;
end
```

Kod dla aproksymacji schodkowej, RK4 na przedziałach

```
function [t,x,uk,nseg]=get_tx_s(tau,u,x0,MDNS)
[ntau,nu]=size(u);n=length(x0);tau=[0;tau];
nt=1+sum(1+floor(MDNS*diff(tau)));
x=zeros(nt,n);uk=zeros(nt,nu);
t=zeros(nt,1);nseg=zeros(ntau,1);kk=1;
for k=1:ntau
    % RK4
    [ttmp,xtmp]=rk4_s(x0,u(k,:)',tau(k+1)-tau(k),MDNS);
    x0=xtmp(end,:)';%warunek poczatkowy do nastepnego
    ni=1+floor(MDNS*(tau(k+1)-tau(k)));%liczba krokow
    x(kk:kk+ni-1,:)=xtmp(1:end-1,:);%zapis x
    t(kk:kk+ni-1)=tau(k)+ttmp(1:end-1,:);%zapis t
    uk(kk:kk+ni-1,:)=kron(u(k,:),ones(ni,1));%zapis uk
    kk=kk+ni;nseg(k)=kk;
end
x(end,:)=xtmp(end,:);% dodaje stan koncowy
uk(end,:)=u(end,:);% dodaje sterowanie na koncu
t(end)=tau(end);%dodaje czas koncowy
% t-wektor czasow [0, t1, ..., tn],
```

Kod dla aproksymacji schodkowej, RK4 dla równań sprzężonych

```
function p=rk4p_s(pf,t,x,uk)
nt=length(t);np=length(pf);x=flipud(x);uk=flipud(uk);
t=t(end)-t;t=flipud(t);p=zeros(nt,np);p(1,:)=pf';
tmp=zeros(np,1);ptmp=pf;tt=0;dp1=zeros(np,1);dp2=zeros(np,1);
dp3=zeros(np,1);dp4=zeros(np,1);
for k=1:nt-1
  h=t(k+1)-t(k);h_2=h/2;h_6=h/6;h_26=2*h_6;
  dp1=-prhs_s(tt,x(k,:)',uk(k+1,:)',ptmp);
  tmp=ptmp+h_2*dp1;tt=tt+h_2;x05=0.5*(x(k,:)+x(k+1,:))';
  dp2=-prhs_s(tt,x05,uk(k+1,:)',tmp);tmp=ptmp+h_2*dp2;
  dp3=-prhs_s(tt,x05,uk(k+1,:)',tmp);tmp=ptmp+h*dp3;tt=tt+h_2;
  dp4=-prhs_s(tt,x(k+1,:)',uk(k+1,:)',tmp);
  ptmp=ptmp+h_6*(dp1+dp4)+h_26*(dp2+dp3);
  p(k+1,:)=ptmp';
end
p=flipud(p);
```

Kod dla aproksymacji schodkowej, obliczenie gradientu

```
function g=get_grad_s(tau,u,x0,xf,W,MDNS)
%solve state equation, forward
[t,x,uk,nseg]=get_tx_s(tau,u,x0,MDNS);
%final cond. psi
pf=[-W*(x(end,1:end-1)'-xf);0];
%solve adjoint equation, backward
ps=rk4p_s(pf,t,x,uk);phi_int=ps(:,end);
ng=length(u);g=zeros(ng,1);nseg=[1;nseg];
for k=1:ng
    %derrivative w.r.t. u_k
    g(k)=phi_int(nseg(k))-phi_int(nseg(k+1));
end
```

Kod dla aproksymacji schodkowej, solver *fmincon*

```
clear;close all;
nu=20;tf=1;umax=1;umin=-1;
tau=(tf/nu)*[1:nu]';uopt=zeros(nu,1);
x0=[0;0;0;0];xf=[pi;0;0];W=0*eye(3);MDNS=1e3;
qh=@(uopt) cost_fun_s(uopt,tau,x0,xf,W,MDNS);
nu=length(uopt);LB=-ones(nu,1);UB=ones(nu,1);%ograniczenia
options=optimoptions('fmincon');
options.SpecifyObjectiveGradient=true;
options.Display='iter';options.Algorithm='interior-point';
uopt=fmincon(qh,uopt,[],[],[],[],LB,UB,[],options);
[t,x,uk,nseg]=get_tx_s(tau,uopt,x0,MDNS);
pf=[-W*(x(end,1:end-1)'-xf);0];
p=rk4p_s(pf,t,x,uk);hmax=get_hmax(t,p,x,uk);
subplot(211);h=plot(t,[x(:,1) x(:,2)/10
    x(:,3)]);set(h,'linewidth',2);
subplot(212);h=plot(t,hmax);set(h,'linewidth',2);
hold on;h=stairs(t,uk);set(h,'linewidth',2);
axis([0 t(end) -1.1 1.1]);hold off
```

Sprawdzenie poprawności obliczenia gradientu względem u_i

```
clear;close all;tau=[.2 .3 .4 .5]';u=1.0*[1;-1;1;1];
x0=[0;0;0;0];xf=[pi;0;0];W=1000*eye(3);MDNS=1e3;
ep=1e-9;gnum=zeros(length(u),1);
[q,g]=cost_fun_s(u,tau,x0,xf,W,MDNS);
for k=1:length(tau)
    ei=zeros(length(u),1);ei(k)=1;
    up=u+ei*ep;um=u-ei*ep;
    fp=cost_fun_s(up,tau,x0,xf,W,MDNS);
    fm=cost_fun_s(um,tau,x0,xf,W,MDNS);
    gnum(k)=(fp-fm)/(2*ep);
end
%porownanie g - r. sprz., gnum - rozn.
[g';gnum']
```

Posługując się opisaną powyżej metodyką, można konstruować inne, adekwatne dla danego zadania, aproksymacje sterowania optymalnego. W szczególności, można jednocześnie optymalizować zarówno wartości sterowania, jak i czasy przełączeń oraz czas końcowy. Możliwe jest też zastosowanie aproksymacji za pomocą łamanej (funkcją przedziałami liniową) oraz rozszerzenie na zdania z większą liczbą sterowań.