On the OKID Method

The objective of this note is to afford the beginner for simulating the observer/Kalman filter identification (OKID) method, where a new approach to improve the transient responses of input time-delay systems is also tested for the input delay-free system.

Notice that equation numbers and figure indices presented in this note are not in order.

* **A new approach to improve the transient response of a time-delay system:**

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 of the time-delay system 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 of the time-delay system during  can be further improved, except for the initial condition .

Example 1: Given the delay-free 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

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



where 

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

Approach 2: The traditional OKID method for the delay-free system with the improved transient response

Let  be created by normal distribution white noise signals mentioned above 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. x-x show the tracking performance of the traditional OKID method with the improved transient response approach.

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

Remark 1: For the delay-free system, the improve approach induces a negative effect.

※ Review of digital redesign approaches for tracker and observer designs

**1. Analog linear quadratic tracker**

Consider a continuous-time system which is controllable and observable described as, (1a)

, (1b)

, (1c)

where  is the state vector,  is the control input,  is the measurable output and ,  and  are system matrices with appropriate dimensions. The optimal state-feedback control law is desired to minimize the following performance index, so we define the quadratic performance index as

, (2)

where ,  and  is the given reference input. This optimal controller [12] is shown by

. (3)

, (4)

, (5)

where the analog state feedback gain , forward gain  and  is the positive-definite and symmetric solution of Ricatti equation as follows

. (6)

The block diagram of the optimally controlled system is shown in Fig. 1.



Fig. 1 Continuous-time control system.

**2. The prediction-based digital redesign method for the tracker design**



Fig. 3 Digitally sampled-data control system.

The derivation of the prediction-based digital controller is described as below. First, we consider a linear controllable continuous-time system, which is described by

, (7a)

, (7b)

where , ,  and  are constant matrices with appropriate dimensions. And the continuous-time state-feedback control law is

, (8)

where  and  have been designed to satisfy some specified performance index discussed in the last section, and  is the reference input. Substituting (8) into (7), we have the closed-loop system as

, (9a)

. (9b)

The sampled-data model in Fig. 3 that corresponds to (7a) and (7b) is shown as

, (10a)

, (10b)

and  is a piecewise-continuous constant input as follows

 for ,

where  is the sampling period. And the sampled-data state-feedback control law is

, (11)

where  and  are digital state-feedback gain and digital forward gain, respectively, corresponding to  and  of continuous-time system. We set , which is a piecewise-constant input, be equal to  in order to track the original reference input . Substituting (11) into (10), we have the closed-loop system as follows

 for , (12a)

. (12b)

Here is a zero-order-hold device used in (11), and the digital controller gains  and  in (11) are obtained from the analog controller gains  and  in (8) for our digital redesign problem. So that, the discrete-time closed-loop state  in (12) can closely match the continuous-time closed-loop state  in (9) at all the sampling instants for a given , that is,  at  for .

From (7), the solution of state  at  is obtained as

. (13)

And if  is a piecewise-continuous constant input, equation (13) can be reduced as



 (14)

where , . Whenever  is a singular matrix, the matrix-valued function  is represented as .

Similarly, the solution of state  at  in (10) is obtained as follows



. (15)

In order to find the predicted state  under the assumption of , it must to make  from (14) and (15). This results the prediction-based digital controller



, (16)

where the state  needs to be predicted based on the available causal signals  and . Substituting (15) into (16), one has

. (17)

Therefore, the desired sampled-data state-feedback control law (11) is obtained from (17) as

 (18)

where , , ,  and .

**3. Digital redesign approach for observer design**

If a linear observable continuous-time system described as

, (26a)

, (26b)

, (26c)

where , , and  are system matrices with appropriate dimensions. One method of estimating the unmeasured state is to construct a full-order model of the plant dynamics

, (27)

where  is the estimate of the actual state . Therefore, if we can obtain the correct initial condition , and set  equal to it, this observer will be satisfactory. However, it is precisely the lack of information about the correct initial condition  that requires the construction of the observer. Otherwise, the estimated state would track the actual state exactly.

To derive the dynamics of the observer, we consider feeding back the difference between the actual output and the estimated output, and correcting the model continuously with this error signal. The equation for this scheme, shown in part of Fig. 7, as follows

, (28)

where  is a proportional gain with appropriate dimension.

We now define the observer error as

. (29)

Differentiating (29) yields

. (30)

Substituting (26) and (28) into (30) yields







. (31)



Fig. 7 Continuous-time system with full-order observer.

Hence, the problem reduces to designing observer such that the error dynamics have their eigenvalues on the far left side of the complex *s*-plane, so that the convergence to the original dynamics can be fast. Since the desirable observer design approach is to use the minimum error energy, an optimal design technique with minimizing performance index is used.

In general, optimal design of controllers for a given controllable and observable linear system is defined as shown in (26), and the optimal state-feedback control law that minimizes the performance index

 (32)

with  and , is obtained as

, (33)

where the optimal feedback gain is , with *P* being the positive definite and symmetric solution of the following Ricatti equation

. (34)

In closed-loop, the optimal control linear system has the form of

, . (35)

Comparing equations (35) and (31), we can see that

,

which has the structure as a state-feedback controller. This is the dual property of linear systems, where the observer gain can be determined as the dual of the feedback controller gain. Thus, the optimal observer gain  can be found by designing the optimal control gain  for the dual system, via  and , so that ; or equivalently, , where *P* is the positive definite and symmetric solution of the Ricatti equation

.

Now since a continuous-time observer for the system as presented in Fig. 7 is defined in (28), next step is to find a digital observer for the system. First, define the discrete-time observer error as

. (36)

such that the discrete-time error dynamics match the continuous-time error dynamics at each sampling instant , or equivalently, assuming that the continuous-time observer is asymptotically stable, the original state and the digital state match .

Using the duality once again, one can find the discrete-time error dynamics of (31) from (15) and (18) as follows

 (37)

where

 (38a)

, (38b)

. (38c)

Further defining , one can write  and with substituting (36) into (37), one becomes

 (39)

By substituting the following identities into (39)

,

,

,

and solving the result for , one obtains the new digitally redesign observer for system (28)

,

or

 (40)

where

, (41a)

, (41b)

, (41c)

with  and  as shown in part of Fig. 8.



Fig. 8 The practically implement full-order observer for the sampled-data linear system.

Approach 3: The digital redesign-based observer and/or Kalman filter for the delay-free system with the improved transient response

Remark 2: 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



Figs. x-x show the tracking performance of the digital redesign-based observer and/or Kalman filter with the improved transient response.

|  |  |
| --- | --- |
|  |  |
|  |  |

Question 1: Why the prediction-based observer and/or Kalman filter has a poor performance at the steady-state than the traditional OKID method? It’s not supposed to be so. Please refer Remark 3 in Example 2 (Page 20).

Possible Reasons:

1. Eigenvalues of  are not small enough.

2.  is non-minimum phase or with poor control zeros than those of .

3. Others. (omitted at here)

Approach 4: The prediction-based observer and/or Kalman filter with the optimal region-pole assignment for the delay-free system

Consider the discrete-time state-space model of a multivariable linear system given by

 (1a)

 (1b)

where , , and  denote the system, input and output matrices, respectively, and , , and  represent the state, input, and output vectors, respectively. To obtain an observer, the term  is added to and subtracted from the right-hand side of state equation (1a), and the system state vector  is considered as an observer state vector , which is provided by a state estimator with an  gain matrix Then, the observer can be constructed as

 (2a)

 (2b)

where  are system parameters obtained by the traditional OKID method [3],  is the observer gain which should be chosen to make the system  as stable as desired,  and  are the estimations of  and , respectively, and the initial state is assigned as  where ‘’ denotes the pseudo-inverse operator.

It is desired to find the best estimate for the state  in the sense that the estimation error

 (7a)

is small as possible. Therefore, from (1a) and (2a), the dynamics of the state estimation error is given by

 (7b)

where the last equality is obtained after inserting  given by (1b). Equation (7) implies that  while , where a reasonable  can be pre-specified by user based on some engineering judgment, and later confirmed from (3b). In general, asymptotical stability of  does not imply that  is also asymptotically stable. An approach similar to the described one in [10] is proposed in the following, to determine the desired observer gain  for the current output-based OKID method in order to have closed-loop observer error poles optimally assigned inside a circle with a pre-specified radius  ().

Determine the transformed matrices , , , to obtain the transformed system

, (8)

and then solve the steady-state Riccati equation

 (9)

, (10)

where  and  are weighting matrices. As a result, eigenvalues of  are guaranteed to be inside the unit circle. The obtained observer gain  is then applied to the original current output-based observer (2a), leading to the closed-loop characteristic equation

. (11)

Therefore, eigenvalues of  are given by eigenvalues of  multiplied by .

Indeed (9) and (10) are dual formulas of the discrete Riccati equation corresponding to a negative feedback-based regulator. Since a positive feedback regulator is used here, (10) has been modified by taking the negative sign of the resultant observer gain. The standard Riccati equation (i.e.,  in (9)) can only yield  (where ) of the  asymptotically stable eigenvalues of  located in the desired region (i.e., inside the  circle, where ) by appropriately tuning the weighting matrices  and  (ex., , , ). On the other hand, the proposed Riccati equation with a general  in (9) and (10) can yield all  asymptotically stable eigenvalues of  and  of the  eigenvalues of  located in the desired region (i.e., inside the  circle, where ) by appropriately tuning , , and . Besides, for a sufficiently small ,  and , the eigenvalues of  approach 0 and 1, respectively.

Simulation results for , , and  are demonstrated in Figs. x-x.

|  |  |  |
| --- | --- | --- |
|  | |  |
|  | |  |
|  |  |

Example 2 Image that there exists a fictitious “Black-Box” system, where the input data  be the high-band signal  (in the frequency domain) of the 123th row of *Lena* ( denoted by  in the spatial domain ) specified in Fig. 3 and the output data be the low-band signal  of the 123th row of *Lena* specified as Fig. 2. How can we apply the OKID method to construct an ARMAX mode-based observer, so that the estimated output  precisely matches the low-band signal  as possible.

*Lena* image has various types of image components, such as horizontal edges, vertical edges, diagonal edges, soft component, and detailed components [34]; therefore, *Lena* is selected as the training image. The Daubechies 9/7 tap biorthogonal, which produce floating point wavelet coefficients, are widely used in image compression techniques to generate a wavelet transform [43]. This transform has proved to provide excellent performance for image compression. Thus, in this study, 9/7 tap biorthogonal filters will provide the transform for our lossy implementation of the new algorithm. The analysis and synthesis filter banks are respectively given as

Analysis low-pass filter:





. (3.)

Analysis high-pass filter:



. (3.)

Synthesis low-pass filter:



. (3.)

Synthesis high-pass filter:





. (3.)

Denote  and  as the low-band image filtered with  in row-wise and the high-band image filtered with  in row-wise, respectively, in frequency domain, where  and  are the position indices of the horizontal row and vertical column of a pixel. The index of pixel position located at the top left-top corner is defined as . Also, denote  and  as the row-wise reconstructed low-band image filtered with  and the row-wise reconstructed high-band image filtered with , respectively, in spatial domain. For a lossless image, the reconstructed image  for  and  is supposed to be identical to the original image  with a near infinity peak signal-to-noise ratio (PSNR), where

 (3.a)

 (3.b)

 and  denote the size of the image,  is the mean-squared error between the source image and reconstructed image. A bigger  value means that the distortion between source image and reconstructed image is smaller.

The design process to construct the universal prediction filter between the low-band signal  and the high-band signal  for the universal images here is a series of steps now presented with some rationalization at each step. To test the characteristics of the proposed approach, for simplicity, take the 123th row of *Lena* (in  pixels), denoted as  for  as a training signal.

Approach 1: The traditional OKID method for the delay-free system

Let the input data  be specified in Fig. 3 and the output data be specified as Fig. 2. Some parameters for OKID method are . The resulting OKID method-based observer is given by

**OKID model ():**

**,** where

,  and ,

,

****

****

****

**Fig. 1**

|  |  |
| --- | --- |
|  |  |
| **Fig. 2** | **Fig. 3** |

|  |  |
| --- | --- |
|  |  |
| **Fig. 5** | **Fig. 6** |

Approach 2: The traditional OKID method for the delay-free system 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

|  |  |
| --- | --- |
|  |  |
| **Fig. 7** | **Fig. 8** |

Approach 3: The digital redesign-based observer and/or Kalman filter for the delay-free system with the improved transient response

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



**Digital-redesign observer**

****

****

****

****

****

****

with ****.

|  |  |
| --- | --- |
|  |  |
| **Fig. 9** | **Fig. 10** |

Remark 3: The prediction-based observer and/or Kalman filter does have a much better performance at the steady-state than the traditional OKID method.

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)])