

《计算机先进控制》实验报告

实验四：广义预测GPC控制设计与仿真

学 院：

专 业：

学生姓名：

学 号：

北京交通大学

实验内容：

设被控对象为如下开环不稳定非最小相位系统：

式中为方差0.01的白噪声。

分控制对象参数已知或未知两种情况进行仿真：

1.控制对象参数已知

取 N1 = d =4、N2 = 8、Nu = 2，控制加权矩阵Γ 为单位阵 I2×2× ，输出柔化系数α = 0.7； 期望输出ω(k) 为幅值为10的方波信号。

算法如下：

a=[1 -2 1.1]; b=[1 2]; c=1; d=4; %对象参数

na=length(a)-1; b=[zeros(1,d-1) b]; nb=length(b)-1; %na、nb 为多项式 A、B 阶次（因 d!=1，对 b 添 0）

aa=conv(a,[1 -1]); naa=na+1; %aa 的阶次

N1=d; N=8; Nu=2; %最小输出长度、预测长度、控制长度

gamma=1\*eye(Nu); alpha=0.7; %控制加权矩阵、输出柔化系数

L=400; %控制步数

uk=zeros(d+nb,1); %输入初值：uk(i)表示 u(k-i)

duk=zeros(d+nb,1); %控制增量初值

yk=zeros(naa,1); %输出初值

w=10\*[ones(L/4,1);-ones(L/4,1);ones(L/4,1);-ones(L/4+d,1)]; %设定值

xi=sqrt(0.01)\*randn(L,1); %白噪声序列

%求解多步 Diophantine 方程并构建 F1、F2、G

[E,F,G]=multidiophantine(aa,b,c,N);

G=G(N1:N,:);

F1=zeros(N-N1+1,Nu); F2=zeros(N-N1+1,nb);

for i=1:N-N1+1

for j=1:min(i,Nu); F1(i,j)=F(i+N1-1,i+N1-1-j+1); end

for j=1:nb; F2(i,j)=F(i+N1-1,i+N1-1+j); end

end

for k=1:L

time(k)=k;

y(k)=-aa(2:naa+1)\*yk+b\*duk(1:nb+1)+xi(k); %采集输出数据,式(7-45)

Yk=[y(k); yk(1:na)]; %构建向量 Y(k)

dUk=duk(1:nb); %构建向量 ΔU(k-j)

%参考轨迹

yr(k)=y(k);%式（7-123）

for i=1:N

yr(k+i)=alpha\*yr(k+i-1)+(1-alpha)\*w(k+d); %式(7-123)

end

Yr=[yr(k+N1:k+N)]'; %构建向量 Yr(k)

%求控制量

dU=inv(F1'\*F1+gamma)\*F1'\*(Yr-F2\*dUk-G\*Yk); %ΔU, 式(7-135)

du(k)=dU(1); u(k)=uk(1)+du(k);

%更新数据

for i=1+nb:-1:2

uk(i)=uk(i-1);

duk(i)=duk(i-1);

end

uk(1)=u(k);

duk(1)=du(k);

for i=naa:-1:2

yk(i)=yk(i-1);

end

yk(1)=y(k);

end

subplot(2,1,1);

plot(time,w(1:L),'r:',time,y);

xlabel('k'); ylabel('w(k)、y(k)');

legend('w(k)','y(k)');

subplot(2,1,2);

plot(time,u);

xlabel('k'); ylabel('u(k)');

子程序如下：

function [E,F,G]=multidiophantine(a,b,c,N)

%功能：多步 Diophanine 方程的求解

%调用格式：[E,F,G]=sindiophantine(a,b,c,N)（注：d=1）

%输入参数：多项式 A、B、C 系数向量及预测步数（共 4 个）

%输出参数：Diophanine 方程的解 E、F、G（共 3 个）

na=length(a)-1; nb=length(b)-1; nc=length(c)-1; %A、B、C 的阶次

%E、F、G 的初值

E=zeros(N); E(1,1)=1; F(1,:)=conv(b,E(1,:));

if na>=nc

G(1,:)=[c(2:nc+1) zeros(1,na-nc)]-a(2:na+1); %令 c(nc+2)=c(nc+3)=...=0

else

G(1,:)=c(2:nc+1)-[a(2:na+1) zeros(1,nc-na)]; %令 a(na+2)=a(na+3)=...=0

end

%求 E、G、F

for j=2:N

for i=1:j-1

E(j,i)=E(j-1,i);

end

E(j,j)=G(j-1,1);%式（7-91）

for i=2:na

G(j,i-1)=G(j-1,i)-G(j-1,1)\*a(i);

end

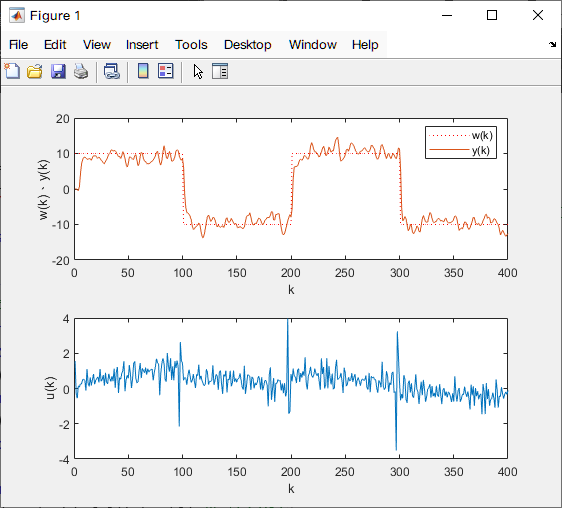
G(j,na)=-G(j-1,1)\*a(na+1); %式(7-91)

F(j,:)=conv(b,E(j,:)); %式(7-91)

end

end

运行结果如下：



2.控制对象参数未知

取初值、，遗忘因子λ=1；控制参数、、，控制加权矩阵Γ为单位阵，输出柔化系数α=0.7；期望输出ω(k)为幅值为10的方波信号。

算法如下：

a=[1 -2 1.1]; b=[1 2]; c=1; d=4; %对象参数

na=length(a)-1; b=[zeros(1,d-1) b]; nb=length(b)-1; %na、nb 为多项式 A、B 阶次（因 d!=1，对 b 添 0）

naa=na+1; %aa 的阶次

N1=d; N=8; Nu=2; %最小输出长度、预测长度、控制长度

gamma=1\*eye(Nu); alpha=0.7; %控制加权矩阵、输出柔化系数

L=400; %控制步数

uk=zeros(d+nb,1); %输入初值：uk(i)表示 u(k-i)

duk=zeros(d+nb,1); %控制增量初值

yk=zeros(na,1); %输出初值

dyk=zeros(na,1); %输出增量初值

w=10\*[ones(L/4,1);-ones(L/4,1);ones(L/4,1);-ones(L/4+d,1)]; %设定值

xi=sqrt(0.01)\*randn(L,1); %白噪声序列

%RLS 初值

thetae\_1=0.001\*ones(na+nb-d+2,1); %不辨识 b 中添加的 0

P=10^6\*eye(na+nb-d+2);

lambda=1; %遗忘因子[0.9 1]

for k=1:L

time(k)=k;

dy(k)=-a(2:na+1)\*dyk(1:na)+b\*duk(1:nb+1)+xi(k); %式(7-45)

y(k)=yk(1)+dy(k); %采集输出数据

Yk=[y(k); yk(1:na)]; %构建向量 Y(k)

dUk=duk(1:nb); %构建向量 ΔU(k-j)

%参考轨迹

yr(k)=y(k); %式(7-123)

for i=1:N

yr(k+i)=alpha\*yr(k+i-1)+(1-alpha)\*w(k+d); %式(7-123)

end

Yr=[yr(k+N1:k+N)]'; %构建向量 Yr(k)

%递推最小二乘法

phi=[-dyk(1:na);duk(d:nb+1)];

K=P\*phi/(lambda+phi'\*P\*phi);

thetae(:,k)=thetae\_1+K\*(dy(k)-phi'\*thetae\_1);

P=(eye(na+nb-d+2)-K\*phi')\*P/lambda;%式(7-138)

%提取辨识参数

ae=[1 thetae(1:na,k)']; be=[zeros(1,d-1) thetae(na+1:na+nb-d+2,k)'];

aae=conv(ae,[1 -1]);

%求解多步 Diophantine 方程并构建 F1、F2、G

[E,F,G]=multidiophantine(aae,be,c,N);

G=G(N1:N,:);

F1=zeros(N-N1+1,Nu); F2=zeros(N-N1+1,nb);

for i=1:N-N1+1

for j=1:min(i,Nu); F1(i,j)=F(i+N1-1,i+N1-1-j+1); end

for j=1:nb; F2(i,j)=F(i+N1-1,i+N1-1+j); end

end

%求控制量

dU=inv(F1'\*F1+gamma)\*F1'\*(Yr-F2\*dUk-G\*Yk); %ΔU 式(7-135)

du(k)=dU(1); u(k)=uk(1)+du(k);

%更新数据

thetae\_1=thetae(:,k);

for i=1+nb:-1:2

uk(i)=uk(i-1);

duk(i)=duk(i-1);

end

uk(1)=u(k);

duk(1)=du(k);

for i=na:-1:2

yk(i)=yk(i-1);

dyk(i)=dyk(i-1);

end

yk(1)=y(k);

dyk(1)=dy(k);

end

figure(1);

subplot(2,1,1);

plot(time,w(1:L),'r:',time,y);

xlabel('k'); ylabel('w(k)、y(k)');

legend('w(k)','y(k)'); axis([0 L -20 20]);

subplot(2,1,2);

plot(time,u);

xlabel('k'); ylabel('u(k)'); axis([0 L -4 4]);

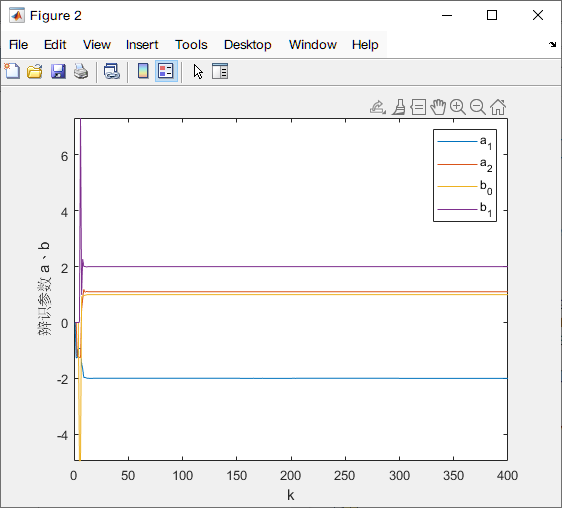
figure(2);

plot(1:L,thetae);

xlabel('k'); ylabel('辨识参数 a、b');

legend('a\_1','a\_2','b\_0','b\_1'); axis([0 L -3 3]);

结果如下：



通过实验可以看到在对象参数未知时，由于使用了递推最小二乘法(RLS)来实时辨识系统参数，使得GPC能够在系统的动态特性不确定或者变化时自动调整控制器参数，使得控制性能保持较好。可以从图中看到随着时间变化，辨识参数逐渐收敛。

但是在实际系统中，RLS算法容易受到噪声影响。如果辨识结果不准确，控制器的性能将会受到很大影响。如果噪声过大，辨识精度下降，控制效果可能变差。而且GPC的性能高度依赖于控制加权矩阵γ和输出柔化系数α的设置。如果这些参数设置不当，可能会导致控制器表现不佳。比如，过大的γ会抑制控制器的动作，导致系统反应迟缓，而过小的γ则可能导致控制器动作过度，造成震荡。