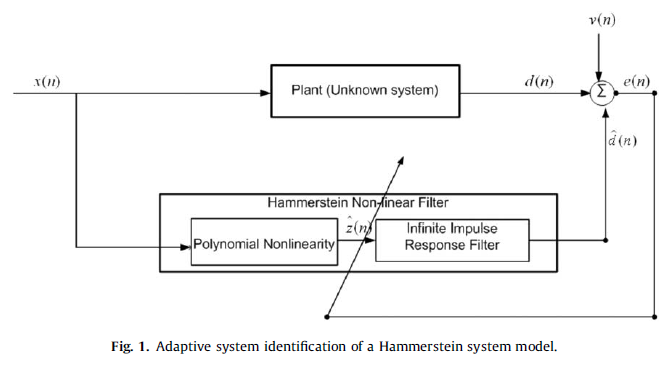
به نام خدا

عنوان: یک الگوریتم شناسایی بر اساس تصویر سازی برای سیستم­های همرشتاین غیر خطی

مقدمه:

سیستم­های همرشتاین غیر خطی نوعی از فیلترهای غیر خطی هستند که در آنها یک سیستم غیرخطی بدون حافظه قبل از یک سیستم دینامیکی خطی قرار می­گیرد (شکل زیر). این سیستم­ها در مدلسازی مسایل پردازش سیگنال کاربرد فراوانی دارد.



شناسایی سیستم­های همرشتاین شامل روش­های غیر پارامتری و پارامتری می­باشد. بیشتر روش­های ارائه شده برای شناسایی غیر تطبیقی هستند و اکثرا نتیجه خود را به صورت کمترین مربعات خطا بیان نمی­کنند.

در این مقاله هدف ما عبارت است از:

1. شناسایی سیستم­های همرشتاین به صورت تطبیقی بر اساس روش تصویر سازی، بدون نیاز به خطی سازی سیستم و یا داشتن دانشی در مورد ورودی سیستم.
2. یافتن مرزی برای پارامتر گام در الگوریتم تطبیقی برای پایداری BIBO.
3. بدست آوردن یک رابطه برای عملکرد متوسط مربعات خطا در الگوریتم ارائه شده بر اساس اصل پایداری انرژی.

بیان مساله

در شکل 1 x، v و d ورودی، نویز و خروجی سیستم هستند.  سیگنال غیر قابل دسترسی داخلی سیستم است. برای قسمت غیر خطی یک مدل چند جمله­ای درجه L به صورت زیر در نظر می­گیریم:



برای قسمت خطی یک مدل IIR با معادله تفاضلی زیر در نظر گرفته می­شود:



که در آن  و  و  ضرایب زیر سیستم­ها را در هر لحظه n نشان می­دهد. همانطور که دیده می­شودM درجه صورت تابع تبدیل، N درجه مخرج تا تبدیل و L درجه چند جمله­ای قسمت غیر خطی است. با فرض  برای نرمالیزاسیون، از (2) داریم:



اگر معادله بالا را به صورت ماتریسی بازنویسی کنیم:





در این پروژه هدف بروزرسانی بردار  با استفاده از سیگنال­های ورودی و خروجی  می­باشد، به صورتی که  تا حد ممکن به  نزدیک باشد.

الگوریتم تطبیقی همرشتاین

هدف مینیمم سازی مربع نرم اقلیدسی بردار  نسبت به Q قید زیر می­باشد:



برای حل مساله بهینه سازی بالا، قدم­های زیر را طی می­کنیم:

1. متغیرهای زیر را مقدار دهی اولیه می­کنیم:





که در آن  ماتریس همانی با ابعاد می­باشد و  مقادیر عددی مثبت هستند و  یک ماتریس صفر می­باشد.

1. برای تمام n نمونه از سیگنال عملیات زیر را انجام می­دهیم:

* 
* 

که در آن :









* 
* 

که در آن I ماتریس همانی است.

* *با استفاده از  بدست آمده در این تکرار، مقدار z(n) را با استفاده رابطه زیر تخمین می­ زنیم:*

**

* *با استفاده از z(n) تخمین زده شده و پارامترهای این تکرار، مقدار خروجی را تعیین می­کنیم:*

**

*یافتن کرانی برای اندازه گام*

*با تعریف سیستم همرشتاین به صورت فضای حالت زیر:*

**

**

*اثبات می­شود که برای پایداری الگوریتم تطبیقی همرشتاین باید:*

**

*تعیین مقدار MSE به صورت تئوری*

*اثبات می­شود که مقدار MSE و EMSE در الگوریتم همرشتاین تطبیقی از روابط زیر بدست می­آیند:*

**

**

*که در آن:*

**

*مثال شبیه سازی*

*در این پروژه یک سیستم همرشتاین را برای بررسی عملکرد الگوریتم در نظر می­گیریم. سیستم مذکور دارای قسمت خطی به صورت زیر است:*

**

*همچنین برای قسمت غیر خطی چند جمله­ای زیر را در نظر می­گیریم:*

**

*خروجی این سیستم با نویز سفید به گونه­ای آغشته می­شود که دارای SNR برابر 30dB باشد. ورودی سیستم را یک بار نویز سفید با میانگین 0 و واریانس 1 در نظر می­گیریم ویک بار نویز رنگی که در آن از فیلتر زیر برای نویز سفید استفاده شده است:*

**

*مقدار گام را با توجه به رابطه (53) برابر مقدار ثابت 1.674e-7 در نظر می­گیریم. همچنین مقدار*  *که فاکتور فراموشی نام دارد را برابر 0.995 در نظر می­گیریم.*

*پیاده سازی الگوریتم در Matlab*

*در ابتدا باید داده­های ورودی-خروجی سیستم همرشتاین مورد نظر را ساخت. برای این منظور برنامه data.m نوشته شده است که توضیحات آن به صورت زیر است:*

flag=2;% 1=white noise, 0=colored noise

*ابتدا از کاربر درخواست می­شود که نوع ورودی (سفید=1، رنگی=0) را وارد کند.*

N=4000;

*تعداد کل نمونه­های مورد نیاز تعیین می­شود*

x=randn(N,1);

*به تعداد نمونه­های مورد نیاز عدد تصادفی با توزیع نرمال استاندارد در بردار x تولید می­شود*

if flag==0

x=filter([1 .5],1,x);

end

*اگر نویز رنگی برای ورودی انتخاب شده بود، نویز سفید با فیلتر رابطه (63) فیلتر می­گردد.*

z=x-.3\*x.^2+.2\*x.^3;

*خروجی قسمت غیر خطی از رابطه (62) محاسبه می­شود.*

a=[1 -.2314 .4318 -.3404 .5184];

b=[1 -1.8 1.62 -1.458 .6561];

dn=filter(b,a,z);

*خروجی قسمت خطی با استفاده از تابع تبدیل (61) محاسبه می­شود.*

d=awgn(dn,30);

*خروجی با نویز مخلوط می­گردد تا SNR آن حدودا برابر 30dB گردد.*

if flag==0

save('colored\_data.mat','x','d')

else

save('white\_data.mat','x','d')

end

*ورودی و خروجی ساخته شده بسته به نوع نویز ورودی در فایلی با نام مناسب ذخیره می­شوند.*

*برنامه اصلی*

*برنامه­های white\_inp.m و colored\_inp.m هردو مشابه هم هستند، فقط در اولی داده­های مربوط به نویز سفید لود می­شود و در دومی داده­های مربوط به نویز رنگی. توضیح خطوط مربوط به این برنامه­ها به صورت زیر است:*

load white\_data

*داده­های مربوطه لود می­شوند.*

M=4;

N=5;

L=3;

*مقادیر درجه صورت و مخرج تابع تبدیل و درجه چند جمله­ای غیر خطی تعیین می­شوند. با توجه به رابطه (3) درجه صورت را باید یکی کمتر از آنچه واقعا هست وارد کرد که ما به جای 5، 4 وارد کرده­ایم.*

lam=.995;

del=.05;

u=1.64e-7;

*مقدار پارامترهای  و  و  وارد می­شوند.*

th=[1 -.2314 .4318 -.3404 .5184 -1.8 1.62 -1.458 .6561 1 -.3 .2]';

*مقدار اصلی پارامترها برای مقایسه در هر مرحله، در متغیر th تعریف می­شود.*

p\_color=[0 0 1;0 0 0;1 0 0;0 1 0];

*رنگ نمودار هر کدام از چهار حالت مورد نظر (Q=1,…, 4) تعیین می­شود.*

for Q=1:4

*تکرار برای Q=1,…,4.*

theta=[zeros(N,1);zeros(M,1);ones(L,1)];

*مقدار دهی اولیه برای پارامترها .*

ak=theta(1:N);bk=theta(N+1:N+M);pk=theta(N+M+1:end);

*مقدار اولیه ضرایب صورت و مخرج قسمت خطی و ضرایب قسمت غیر خطی.*

R=eye(M+N+L);

*مقدار اولیه ماتریس  تعیین می­شود.*

S=zeros(M+N+L,Q);

*مقدار اولیه ماتریس  تعیین می­شود.*

sp=200;

*یک نقطه برای شروع الگوریتم از آن نمونه به خصوص. این مقدار باید از M+N+L بزرگتر باشد که ما آنرا 200 انتخاب کردیم.*

dh=zeros(size(d));

zh=zeros(size(d));

phi=cell(sp,1);

for i=1:length(d)

phi{i}=zeros(N+M+L,Q);

end

ps=cell(sp,1);

for i=1:length(d)

ps{i}=zeros(N+M+L,1);

end

*متغیرهای مورد استفاده در الگوریتم مقدار دهی اولیه صفر می­شوند تا بعداً در الگوریتم مقادیر آنها تعیین گردد.*

for n=sp:length(d)

*شروع الگوریتم برای تمام نمونه­های سیگنال­های ورودی-خروجی*

e=(d(n-1:-1:n-Q)-S'\*theta);

*محاسبه *

temp=[-dh(n-1-1:-1:n-1-N);zh(n-1-1:-1:n-1-M)];

for l=1:L

temp=[temp;sum(x(n-1:-1:n-1-M).^l)];

end

ps{n-1}=temp;

*محاسبه .*

for q=1:Q

phi{n-1}(:,q)=ps{n-q};

end

for k=1:N

phi{n-1}=phi{n-1}-ak(k)\*phi{n-1-k};

end

*محاسبه .*

R1=pinv(R);

R1=lam\*R1+(1-lam)\*phi{n-1}\*phi{n-1}';

R=pinv(R1);

*محاسبه  از رابطه (27).*

theta=theta+u\*R\*phi{n-1}\*pinv(del\*eye(Q)+u\*S'\*phi{n-1})\*e;

*محاسبه  از رابطه (29).*

ak=theta(1:N);bk=theta(N+1:N+M);pk=theta(N+M+1:end);

*جداسازی  و  و  از روی .*

zh(n)=sum(x(n).^(1:L).\*pk');

*محاسبه  با رابطه .*

s=[-dh(n-1:-1:n-N);zh(n-1:-1:n-M);x(n).^(1:L)'];

*تشکیل .*

S=[s S(:,1:end-1)];

*به روز رسانی .*

dh(n)=s'\*theta;

*محاسبه  با رابطه .*

MSD(n)=mse(th-theta);

end

*محاسبه MSD خطای بین مقدار اصلی پارامترها و مقدار تخمین زده شده توسط الگوریتم و پایان حلقه تکرار برای شمارش نمونه­ها.*

MSD=MSD(sp+N+M+L+1:end);

plot(MSD),'Color',p\_color(Q,:));hold on

end

*جداسازی قسمت معنی دار نمودار MSD و رسم آن با استفاده از رنگ مربوطه و پایان حلقه مربوط به شمارش Q.*

legend('Q=1','Q=2','Q=3','Q=4')

axis tight

xlabel('Number of iterations(n)')

ylabel('Mean Square Deviation')

*برچسب گذاری و تنظیمات نمودار رسم شده.*

*در برنامه ErrorMse.m مقدار خطا با استفاده از رابطه تئوری محاسبه می­شود. این برنامه مشابه برنامه­ای است که در بالا توضیح داده شد، فقط خطوط زیر در آن اضافه شده است.*

akk(n,:)=ak';

*ذخیره سازی ضرایب مخرج تابع تبدیل در هر تکرار*

MSE(n)=mse(dh(sp:n)-d(sp:n));

*محاسبه MSE بین داده­های واقعی و تخمین زده شده توسط الگوریتم.*

C=C+1/(length(d)-sp)\*pinv(del\*eye(Q)+u\*phi{n-1}'\*S)...

\*phi{n-1}'\*R\*phi{n-1}\*pinv(del\*eye(Q)+u\*phi{n-1}'\*S);

*محاسبه ماتریس .*

G=G+1/(length(d)-sp)\*pinv(del\*eye(Q)+u\*phi{n-1}'\*S);

*محاسبه ماتریس .*

zet(n)=u;

for f=1:10:n-2

zet(n)=zet(n)-(-1)^f\*prod(sum(akk(n-10:n,:)));

end

*محاسبه مقدار .*

TMSE=-u\*.2^2\*trace(C)/(trace(u\*C)-2\*zet(n)\*trace(G))+.2^2;

*محاسبه مقدار  از رابطه (60) .*

plot(MSE,'k');hold on

plot(TMSE(1,length(MSE)),'r')

legend('MSE for Q=3','Theoretical MSE')

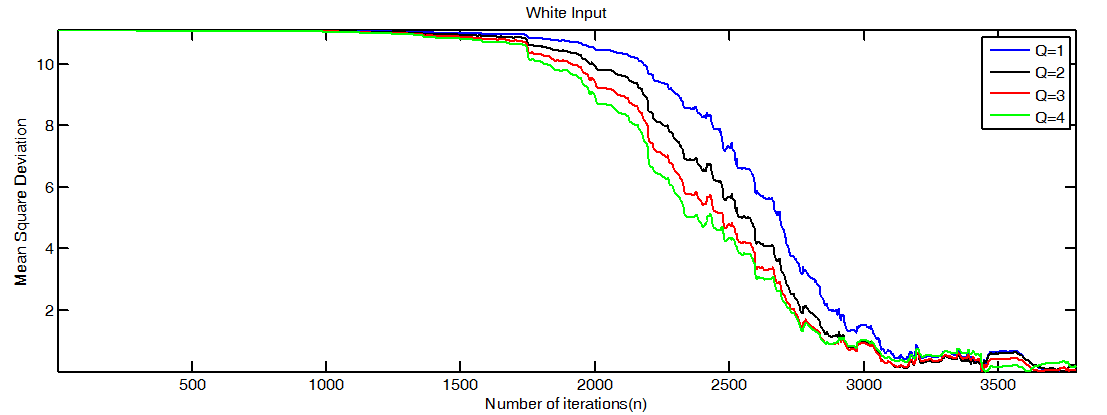
xlabel('Number of iterations(n)')

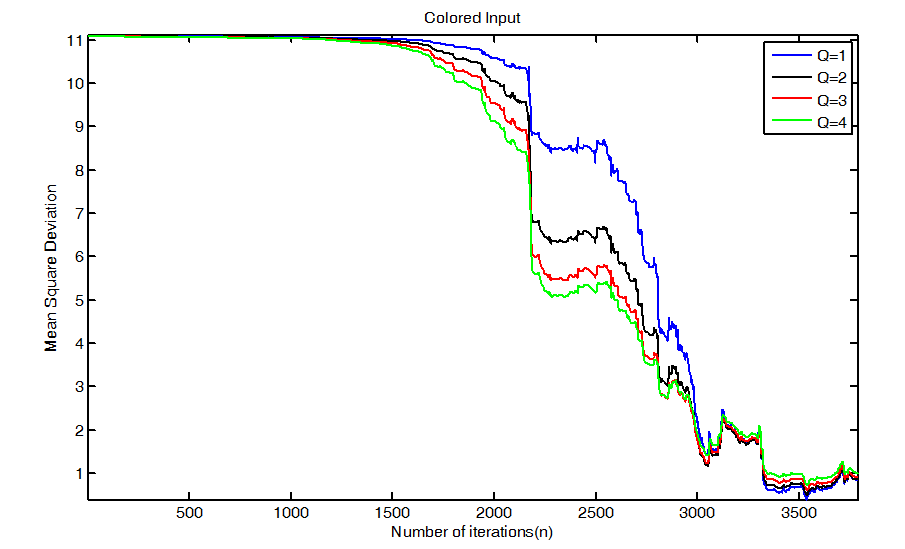
ylabel('Mean Square Error')

رسم MSE و MSE تئوری بر روی یک نمودار.

*نتایج*

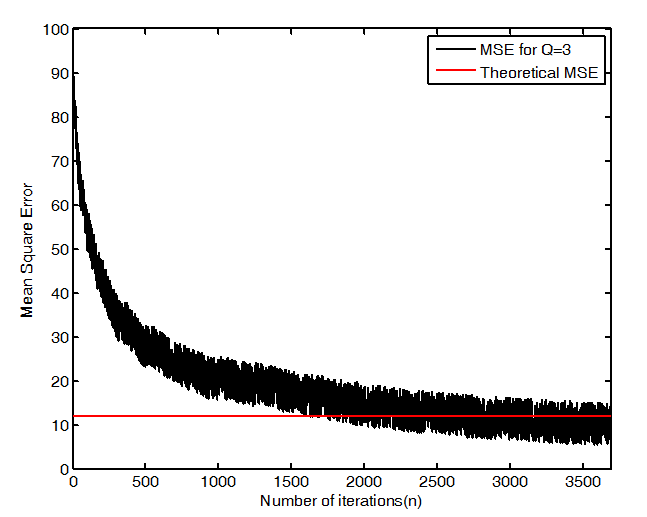
*با اجرای برنامه­ها برای ورودی نویز سفید و رنگی، دو نمودار زیر رسم می­گردد:*

**

**

*همانطور که دیده می­شود خطای پارامترها با پیشرفت تکرارها به سمت 0 می­رود که نشان از موفقیت الگوریتم در شناسایی پارامترها دارد. همچنین برای هردو ورودی می­بینیم که همگرایی الگوریتم با افزایش Q سریعتر می­گردد.*

*همچنین با اجرای برنامه برای MSE تئوری نمودار زیر رسم می­شود:*

**

*همانطور که دیده می­شود، MSEهای واقعی و تئوری با پیشرفت تکرارها به هم نزدیک می­شوند.*

*یک مثال عملی*

*الگوریتم ارائه شده بر روی یک مثال عملی نیز پیاده سازی شده. این الگوریتم برای شناسایی یک مدل برای دینامیک رفلکس کشش در ماهیچه انسان استفاده شده است. با انتخاب مرتبه 8 برای قسمت خطی و مرتبه 5 برای قسمت غیر خطی، خطای آموزش الگوریتم در شکل زیر قابل مشاهده است (شکل از اصل مقاله است و به علت نبود داده­ها شبیه سازی نشده­اند).*

**