On the OKID Method

The objective of this note is to afford the beginner for learning the observer/Kalman filter identification (OKID) method, where a new approach to improve the transient responses of i) the traditional OKID method and ii) the digital redesign-based observer and/or Kalman filter are also respectively presented.

* **A new approach to improve the transient response:**

Let the dimension of the system matrix . A good initial state  for the observer/Kalman can be obtained as follows, so that the transient response during  can be further improved.

Step 1: During the first round to test the performance of OKID method-based observer/Kalman filter, set the initial condition be . Collect the state  or any one of steady-state .

Step 2: Select  (or any steady-state ) as the initial state  for the practical observer/Kalman filter to be used. Then, repeat the control process started at  again. As a result, the transient response during  can be further improved, except for the initial condition .

Example 1: Given the system presented by



where

 



the corresponding discrete-time model for the sampling time  is given by



where

 

Eigenvalues of  are .

Approach 1: The traditional OKID method with the improved transient response

Let  be created by normal distribution white noise signals and some parameters for OKID method are . The resulting OKID method-based observer is given by



where  is selected as the collected state at  during the first round for testing the performance of OKID method and

 Notice that  Figs. 1-4 show the tracking performance of the traditional OKID method with the improved transient response. 







Approach 2: The digital redesign-based observer and/or Kalman filter with the improved transient response

Remark 1: The traditional OKID method uses the past output measurement  to estimate the state ; however, the digital redesign-based observer and/or Kalman filter uses the current output measurement  to estimate the state .



where  is selected as the collected state at  during the first round for testing the performance of OKID method and



 is determined by solving the discrete-time LQR for the fictitious system presented by the triple  where  or some other approach, such as the digital redesign approach (omitted at here). Figs. 5-8 show the tracking performance of the traditional OKID method with the improved transient response.







Matlab code

Step1: Produce input and output

clc; clear all; close all;

%% System matrix

A = [-1.0, 0.0, 0.00,-0.5; % ¨t²Î°Ñ¼ÆA

0.0,-0.5,-0.25,-0.5;

0.0, 0.0,-0.50, 0.0;

0.0, 0.0, 0.00,-0.5];

B =[ 0.50, 0.50 1.0; % ¨t²Î°Ñ¼ÆB

-0.25,-0.25 0.7;

0.50, 0.50 -0.9;

0.50,-0.50 0.5];

C = [ 1, 1, 0,-1.5; % ¨t²Î°Ñ¼ÆC

0, 1, 0,-1.0];

D = [0, 0 ,0; % ¨t²Î°Ñ¼ÆD

0, 0, 0];

initial\_state=[0.1 % ª¬ºAªì©l­È

0.2

0.3

0.4];

[p m] = size(D); % p¬°¿é¥Xºû«×¡Am¬°¿é¤Jºû«×

n = size(A); % n¬°ª¬ºAºû«×

Ts = 0.1; % Â÷´²±Ä¼Ë®É¶¡

Tend = 10;

[G,H] = c2d(A,B,Ts); % ±N¨t²Î°Ñ¼Æ°µÂ÷´²¤Æ

%% Reference

t\_ds = 0:Ts:Tend;

u = 0.2\*randn(m,size(t\_ds,2));

ii = 0;

x(:,1) = initial\_state;

for t = 1 : size(t\_ds,2)

ii = ii+1;

x(:,ii+1) = G\*x(:,ii) + H\*u(:,ii);

y(:,ii) = C\*x(:,ii);

end

length = size(u,2);

save IODATA.mat u y length

Step2: Produce  initial

clc; clear all; close all; warning off; pause(0.01)

load IODATA.mat

%%%%% OKID ¹Lµ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[G\_ok,H\_ok,C\_ok,D\_ok,L\_ok] = Auxi\_OKID\_JXL(u,y,2,3,0);

%%%%% ­«²¹Lµ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[th,xh,yh,eh,Gd,Hd,Ld,x\_Initial] = Auxi\_OKID\_Process(u,y,G\_ok,H\_ok,C\_ok,D\_ok,L\_ok,(10/101),1,1e8);

figure(1); plot(1:length,y(1,:),'.b-',1:length,yh(1,:),'xr:'); xlim([0,length]);

ylabel('y\_1 vs. yh\_1');

xlabel('Time (sec)');

figure(2); plot(1:length,y(1,:)-yh(1,:),'k'); xlim([0,length]);

ylabel('y\_1 ¡Ð yh\_1');

xlabel('Time (sec)');

figure(3); plot(1:length,y(2,:),'.b-',1:length,yh(2,:),'xr:'); xlim([0,length]);

ylabel('y\_2 vs. yh\_2');

xlabel('Time (sec)');

figure(4); plot(1:length,y(2,:)-yh(2,:),'k'); xlim([0,length]);

ylabel('y\_2 ¡Ð yh\_2');

xlabel('Time (sec)');

save x\_Initial.mat x\_Initial

Step3: Result

clc; clear all; close all; warning off; pause(0.01)

load IODATA.mat

load x\_Initial.mat

%%%%% OKID ¹Lµ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[G\_ok,H\_ok,C\_ok,D\_ok,L\_ok] = Auxi\_OKID\_JXL(u,y,2,3,0);

%%%%% ­«²¹Lµ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[th,xh,yh,eh,Gd,Hd,Ld,x\_Initial] = Auxi\_OKID\_Process\_step2(u,y,G\_ok,H\_ok,C\_ok,D\_ok,L\_ok,(10/101),1,1e8,x\_Initial);

figure(1); plot(1:length,y(1,:),'.b-',1:length,yh(1,:),'xr:'); xlim([0,length]);

ylabel('y\_1 vs. yh\_1');

xlabel('Time (sec)');

figure(2); plot(1:length,y(1,:)-yh(1,:),'k'); xlim([0,length]);

ylabel('y\_1 ¡Ð yh\_1');

xlabel('Time (sec)');

figure(3); plot(1:length,y(2,:),'.b-',1:length,yh(2,:),'xr:'); xlim([0,length]);

ylabel('y\_2 vs. yh\_2');

xlabel('Time (sec)');

figure(4); plot(1:length,y(2,:)-yh(2,:),'k'); xlim([0,length]);

ylabel('y\_2 ¡Ð yh\_2');

xlabel('Time (sec)');

save x\_Initial.mat x\_Initial

Sub Program

function [G,H,C,D,Lo,Singu,Vn,Sn] = Auxi\_OKID\_JXL(ID\_u,ID\_y,q,Mult\_Num,D\_exist)

% Singu\_Value = diag(Singu); % ¤è«K½Æ»s¨Ï¥Î(¨ú¥X©\_²§­È)

% ¢d¢d ¿é¥X¤Jºû«×½T©w ¢d¢d

[m,u\_l] = size(ID\_u); % m ¬°¿é¤J­Ó¼Æ

[p,y\_l] = size(ID\_y); % p ¬°¿é¥X­Ó¼Æ

% ¢d¢d ¬ÛÃö°Ñ¼Æ¹w³]­È³]©w(Àq»³]©w) ¢d¢d

Check\_q = exist('q');

if (Check\_q == 0 | q == 0)

q = 1;

end

% -----

Check\_Mult\_Num = exist('Mult\_Num');

if (Check\_Mult\_Num == 0 | Mult\_Num == 0)

Mult\_Num = 2;

end

% -----

Check\_D\_exist = exist('D\_exist');

if (Check\_D\_exist == 0 | D\_exist == 0)

D\_exist = 0;

end

% ¢d¢d ±N¿é¥X¤J ID\_Data ¥H­Ë©ñ¸mªº¤è¦¡²Õ¦¨ 2.7e ¦¡ ¢d¢d

v\_bar = [ID\_u; ID\_y]; % 2.4 ¦¡

V\_bar = [];

for i = 1:q

V\_bar = [v\_bar(:,i:end-((q+1)-i)); V\_bar]; % 2.7e ¦¡

end

% ¢d¢d D ¶µªº¦³µL¡A¨M©w 2.7e ¦¡ªº u(q),... ¨º¤@¦Cªº¥h¯d ¢d¢d

% ¢d¢d °Ñ¦Ò 2.7e ¦¡»P 2.34b ¦¡¶¡ªº®t²§©Ê ¢d¢d

y\_bar = ID\_y(:,q+1:end); % 2.7b ¦¡

if (D\_exist == 0)

Y\_bar = y\_bar\*pinv(V\_bar);

D = zeros(p,m);

Y\_bar = [D,Y\_bar];

else

V\_bar = [ID\_u(:,q+1:end); V\_bar]; % ¼W¥[ u(q),... ¨º¤@¦C

Y\_bar = y\_bar\*pinv(V\_bar);

D = Y\_bar(:,1:m);

end

Y\_bar(:,1:m) = []; % ¨D¥X D ¶µ«á§R°£¡A¨ä¾l¬° 2.10b

% ¢d¢d ±N Y\_bar ¤À¸Ñ¬° Y\_bar\_1 »P Y\_bar\_2 ¢d¢d

for i = 1:q

Y\_bar\_1(:,:,i) = Y\_bar(:,(i-1)\*(p+m)+1:(i-1)\*(p+m)+m);

Y\_bar\_2(:,:,i) = -Y\_bar(:,(i-1)\*(p+m)+(m+1):(i-1)\*(p+m)+(m+p));

end

% ¢d¢d 2.12 ¦¡ Yk ªº­pºâ ¢d¢d

% ¢d¢d ¬°¤F«K©ó­pºâ¡A±N Yk ªº k ³]¬° k+1¡A¨ä¤¤¥O Y1 ¬° D ¢d¢d

% ¢d¢d «áÄò¦A±N k+1 ³]¦^ k (µ¦¡ Y(:,:,1) = []) ¢d¢d

Y(:,:,1) = D;

for k = 1:2\*(Mult\_Num+1)

Left\_temp = []; Right\_temp = []; Sum\_temp = [];

if (k <= q) % 2.12b ¦¡

for j = 1:k

Left\_temp = [Left\_temp Y\_bar\_2(:,:,j)];

Right\_temp = [Y(:,:,j); Right\_temp];

end

Sum\_temp = Left\_temp\*Right\_temp;

Y(:,:,k+1) = Y\_bar\_1(:,:,k)-Sum\_temp;

else % 2.12c ¦¡

for j = 1:q

Left\_temp = [Left\_temp Y\_bar\_2(:,:,j)];

Right\_temp = [Right\_temp; Y(:,:,k-j+1)];

end

Sum\_temp = Left\_temp\*Right\_temp;

Y(:,:,k+1) = -Sum\_temp;

end

end

Y(:,:,1) = []; % §R°£ D ¸Ó¶µ

% ¢d¢d 2.14 ¦¡ Yok ªº­pºâ ¢d¢d

Y\_o(:,:,1) = Y\_bar\_2(:,:,1);

for k = 2:2\*(Mult\_Num+1)

Left\_temp = []; Right\_temp = []; Sum\_temp = [];

if (k <= q) % 2.14b ¦¡

for j = 1:(k-1)

Left\_temp = [Left\_temp Y\_bar\_2(:,:,j)];

Right\_temp = [Y\_o(:,:,j); Right\_temp];

end

Sum\_temp = Left\_temp\*Right\_temp;

Y\_o(:,:,k) = Y\_bar\_2(:,:,k)-Sum\_temp;

else % 2.14c ¦¡

for j = 1:q

Left\_temp = [Left\_temp Y\_bar\_2(:,:,j)];

Right\_temp = [Right\_temp; Y\_o(:,:,k-j)];

end

Sum\_temp = Left\_temp\*Right\_temp;

Y\_o(:,:,k) = -Sum\_temp;

end

end

% ¢d¢d 2.15 ¦¡ªº­pºâ ¢d¢d

H\_bar = [];

for i = 1:Mult\_Num+1

H\_bar\_temp = [];

for j = 1:Mult\_Num+2

temp = [Y(:,:,(i-1)+j) Y\_o(:,:,(i-1)+j)];

H\_bar\_temp = [H\_bar\_temp temp];

end

H\_bar = [H\_bar; H\_bar\_temp];

end

H\_bar\_0 = H\_bar(:,1:(p+m)\*(Mult\_Num+1)); % ·í k = 1 ®É¡AH\_bar ªº½d³ò

H\_bar\_1 = H\_bar(:,(p+m)+1:end); % ·í k = 2 ®É¡AH\_bar ªº½d³ò

% ¢d¢d H\_bar\_0 ªº©\_²§­È¤À¸Ñ ¢d¢d

[V,Singu,S] = svd(H\_bar\_0);

n\_min = q\*p;

Vn = V(:,1:n\_min);

Sn = S(:,1:n\_min);

Singun = [Singu(1:n\_min,1:n\_min)];

Singu\_Value = diag(Singu);

% ¢d¢d 2.17a ¦¡ ¢d¢d

G = (Singun^-0.5)\*Vn'\*H\_bar\_1\*Sn\*(Singun^-0.5);

% ¢d¢d 2.17b ¦¡ ¢d¢d

InputMatrix\_temp = (Singun^0.5)\*Sn';

H = InputMatrix\_temp(:,1:m);

Lo = InputMatrix\_temp(:,(m+1):(m+p));

% ¢d¢d 2.17c ¦¡ ¢d¢d

OutputMatrix\_temp = Vn\*(Singun^0.5);

C = OutputMatrix\_temp(1:p,:);

function [th,xh,yh,e,Gd,Hd,Lo,x\_in] = Auxi\_OKID\_Process(u,y,G,H,C,D,Lo,Ts,Predict\_OKID,Qo)

% ¢d¢d ¬ÛÃö°Ñ¼Æ¹w³]­È³]©w ¢d¢d

Check\_POKID = exist('Predict\_OKID');

if (Check\_POKID == 0 | Predict\_OKID == 0)

Predict\_OKID = 0;

end

Check\_Qo = exist('Qo');

if (Check\_Qo == 0)

Qo = 1e6;

end

[n,m] = size(H);

[p,n] = size(C);

% ¢d¢d ¹w´ú«¬¨t²Î¡B¿é¤J©MÆ[´ú¾¹°Ñ¼Æ³]©w ¢d¢d

if Predict\_OKID == 1

[A,B] = d2c(G,H,Ts);

Qo = Qo\*eye(n);

Ro = eye(p);

[Lc\_temp,Po] = lqr(A',C',Qo,Ro);

Lc = Lc\_temp';

Lo = (G-eye(n))\*inv(A)\*Lc\*inv(eye(p)+C\*(G-eye(n))\*inv(A)\*Lc);

end

Gd = (G-Lo\*C\*G);

Hd = (H-Lo\*C\*H);

% ¢d¢d ªì©l­È³]©w ¢d¢d

Num\_Sample = length(u);

xh(:,1) = pinv(C)\*y(:,1);

yh(:,1) = C\*xh(:,1)+D\*u(:,1);

e(:,1) = y(:,1)-yh(:,1);

% ¢d¢d ¼ÒÀÀ¹Lµ ¢d¢d

for i = 2:Num\_Sample

if Predict\_OKID == 0 % «D¹w´ú«¬

xh(:,i) = G\*xh(:,i-1)+H\*u(:,i-1)-Lo\*e(:,i-1);

elseif Predict\_OKID == 1 % ¹w´ú«¬

xh(:,i) = Gd\*xh(:,i-1)+Hd\*u(:,i-1)+Lo\*y(:,i);

end

yh(:,i) = C\*xh(:,i)+D\*u(:,i);

e(:,i) = y(:,i)-yh(:,i);

th(:,i) = (i-1)\*Ts;

end

% ¢d¢d ¤ñ¹ïª¬ºAºû«×¤Îµ§¼Æ ¢d¢d

x\_in = xh(:,10);

disp(' Row Col')

disp(['xh',char(9),num2str(size(xh)),char(10),...

'ue',char(9),num2str(size(u)),char(10),...

'yh',char(9),num2str(size(yh)),char(10),...

'th',char(9),num2str(size(th)),char(10)])