Amirkabir University of Technology
(Tehran Polytechnic)

Electrical Engineering Department

گزارشکار آزمایشگاه مقدمه‌ای بر هوش محاسباتی

آزمایش شماره‌های 6

شناسایی سیستم با کمک شبکه عصبی - Identification

نام استاد: محمدحسین امینی

نام دانشجو: محمدعرشیا ثمودی - 9723021

آزمایش ششم

هدف آزمایش: شناسایی سیستم با کمک شبکه عصبی

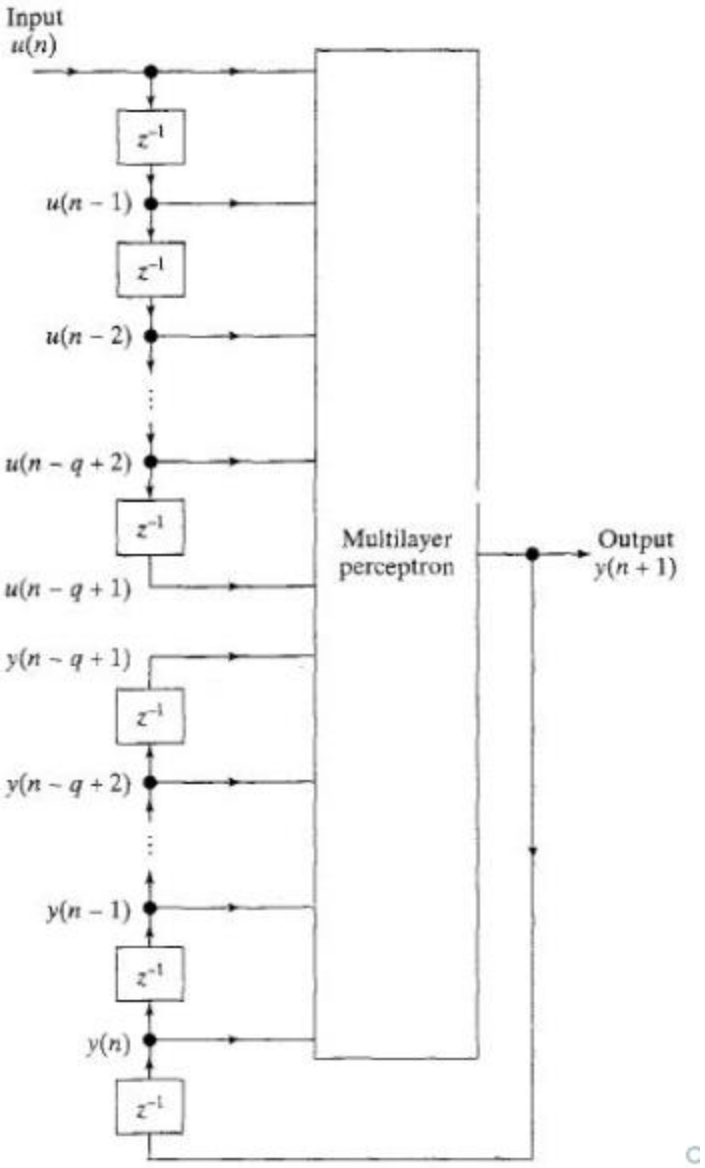
شرح آزمایش:

1. مقدمه

یکی از قابلیت شبکه های عصبی چند لایه، شناسایی سیستم‌های ناشناخته می‌باشد. شناسایی سیستم ها یکی از پرکاربرد ترین مسایل در علم کنترل است.

با استفاده از شبکه های عصبی چندلایه پیاده سازی شده در جلسات قبل، می توان سیستم های ناشناخته را به صورت زمان گسسته یا زمان پیوسته شناسایی کرد. در این آزمایش به شناسایی سیستم های گسسته و پیوسته به وسیله شناساگر استاتیکی پرداخته می‌شود.

شناساگر استاتیکی؛ در این نوع مدل شناساگر ، ورودی‌های شبکه عصبی ورودی سیستم و ورودی‌های تاخیر یافته و خروجی و تاخیر یافته های آن می‌باشد. پس از آموزش شبکه با این ورودی ها، خروجی شبکه تخمین زده می‌شود.

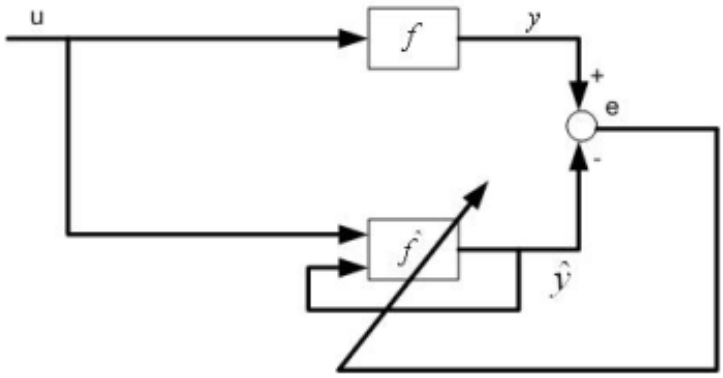


شکل 1. مدل استاتیکی تشخیص سیستم

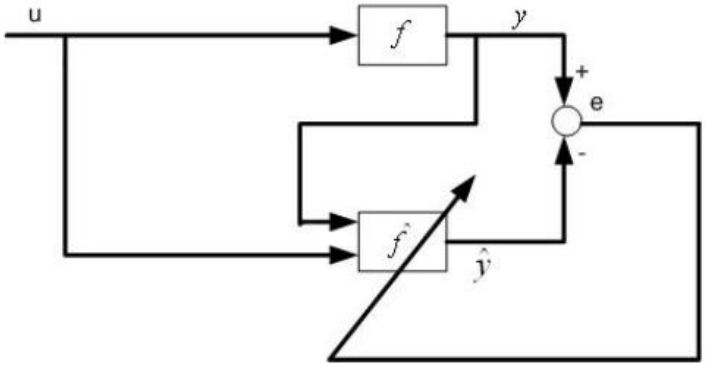
مدل های شناسایی سیستم پیوسته:

1.مدل موازی
2.مدل سری موازی

تفاوت این دو مدل در فیدبک داده شده به تخمینگر است که میتوانید در اشکال زیر مشاهده کنید:



شکل 2. مدل استاتیکی موازی



شکل 3. مدل استاتیکی سری-موازی

تخمین مدل استاتیکی توسط واحد گرافیکی NNTOOL در متلب

هدف آزمایش تخمین سیستم است و برای تخمین درست، ورودی باید غنی باشد چراکه وجود هارمونیک‌های متفاوت در خروجی باعث یادگیری بهتر سیستم می‌شود. انواع ورودی‌های ما نظیر ضربه، پله، سینوسی و نویز سفید هستند.

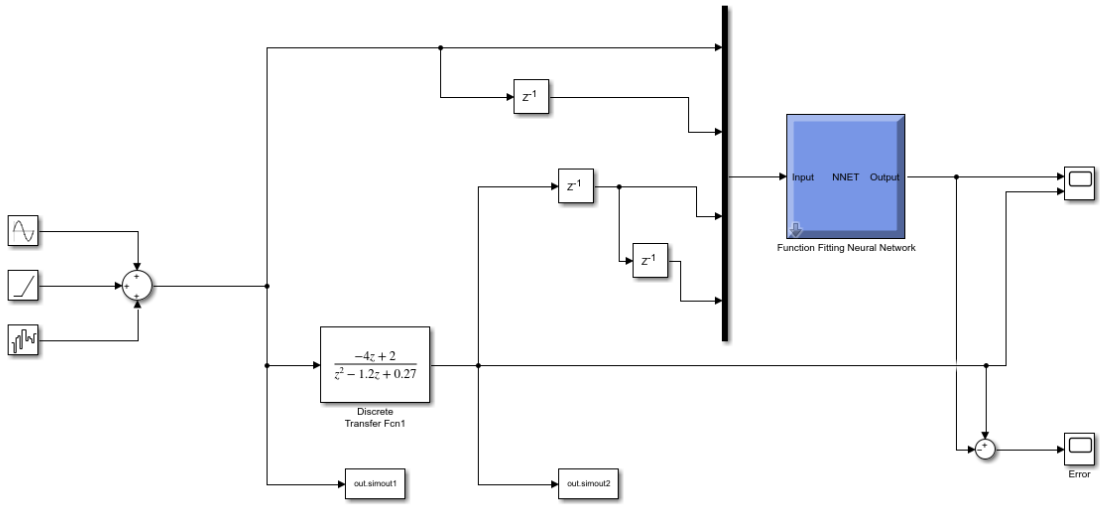
نویز سفید ورودی مناسبی نیست چرا که سرشار از هارمونیک است و به سیستم آسیب می‌زند. بنابراین برای یادگیری سیستم، ورودی را چندین هارمونیک و نویز سفید محدود قرار میدهیم تا سیستم با ورودی‌های غنی، آزموده شود.

ابتدا باید تابع تبدیل خود را بع فرم معادله دیفرنس دربیاوریم تا فرم تاخیر در ورودی و خروجی را بدست آوریم.

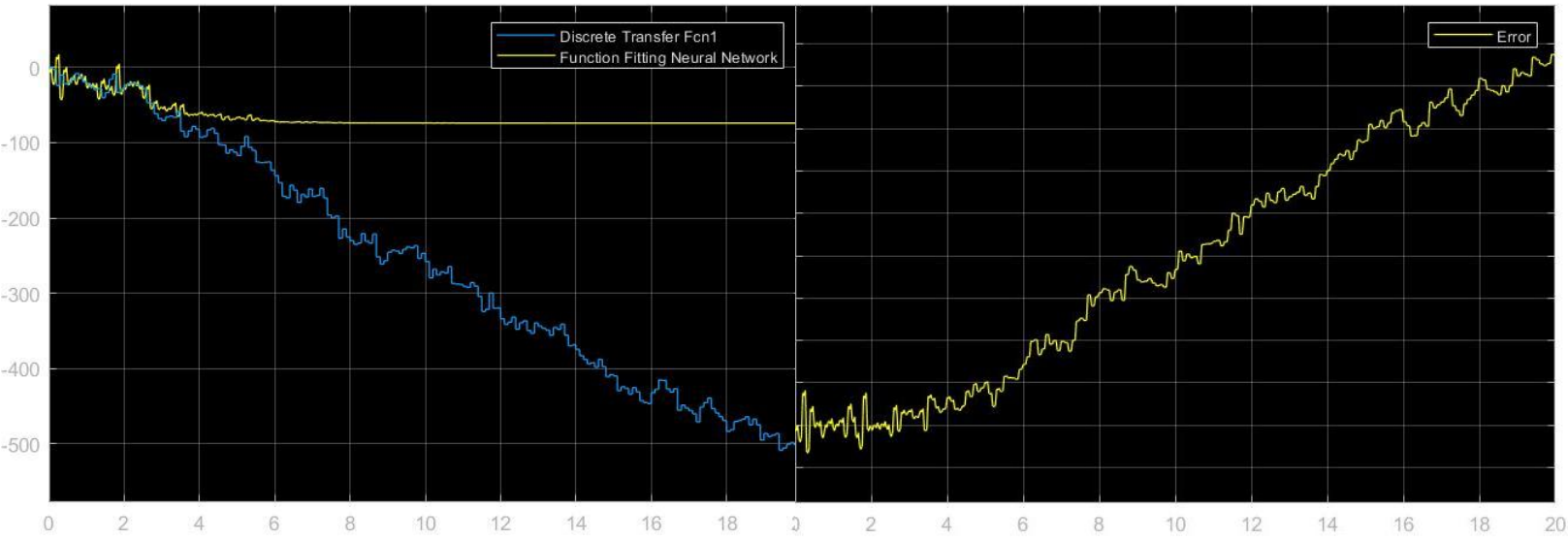
$$H(z) = \frac{-4 + 2z^{-1}}{1 - 1.2z^{-1} + 0.27z^{-2}} \Rightarrow y[n] = -4x[n] + 2x[n-1] + 1.2y[n-1] + 0.27y[n-2]$$

در سیمولینک متلب مطابق با معادله بالا خروجی و ورودی و تاخیریافته‌های آن‌ها را به واحد Mux می‌دهیم و خروجی Mux را به یک Fitting Neural Network میدهیم تا خروجی را به ما بازگرداند و آنگاه می‌توانیم با خروجی تابع تبدیل قیاس کنیم تا ببینم آیا سیستم توانایی پیروی از ورودی را دارد یا خیر.

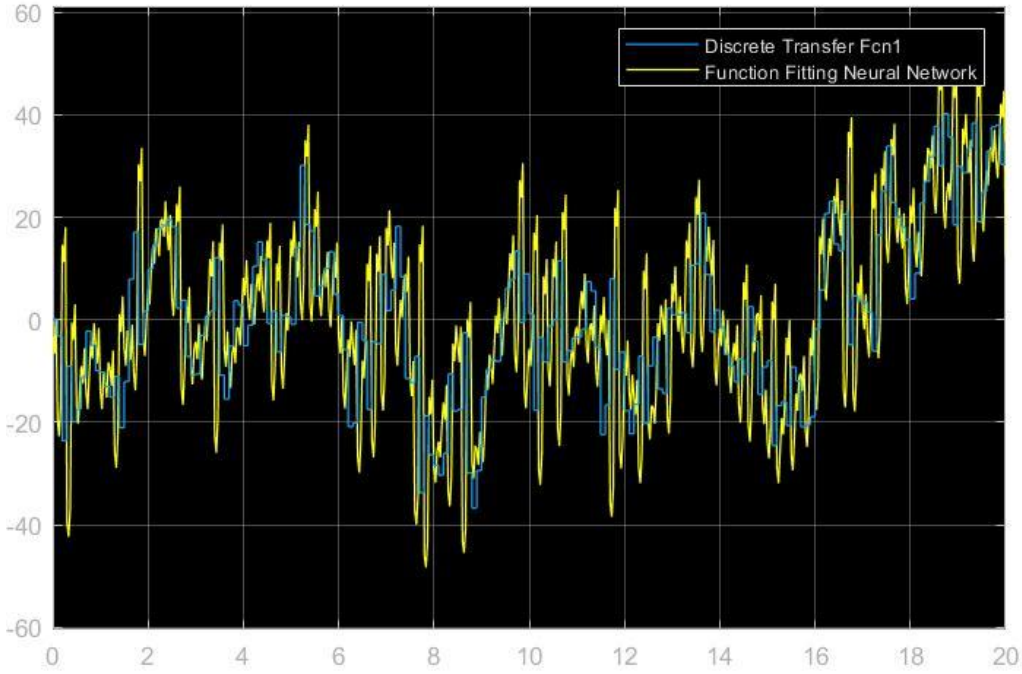
برای ایجاد یک بلوک Fitting Neural Network کافی است تا دستور nntool را در متلب اجرا کرده و پس از تعیین نوع شبکه عصبی، آن را به صورت یک بلوک سیمولینک استخراج کنید.



به منظور ورودی یک تابع نویز سفید، یک هارمونیک و تابع رمپ را به سیستم می‌دهیم. می‌دانیم سیستم تابع رمپ را به خوبی دنبال نمی‌کند چراکه تابع رمپ ذاتا ماهیت ناپایدار دارد.



بار دیگر اینبار بدون رمپ تکرار میکنیم:



این بار می‌بینیم که خروجی شبکه عصبی تخمین بهتری دارد و به خروجی تابع تبدیل نزدیک‌تر است.

تخمین مدل استاتیکی توسط کدزنی NNTOOL در متلب

```
fs = 100;
Ts = inv(fs);
t = 0:inv(fs):20-Ts;
f1 = 10;

% inputs

inp1 = t;                                                    % Ramp
inp2 = sin(2*pi*f1*t);                                       % sineWave with f1 frequency
Noise_Power = 0.5; Noise = wgn(1,length(t),Noise_Power);     % White Guassian Noise With power 0.2
input = inp1 + inp2 + Noise;

% H(z) Numerator and Denuminator
TFN = [-4 2];
TFD = [1 -1.2 0.27];

% Filtered Output
output = filter(TFN,TFD,input);

% Delayed input and output as NN inputs
```

```
a1 = [0 ,input(1:end-1)];
a3 = [0 ,output(1:end-1)];
a4 = [0 ,0 ,output(1:end-2)];

x = [input; a1; a3; a4];

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.

% Create a Fitting Network
% The Single Layer Network with 10 Neurons
hiddenLayerSize = [10];

net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideParam.trainRatio = 80/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 10/100;
%net.divideFcn = 'divideind';

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean Squared Error

% Viewing Network
view(net)

% Train Network
[net,tr] = train(net,x,output);

% Testing Network
network_outputs = net(x);
network_error = gsubtract(output,network_outputs);
network_performance = perform(net,output,network_outputs);

% Plotting
figure(1)
subplot(3,1,1)
plot(t,output);
xlabel('Time(secons)'); ylabel('Amplitude'); title('Filtered Output');

subplot(3,1,2)
plot(t,network_outputs,'r-');
xlabel('Time(secons)'); ylabel('Amplitude'); title('Neural Network Output. Noise Power
'+string(Noise_Power));

subplot(3,1,3)
```

```
plot(t,network_error,'k-');
xlabel('Time(secons)'); ylabel('Amplitude'); title('Error');
```