

در این تمرین ابتدا توابع را پیدا سازی کردیم و درصد داده ها را با تست و ترین با سایز 30 و 70 درصد جداسازی کردیم.

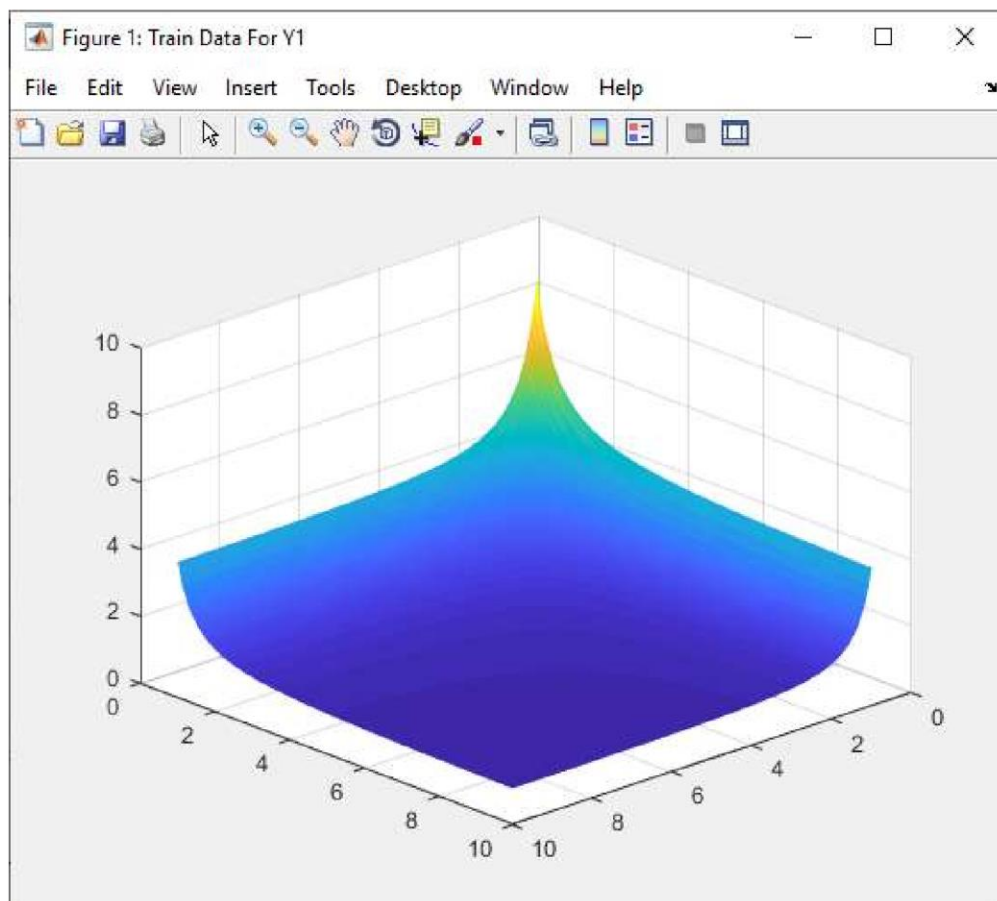
تابع y_1

```
function [x1_train,x2_train,y_train] = generateDatasetY1(P)
    data = linspace(1,10,P);

    x1_train = data;
    x2_train = data';

    y_train = (1+(x1_train.^-2) + (x2_train .^ -1.5)).^2;
end
```

خروجی تابع y_1



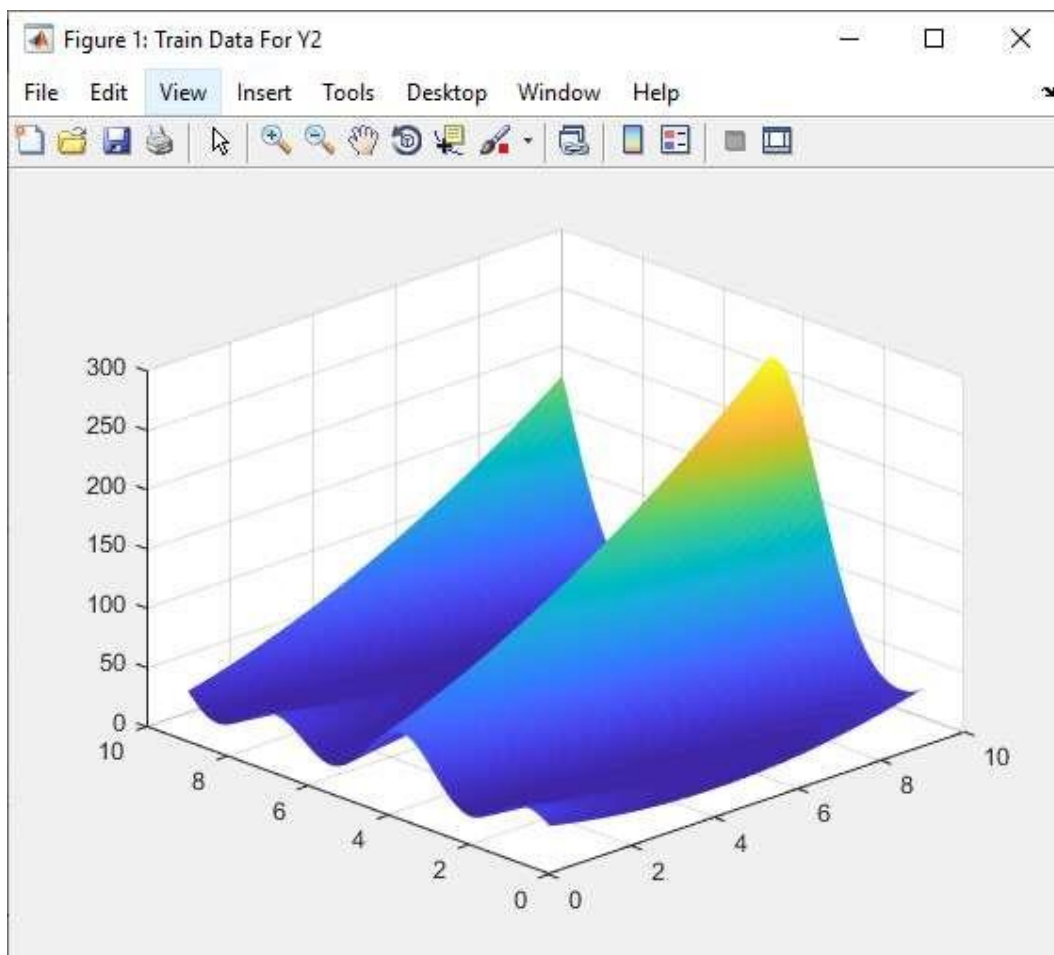
تابع y_2

```
function [x1_train,x2_train,y_train] = generateDatasetY2(P)
    data = linspace(1,10,P);

    x1_train = data;
    x2_train = data';

    y_train =(x1_train-6*sin(x2_train)).^2;
end
```

خروجی تابع y_2



در ادامه ماتریکس که ایجاد شده است از این توابع را به شکل یک بعدی تبدیل میکنیم تا راحت تر به شبکه بدهیم

```
i=1;
for p=1:train_size
    for p2=1:train_size
        x1(i)=x1_train(p);
        x2(i)=x2_train(p2);
        y(i)=y_train(p,p2);
        i=i+1;
    end
end

Inputs=[x1' x2'];
Targets=y';
```

در ادامه مدل FIS یا همان مدل مدانی را به کمک تابع CreateFisUsingLookupTable ایجاد میکنیم که در ورودی این تابع ابتدا مقدار های ترین x2 , x1 به همراه target با تعداد membership functions و نوع membership functions به تابع داده میشود.

```
%% Create FIS

nmf=[20 20 20];

mftype={'trimf','trimf','trimf'};

fis=CreateFisUsingLookupTable([Inputs Targets],nmf,mftype);
```

(در جلو تر در مورد این تابع توضیح داده میشود)
بعد ساخت مدل , داده های تست را هم یک بعدی میکنیم و به کمک تابع `evalfis` داده های پیش بینی شده (`y_hat` یا `y_pred`) را به کمک مدل ساخته شده , بدست میاوریم.

```
Inputs_test=[x1_testt' x2_testt'];  
Targets_test=y_testt';  
  
Outputs=evalfis(Inputs_test,fis);
```

بعد از پیش بینی داده ها , خروجی را به شکل ماتریس باز میگردانیم تا بتوانیم تحلیل و نمایشی از آن داشته باشیم .

```
i=1;  
for p=1:test_size  
    for p2=1:test_size  
        y_pred(p,p2)=Outputs(i);  
        i=i+1;  
    end  
end
```

بعد از این مرحله معیار های ارزیابی را بر روی خروجی بررسی میکنیم در ابتدا `average` مقدار `y` و `y_hat` بدست میاید بعد به توابع معیار داده میشود.

```

avrage_y = avrage(x1_test,x2_test,y_test);
avrage_yhat = avrage(x1_test,x2_test,y_pred);

disp("FVU = " + FVU(x1_test,x2_test,y_test,y_pred,avrage_yhat));
disp("CORR = " + CORR(x1_test,x2_test,y_test,y_pred,avrage_y,avrage_yhat));

```

تابع **avrage** که به کمک دو **for** و تقسیم بر سائز داده ها مقدار **avrage** داده ها را بدست می‌آورد

```


function [avrage] = avrage(x1,x2,y)
    total=0;
    size=0;

    for i=1:length(x1')
        for j=1:length(x2)
            total=total+y(j,i);
            size=size+1;
        end
    end

    avrage=total/size;
end

```

تابع **FVU** که در این تابع طبق فرمول داده شده در توضیحات تمرین عمل شده در ابتدا دو **for** برای دو **sigma** بدست آمده بعد مقادیر آنها تقسیم شده اند .



```
function [out] = FVU(x1,x2,y,yyhat,avrage)
    sigma1=0;
    sigma2=0;

    for i=1:length(x1')
        for j=1:length(x2)
            sigma1=sigma1 + (yyhat(j,i) - y(j,i))^2;
            sigma2=sigma2 + (y(j,i) - avrage)^2;
        end
    end

    out=sigma1/sigma2;
end
```

تابع **CORR** که در این تابع مثل روش FVU با دو for و محاسبه سه sigma و گرفتن رادیکال و تقسیم شان عمل میشود.

```
function [out] = CORR(x1,x2,y,yyhat,avrage_y,avrage_yhat)
    sigma1=0;
    sigma2=0;
    sigma3=0;

    for i=1:length(x1')
        for j=1:length(x2)
            sigma1=sigma1 + ((y(j,i)-avrage_y)*(yyhat(j,i)-avrage_yhat));
            sigma2=sigma2 + (y(j,i)-avrage_y)^2;
            sigma3=sigma3 + (yyhat(j,i)-avrage_yhat)^2;
        end
    end

    down_sigmas=sqrt(sigma2*sigma3);

    out=sigma1/down_sigmas;
end
```

توضیحات در مورد تابع ساخت FIS با عنوان CreateFisUsingLookupTable

در این تابع ابتدا membership function ها با مقادیر داده شده از ورودی تابع به کمک تابع CreateMembershipFunctions برای ورودی ها و خروجی شبکه ساخته می شوند. در این تابع رنج membership function ها با for نسبت به سایز کمترین و بیشترین و تعداد داده ها تنظیم میشود.


```

A=cell(nx,1);
for i=1:nx
    A{i}=CreateMembershipFunctions(x(:,i),nmf(i),mftype{i});
end

B=CreateMembershipFunctions(y,nmf(end),mftype{end});

```

در مرحله بعد ماتریکس Rules های مدل و متغیر تعریف و ایجاد میشود

```

nRules=numel(S);
for r=1:nRules
    S{r}=zeros(1,nmf(end));
end

```

و بعد از ساخت متغیر امتیاز های x_1 , x_2 و y بر اساس membership function های آن محاسبه میشود


```

for r=1:nRules
    xind=cell(1,nx);
    [xind{1:nx}]=ind2sub(SSize,r);
    xind=cell2mat(xind);
    XIND(r,:)=xind;
    for yind=1:nmf(end)
        bmf=B{yind,1};
        bparam=B{yind,2};
        s=zeros(1,P);
        for p=1:P
            s(p)=feval(bmf,y(p),bparam);
            for i=1:nx
                amf=A{i}{xind(i),1};
                aparam=A{i}{xind(i),2};
                s(p)=s(p)*feval(amf,x(p,i),aparam);
            end
        end
        S{r}(yind)=sum(s);
    end
end
end

```

این مرحله جز مرحله سنگین پیاده سازی از نظر کارایی آموزش شبکه هستش که ممکن است برای دیتا زیاد فرایند آموزش زیاد زمان ببرد. این قسمت از 4 حلقه تشکیل شده حلقه دیتاها , حلقه تعداد memberships function های ورودی , حلقه تعداد memberships function های خروجی و حلقه تعداد بعد ورودی , در داخلی ترین حلقه امتیاز دیتا با کل membership function بررسی میشود و یک امتیاز بدست آورده می شود و آن امتیاز در واقع همان Rules ها است که برای هر Rules به Rules دیگر متصل میشود.

در این مرحله هم یکسری Rules های زیادی ایجاد میشود که در مرحله بعد حذف میشود.

```

%%Delete Extra Rules
R=zeros(size(S));
Keep=true(size(S));
for r=1:nRules
    [Smax, R(r)]=max(S{r});
    if Smax==0
        Keep(r)=false;
    end
end
end

```

در مرحله آخر مدل mamdani را ایجاد میکنیم و Rules ها و membership function ها را که بالاتر ایجاد کرده بودیم در اینجا به مدل ست میکنیم.

```

fis=newfis('Lookup Table FIS','mamdani');

for i=1:nx
    fis=addvar(fis,'input',['x' num2str(i)],[min(x(:,i)) max(x(:,i))]);
    for j=1:nmf(i)
        fis=addmf(fis,'input',i,...
            ['A(' num2str(i) ',' num2str(j) ')'],...
            A{i}{j,1},A{i}{j,2});
    end
end

fis=addvar(fis,'output','y',[min(y) max(y)]);
for j=1:nmf(end)
    fis=addmf(fis,'output',1,['B' num2str(j)],B{j,1},B{j,2});
end

fis=addrule(fis,Rules);

```

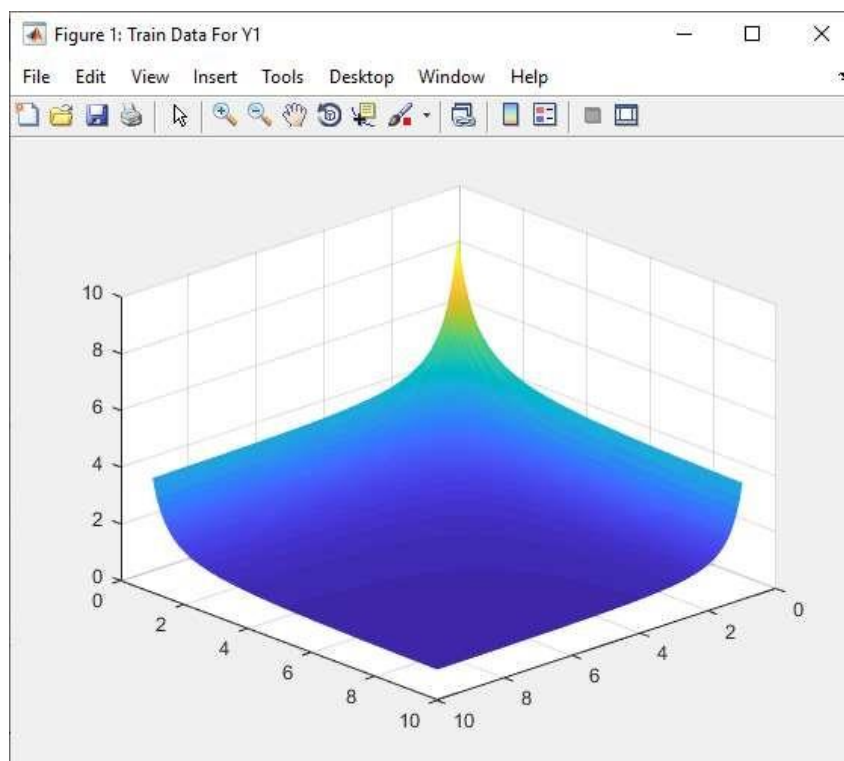
مرحله تست

در این مرحله مشکلی که با آن برخورد کردم در مرحله آموزش شبکه بود وقتی دیتا زیاد به شبکه داده میشد (2000 داده تمرین) بخاطر حلقه هایی که نیاز بود استفاده بشه در `for` دیتا ها که ماتریکس 2000×2000 به همراه `for` تعداد `memberships function` های ورودی که حدودا 250 تا بود با `for` تعداد `memberships function` های خروجی که 7 تا بود با در نظر گرفتن `for` بعد ورودی که 2 تا بود نیاز به زمان زیادی برای فرایند آموزش بود بخاطر همین موضوع دیتا ها را از 2000 تا به 100 تا تغییر دادم که البته از نظر آموزش شبکه در خروجی تغییری نداشت فقط تعداد دیتا کم شد و باعث شد سریعتر شبکه آموزش ببیند .

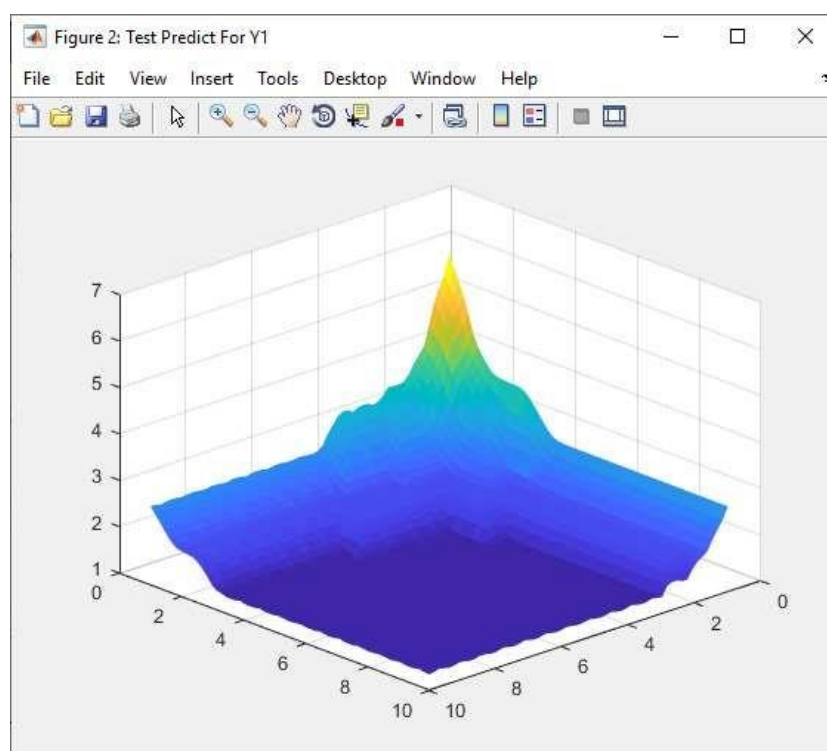
تابع `y1` با دیتای 100 و با در نظر گرفتن 70 داده آموزشی و 30 داده تستی `memberships function` ها به همراه تعداد مقادیر

خروجی	ورودی دوم	ورودی اول
<code>trimf</code>	<code>trimf</code>	<code>gaussmf</code>
10	15	15

خروجی اصلی مدل



خروجی پیش بینی شده



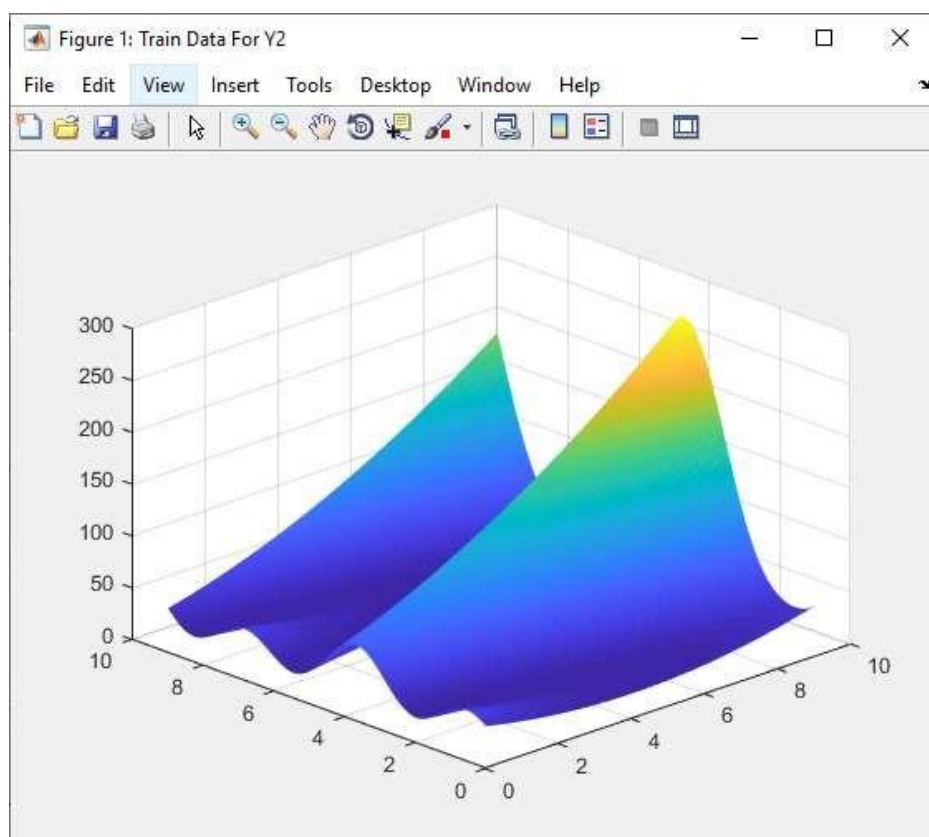
معیار های ارزیابی تابع y_1

FVU = 0.01701
CORR = 0.9871

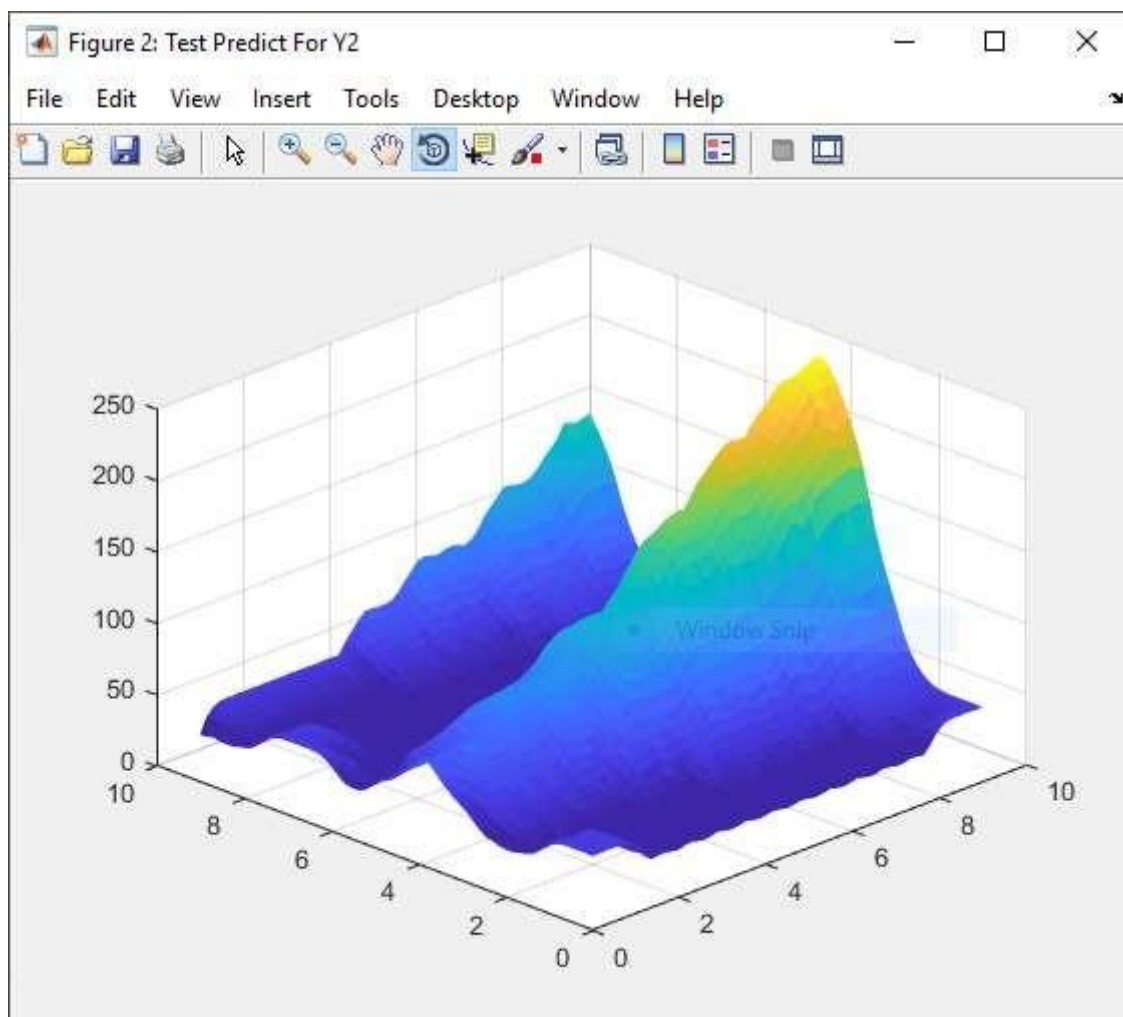
تابع y2 با دیتای 100 و با در نظر گرفتن 70 داده آموزشی و 30 داده تستی
memberships function ها به همراه تعداد مقادیر

ورودی اول	ورودی دوم	خروجی
gaussmf	trimf	gaussmf
15	15	10

خروجی اصلی مدل



خروجی پیش بینی شده



معیار های ارزیابی تابع y_2

$FVU = 0.04527$

$CORR = 0.99089$