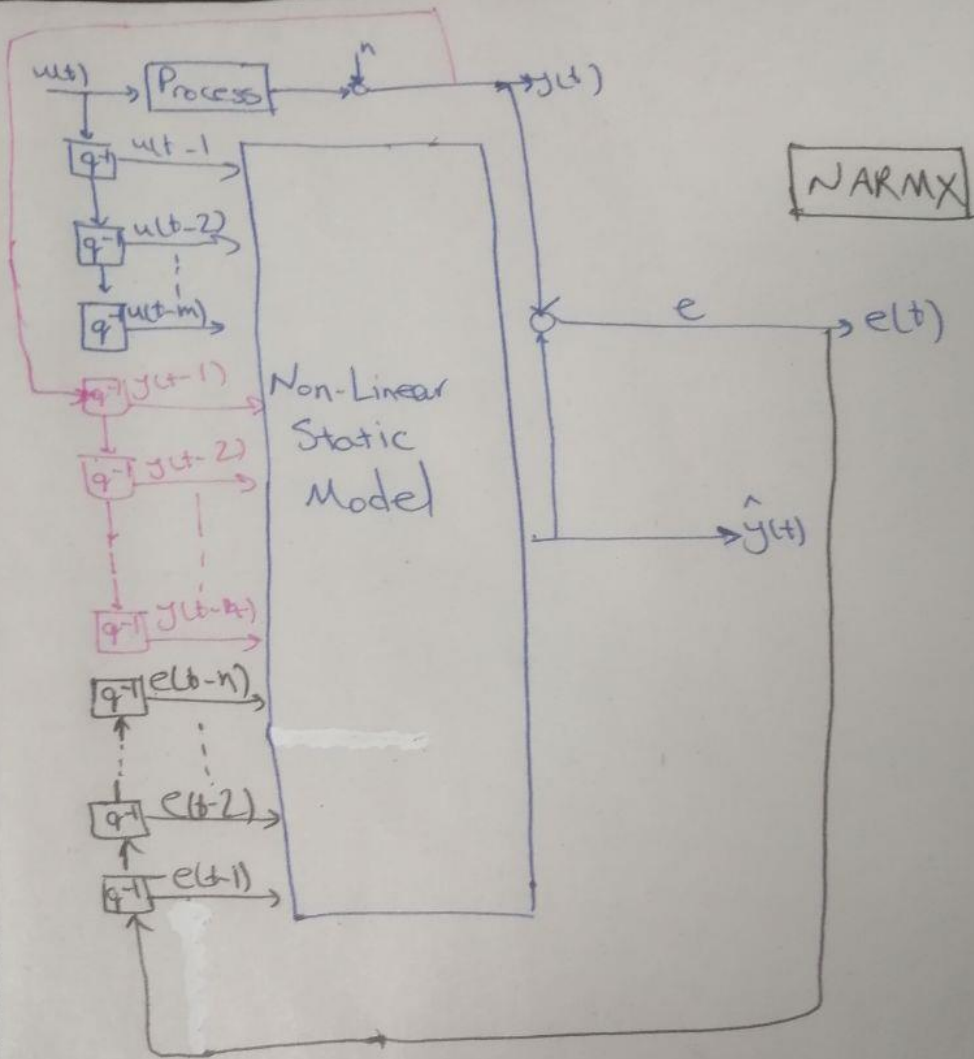


Mohamad Alikhani

40109494




۱ هماهنگ نشده

نام: _____

نام خانوادگی: _____

امتحان درس: _____

تاریخ امتحان: _____



دوره: _____

رشته: _____

کل نمره نهایی: _____

شماره دانشجویی: _____

نام استاد: _____

نام: _____

نام خانوادگی: _____

امتحان درس: _____

تاریخ امتحان: _____

$$MRBF: \hat{y} = \frac{\sum_{i=1}^n w_i \varphi_i(\|u - c_i\| \xi_i)}{\sum_{i=1}^n \varphi_i(\|u - c_i\| \xi_i)}, \quad \hat{y} = \sum_{i=1}^n w_i \hat{\varphi}_i$$

$$\tilde{\varphi}_i = \frac{\varphi_i(\|u - c_i\| \xi_i)}{\sum_{j=1}^n \varphi_j(\|u - c_j\| \xi_j)}, \quad \sum_{i=1}^n \tilde{\varphi}_i = 1$$

این ماستی به کام زبون ها خواهد داشت
 در نتیجه در هنگام آموزش با تغییر F یک از زبون ها $\tilde{\varphi}$ مربوط به قبیل زبون ماستی از
 قرار می گیرند، این امر باعث کاهش ضرایب نسبت به تغییرات انحراف می شود
 RBF خواهد شد (در حالت RBF ، تغییرات مربوط به انحراف می کار یک زبون شد خواهد شد
 که در حالت $MRBF$ این میزان تغییرات فرمای زبون ما به بخش می شود) در نتیجه رفتار زیرومیلی
 $Extrapolate$ دارا از از دخی را بین زدگی نخواهد بود، و رفتار انحراف خواهد شد.

به عبارت دیگر نرمالیزه کردن سبب این می شود که اگر شبکه برای نام ورودی ها یکسان باشد

نرمالیزه کردن داده ها به معنای تبدیل آن ها به یک مقیاس مشخص است که می تواند باعث بهبود عملکرد شبکه شود. زمانی که از RBF در لایه های میانی استفاده می شود، نرمالیزه کردن داده ها می تواند باعث شود که تفاوت های اندازه گیری بین ورودی ها کاهش یابد و شبکه بهتر بتواند الگوهای پیچیده تری را یاد بگیرد.

بنابراین، در ساختارهایی مانند RBF، نرمالیزه کردن در لایه‌های میانی متداول است زیرا می‌تواند بهبود عملکرد شبکه و یادگیری الگوهای پیچیده‌تر کمک کند.

عمل `normalize` کردن تفسیرپذیری را کم می‌کند اما در مقابل، باعث می‌شود که داده‌ها در بازه یکسان قرار بگیرند. این عمل زمانی مهم می‌شود که داده‌ها در بازه‌های غیریکسان قرار داشته باشند و با عمل نرمالیزاسیون، داده‌ها در بازه و ارزش یکسان قرار بگیرند و شناسایی برای رگرسورهای مختلف به صورت یکسان صورت بگیرد. در غیر این صورت اگر داده‌ها در بازه‌های مختلف باشند مشکلات زیادی ایجاد می‌شود.

۲ هماهنگ نشده

برای ادغام دو مدل محلی در ساختار LLM و روش آموزشی LoLiMoT، می‌توان از روش‌های ترکیبی مانند ادغام مدل‌ها (model ensemble) و یا ادغام نتایج (result fusion) استفاده کرد. در ادغام مدل‌ها، دو مدل محلی به صورت موازی آموزش داده می‌شوند و سپس خروجی‌های آن‌ها توسط یک مدل مرکزی ترکیب می‌شوند. در ادغام نتایج، خروجی‌های دو مدل محلی به صورت موازی تولید می‌شوند و سپس توسط یک الگوریتم ترکیبی (مانند میانگین وزنی یا رای‌گیری) ترکیب می‌شوند.

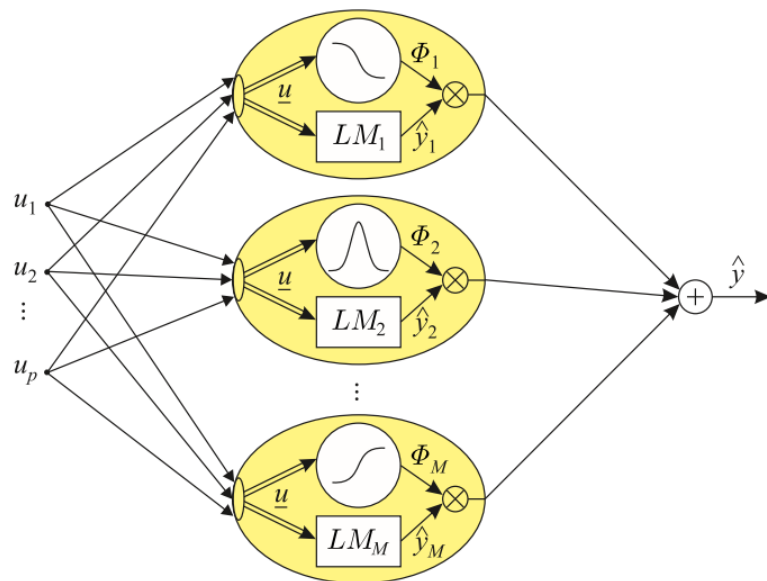
برای یک سیستم دو ورودی - یک خروجی، می‌توان از دو مدل محلی استفاده کرد که هر کدام از ورودی‌ها را به صورت جداگانه پردازش کنند و سپس خروجی‌های آن‌ها ترکیب شوند. به عنوان مثال، در یک سیستم تشخیص اشیاء، یک مدل محلی می‌تواند تصاویر را پردازش کرده و دیگری می‌تواند داده‌های سنسوری دیگر (مانند داده‌های صوتی یا لرزه) را پردازش کند. سپس خروجی‌های این دو مدل محلی توسط یک الگوریتم ترکیبی مانند میانگین وزنی یا رای‌گیری ترکیب می‌شوند تا خروجی نهایی سیستم تولید شود.

شرایط لازم برای ادغام دو مدل محلی شامل هماهنگی و توافق بر روی فرایند ترکیب خروجی‌ها، انطباق و همخوانی ورودی‌ها با مدل‌های محلی و تعیین وزن‌های مناسب برای ترکیب خروجی‌ها می‌باشد.

پیشنهاد ما برای یک سیستم دو ورودی - یک خروجی این است که از دو مدل محلی متفاوت با روش‌های آموزش متنوع استفاده شود تا اطمینان حاصل شود که تنها یک مدل محلی نیست که بهینه‌ترین خروجی را تولید می‌کند. سپس با استفاده از روش ادغام مناسب، خروجی‌های این دو مدل ترکیب شوند تا خروجی نهایی سیستم تولید شود.

در مدل local هر نتورک از دو بخش تشکیل شده است. اول LM و بخش دوم یک تابع برای validation است. پس یک خروجی \hat{y} با دو ورودی $u=[u_1;u_2]$ یک superposition از M تا مدل وزن دار است. (M تعداد نورون هاست).

فانکشن validity قانون‌ها را ارائه داده و هر کدام از مدل‌های local این قانون‌ها را برای بدست آوردن \hat{y} اجرا می‌کنند. برای transition مدل‌های محلی فانکشن‌ها بین ۰ و ۱ هستند.



$$\hat{y}(t) = f(u(t-1), u(t-2), \hat{y}(t-1))$$

$$\rightarrow a_1 u(t-1) + a_2 u(t-2) + a_3 \hat{y}(t-1) + a_4 u^2(t-1) + a_5 u^2(t-2) + a_6 \hat{y}^2(t-1) \\ + a_7 u(t-1)u(t-2) + a_8 u(t-1)\hat{y}(t-1) + a_9 u(t-2)\hat{y}(t-1)$$

The cost function:

$$j = \frac{1}{2} e^2$$

Second power coefficient:

$$a_4(t) = a_4(t-1) - \eta \frac{\partial j}{\partial a_4(t-1)} = a_4(k) - \eta \frac{\partial j}{\partial e} \times \frac{\partial e}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial a_4} = a_4(k) + \eta e u^2(t-1)$$

$$a_5(t) = a_5(t-1) - \eta \frac{\partial j}{\partial a_5(t-1)} = a_5(k) - \eta \frac{\partial j}{\partial e} \times \frac{\partial e}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial a_5} = a_5(k) + \eta e u^2(t-2)$$

$$a_6(t) = a_6(t-1) - \eta \frac{\partial j}{\partial a_6(t-1)} = a_5(k) - \eta \frac{\partial j}{\partial e} \times \frac{\partial e}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial a_6}$$

$$\frac{\partial \hat{y}}{\partial a_6} = \frac{\partial f}{\partial a_6} + \underbrace{\frac{\partial f}{\partial \hat{y}(t-1)} \times \frac{\partial \hat{y}(t-1)}{\partial a_6}}_{BPTT}$$

The BPTT (Backpropagation through time) depends on the responses of the past samples, in another word, it is recursive. And the final equation for the algorithm is:

$$\frac{\partial \hat{y}(t-i)}{\partial a_6} = \frac{\partial f}{\partial a_6} + \frac{\partial f}{\partial \hat{y}(t-i-1)} \times \frac{\partial \hat{y}(t-i-1)}{\partial a_6}$$

And we continue upon reaching:

$$\hat{y}(0) = 0 \text{ or } \frac{\partial \hat{y}(0)}{\partial a_6} = 0$$

As is shown, "Backpropagation Through Time" happens, and updating parameters requires the previous steps' values. This happens in recurrent networks, and we do the propagation for a few steps back and ignore the rest of the steps. This method of countering this issue is called truncated BPTT.

سوال ۴ هماهنگ نشده

Contributed by:

Jairo Espinosa
ESAT-SISTA KULEUVEN
Kardinaal Mercierlaan 94

B-3001 Heverlee Belgium

jairo.espinosa@esat.kuleuven.ac.be

Description:

The data comes from a model of a Steam Generator at Abbott Power Plant in Champaign IL.
The model is described in the paper of Pellegrinetti [1].

Sampling:

3 sec

Number:

9600

Inputs:

u1: Fuel scaled 0-1
u2: Air scaled 0-1
u3: Reference level inches
u4: Disturbance defined by the load level

Outputs:

y1: Drum pressure PSI
y2: Excess Oxygen in exhaust gases %
y3: Level of water in the drum
y4: Steam Flow Kg./s

References:

[1] G. Pellegrinetti and J. Benstman, Nonlinear Control Oriented Boiler Modeling -A Benchmark Problem for Controller Design, IEEE Tran. Control Systems Tech. Vol.4No.1 Jan.1996

[2] J. Espinosa and J. Vandewalle Predictive Control Using Fuzzy Models Applied to a Steam Generating Unit, Submitted to FLINS 98 3rd. International Workshop on Fuzzy Logic Systems and Intelligent Technologies for Nuclear Science and Industry

Properties:

To make possible the open loop identification the water level was stabilized by applying to the water flow input a feedforward action proportional to the steam flow with value 0.0403 and a PI action with values $K_p=0.258$ $T_i=1.1026e-4$ the reference of this controller is the input u3.

Columns:

Column 1: time-steps
Column 2: input fuel
Column 3: input air
Column 4: input level ref.
Column 5: input disturbance
Column 6: output drum pressure
Column 7: output excess oxygen
Column 8: output water level
Column 9: output steam flow

Category:

Process industry systems

Where:

<ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/espinosa/datasets/powplant.dat>

ابتدا دیتا را در متلب بارگذاری میکنیم:

```
%% data
load steamgen
c=steamgen;
p=[c((87-7):(9600-7),2)';c(87:9600,3)';c(87:9600,4)';c((87-1):(9600-1),5)';c((1):(9600-86),5)';c((87-9):(9600-9),6)';c((87-6):(9600-6),7)';c((87-49):(9600-49),8)'];
T=c(87:9600,9)';
num_traindata=3500;
num_testdata=4000;
```

با استفاده از دستور های زیر شبکه را نرمالیزه میکنیم:

```
%% normalizing
[p2,ps] = mapminmax(p);
[t2,ts] = mapminmax(T);
```

شبکه MLP تک لایه

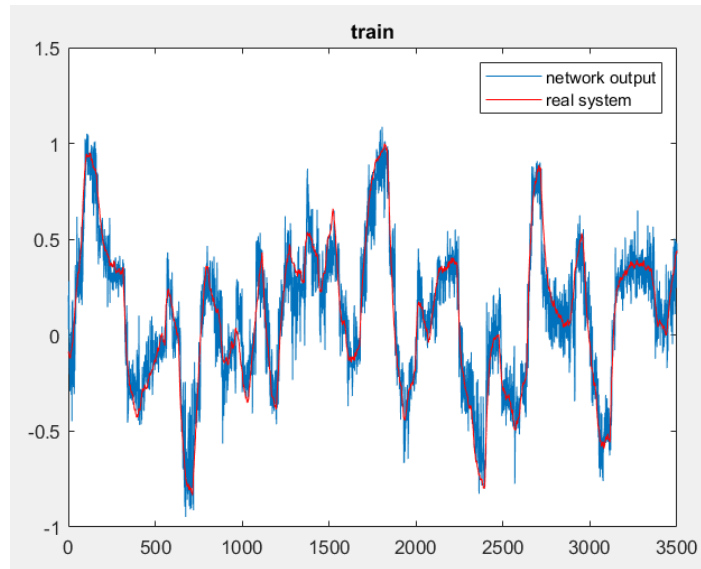
شبکه عصبی MLP تک لایه با ۲۵ نرون لایه پنهان

```
m=25;
epochs =150;
net=newff(minmax(p2),[20 1],{'tansig' 'purelin'},'trainlm')
net.trainParam.epochs = epochs;
net.trainParam.lr=0.00001;
[net,tr]=train(net,p2(:,1:num_traindata),t2(1:num_traindata));
% train
a2 = sim(net,p2(:,1:num_traindata));
e=a2-t2(1:num_traindata);
msetrain=mse(e)
% interpolate
a3 = sim(net,p2(:,(num_traindata+1):num_testdata));
e1=a3-t2((num_traindata+1):num_testdata);
msetest=mse(e1)
```

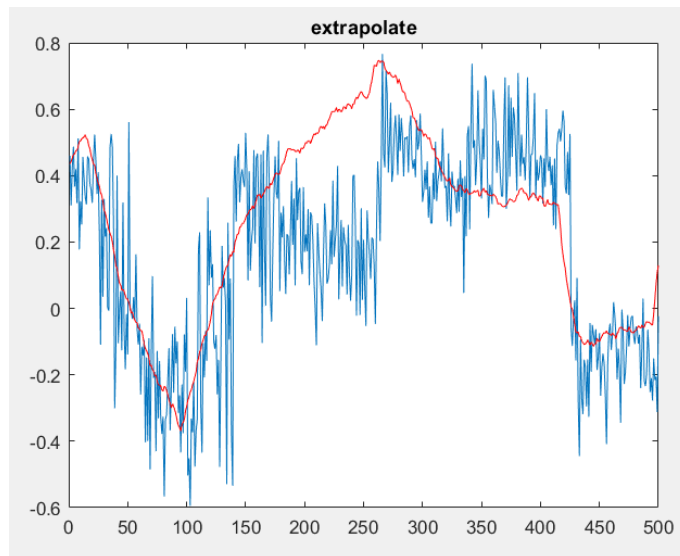
پلات نهایی

در کل دو عدد پلات برای این مساله برای شناسایی هر سیستم MISO داریم:

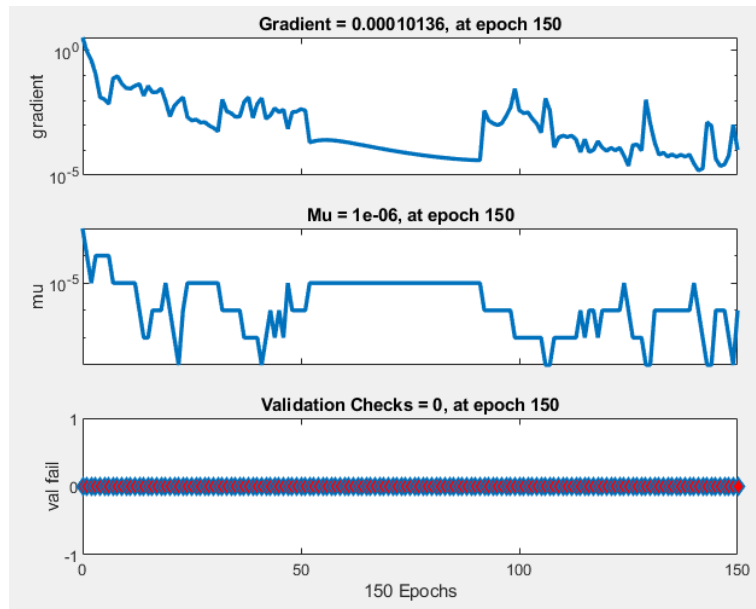
۴ ورودی خروجی اول



در شکل بالا دیده میشود که عمل آموزش نسبتاً خوب انجام شده است.

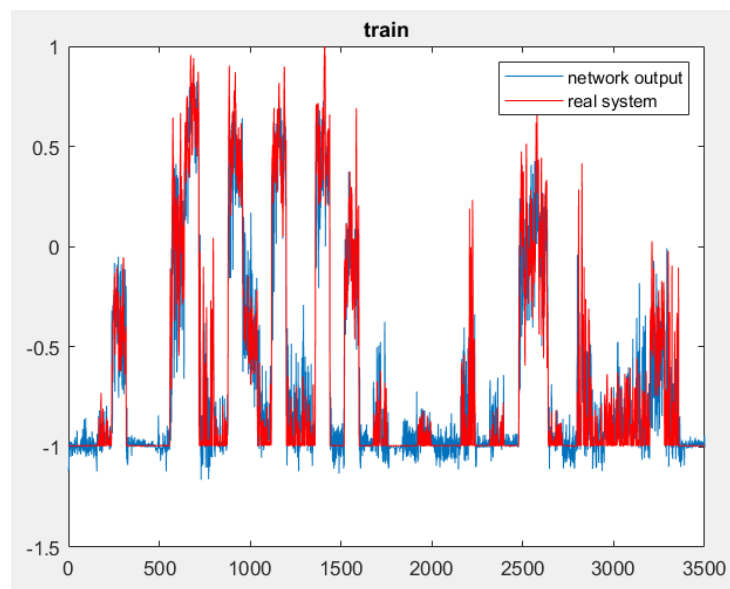


به دلیل سریع بودن دینامیک های خروجی دنبال کنندگی خروجی به خوبی انجام نشده است.

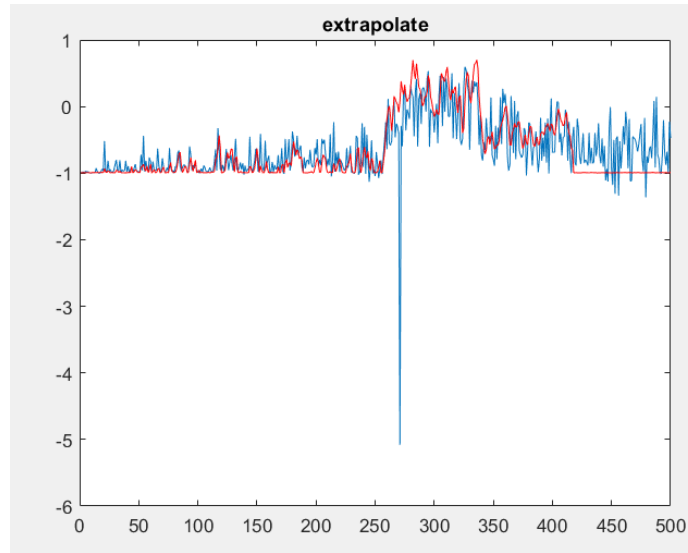


نمودارهای خطا نوسانات زیادی داشته است. بخشی از آن میتواند به خاطر نویز گرادیان باشد و یا میتواند به علت بزرگ بودن نرخ آموزش باشد. و یا اینکه شبکه مناسب نیست.

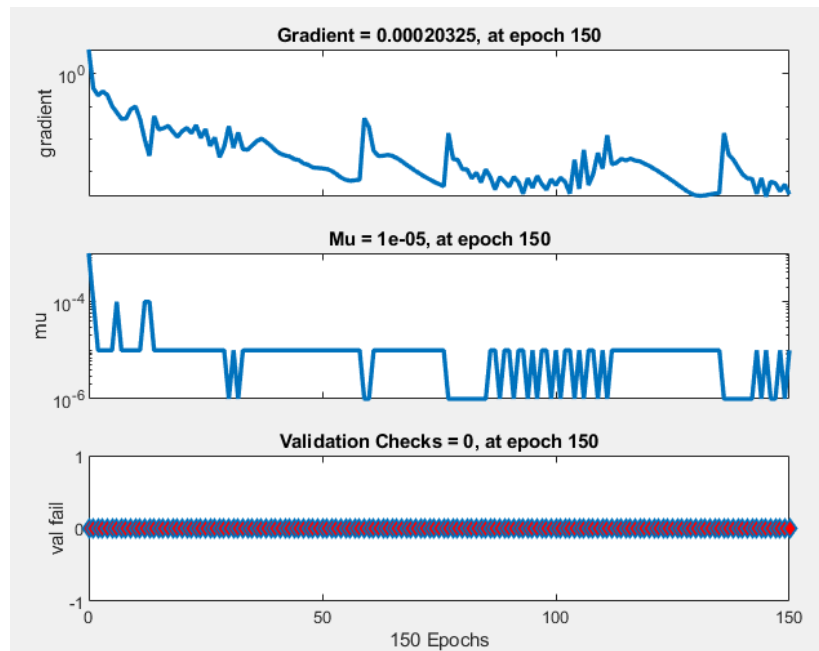
۴ ورودی خروجی دوم



همینطور که مشاهده میشود به علت دینامیکهای سنگین خروجی دوم، موفقیت شناسایی با MLP تک لایه کمی دور از انتظار به نظر میرسد.

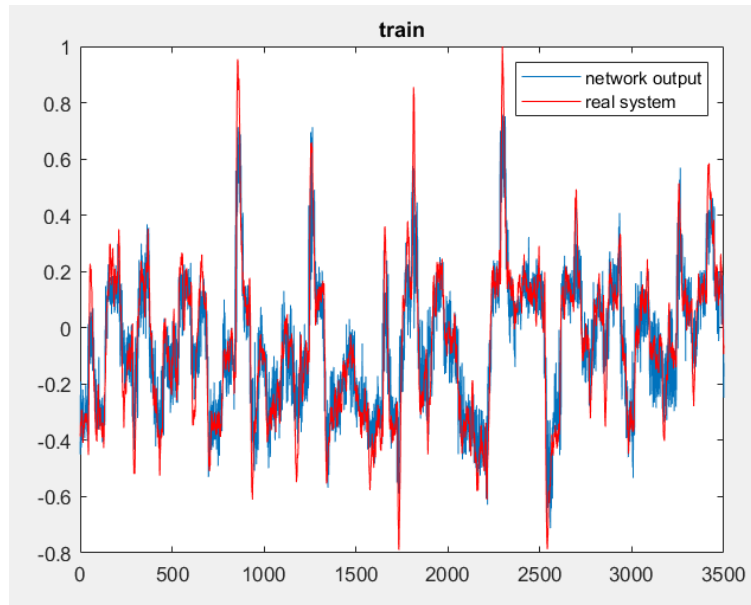


با توجه به نمودار میتوان اینگونه نتیجه گرفت آموزش به درستی انجام نشده است. و میتوانستیم با افزایش پیچیدگی شبکه جواب بهتری بگیریم.

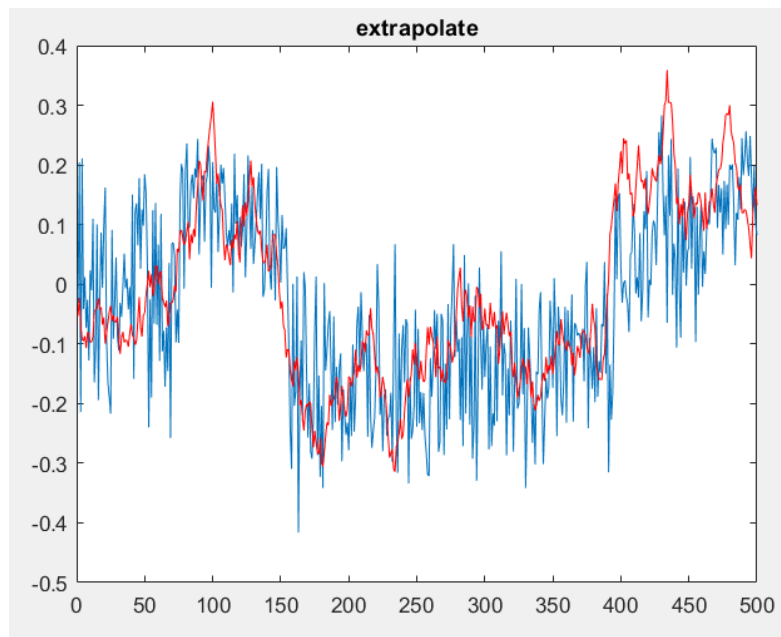


در ایپاک های ۶۰، ۷۵ و ۱۳۰ دارای افزایش خطا هستیم. که این عمر زیبنده نیست 😊

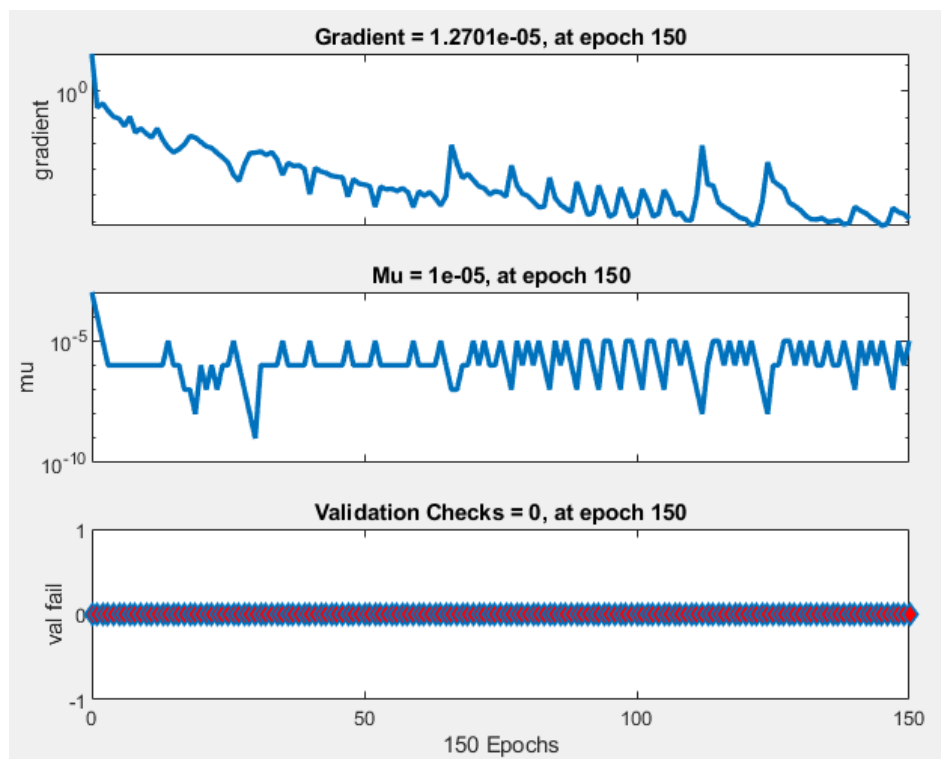
۴ ورودی خروجی سوم



آموزش به صورت قابل قبولی صورت گرفته است

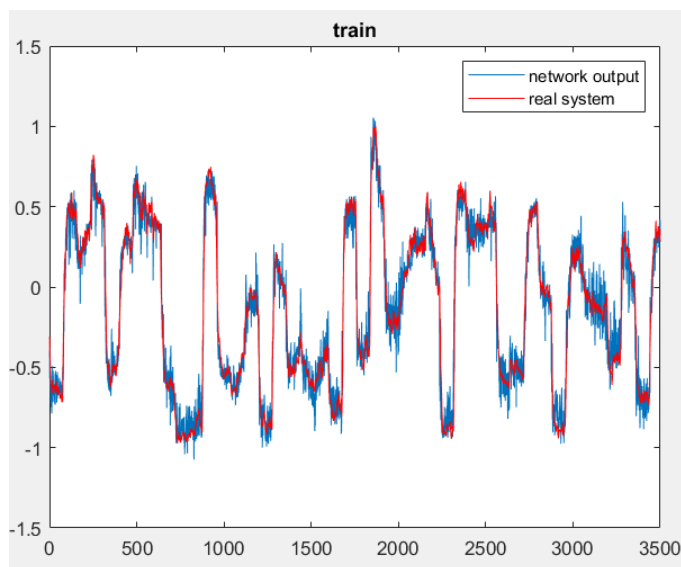


نتیجه حاصل از آموزش خروجی ۳.

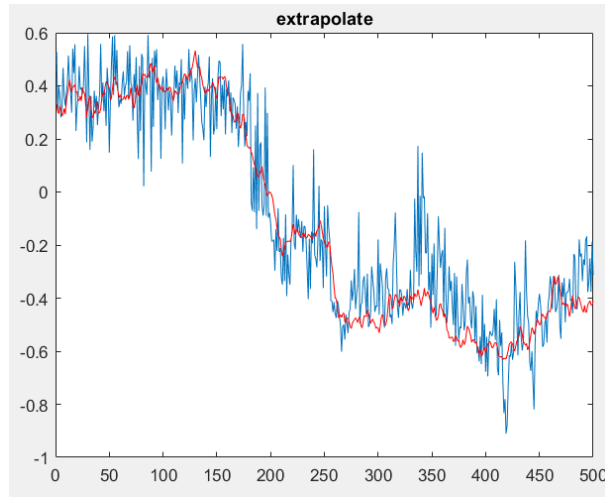


از ایپاک ۱۰۰ به بعد مشاهده می‌شود که در دو مرحله میزان خطا افزایش یافته است.

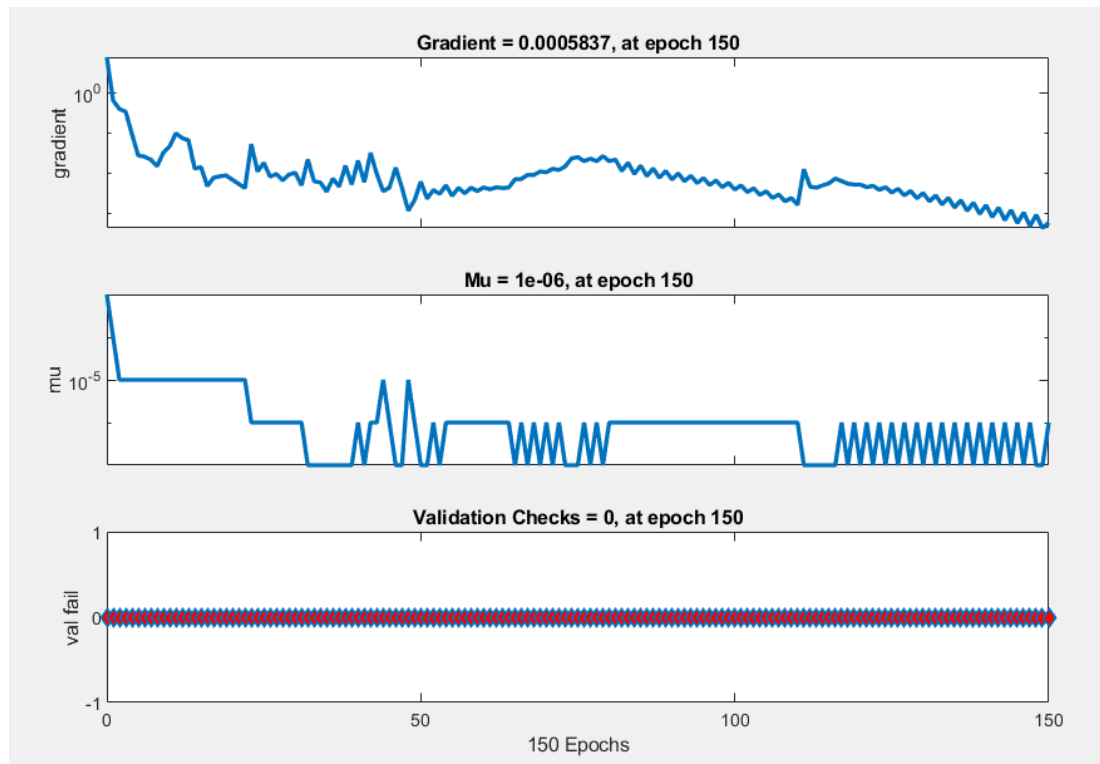
۴ ورودی خروجی چهارم



نتیجه آموزش شبکه با داده های خروجی ۴



خروجی حاصل از مرحله تست.



خطای نهایی برای خروجی ۴

چون سیستم یک سیستم صنعتی واقعی بود نمیتوانستیم با یک لایه MLP آن را شناسایی کنیم، نیاز به شبکه پیچیده تری داریم. در همه موارد شبکه underfit شده است به این دلیل که شبکه برای داده های موجود زیادی ضعیف است.

شبکه MLP دو لایه

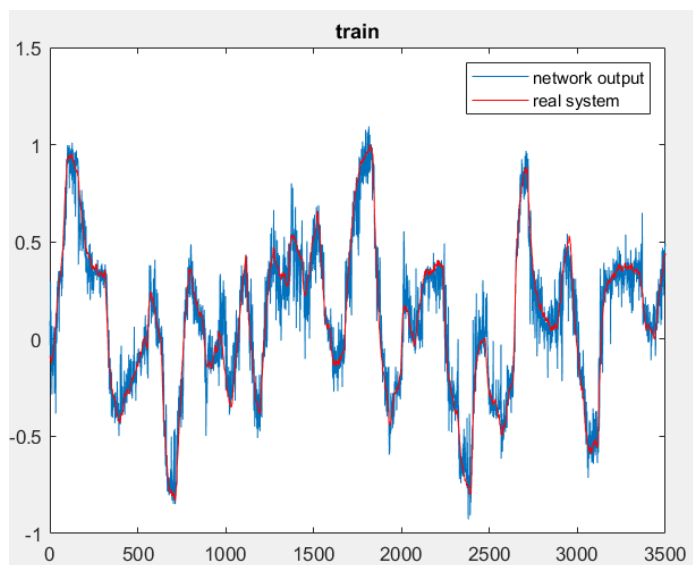
با توجه به underfit شدن شبکه تک لایه، انتظار داریم نتایج حاصل از شبکه دولایه بهتر شوند.

```
%% newff
epochs =150;
net=newff(minmax(p2),[8 25 1],{'tansig' 'tansig'
'purelin'},'trainlm')
net.trainParam.epochs = epochs;
net.trainParam.lr=0.00001;
[net,tr]=train(net,p2(:,1:num_traindata),t2(1:num_traindata));
a2 = sim(net,p2(:,1:num_traindata));
e=a2-t2(1:num_traindata);
mse_train=mse(e)
a3 = sim(net,p2(:,(num_traindata+1):num_testdata));
e1=a3-t2((num_traindata+1):num_testdata);
mse_test=mse(e1)
```

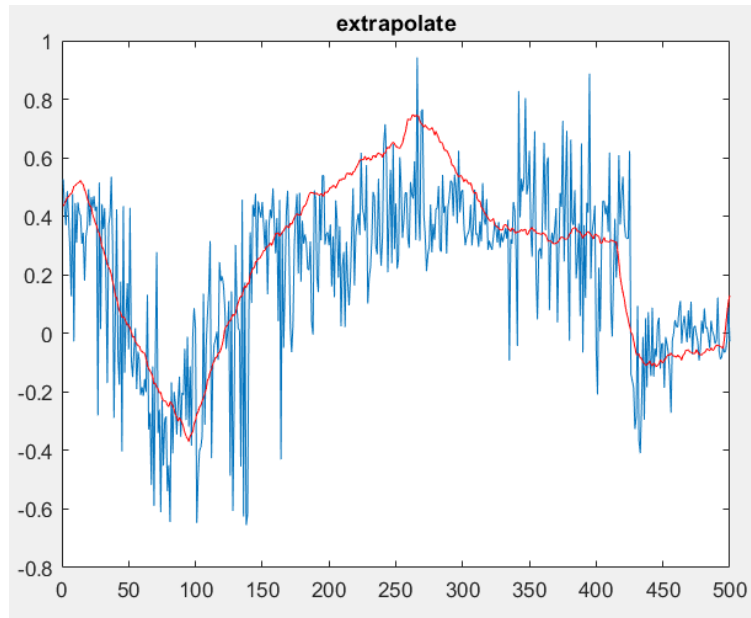
۸ نرون لایه اول

۲۵ نرون لایه دوم

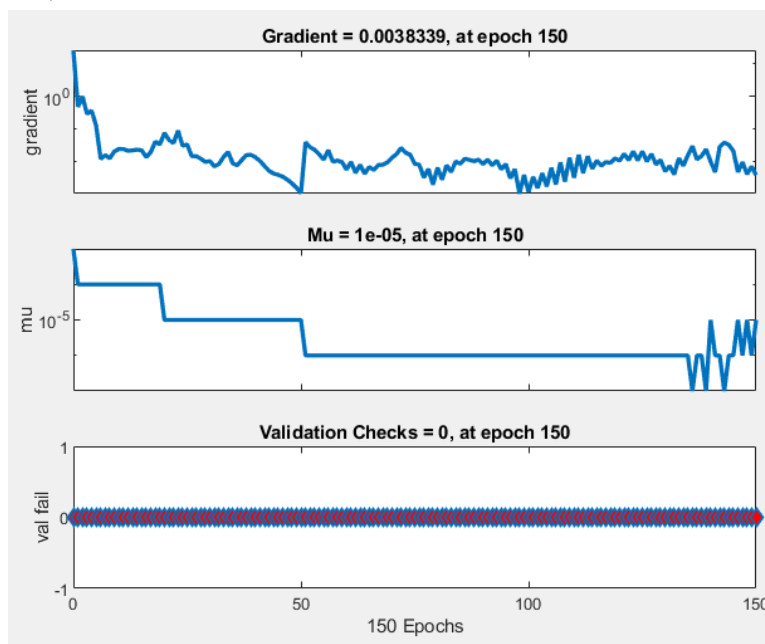
۴ ورودی خروجی اول



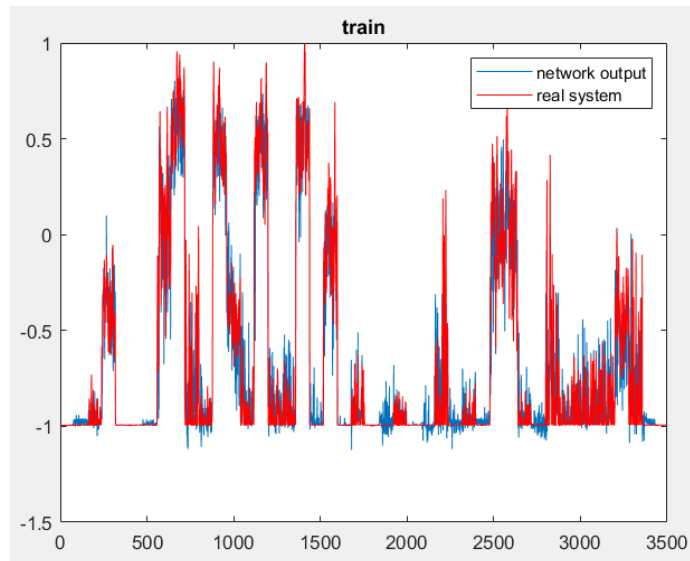
نتایج حاصل از آموزش شبکه دولایه به روی خروجی اول



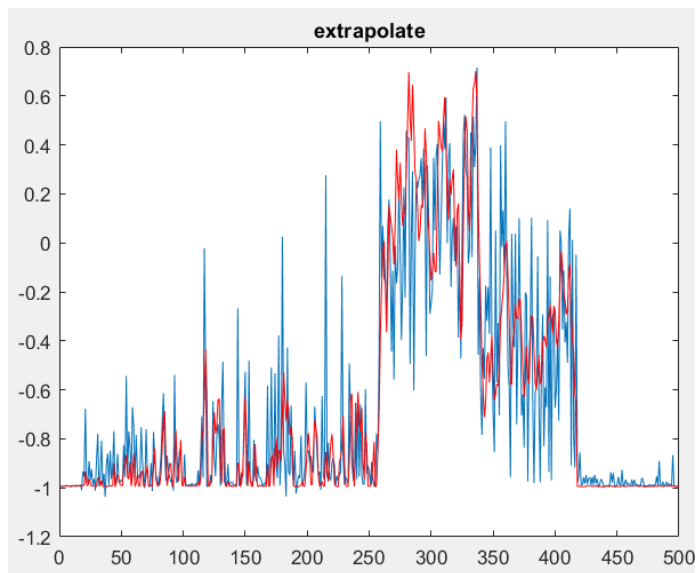
نتیجه شبکه بر روی داده های تست. بهتر از حالت قبل انجام شده.

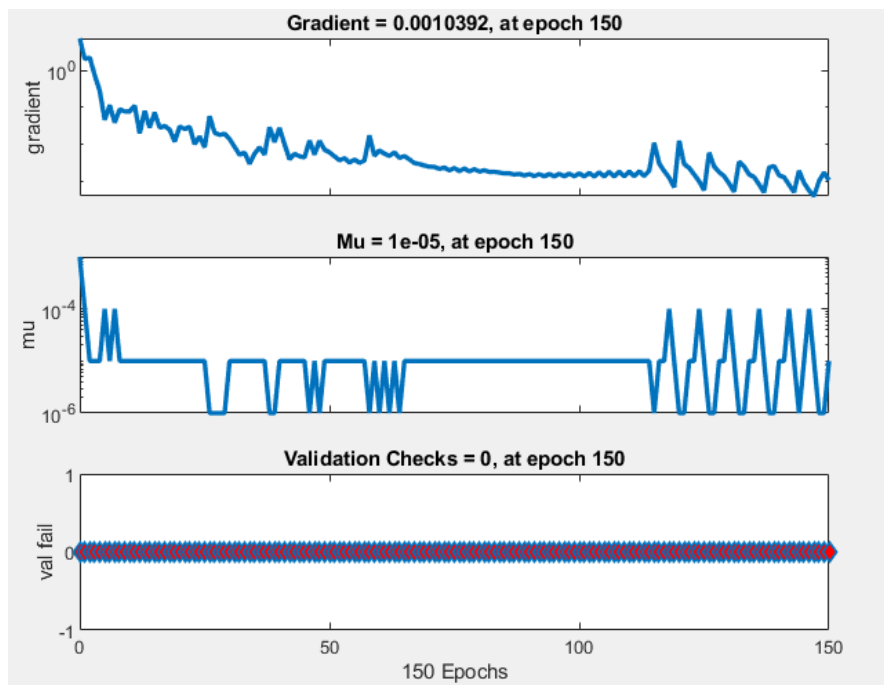


حتی منحنی خطا هم بهتر و نرم تر شده است
۴ ورودی خروجی دوم



نتایج آموزش روی خروجی ۲، با توجه به دینامیک سنگین داده میتوان نتیجه گرفت که حتی شبکه دولایه هم برای آموزش این خروجی کافی نیست. البته با مقایسه این نمودار و نمودار ورودی اول مشخص میشود که تا حدی توانسته دینامیک های خروجی را دنبال کند.

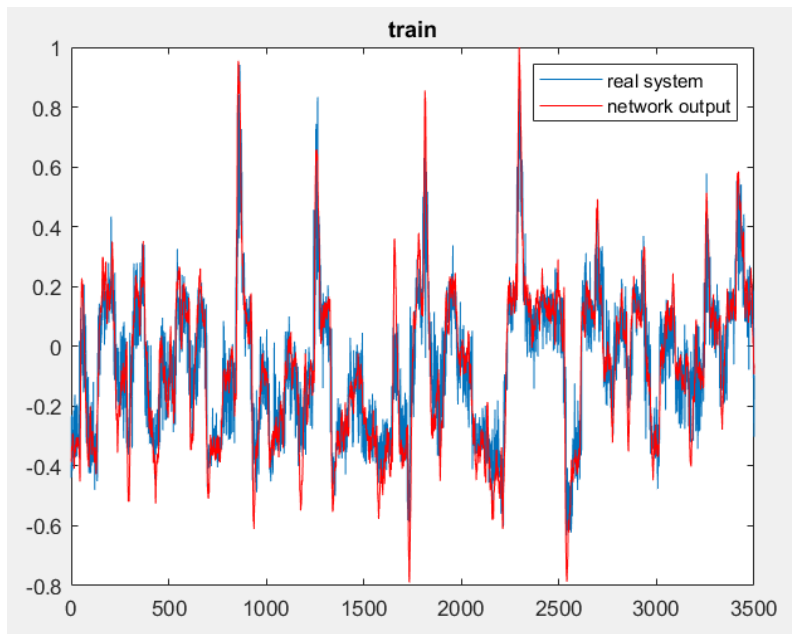




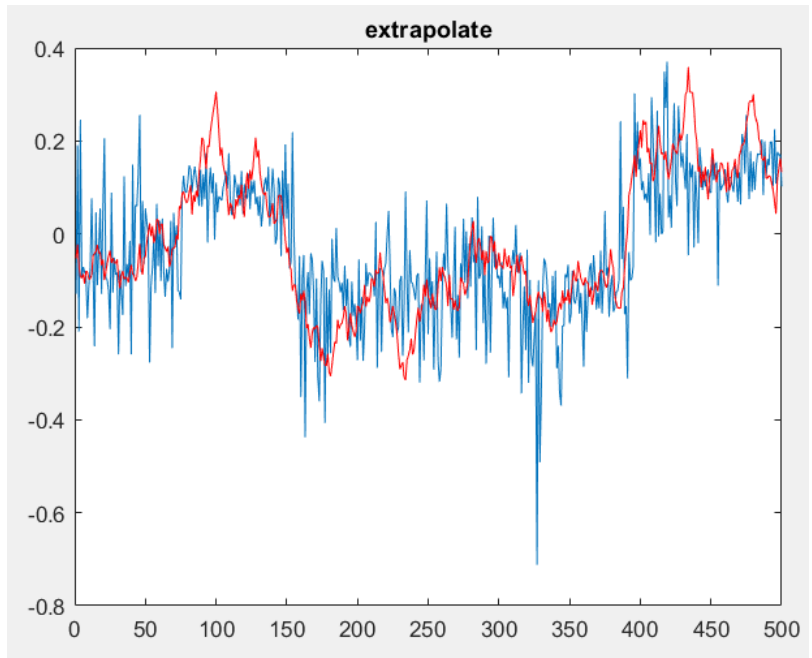
با این وجود، خطا تا حد مناسبی کاهش پیدا کرده است

توجه شود که در راهنمای شکل های آموزش اشتباه شده بود. از الان به بعد تصحیح شده است.

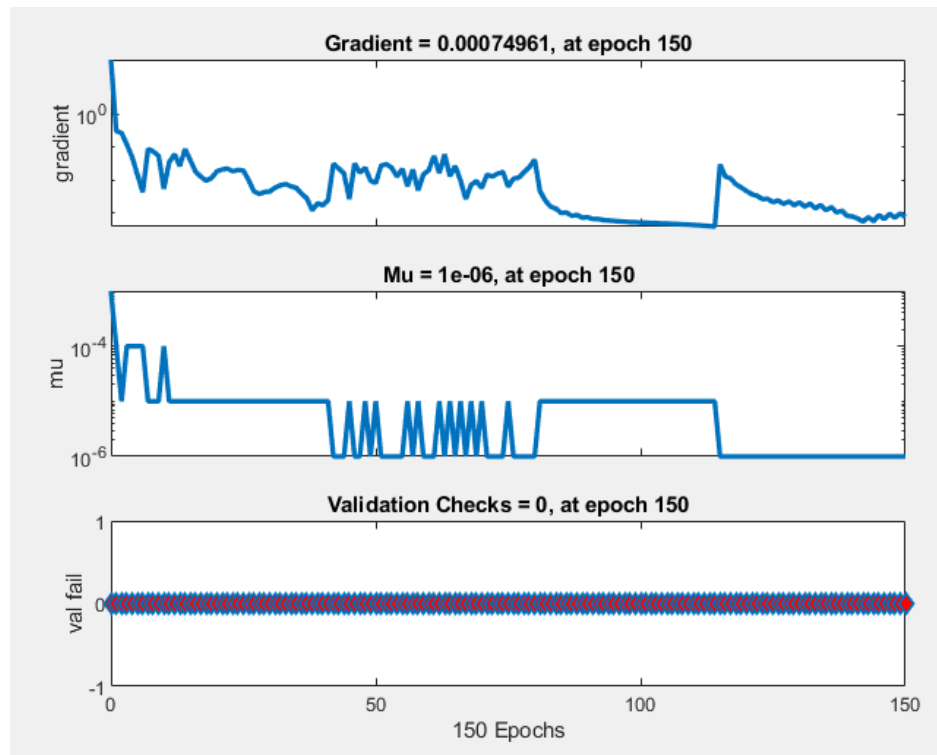
۴ ورودی خروجی سوم



نمودار آموزش



نمودار تست



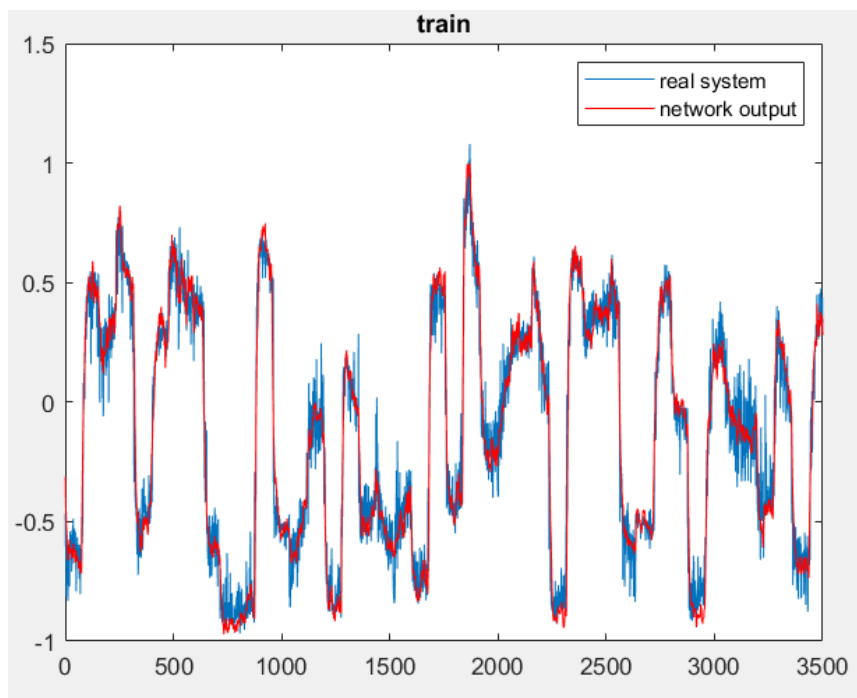
خطا به حد قابل ملاحظه ای کاهش یافته است (در حدود ۰,۰۰۱) و این چیز خوبی است. این

موضوع نشاندهنده آن است که:

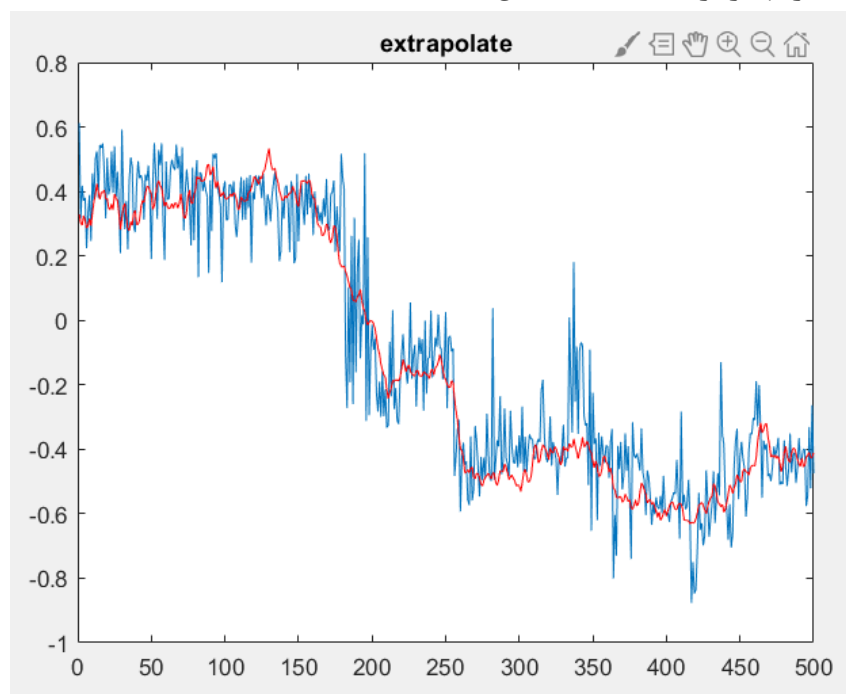
We are on the right track

۴ ورودی خروجی چهارم

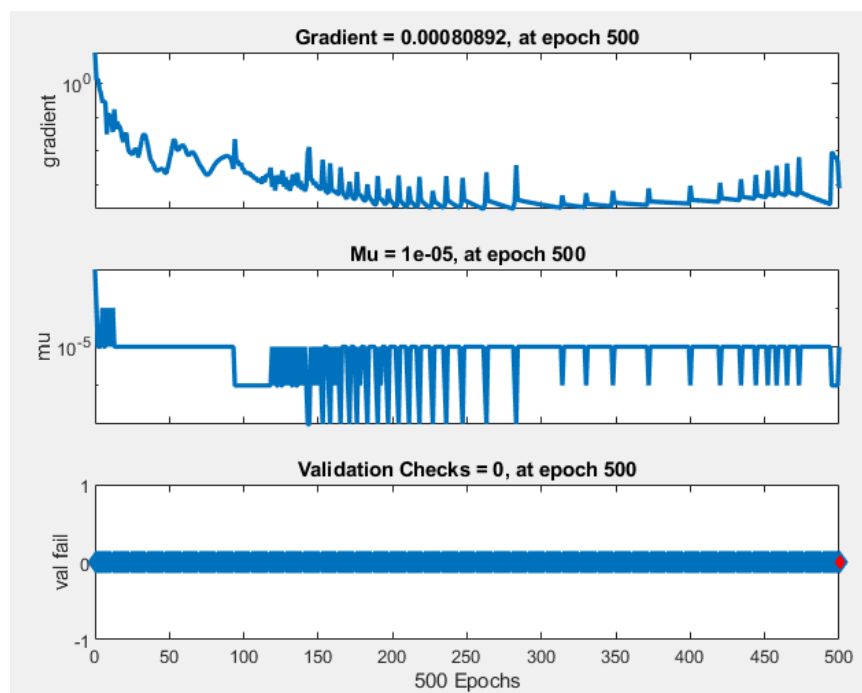
حال به سراغ یک ابتکار میرویم. آیا افزایش تعداد ایپاک سبب بهتر شدن نتیجه میشود؟
ایپاک ۵۰۰



نتیجه آموزش به نظر بهتر از حالت های قبل است.



نتیجه تست، تقریبا به همان منوال سابق است.



خطا در بازه ۰,۰۱ و ۰,۰۰۱ است

نتیجه با افزایش تعداد ایپاک بهتر نشده است.

RBF

با خط های پایین داده را نرمالیزه میکنیم

```
%% normalizing
for i=1: num_testdata
u(i,1)=(U(i,1)-min(U(:,1)))/(max(U(:,1))-min(U(:,1)));
u(i,2)=(U(i,2)-min(U(:,2)))/(max(U(:,2))-min(U(:,2)));
u(i,3)=(U(i,3)-min(U(:,3)))/(max(U(:,3))-min(U(:,3)));
u(i,4)=(U(i,4)-min(U(:,4)))/(max(U(:,4))-min(U(:,4)));
u(i,5)=(U(i,5)-min(U(:,5)))/(max(U(:,5))-min(U(:,5)));
u(i,6)=(U(i,6)-min(U(:,6)))/(max(U(:,6))-min(U(:,6)));
u(i,7)=(U(i,7)-min(U(:,7)))/(max(U(:,7))-min(U(:,7)));
z(i)=(T(i)-min(T))/(max(T)-min(T));
end
```

کد های مزبوط به شبکه RBF

```
%% rbf
m=30;
epoch=10;
etha1=.01;
etha=0.8;
sigma1=.5;
sigma2=.5;
sigma3=.5;
sigma4=.5;
sigma5=.5;
sigma6=.5;
sigma7=.5;
c=1+5*rand(7,m);
w=-1+2*rand(m,1);
p=1000*eye(m);
for k=1:epoch
% data=randperm(121);
for i=1:num_traindata
for j=1:m
t1(1,j)= ((u(i,1)-c(1,j))/sigma1)^2;
t2(1,j)=((u(i,2)-c(2,j))/sigma2)^2;
t3(1,j)=((u(i,3)-c(3,j))/sigma3)^2;
t4(1,j)=((u(i,4)-c(4,j))/sigma4)^2;
t5(1,j)=((u(i,5)-c(5,j))/sigma5)^2;
t6(1,j)=((u(i,6)-c(6,j))/sigma6)^2;
t7(1,j)=((u(i,7)-c(7,j))/sigma7)^2;
phi(1,j)=exp(-
.5*(sqrt(t1(j)+t2(j)+t3(j)+t4(j)+t5(j)+t6(j)+t7(j))));
end

xt=phi';
yhat(i)=xt'*w;
e1(i)=z(i)-yhat(i);
```



```

    for j=1:m
        c(:,j)=c(:,j)+etha*e1(i)*w(j,1)*[(u(i,1)-
c(1,j))/(sigma1^2);(u(i,2)-c(2,j))/(sigma2^2);(u(i,3)-
c(3,j))/sigma3^2;(u(i,4)-c(4,j))/(sigma4^2);(u(i,5)-
c(5,j))/(sigma2^5);(u(i,6)-c(6,j))/sigma6^2;(u(i,7)-
c(7,j))/sigma7^2]*phi(j);

sigma1=sigma1+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma1)^3)*((u(i,1)-
c(1,j))^2);

sigma2=sigma2+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma2)^3)*((u(i,2)-
c(2,j))^2);

sigma3=sigma3+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma3)^3)*((u(i,3)-
c(3,j))^2);

sigma4=sigma4+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma4)^3)*((u(i,4)-
c(4,j))^2);

sigma5=sigma5+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma5)^3)*((u(i,5)-
c(5,j))^2);

sigma6=sigma6+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma6)^3)*((u(i,6)-
c(6,j))^2);

sigma7=sigma7+etha1*e1(i)*w(j,1)*phi(j)*(1/(sigma7)^3)*((u(i,7)-
c(7,j))^2);
    end
    gama=p*xt/(1+xt'*p*xt);
    p=(eye(m)-gama*xt')*p;
    t(i)= trace(p);
    if t(i)>1000
        p=1000*eye(m);
    end
    w=w+gama*e1(i);

end
mse_train(k)=mse(e1)
for h=(num_traindata+1):num_testdata
    for g=1:m
        phi1(1,g)=exp(-.5*(sqrt(((u(h,1)-c(1,g))/sigma1)^2+((u(h,2)-
c(2,g))/sigma2)^2+((u(h,3)-c(3,g))/sigma3)^2+((u(h,4)-
c(4,g))/sigma4)^2+((u(h,5)-c(5,g))/sigma5)^2+((u(h,6)-
c(6,g))/sigma6)^2+((u(h,7)-c(7,g))/sigma7)^2)));
        end

        xt1=phi1';
        yhat(h)=xt1'*w;
        e2(h)=z(h)-yhat(h);
        gama=p*xt1/(1+xt1'*p*xt1);
        p=(eye(m)-gama*xt1')*p;
        t(h)= trace(p);
    end
end

```

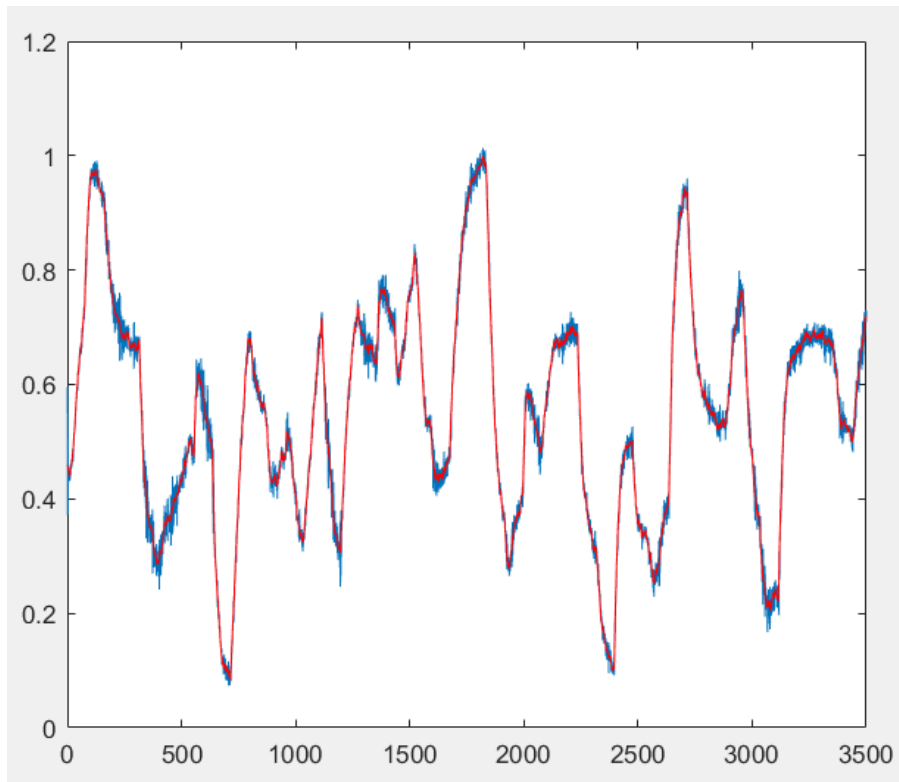
```

if t(h)>1000
    p=1000*eye(m);
end
w=w+gama*e2(h);

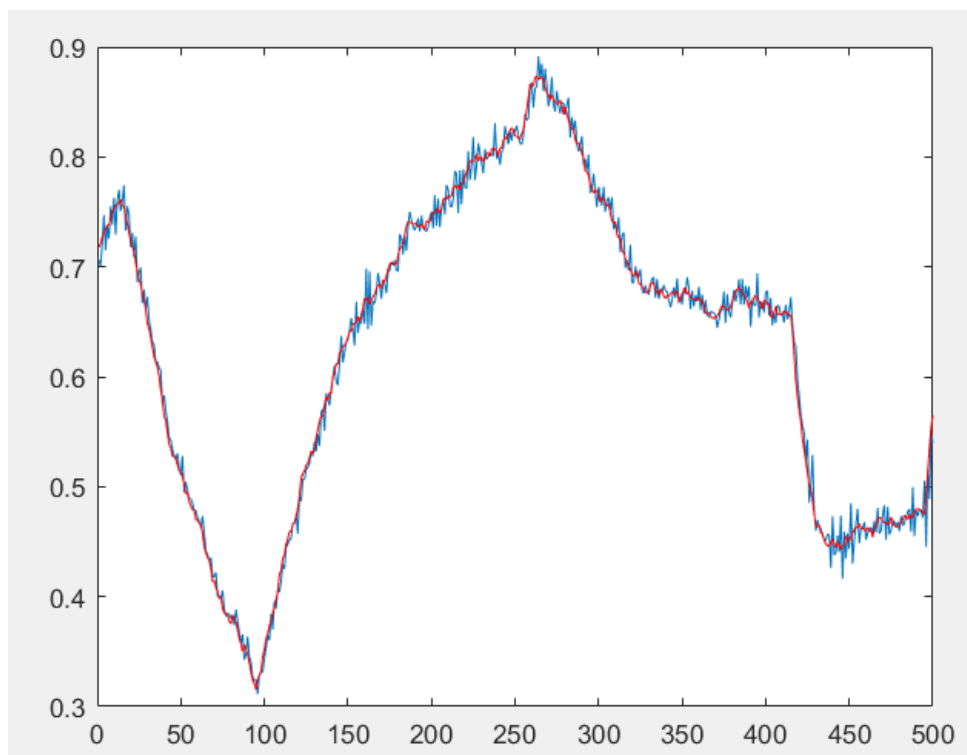
end
mse_test(k)=mse(e2((num_traindata+1):num_testdata))
end

```

خروجی اول



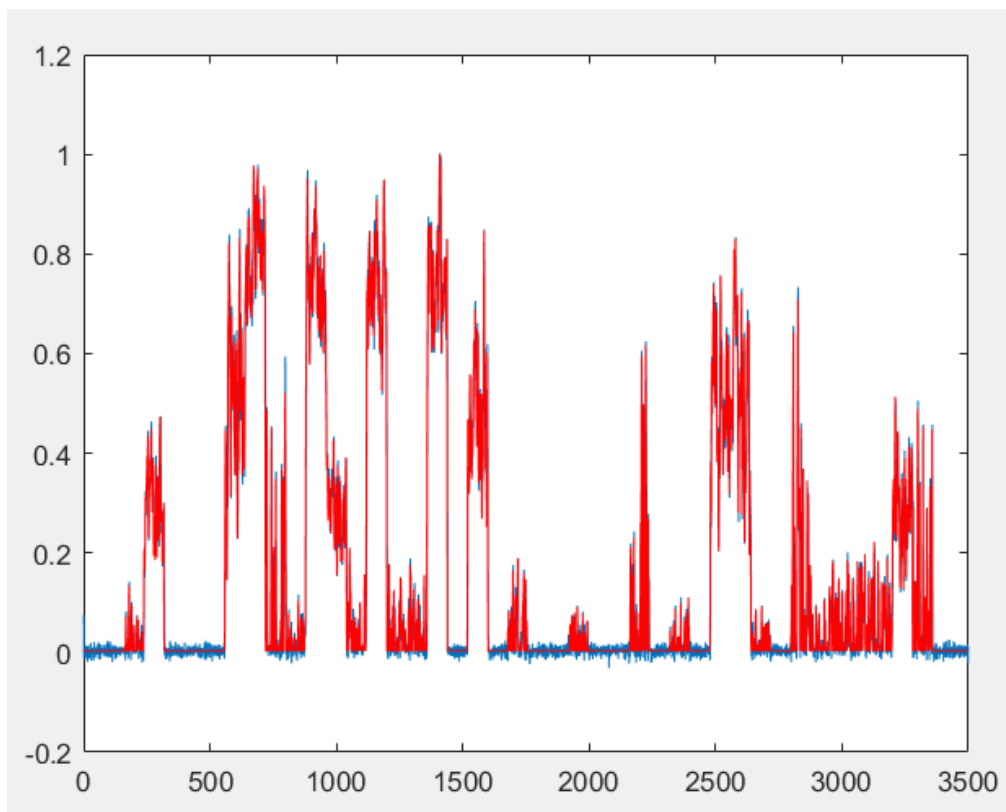
به نظر شبکه خیلی بهتر از حالت MLP جواب میدهد



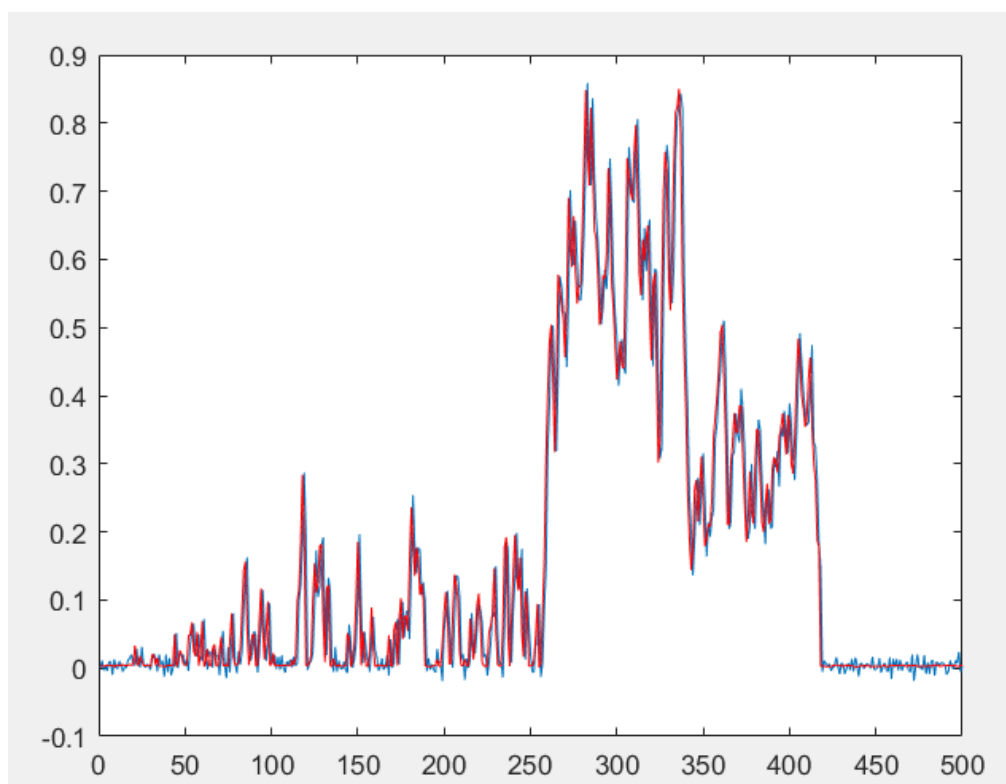
نتایج تست هم به نظر بهتر شده اند

آبی نمودار واقعی است
و قرمز نمودار تخمینی است

خروجی دوم

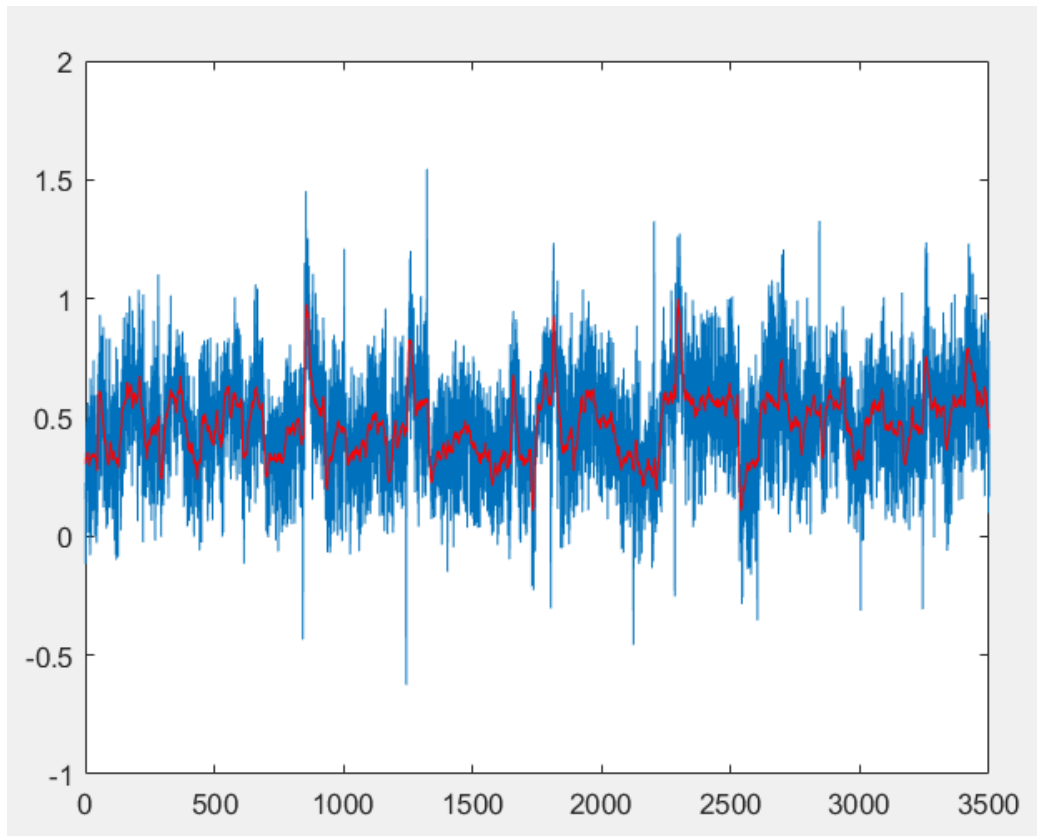


داده های آموزش تا حد زیادی نمودار تخمین و نمودار واقعی روی هم افتاده اند.



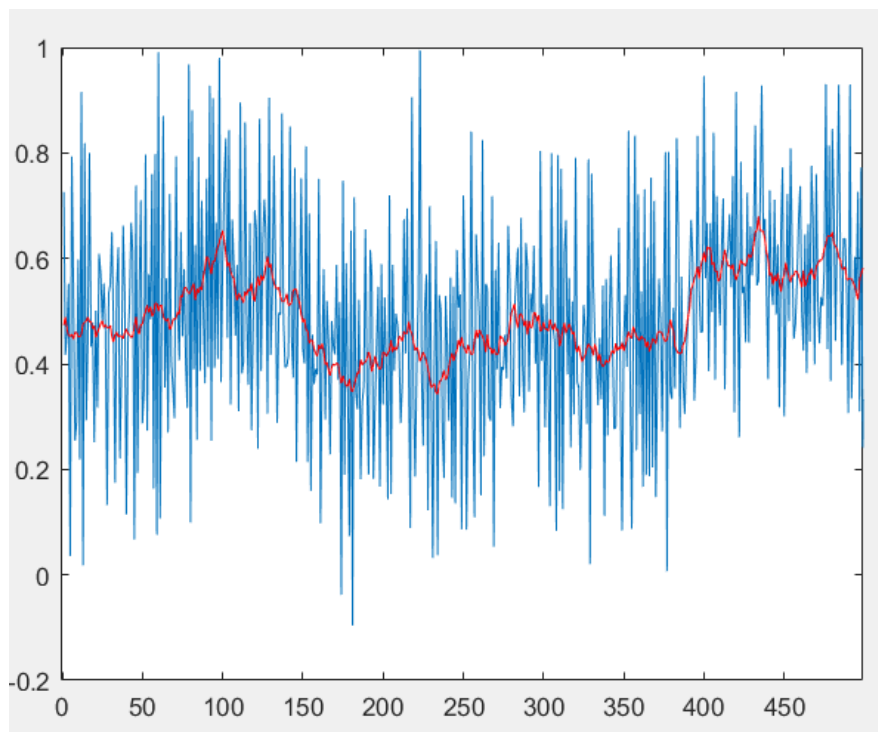
از داده تست مشاهده میشود که عملکرد شبکه rbf از MLP خیلی بهتر است.

خروجی سوم



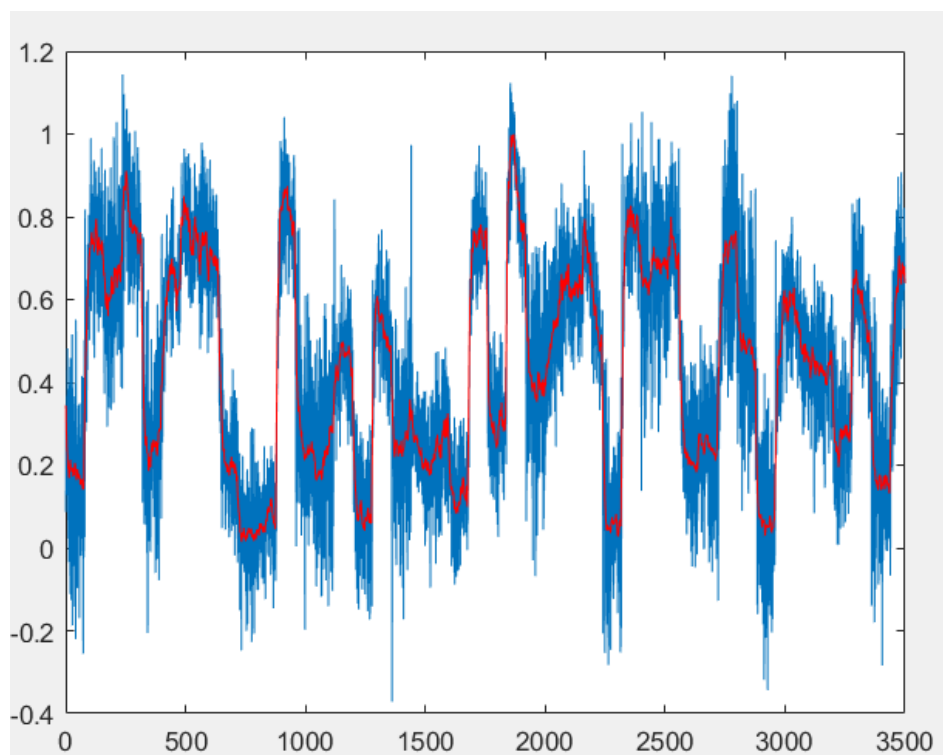
داده آموزش

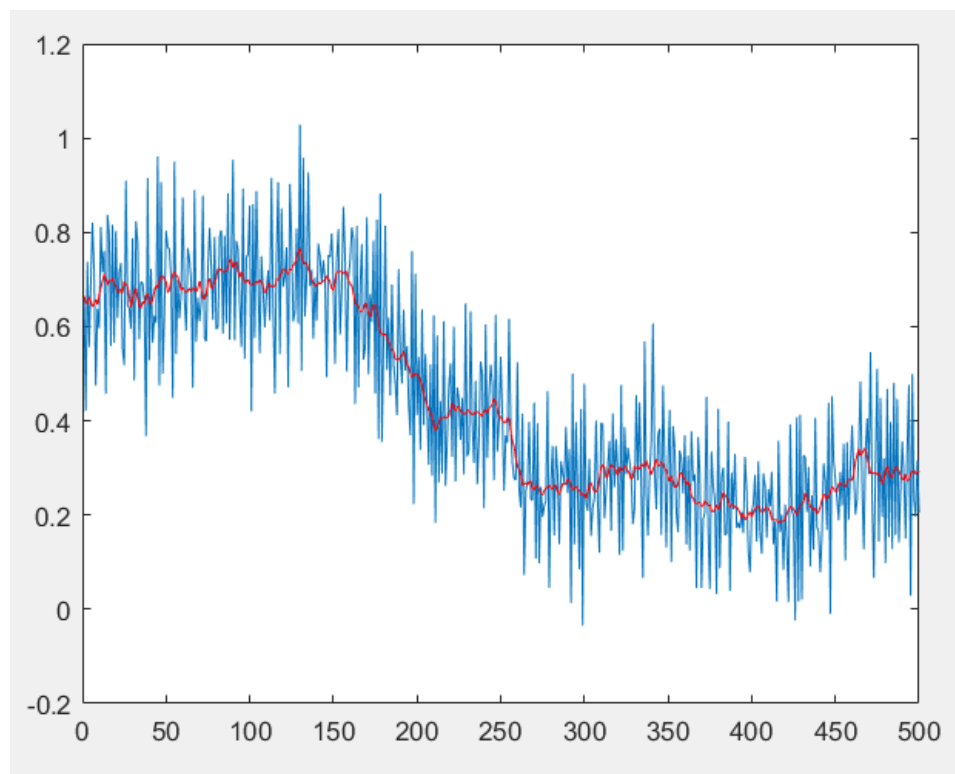
به مانند حالت قبل این خروجی سخت ترین خروجی برای مدل سازی است



از نتایج تست هم پیدا ست که مدلستزی به خوبی صورت نگرفته است

خروجی چهارم





نتیجه تست به روی داده های خروجی ۴. همچنان داده ها به خوبی مدل نشده اند. چون دینامیک هائی خروجی از حالت های قبل سخت تر هستند