**HW5**

**1)**

x1\_dot = x2

x2\_dot = x3

x3\_dot = alpha(x1^2\*sinx2 + x2^2\*sinx3)+(x1+x3^2)\*cosx3 + u(t) +d(t)

alpha = 1 ± 0.1

|d(t)| <=0.2

**使用線性回饋的sliding mode control 修正**

假設 s = A\*x3 + B\*x2 + C\*x3 (A,B可以決定狀態彎曲程度)

本處另 B=4、 C=20 ( 固定A = 1 )

可以使系統打在(D^2+4\*D+20)\*x1=0的平面後進行收斂，並且由於此平面的eigen value為 -2±4i ，所以會以旋轉向原點收斂。

當 s\_dot >1

k = K;

當 s\_dot <-1

k = -K;

當 1>=s\_dot>=-1

k = K\*s;

ux = A\*( x3\_dot) - (B\*( x2\_dot) + C\*( x1\_dot) +k);

但是由於系統有震盪誤差，所以當K的值沒有大於可以收斂邊緣很多的話，會不易觀測出系統跑入平面的感覺，這是因為收斂不夠快，無法快速壓制誤差。

經測試: 550<x<1730 or 3985<x:收斂 3985>x>1730 or 550>x:發散

|  |  |  |
| --- | --- | --- |
|  | K = 550 | K = 5000 |
| 狀態空間 |  |  |
| 能量因子隨時間變化 |  |  |
| 狀態收斂 |  |  |

**主程式:**

%% main code

% k:const.

K = 5000;

delt = 0.0001;

totTime = 10;

totalStep = totTime/delt ;

tarr= [0 : 1 : totalStep]\*delt;

x = [0 4 0]'

xarr1 = zeros(totalStep+1,1);xarr2 = xarr1;xarr3 = xarr1;

xarr1(1) = x(1);xarr2(1) = x(2);xarr3(1) = x(3);

x\_1 = xarr1(1);x\_2 = xarr2(1);x\_3 = xarr3(1);

Bx = [0;0;1];

for i=1:totalStep

%get random of alpha, the range is 1~1.1

alpha = 1 + rand\*0.1;

tmp1 = (alpha\*x\_1\*sin(x\_2)+cos(x\_3));

tmp2 = (alpha\*x\_2\*sin(x\_3));

tmp3 = (x\_3^2\*cos(x\_3));

Ax = [ 0 1 0;

0 0 1;

tmp1 tmp2 tmp3

];

A = 1;

B = 4;

C = 20;

s = A\*x\_3 + B\*x\_2 + C\*x\_1;

if s>0

if s>=1

k = K;

else

k = K\*s;

end

else

if s<=-1

k = -1\*K;

else

k = K\*s;

end

end

ss(i+1) = s;

kk(i+1) = k;

ux = A\*(tmp1\*x\_1 + tmp2\*x\_2 + tmp3\*x\_3) - (B\*(x\_3) + C\*(x\_2) +k);

if ux>2200

ux = 2200;

elseif ux<-2200

ux = -2200;

end

uu(i+1) = ux;

xNew = zeros(size(x));

ut = Bx\*ux;

dt = 0.4\*rand()-0.2;

x\_dot = Ax\*x + ut + dt;

xNew = x + x\_dot\*delt;

xN = xNew;

xarr1(i+1) = xN(1);xarr2(i+1) = xN(2);xarr3(i+1) = xN(3);

x = xN;

x\_1 = xN(1);x\_2 = xN(2);x\_3 = xN(3);

end

figure(1);plot(tarr,xarr1,'-r',tarr,xarr2,'-g',tarr,xarr3,'-b');legend('x1','x2','x3');

xlabel('Time');ylabel('x');

figure(2)

plot3(xarr1,xarr2,xarr3,'-b');hold on;

plot3(xarr1(1),xarr2(1),xarr3(1),'o',xarr1(i+1),xarr2(i+1),xarr3(i+1),'x');

xlabel('x1');ylabel('x2');zlabel('x3');

figure(3)

subplot(3,1,1);

plot(tarr,uu,'-b');

xlabel('Time');ylabel('u');

subplot(3,1,2);

plot(tarr,kk,'-b');

xlabel('Time');ylabel('k');

subplot(3,1,3);

one = ones(1,totalStep+1);

minusone = -1\*ones(1,totalStep+1);

plot(tarr,ss,'-b',tarr,one,'g',tarr,minusone,'g');

legend('s','up layer','low layer');

xlabel('Time');ylabel('s');

figure(4)

subplot(3,1,1);

plot(xarr1,xarr2,'-b');hold on;

plot(xarr1(1),xarr2(1),'o',xarr1(i+1),xarr2(i+1),'x');

xlabel('x1');ylabel('x2');

subplot(3,1,2);

plot(xarr2,xarr3,'-b');hold on;

plot(xarr2(1),xarr3(1),'o',xarr2(i+1),xarr3(i+1),'x');

xlabel('x2');ylabel('x3');

subplot(3,1,3);

plot(xarr1,xarr3,'-b');hold on;

plot(xarr1(1),xarr3(1),'o',xarr1(i+1),xarr3(i+1),'x');

xlabel('x1');ylabel('x3');

**2)**

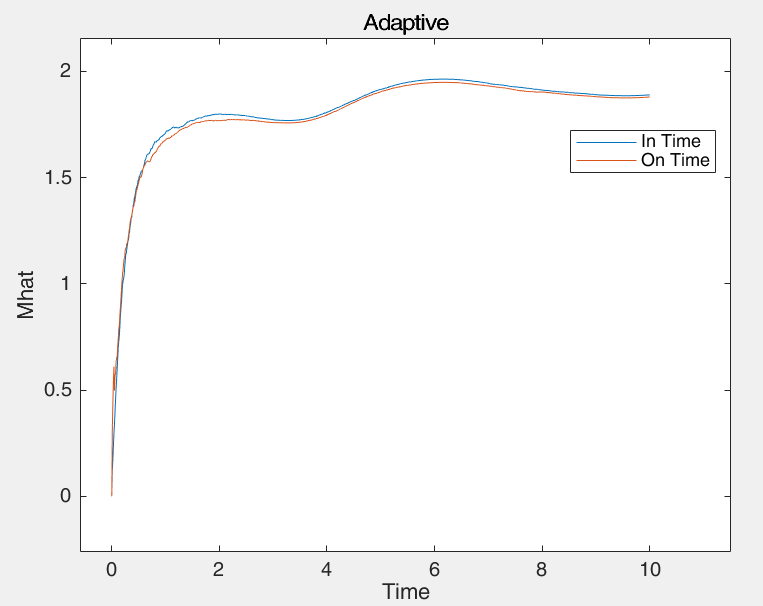
在in time疊代運算，資料會與前面時間資料積分。

在on time的初始M\_hat 設定為 0去計算誤差值，並且疊代運算，資料會與上一筆質量資料有關。

會取最後一筆資料作為估算值:

m1 = 1.8898

m2 = 1.8800



**主程式:**

%% main code

%Set M0 = 2

m0 = 2;

delt = 0.01;

totTime = 10;

totalStep = totTime/delt ;

tarr= [0 : 1 : totalStep]\*delt;

%% off line

uarr = sin(tarr);

nTot = length(uarr);

darr = 0.1\*rand(1,nTot);

yarr = uarr/m0 + darr;

UY\_integral = zeros(1,totalStep+1);

YY\_integral = zeros(1,totalStep+1);

for i = 2:totalStep+1

UY\_integral(i) = UY\_integral(i-1) + uarr(i)\*yarr(i)\*delt;

YY\_integral(i) = YY\_integral(i-1) + yarr(i)\*yarr(i)\*delt;

m1(i) = UY\_integral(i)/YY\_integral(i);

end

%% on line

uarr = sin(tarr);

nTot = length(uarr);

darr = 0.1\*rand(1,nTot);

yarr = uarr/m0 + darr;

YY\_integral = zeros(1,totalStep+1);

m2 = zeros(1,totalStep+1);

m2(1) = 0;

for i = 2:totalStep+1

YY\_integral(i) = YY\_integral(i-1) + yarr(i)\*yarr(i)\*delt;

error = (-uarr(i)+m2(i-1)\*yarr(i));

m\_dot = -error\*yarr(i)/YY\_integral(i);

m2(i) = m2(i-1) + m\_dot\*delt;

end

figure(1);

plot(tarr,m1);

hold on;

plot(tarr,m2);

title("Adaptive");

xlabel('Time');ylabel('Mhat');

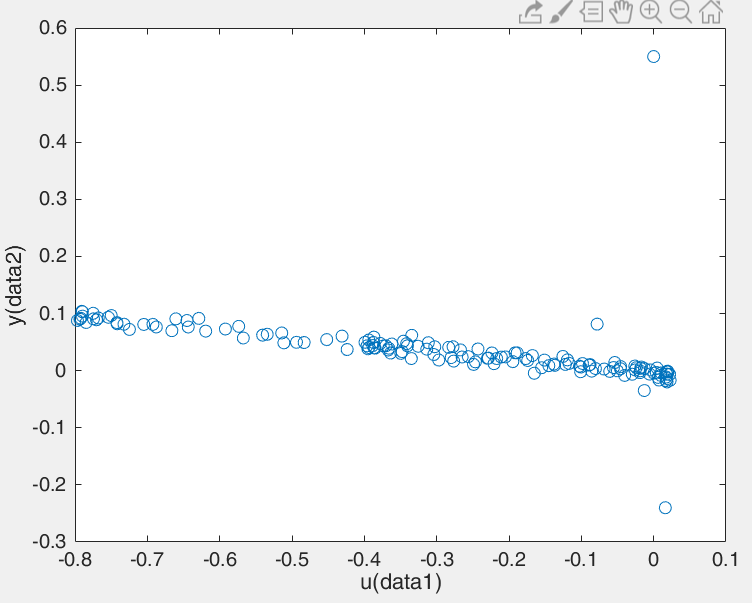
legend("In Time","On Time");

m1(totalStep+1)

m2(totalStep+1)

**3)**

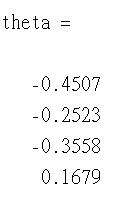
資料作圖顯示:



由於Theta =

運算時使用的是

對時間用k資料進行推移運算，並進行sigma加總起來



**主程式:**

%% main code.

[u, y] = textread('hw6Data.txt', '%f%f');

len = length(u);

plot(u,y,'o');

xlabel("u(data1)")

ylabel("y(data2)")

tmp = 0;

tmp2 = 0;

for i=4:len

phi = zeros(4,1);

phi(1) = y(i-1);

phi(2) = y(i-2);

phi(3) = u(i-1);

phi(4) = u(i-2);

tmp = tmp + phi\*phi';

tmp2 = tmp2 + y(i)\*phi;

end

tmp3 = inv(tmp);

%[-a1 -a2 b1 b2]

theta = tmp3\*tmp2