به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش تمرین شماره ۳

شهاب مقدادی نیشابوری

۴۰۰۰۹۴۴۳

استاد : آقای دکتر مهدی علیاری

بهمن ۱۴۰۳

فهرست مطالب

لینک کد ها:

https://drive.google.com/drive/folders/1pyWC_Ck__C_sj94f4KmB7943yRI6327V?usp=sharing

بخش۱: سوالات تمرین

سوال اول

در این کد، مراحل مختلفی برای طراحی و شبیه‌سازی یک کنترل‌کننده PID برای یک سیستم با تاخیر انجام شده است. ابتدا باید اشاره کرد که در سیستم‌های بدون تاخیر، مانند سیستم اصلی امکان طراحی کنترل‌کننده PID با روش زیگلر-نیکولز وجود ندارد. دلیل این امر آن است که این سیستم‌ها فاز کافی برای رسیدن به ۱۸۰- درجه ندارند، که برای اعمال روش زیگلر-نیکولز ضروری است. با افزودن تاخیر به سیستم، فاز سیستم کاهش می‌یابد و شرایط لازم برای استفاده از این روش فراهم می‌شود.

در این بخش، سیستم اصلی و تاخیر تعریف شده‌اند. سیستم اصلی با تابع انتقال مشخص شده است که شامل یک ترم مشتق‌گیرنده و یک قطب است. سپس، یک تاخیر زمانی به سیستم اضافه شده است. این تاخیر به‌دلیل تأثیرات واقعی در سیستم‌های کنترل، مانند زمان پاسخ سنسورها یا محرک‌ها، در نظر گرفته می‌شود.

```
s = tf('s');
T = s / (s + 1);
theta = 3; %delay
```

از آن‌جایی که تاخیر زمانی به‌صورت مستقیم در محاسبات قابل استفاده نیست، از تقریب پادِ استفاده شده است. این تقریب، تاخیر زمانی را به یک تابع گویا تبدیل می‌کند که برای شبیه‌سازی و تحلیل سیستم مناسب‌تر است. در اینجا از تقریب پادِ مرتبه دوم استفاده شده است تا دقت بیشتری داشته باشد.

```
[num_pade, den_pade] = pade(theta, 2);
Pade_approx = tf(num_pade, den_pade);
T_delayed = T * Pade_approx;
```

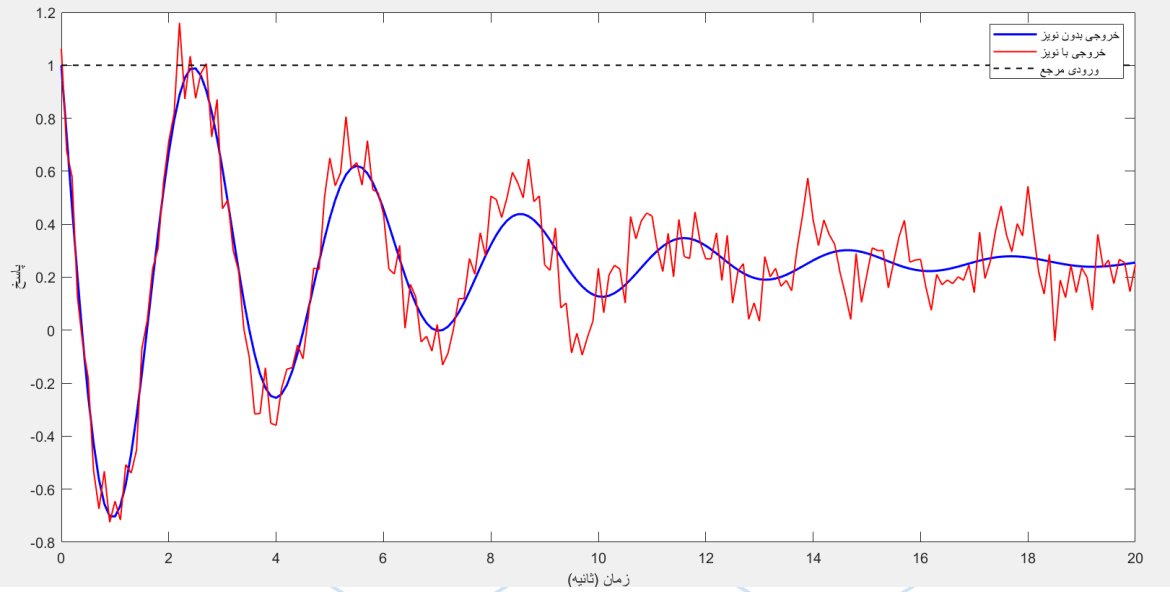
سپس، از تابع margin برای محاسبه Gain Margin و Phase Crossover Frequency استفاده شده است. این مقادیر برای تعیین پارامترهای کنترل‌کننده PID با استفاده از روش زیگلر-نیکولز ضروری هستند . Gain Marginنشان‌دهنده میزان بهره‌ای است که سیستم قبل از ناپایدار شدن می‌تواند تحمل کند، و Phase Crossover Frequencyفرکانسی است که در آن فاز سیستم به ۱۸۰- درجه می‌رسد.

با استفاده از مقادیر Ku وTu ، پارامترهای کنترل‌کننده PID محاسبه می‌شوند. این پارامترها شامل بهره تناسبی، زمان انتگرال‌گیرنده و زمان مشتق‌گیرنده هستند. این مقادیر با استفاده از فرمول‌های استاندارد روش زیگلر-نیکولز تعیین می‌شوند تا سیستم به‌طور بهینه کنترل شود.

```
[Gm, Pm, Wcg, Wcp] = margin(T_delayed);
% PID parameters
Kp = 0.6 * Ku;
Ti = Tu / 2;
Td = Tu / 8;
C = pid(Kp, Kp/Ti, Kp*Td);
```

در نهایت، پاسخ پله سیستم حلقه بسته شبیه‌سازی و رسم می‌شود. پاسخ پله نشان‌دهنده رفتار سیستم در برابر یک تغییر ناگهانی در ورودی است. این نمودار به ما کمک می‌کند تا عملکرد کنترل‌کننده PID را بررسی کنیم و مطمئن شویم که سیستم به‌طور پایدار و مطلوب پاسخ می‌دهد.

سپس در مرحله بعد، نویز سفید با دامنه ۰.۱ به سیستم اضافه کرده و رفتار آن را مشاهده میکنیم:



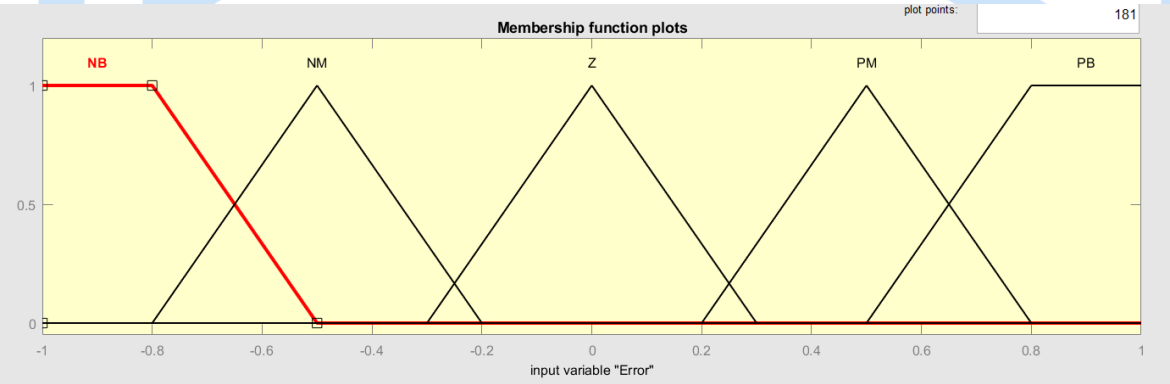
در مرحله بعد، یک کنترلر فازی طراحی میکنیم، کنترلر فازی طراحی‌شده بر اساس یک سیستم استنتاج ممدانی است که دارای دو ورودی و یک خروجی است. ورودی‌های این کنترلر خطای سیستم و تغییرات آن هستند که در بازه‌های معین تعریف شده‌اند. هر ورودی دارای پنج تابع عضویت است که به‌صورت توابع مثلثی و ذوزنقه‌ای توزیع شده‌اند. خروجی کنترلر نیز دارای پنج تابع عضویت است که میزان اعمال کنترل

را مشخص می‌کند. این توابع عضویت برای توصیف رفتار فازی سیستم تنظیم شده‌اند تا کنترلر بتواند در شرایط مختلف ورودی پاسخ مناسبی ارائه دهد.

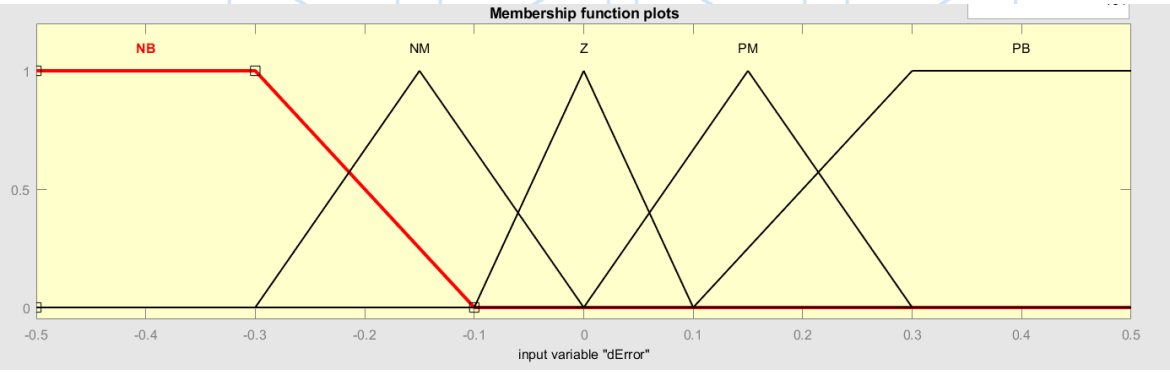
قوانین فازی کنترلر در قالب یک ماتریس تعریف شده‌اند که هر سطر آن نشان‌دهنده یک قاعده فازی است. این قواعد به‌صورت "اگر-آنگاه" تنظیم شده‌اند و ترکیب مقادیر خطا و تغییرات آن را به خروجی کنترلر مرتبط می‌سازند. هر قانون شامل یک درجه تأثیرگذاری نیز هست که مشخص می‌کند هر قاعده تا چه اندازه در تصمیم‌گیری نهایی مشارکت دارد. این سیستم استنتاج فازی بر اساس قوانین تعریف‌شده، مقدار مناسب سیگنال کنترلی را برای هر لحظه از زمان تولید می‌کند.

در فرآیند شبیه‌سازی، کنترلر فازی بر روی یک سیستم تأخیر زمانی اعمال شده است و خروجی آن با مقدار مرجع مقایسه می‌شود. کنترلر مقدار خطا و تغییرات آن را در هر لحظه محاسبه کرده و با استفاده از سیستم استنتاج فازی مقدار کنترل مناسب را تعیین می‌کند. پاسخ سیستم تحت کنترل فازی در مقایسه با سیگنال ورودی مرجع رسم شده است تا عملکرد آن ارزیابی شود. این شبیه‌سازی نشان می‌دهد که کنترلر فازی توانایی کاهش خطا و بهبود پایداری سیستم را دارد.

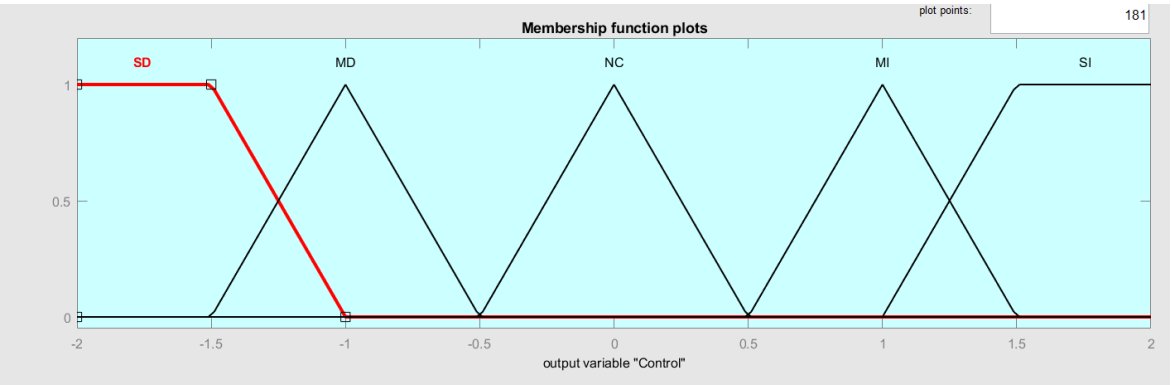
شکل توابع عضویت مدل فازی برای خطا:



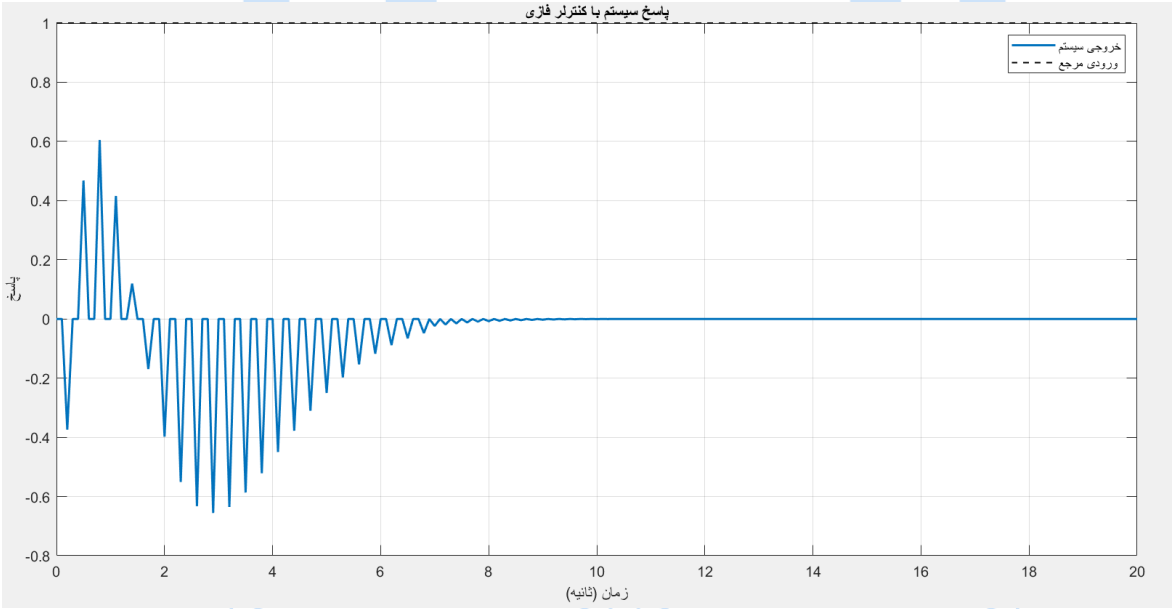
شکل توابع عضویت برای مشتق خطا:



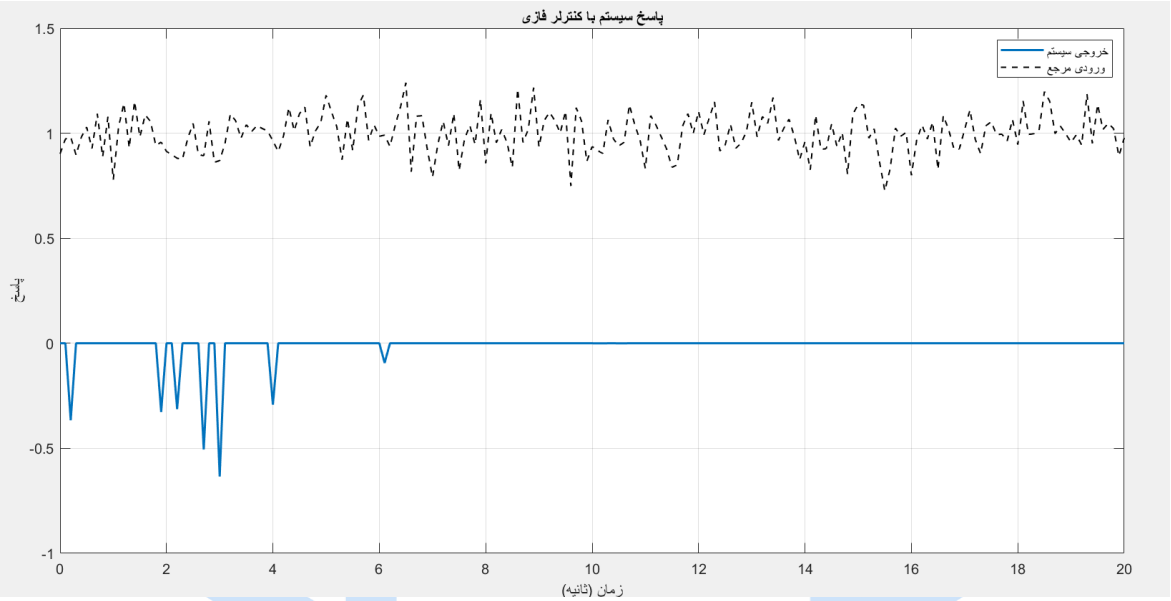
شکل توابع عضویت خروجی:



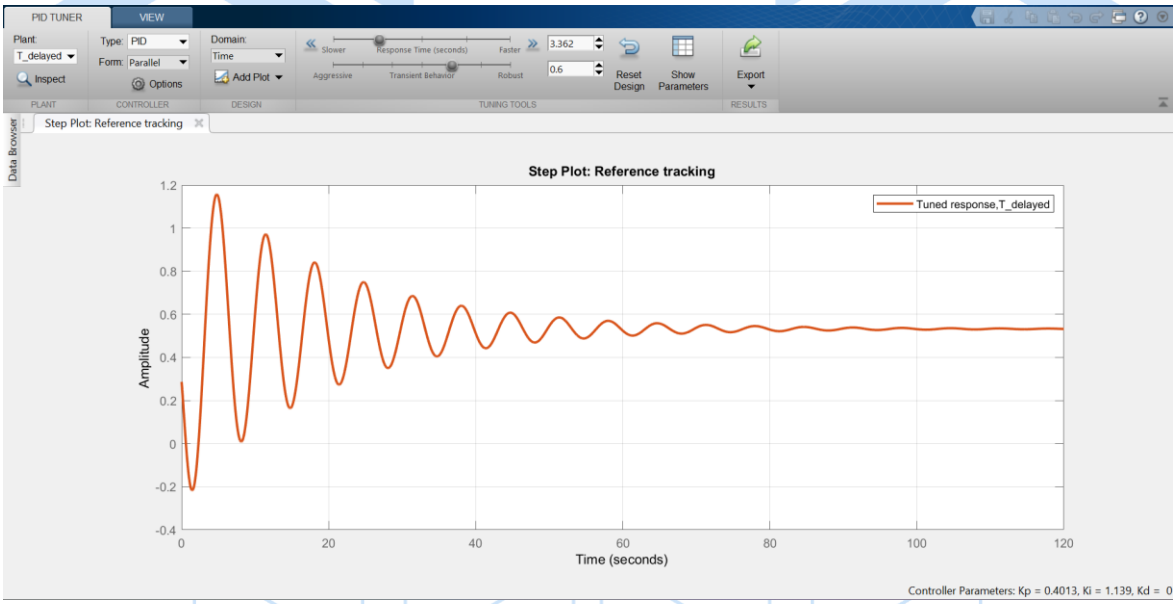
خروجی حلقه کنترلی با Fuzzy PID بدون حضور نویز سفید:



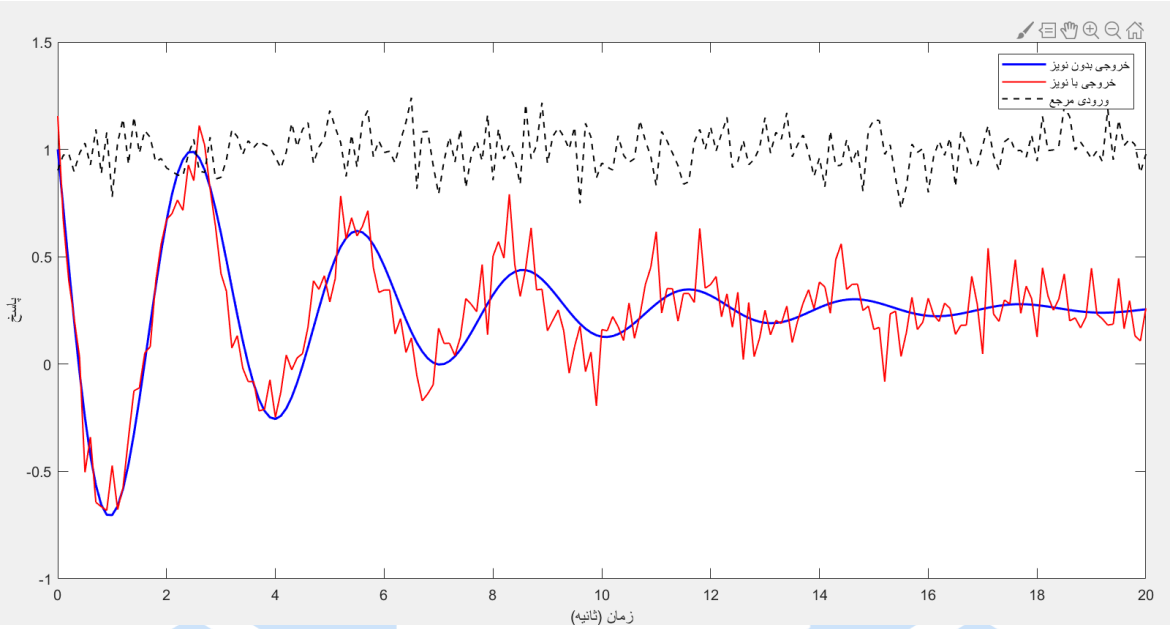
خروجی با حضور نویز سفید:



سپس با استفاده از PID Tuner خود متلب، برای این سیستم یک PID طراحی میکنیم:



سپس همانند قبل، سیستم را طراحی کرده و خروجی را با پله ثابت، و با پله همراه با نویز پلات میکنیم:



کنترلر Fuzzy-PIDدر مقایسه با PID Tunerمتلب و روش زیگلر-نیکولز عملکرد بهتری دارد زیرا قابلیت تنظیم تطبیقی دارد و می‌تواند غیرخطی بودن سیستم را بهتر مدیریت کند. در روش زیگلر-نیکولز، ضرایب PIDبر اساس یک رویکرد تجربی تنظیم می‌شوند که ممکن است برای تمام شرایط عملیاتی مناسب نباشد. همچنین PID Tunerمتلب از مدل‌های خطی برای تنظیم ضرایب استفاده می‌کند که در مواجهه با تغییرات غیرخطی یا نویز، کارایی کمتری دارد. در مقابل، کنترلر Fuzzy-PIDبا استفاده از منطق فازی قادر است به‌صورت پویا و هوشمندانه ضرایب کنترل را بر اساس شرایط لحظه‌ای سیستم تنظیم کند و در نتیجه، باعث بهبود پاسخ زمانی، کاهش نوسانات، و افزایش پایداری سیستم می‌شود.

سوال دوم

در این سوال خواسته شده تا با استفاده از یک سیستم فازی، یک ماشین در یک پارکینگ در مختصات x برابر با ۱۰ و زاویه ۰ درجه(به صورت صاف) پاک شود. برای حل سوال از دو مجموعه داده موجود در گیت هاب درس کمک گرفته شد و با کمک آنها شبکه فازی با کمک anfis ترین شدند.

توضیحات کد:

دو مجموعه داده شامل اطلاعات مربوط به موقعیت کامیون در طول زمان ایجاد شده است. هر سطر نشان‌دهنده یک مرحله از حرکت کامیون است. ستون اول شماره مرحله، ستون دوم مقدار x، ستون سوم مقدار زاویه و ستون چهارم خروجی سیستم(زاویه فرمان) را نشان می‌دهد.

داده‌های مربوط به ورودی و خروجی از دو مجموعه داده استخراج شده و در قالب inputDataو outputDataذخیره می‌شوند. داده‌های x1و x2به عنوان ورودی‌های مدل و yبه عنوان خروجی مورد استفاده قرار می‌گیرد.

```
x1 = [dataset_1(:,2); dataset_2(:,2)]
x2 = [dataset_1(:,3); dataset_2(:,3)]
y = [dataset_1(:,4); dataset_2(:,4)]
inputData = [x1 x2];
```

یک سیستم فازی اولیه با استفاده از روش *Grid Partitioning*ایجاد می‌شود. برای هر ورودی پنج تابع عضویت از نوع گاوسی در نظر گرفته شده است. محدوده خروجی سیستم به بازه‌ای از -۹۰ تا ۹۰ تغییر داده می‌شود.

```
opt = genfisOptions('GridPartition');
opt.NumMembershipFunctions = [5 5];
opt.InputMembershipFunctionType = ["gaussmf" "gaussmf"];
%opt.OutputRange = [min(outputData), max(outputData)];
fis = genfis(inputData,outputData, opt);
fis.output.range = [-90, 90];  % Change the output range to -90 to 90
```

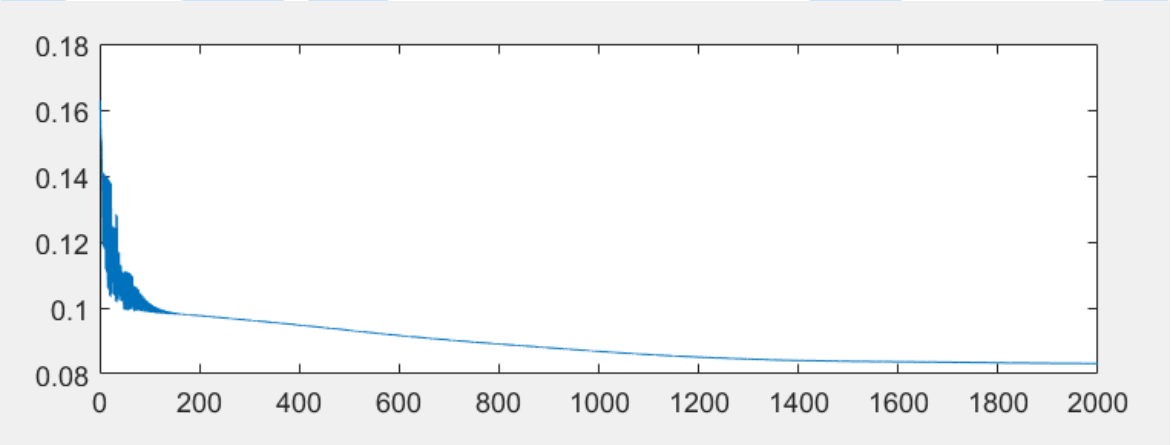
مدل ANFIS آموزش داده می‌شود. تعداد دوره‌های آموزش ۲۰۰۰ تعیین شده است و گام یادگیری اولیه برابر با ۰.۲ است که بر اساس چندین بار اجرای برنامه انتخاب شده تا هم سرعت شروع بهینه باشد و هم مقدار افزایش گام یادگیری کنترل شود.

```
trainingData = [inputData outputData];
options = anfisOptions('InitialFIS',fis);
options.EpochNumber = 2000;
options.InitialStepSize = 0.2;
[fis,trainError,stepSize] = anfis(trainingData,options);
```

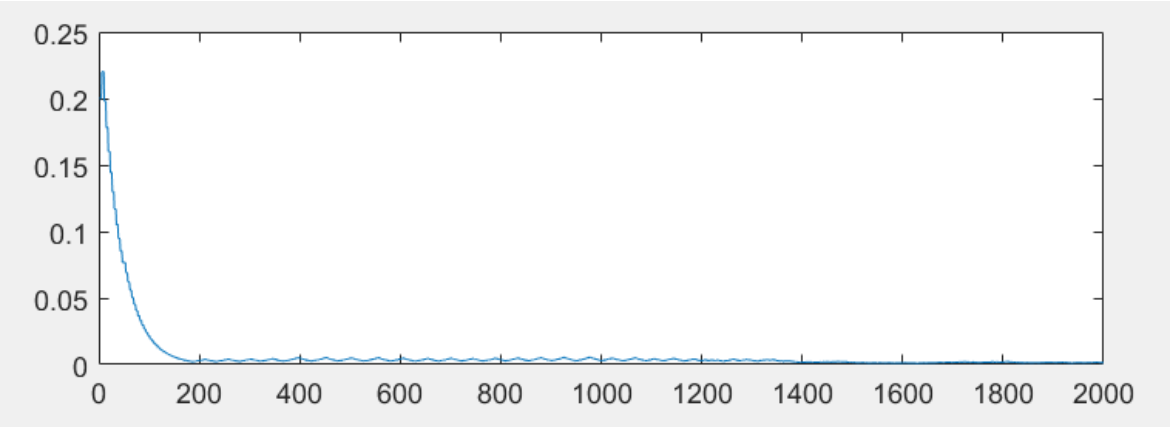
نمودار خطای آموزش و گام یادگیری در دو زیرنمودار نمایش داده می‌شود تا روند تغییرات مدل طی فرایند آموزش بررسی شود.

```
figure;
subplot(2, 1, 1);
plot(trainError);
subplot(2, 1, 2);
plot(stepSize);
```

نمودار خطا:



نمودار گام آموزش:



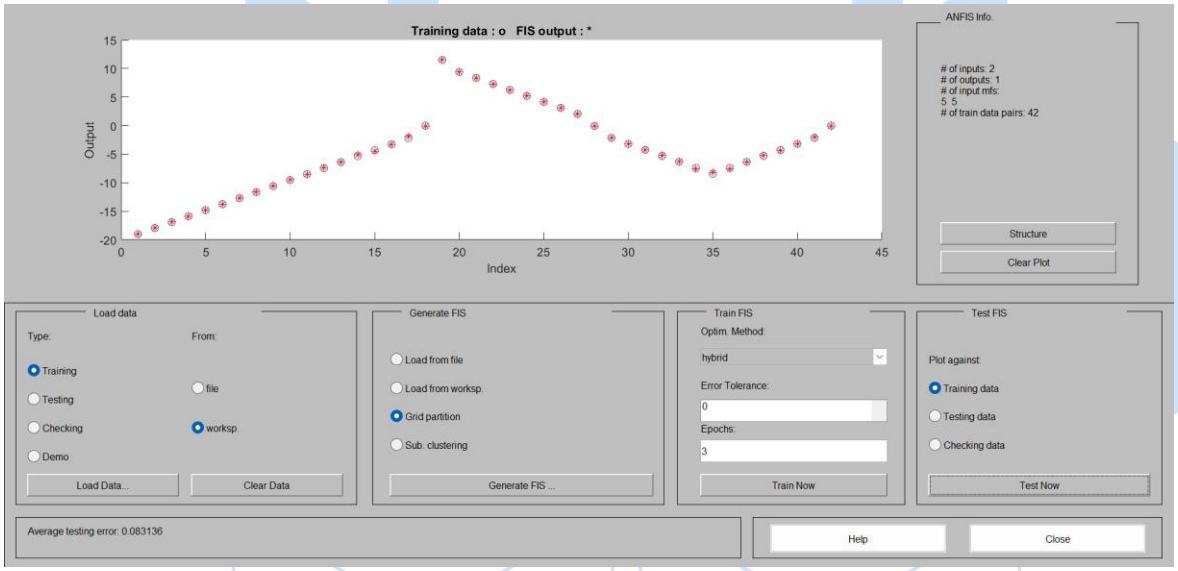
متغیرهای اولیه مربوط به موقعیت کامیون و زاویه آن مقداردهی می‌شوند. همچنین، آرایه‌هایی برای ذخیره مقادیر این متغیرها در طول شبیه‌سازی تعریف می‌شوند.

شبیه‌سازی حرکت کامیون انجام می‌شود. در هر مرحله، زاویه فرمان کامیون با استفاده از مدل ANFIS محاسبه می‌شود و سپس مقدار جدید موقعیت و زاویه کامیون تعیین می‌شود. این فرایند ادامه پیدا می‌کند تا زمانی که موقعیت کامیون به مقدار مشخصی برسد که نشان‌دهنده پایان فرایند پارک است. با مشاهده این بخش مشخص میشود که برنامه به صورت کامل حل شده و مساله در جاهایی که در آن اطلاعات موجود است حل شده است.

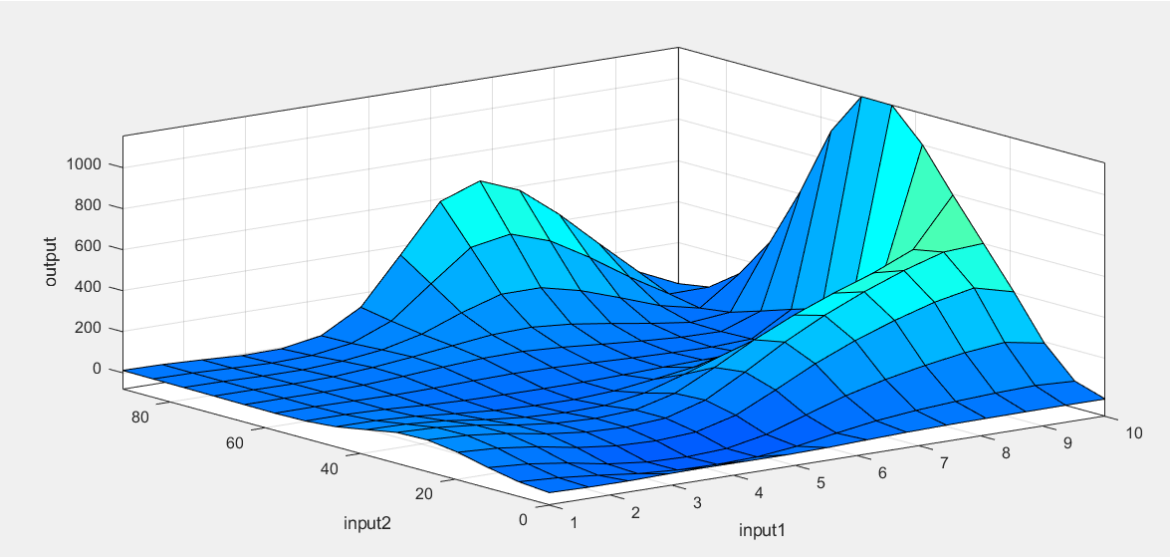
```
x = 1;
Phi = 0;
theta = 0;
y = 0;
xs = [];
Phis = [];
ys = [];
thetas = [];
xs(end+1) = x;
Phis(end+1) = Phi;
thetas(end+1) = theta;
```

```
for i = 1 : 500
    theta = evalfis(fis, [x Phi]);
    [x, Phi] = next_phase(x, Phi, theta);
    xs(end+1) = x;
    Phis(end+1) = Phi;
    thetas(end+1) = theta;
    if (x >= 9.5)
        break
    end
end
```

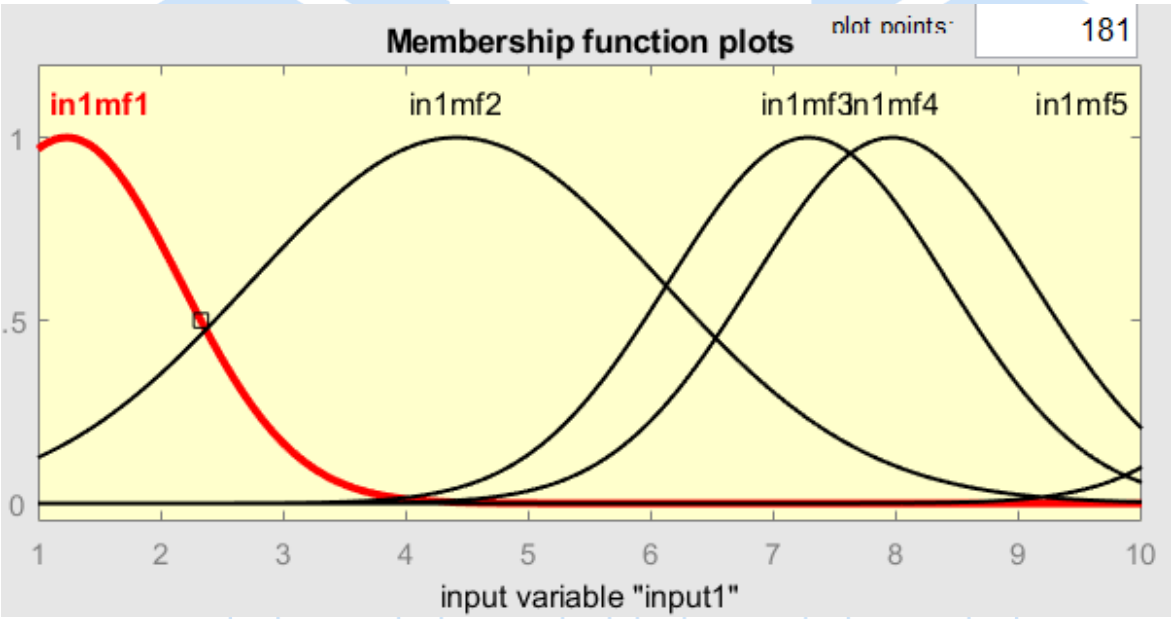
عملکرد مدل:



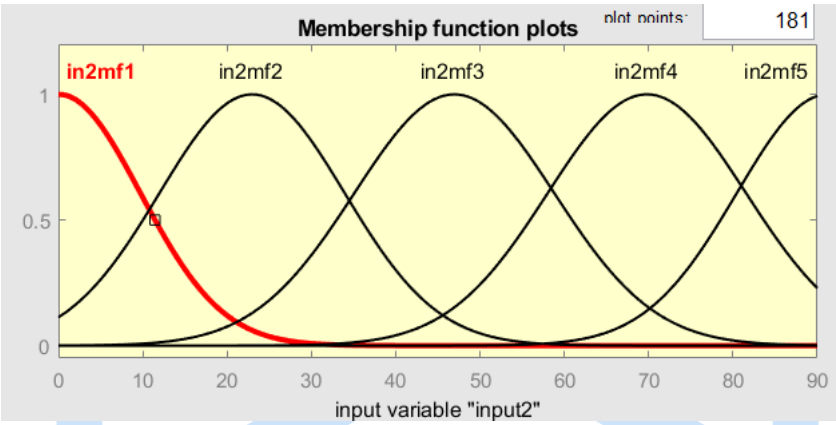
surface ورودی های موقعیت و زاویه کنونی نسبت به مبدا و خروجی زاویه فرمان:



توابع عضویت مربوط به موقعیت:



توابع عضویت مربوط به زاویه نسبت به مبدا:



سوال سوم

بخش اول:

در این سوال خواسته شده تا با مدل ANFIS، یک مدل ball and beam شناسایی شود. دیتای ما حاوی ۱۰۰۰ سطر و دو ستون میباشد. ستون اول ورودی است که برابر با زاویه میله، و ستون دوم خروجی و برابر با موقعیت توپ روی میله میباشد.

در این مرحله، داده‌های مربوط به سیستم از فایل ballbeam.datبارگذاری می‌شود. ستون اول x زاویه‌ی تیر و ستون دوم outputDataموقعیت توپ روی تیر است. این دو متغیر ورودی و خروجی مدل هستند.

```
data = load('ballbeam.dat')
x = data(:,1)
outputData = data(:,2);
```

در این قسمت، داده‌ها برای مدل آماده‌سازی می‌شوند. مقدار x0نشان‌دهنده زاویه تیر در لحظه فعلی، x1مقدار زاویه در یک لحظه قبل، و x2مقدار زاویه در دو لحظه قبل است. همچنین، x3مقدار قبلی موقعیت توپ را نشان می‌دهد. خروجی مدل، outputData، مقدار جدید موقعیت توپ است که باید پیش‌بینی شود. این داده‌ها به گونه‌ای انتخاب شده‌اند که مدل بتواند موقعیت توپ را بر اساس زاویه تیر و موقعیت‌های قبلی پیش‌بینی کند. در نهایت، ماتریس inputDataشامل سه مقدار قبلی زاویه و مقدار قبلی موقعیت توپ است که به عنوان ورودی‌های ANFIS استفاده خواهند شد.

```
x3 = outputData(2: end-1);
outputData = outputData(3: end);
x0 = x(3: end);
x1 = x(2: end-1);
x2 = x(1: end-2);
inputData = [x0 x1 x2 x3];
```

یک سیستم فازی اولیه با استفاده از روش *Grid Partitioning* ساخته می‌شود. برای هر یک از چهار ورودی، ۳ تابع عضویت در نظر گرفته شده است. تمامی توابع عضویت از نوع *گاوسی* (gaussmf)هستند. این سیستم فازی اولیه قرار است توسط ANFIS آموزش ببیند.

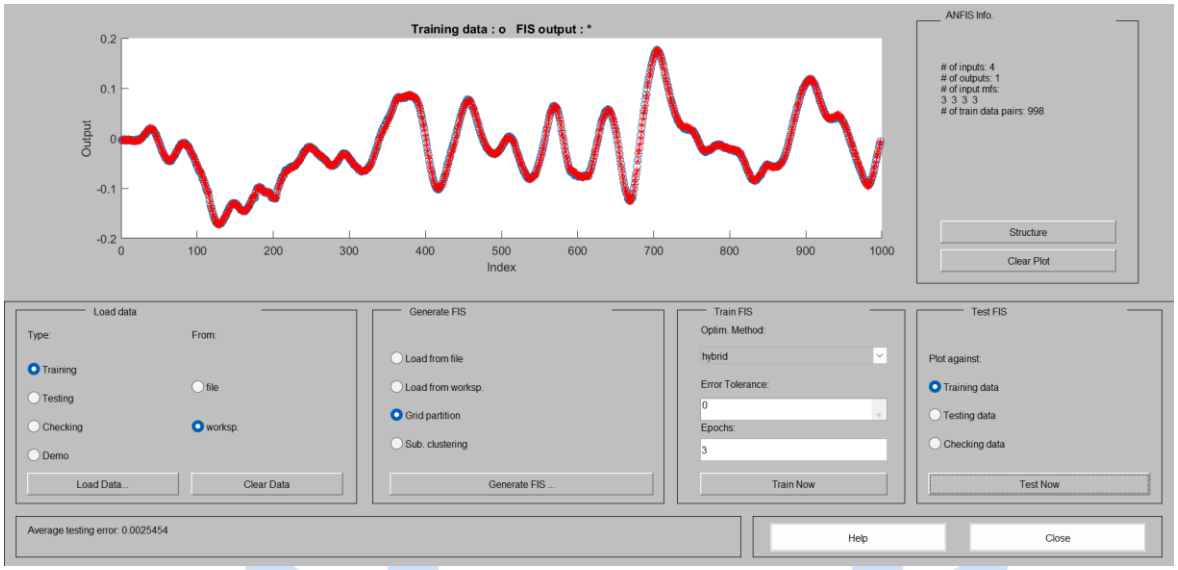
```
opt = genfisOptions('GridPartition');
opt.NumMembershipFunctions = [3 3 3 3];
opt.InputMembershipFunctionType = ["gaussmf" "gaussmf" "gaussmf" "gaussmf"];
fis = genfis(inputData,outputData, opt);
```

مدل ANFIS با استفاده از داده‌های آموزشی (trainingData) آموزش داده می‌شود InitialFIS . سیستم فازی اولیه است که در مرحله قبل ساخته شد EpochNumber .برابر ۲۰ تعیین شده است، به این معنی که مدل برای ۲۰ دوره تکرار (Epoch) آموزش می‌بیند. مقدار اولیه گام یادگیری (InitialStepSize)برابر ۰.۰۲ تنظیم شده است و مقدار کاهش گام یادگیری (StepSizeDecreaseRate)نصف مقدار پیش‌فرض در نظر گرفته شده است تا یادگیری پایدارتر شود. در نهایت، anfisمدل را آموزش داده و مقادیر fisمدل نهایی آموزش‌دیده، trainError(خطای آموزش)، و stepSize(گام یادگیری) را خروجی می‌دهد.

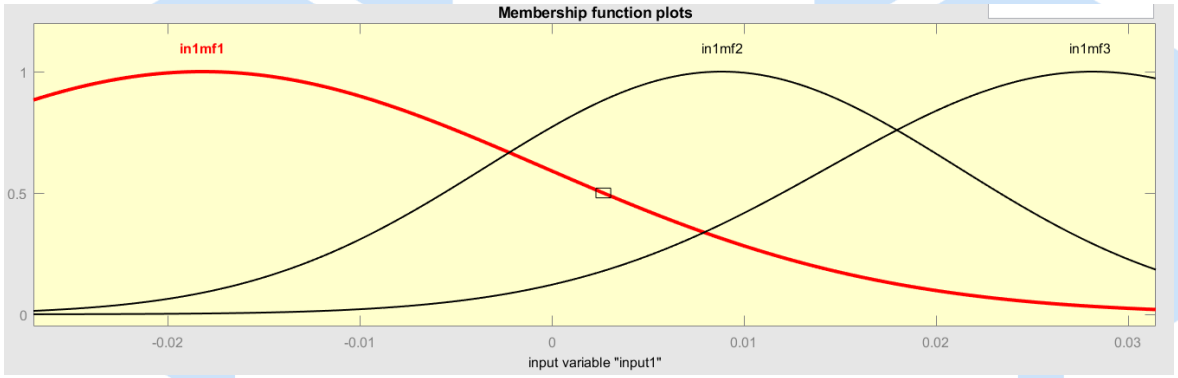
```
trainingData = [inputData outputData];
options = anfisOptions('InitialFIS',fis);
options.EpochNumber = 20;
options.InitialStepSize = 0.02;
options.StepSizeDecreaseRate = options.StepSizeDecreaseRate/2;
[fis,trainError,stepSize] = anfis(trainingData,options);
```

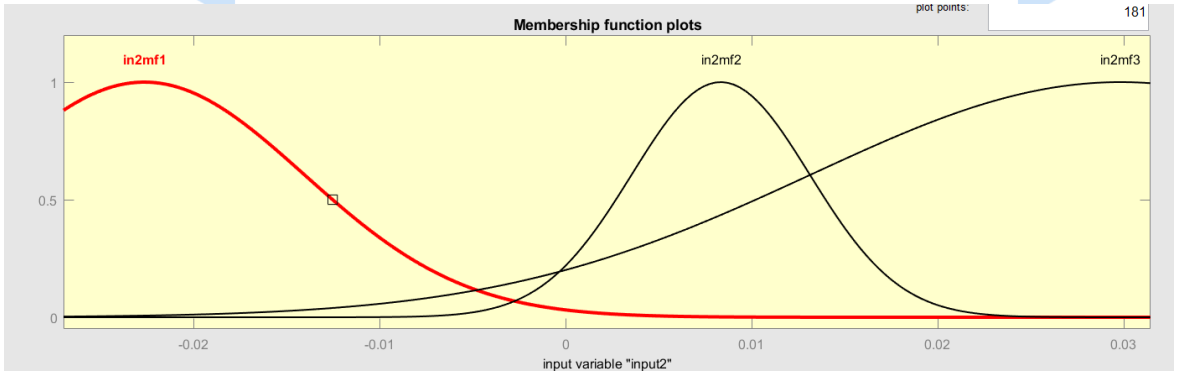
در بخش انتخاب گام یادگیری، مقادیر مربوط به گام‌های اولیه و نرخ کاهش گام به صورت تجربی و از چندین بار اجرای برنامه به دست آمده‌اند. گام یادگیری اولیه برابر ۰.۰۲ انتخاب شده است، که از آزمایشات قبلی به عنوان مقدار بهینه برای شروع فرایند یادگیری به دست آمده است. همچنین، نرخ کاهش گام به نصف مقدار پیش‌فرض تنظیم شده است تا سرعت یادگیری در طول آموزش کاهش یابد و به دقت بهتری دست پیدا کند. این مقادیر به گونه‌ای تنظیم شده‌اند که مدل به تدریج به بهترین نتایج ممکن دست یابد و از همگرایی سریع و نوسانات زیاد جلوگیری شود.

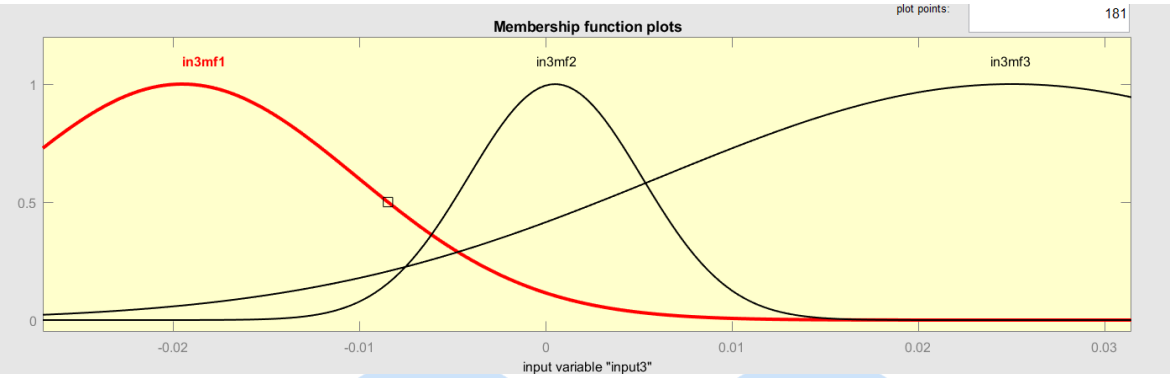
شکل خروجی:

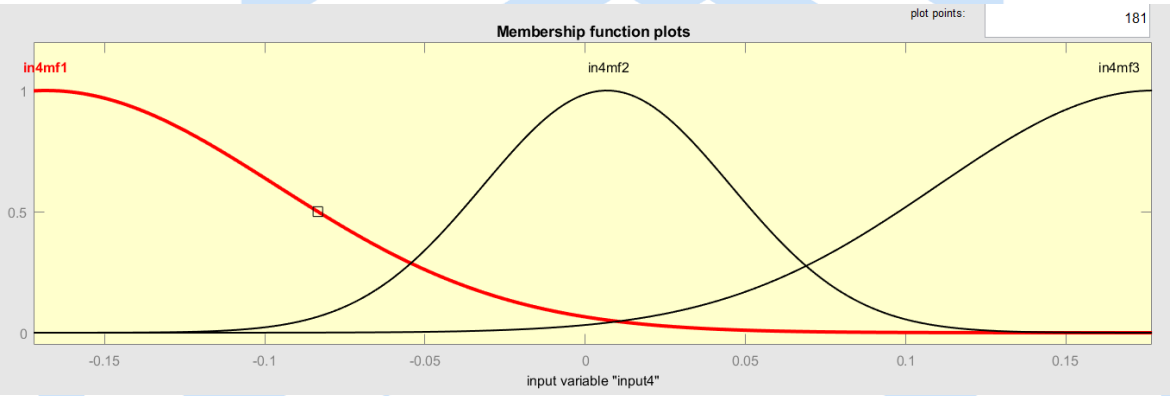


شکل توابع تعلق:









بخش امتیازی:

در مرحله اول مجموعه داده steamgen.dat بارگذاری می‌شود. سپس داده‌ها به صورت ستونی نرمال‌سازی می‌شوند. این کار به این صورت انجام می‌شود که از هر مقدار، میانگین ستون مربوطه کم شده و سپس بر انحراف معیار همان ستون تقسیم می‌شود. این نرمال‌سازی باعث می‌شود که مقیاس تمامی ویژگی‌ها یکسان شود و تأثیر مقادیر بزرگ‌تر بر مدل کاهش یابد.

در مرحله بعد ویژگی‌های ورودی و خروجی از داده‌ها استخراج می‌شوند. متغیر x1 شامل داده‌های سطرهای دوم تا انتها و ستون‌های دوم تا چهار ستون قبل از آخر است. متغیر x2 نیز شامل داده‌های یک سطر قبل از x1 است. سپس با ترکیب این دو متغیر، ماتریس x تشکیل می‌شود که به عنوان ورودی مدل استفاده خواهد شد. خروجی‌های مدل نیز در چهار بردار y1 تا y4 ذخیره می‌شوند که هرکدام شامل داده‌های مربوط به یکی از چهار ستون پایانی مجموعه داده اصلی هستند.

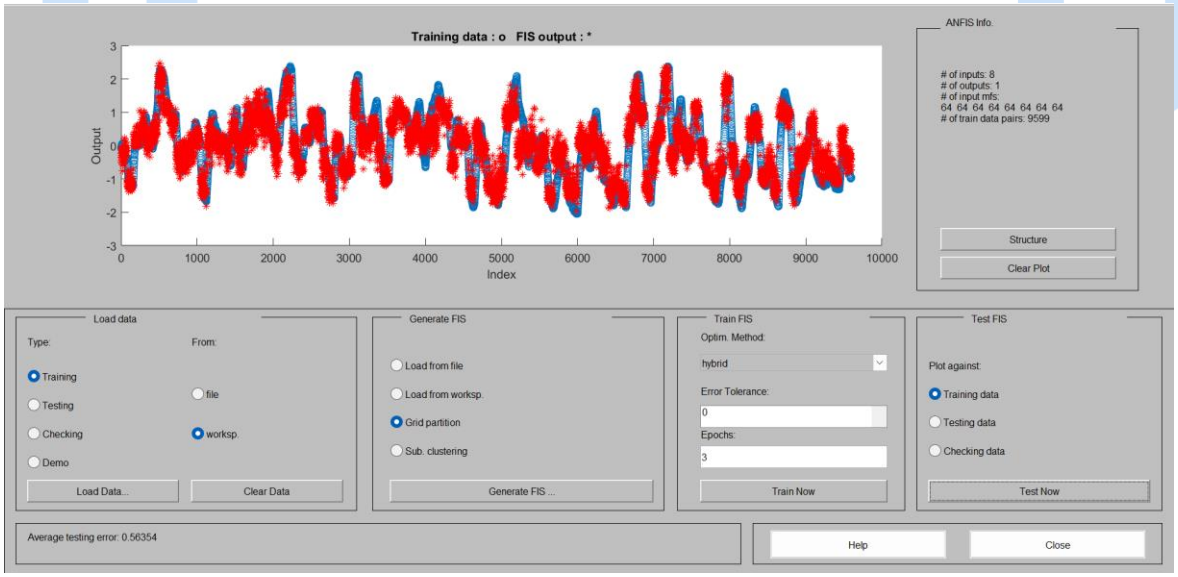
برای ایجاد یک سیستم استنتاج فازی اولیه از روش خوشه‌بندی تفاضلی استفاده می‌شود. ابتدا گزینه‌های مربوط به این روش با استفاده از genfisOptions تنظیم می‌شود. سپس چهار مدل فازی اولیه، یکی برای هر خروجی، با استفاده از genfis ایجاد می‌شوند. این مدل‌ها بر اساس داده‌های ورودی x و خروجی‌های y1 تا y4 ساخته می‌شوند و در واقع نقطه شروعی برای آموزش ANFIS هستند.

سپس مجموعه داده‌های آموزشی ساخته می‌شوند. برای این کار، هر خروجی به همراه ورودی‌های مربوطه در یک ماتریس قرار می‌گیرد. این داده‌ها بعداً برای آموزش مدل‌های ANFIS استفاده خواهند شد.

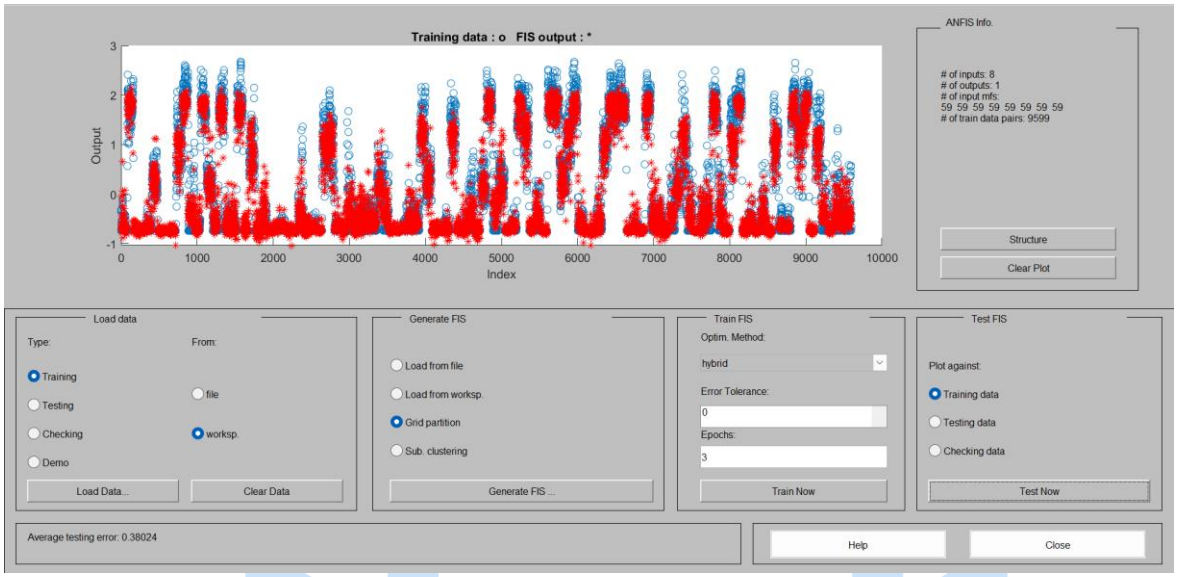
در ادامه گزینه‌های مربوط به آموزش ANFIS تنظیم می‌شوند. در این بخش مشخص می‌شود که مدل اولیه همان مدل فازی ایجادشده در مرحله قبل باشد. همچنین تعداد تکرارهای آموزش برابر ۱۰ در نظر گرفته می‌شود. این مقدار تعیین می‌کند که الگوریتم چند بار روی مجموعه داده اجرا شود تا پارامترهای مدل بهینه شوند.

در نهایت مدل‌های ANFIS برای هر چهار خروجی آموزش داده می‌شوند. برای این کار داده‌های آموزشی مربوط به هر خروجی به همراه گزینه‌های از پیش تعیین‌شده به تابع anfis داده می‌شود. خروجی این فرآیند شامل مدل نهایی پس از آموزش، مقدار خطای آموزشی در هر مرحله و مقادیر مربوط به اندازه گام یادگیری است. پس از این مرحله مدل‌ها آماده‌اند تا برای پیش‌بینی خروجی‌های جدید مورد استفاده قرار بگیرند. لازم به ذکر است که علت جداکردن خروجی ها این است که anfis قادر به شناسایی تنها یک خروجی میباشد.

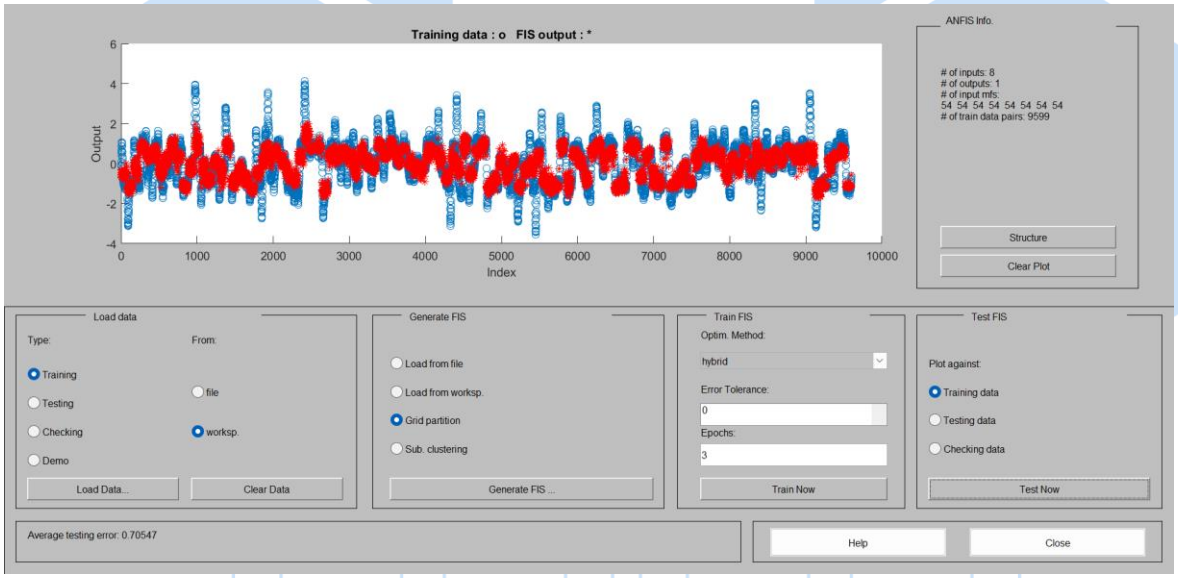
شکل خروجی برای خروجی اول:



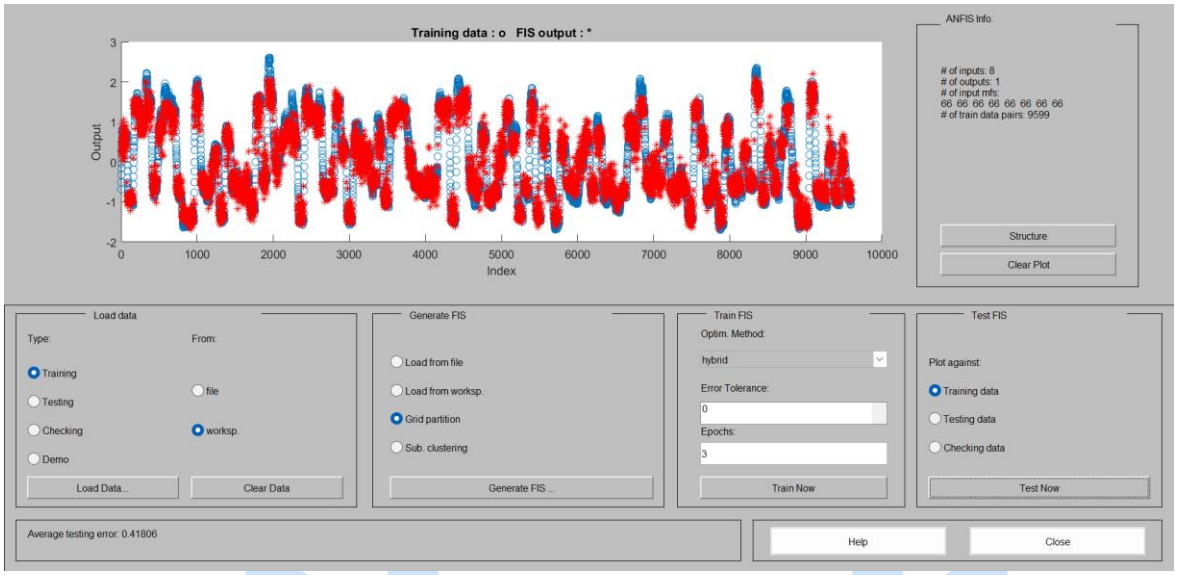
شکل خروجی برای متغیر دوم:



شکل خروجی برای متغیر سوم:



شکل خروجی برای متغیر چهارم:



لازم به ذکر است که به علت زیاد بودن نمونه ها و همچنین زیاد بودن متغیر های ورودی، امکان افزایش تعداد ایپاک به علت در دسترس نبودن سخت افزار لازم نبود، به همین تقریب ایده آلی برای متغیر های خروجی به دست نیامد، اما تقریب به دست آمده نیز از دقت قابل قبولی برخوردار است.

سوال چهارم:

در ابتدا، چندین پارامتر مهم برای تنظیم مدل و فرآیند آموزش تعریف می‌شود. تعداد قوانین فازی ۹ انتخاب شده است و تعداد نقاط آموزشی به ۲۵۰ تعریف شده است. تعداد کل داده‌ها ۷۰۰ است و نرخ یادگیری ۰.۰۵ تنظیم شده است. همچنین آرایه‌هایی برای ذخیره مقادیر مختلف مانند وضعیت‌ها (state_values)، خروجی‌ها(output_values) ، مقادیر گسترش(spread_values) ، ورودی‌ها (input_signal)، و خروجی‌های مدل (model_output) تعریف می‌شوند.

```
num_rules = 9;
num_training_points = 250;
total_points = 700;
learning_rate = 0.05;
```

سپس، مقادیر اولیه برای آرایه‌ها و متغیرهای استفاده‌شده در مدل تخصیص داده می‌شود. ورودی‌ها، خروجی‌های واقعی سیستم و مقادیر فازی ابتدا با مقادیر تصادفی و با استفاده از فرمول‌های سینوسی برای شبیه‌سازی عملکرد سیستم تعیین می‌شوند. این مقادیر به‌صورت اولیه برای فرآیند آموزش و تست استفاده می‌شوند.

```
state_values = zeros(num_training_points, num_rules);
output_values = zeros(num_training_points, num_rules);
spread_values = zeros(num_training_points, num_rules);
true_output = zeros(total_points, 1);
input_signal = zeros(total_points, 1);
state_input = zeros(total_points, 1);
predicted_output = zeros(total_points, 1);
model_output = zeros(total_points, 1);
z_values = zeros(total_points, 1);
rule_activations = zeros(total_points, 1);
```

در مرحله بعد، مقادیر اولیه برای وضعیت‌های مختلف (state_values) و خروجی‌های مدل (output_values)تعیین می‌شود. این مقادیر به‌صورت یکنواخت از -۱ تا ۱ در نظر گرفته می‌شوند تا تمام قوانین در دامنه ورودی به‌درستی پوشش داده شوند. همچنین مقادیر گسترش (spread_values) برای هر قانون به‌صورت یکنواخت محاسبه می‌شود.

در این بخش، مدل برای ۲۵۰ ایپاک آموزش داده می‌شود. برای هر ایپاک، ورودی جدید به‌طور تصادفی تولید می‌شود و مقادیر خروجی مدل محاسبه می‌شود. سپس مقادیر قوانین فازی با استفاده از وزن‌ها به‌روزرسانی می‌شوند. این به‌روزرسانی‌ها برای هر قانون به‌طور جداگانه انجام شده و در آرایه‌های state_values, output_values, و spread_values ذخیره می‌شوند. پیش‌بینی مدل بر اساس ترکیب قوانین فازی برای هر ورودی به‌دست می‌آید.

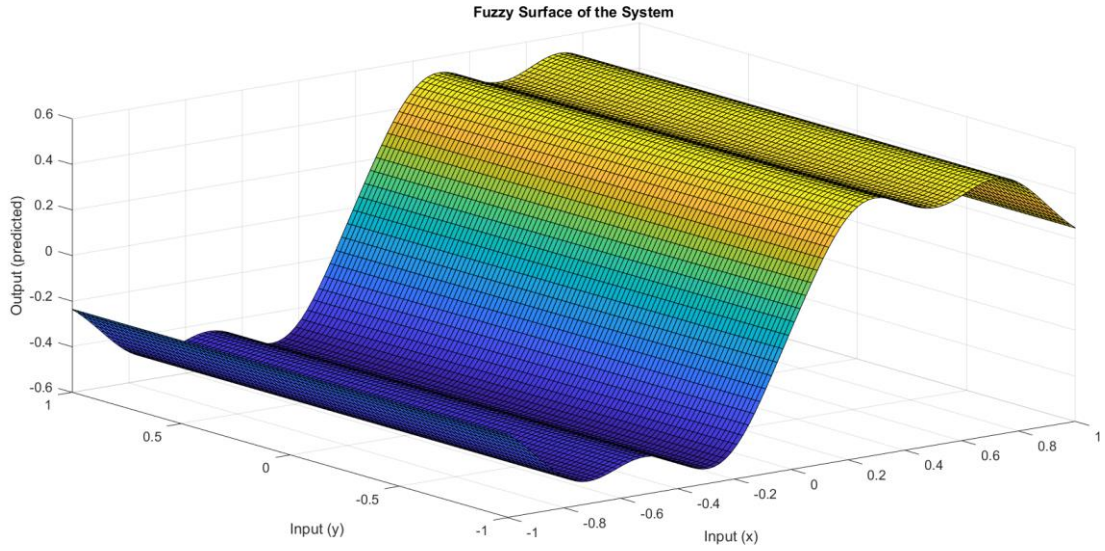
در ادامه، پیش‌بینی مدل برای هر ورودی جدید محاسبه می‌شود. ابتدا فعال‌سازی قوانین فازی با استفاده از ورودی‌ها و مقادیر گسترش محاسبه می‌شود. سپس به‌وسیله میانگین وزنی خروجی‌ها، پیش‌بینی مدل به‌دست می‌آید. این پیش‌بینی‌ها در آرایه predicted_output ذخیره می‌شوند و برای مقایسه با مقادیر واقعی استفاده می‌شوند.

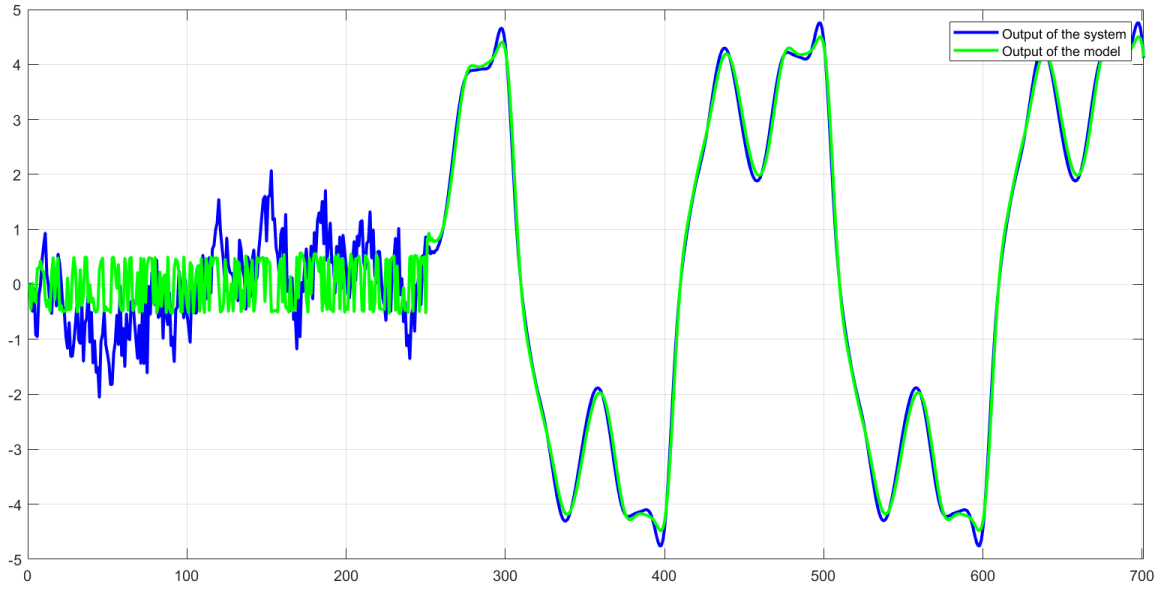
در مرحله تست، مدل با داده‌های جدید آزمایش می‌شود. برای هر ورودی جدید که از ایپاک ۲۵۰ به بعد تولید می‌شود، پیش‌بینی مدل محاسبه می‌شود. همانند مرحله آموزش، برای هر ورودی جدید، فعال‌سازی قوانین فازی محاسبه شده و سپس پیش‌بینی مدل به‌دست می‌آید. این پیش‌بینی‌ها برای مقایسه با خروجی واقعی سیستم استفاده می‌شوند.

در این قسمت، خطای مدل با استفاده از معیار RMSE (ریشه میانگین مربعات خطا) محاسبه می‌شود. این معیار برای ارزیابی دقت مدل در پیش‌بینی خروجی‌ها نسبت به مقادیر واقعی استفاده می‌شود. مقدار این خطا برای داده‌های تست محاسبه شده و در کنسول نمایش داده می‌شود.

در این بخش، دو نمودار رسم می‌شود. اولین نمودار نشان‌دهنده مقایسه بین خروجی واقعی سیستم و خروجی مدل در طول زمان است. این نمودار با استفاده از رنگ آبی برای خروجی واقعی و رنگ سبز برای خروجی مدل ترسیم می‌شود. در دومین نمودار، یک سطح سه‌بعدی از نتایج مدل بر اساس ورودی‌ها رسم می‌شود. این نمودار نشان‌دهنده سطح فازی سیستم است و به‌عنوان نمایشی از رابطه بین ورودی‌ها و خروجی مدل استفاده می‌شود.

```
RMSE for predicted data: 0.1465
```


Fuzzy Surface of the System
Output (predicted)
Input (y)
Input (x)


Output of the system
Output of the model

سوال پنجم:

برای این سوال از ما خواسته شده تا یک تسک regression را با دو متد ANFIS و RBF بر روی یک دیتاست آزمایش کرده، و بهترین مدل به همراه علت بهتر بودن آن را ارایه دهیم.

در ابتدا، داده‌ها از فایل اکسل خوانده می‌شوند و به یک آرایه تبدیل می‌شوند. سپس دو ستون اول که حاوی اطلاعات غیرضروری هستند حذف می‌شوند. در ادامه، داده‌هایی که دارای مقدار نامعتبر منفی دویست هستند شناسایی و حذف می‌شوند تا کیفیت داده‌های ورودی بهبود یابد. پس از این مرحله، برای نرمال‌سازی داده‌ها، میانگین هر ستون از مقدار هر نمونه کم شده و بر انحراف معیار آن ستون تقسیم می‌شود. این کار باعث می‌شود که تمامی ویژگی‌ها مقیاس‌بندی یکنواختی داشته باشند و مدل‌های یادگیری عملکرد بهتری ارائه دهند.

سپس مجموعه داده به سه بخش تقسیم می‌شود. شصت درصد داده‌ها برای آموزش، بیست درصد برای تست و بیست درصد باقی‌مانده برای اعتبارسنجی در نظر گرفته می‌شوند. برای اطمینان از تقسیم‌بندی مناسب، ابتدا تعداد نمونه‌های هر بخش بر اساس تعداد کل نمونه‌ها تعیین شده و سپس از طریق تولید اعداد تصادفی و انتخاب شاخص‌ها، داده‌های تست و اعتبارسنجی به صورت تصادفی از مجموعه اصلی جدا می‌شوند. در نهایت، داده‌های باقی‌مانده به عنوان داده‌های آموزشی اختصاص داده می‌شوند.

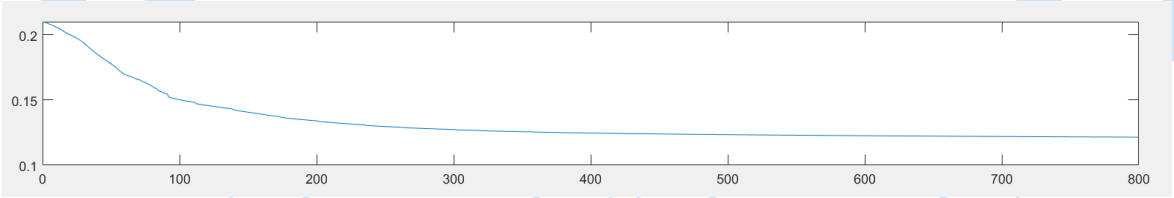
در این بخش، متغیر خروجی که در ستون هشتم داده‌های آموزشی قرار دارد جدا می‌شود و باقی‌مانده داده‌ها به عنوان ورودی استفاده می‌شوند.

در این قسمت، سیستم فازی اولیه بر اساس خوشه‌بندی کاهشی تولید می‌شود. از genfisOptions برای تنظیم پارامترهای تولید سیستم فازی استفاده شده و سپس genfis با داده‌های ورودی و خروجی اجرا می‌شود تا یک سیستم فازی اولیه ساخته شود. خوشه‌بندی کاهشی باعث می‌شود که تعداد قوانین فازی بهینه باشد و از پیچیدگی بیش‌ازحد مدل جلوگیری شود.

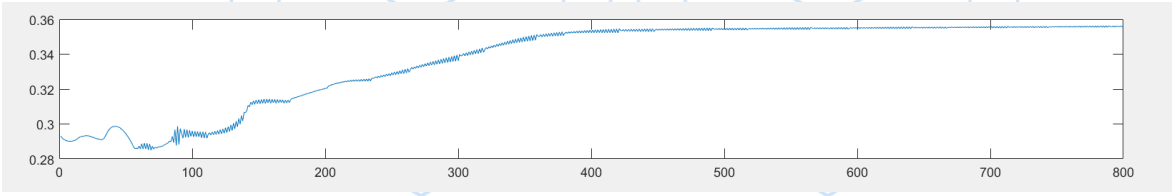
در این بخش پس از تولید سیستم فازی، داده‌های تست از مجموعه تستی استخراج شده و متغیر خروجی آن جدا می‌شود. سپس مجموعه تستی جدیدی ایجاد می‌شود که شامل ورودی‌های تست و خروجی واقعی آن‌ها است. در ادامه، تنظیمات مربوط به ANFIS انجام می‌شود. تعداد تکرارهای آموزش هشتصد در نظر گرفته شده و نرخ اولیه گام برابر با دو صدم تنظیم شده است. همچنین نرخ کاهش گام در هر مرحله نصف مقدار پیش‌فرض آن انتخاب شده تا همگام‌سازی مدل به شکل بهینه‌تری انجام شود. در نهایت، داده‌های تست به عنوان مجموعه اعتبارسنجی به anfisOptions اضافه شده و مدل ANFIS آموزش داده می‌شود. داده‌های تست که اضافه شدند به این دلیل بود که بهترین مدل روی داده‌های آموزش، بهترین خروجی را روی داده‌های تست نداشت بنابرین مجبور به اضافه کردن این بخش شدم.

پس از آموزش، نمودارهای مربوط به خطای آموزش، خطای اعتبارسنجی و مقدار گام یادگیری رسم می‌شوند. در نمودار اول خطای آموزشی در طول تکرارهای مختلف نمایش داده شده است. نمودار دوم روند تغییرات خطای اعتبارسنجی را نشان می‌دهد که از آن می‌توان برای بررسی احتمال بیش‌برازش استفاده کرد. در نمودار سوم نیز مقدار گام یادگیری در طول فرآیند آموزش به نمایش درآمده است. این نمودار نشان می‌دهد که چگونه اندازه گام در طول آموزش تغییر می‌کند و کاهش تدریجی آن تا رسیدن به مقدار بهینه انجام شده است.

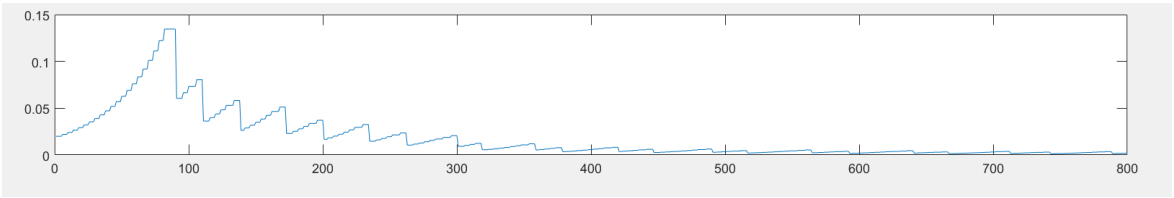
نمودار خطای آموزش:



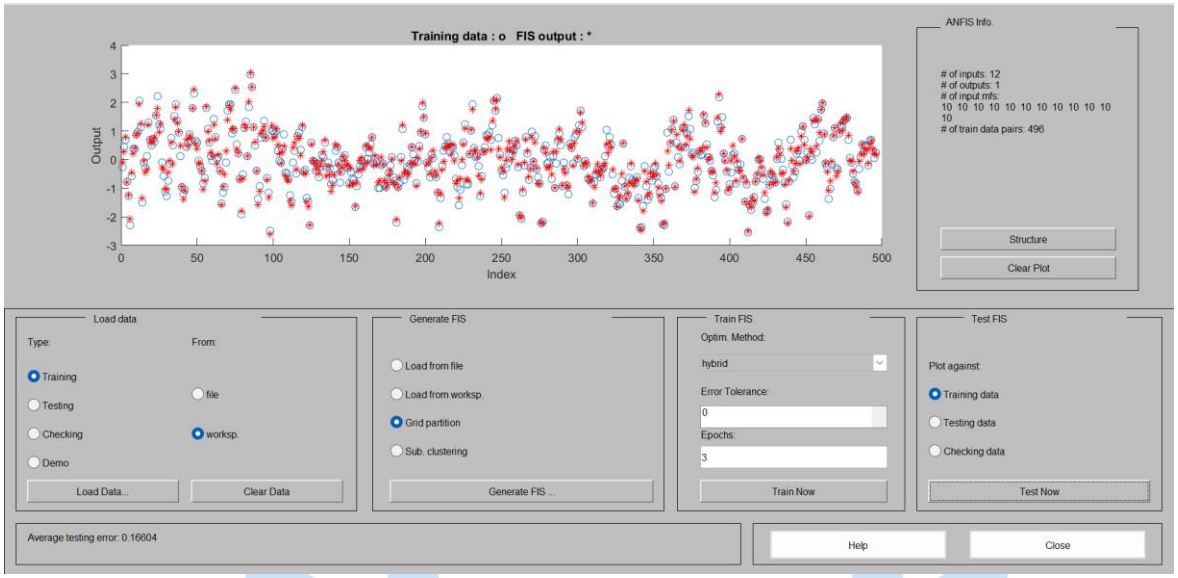
نمودار خطای داده های تست:



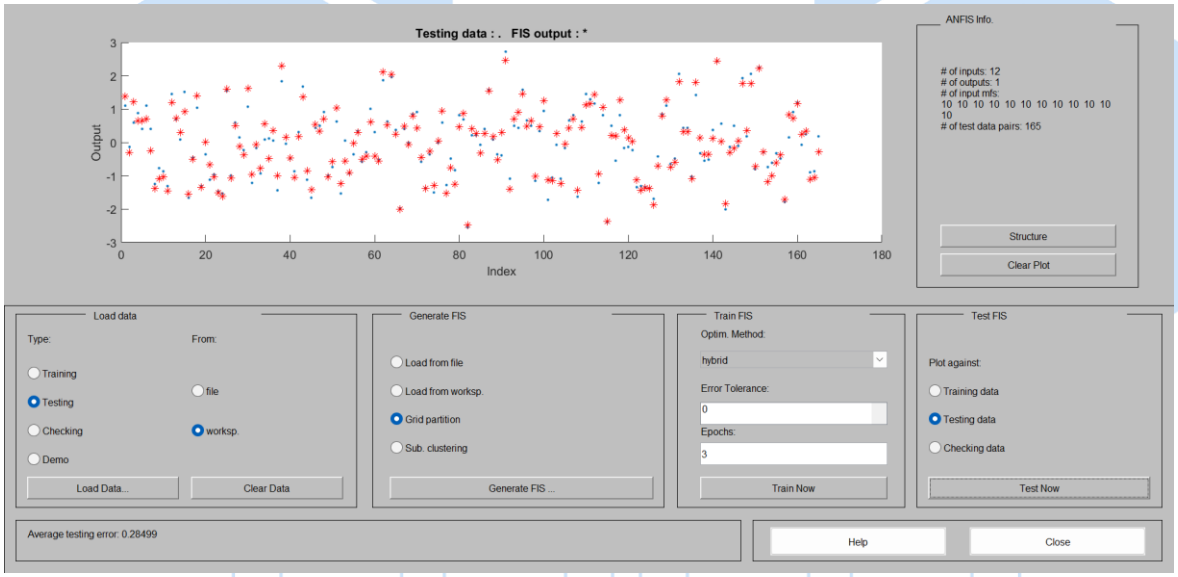
نمودار گام آموزش:



عملکرد مدل نهایی روی داده های آموزش:



عملکرد مدل نهایی روی داده های تست:



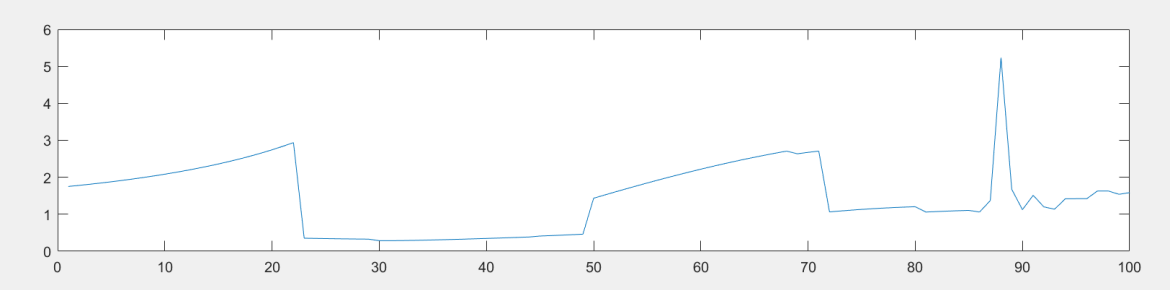
در این مرحله، مدل ANFIS نهایی روی داده‌های آموزشی و تست اعمال شده و خروجی‌های پیش‌بینی‌شده تولید می‌شوند. سپس مقدار میانگین مربعات خطا برای هر دو مجموعه محاسبه شده است. این مقدار نشان می‌دهد که مدل ANFIS چه دقتی در پیش‌بینی داده‌های جدید دارد. میانگین مربعات خطا معیاری استاندارد برای ارزیابی مدل‌های رگرسیون بوده و مقدار کمتر آن به معنی دقت بالاتر مدل است.

```
mseTrain0 =

    0.0276

>> mseTest0

mseTest0 =

    0.0812
```

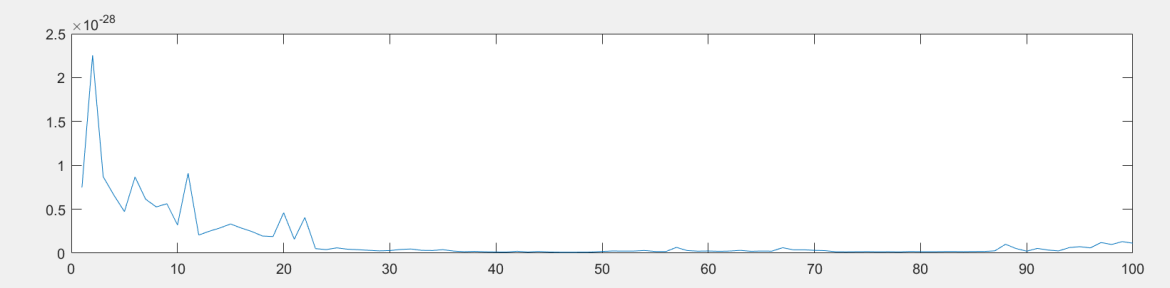
برای مقایسه عملکرد شبکه RBF باANFIS ، یک شبکه تابع پایه شعاعی با استفاده از newrbeآموزش داده شده است. این شبکه با مقادیر مختلفی از پارامتر پخش آزمایش می‌شود تا مقدار بهینه‌ای برای آن پیدا شود. مقدار گسترش از یک تا یک صدم با کاهش‌های تدریجی مورد آزمایش قرار گرفته و برای هر مقدار، یک شبکه جدید ساخته شده است. عملکرد هر مدل روی داده‌های آموزشی و تست ارزیابی شده و مقدار میانگین مربعات خطای آن ذخیره شده است. مدل با کمترین خطای تست به عنوان بهترین شبکه انتخاب شده است.

در این مرحله، مقادیر میانگین مربعات خطا برای مدل‌های مختلف شبکه RBF رسم شده است. نمودار اول روند تغییرات خطای تست را در مقادیر مختلف گسترش نشان می‌دهد و نمودار دوم مربوط به تغییرات خطای آموزش است. این نمودارها نشان می‌دهند که چگونه تغییر پارامتر گسترش روی عملکرد شبکه تأثیر گذاشته است. در نهایت، مدلی که کمترین مقدار خطای تست را داشته به عنوان مدل بهینه انتخاب شده است. بررسی خروجی‌های این مدل نشان داد که مقادیر پیش‌بینی شده با مقادیر واقعی همخوانی مناسبی دارند و بنابراین مسئله به طور کامل حل شده است.

نمودار خطا روی داده های تست:



نمودار خطا روی داده های آموزش:



در نهایت، عملکرد مدل با کمترین خطا روی داده های تست به این صورت میباشد:

```
>> disp(minErrTr);
disp(minErrTs);
   2.9064e-30

    0.2848
```

و تعداد مراکز این مدل نیز برابر با 496 عدد میباشد.

نتیجه گیری:

مدل ANFIS توانست با خطای کم ۰.۰۲ روی داده‌های آموزش و ۰.۰۸ روی داده‌های تست عملکرد خوبی از خود نشان دهد. این مدل با استفاده از منطق فازی و الگوریتم‌های آموزشی تطبیقی به خوبی با پیچیدگی‌های داده‌های ورودی سازگار شد و نتایج مناسبی را در پیش‌بینی خروجی‌ها به دست آورد. در این مدل، به دلیل استفاده از داده‌های اعتبارسنجی برای تنظیم دقیق‌تر پارامترها، توانایی تعمیم به داده‌های جدید بهتر از مدل‌های ساده‌تر بهبود یافت.

در مقابل، مدل RBF دارای خطای صفر روی داده‌های آموزش و خطای ۰.۲ روی داده‌های تست بود. هرچند که مدل RBF موفق شد تا پیش‌بینی دقیقی روی داده‌های آموزش انجام دهد، اما زمانی که با داده‌های تست مواجه شد، نتایج کمتری ارائه کرد. دلیل این امر ممکن است مربوط به میزان پیچیدگی بالای داده‌ها باشد که مدل RBF قادر به تعمیم دادن آن‌ها به خوبی نبوده است. همچنین تعداد مراکز شبکه RBF که معادل تعداد نمونه‌های آموزشی بود، نشان می‌دهد که این مدل بیش از حد به داده‌های آموزش تطبیق پیدا کرده است و در نتیجه در داده‌های تست عملکرد ضعیف‌تری داشته است.

در نهایت، دلیل بهتر بودن عملکرد مدل ANFIS این است که این مدل با استفاده از ترکیب منطق فازی و فرآیند آموزش تطبیقی، قادر به تعمیم بهتر به داده‌های جدید بوده و از طریق اعتبارسنجی، تنظیمات بهینه را برای کاهش خطا در داده‌های تست اعمال کرده است. این در حالی است که مدل RBFبه دلیل تعداد زیاد مراکز و تطبیق بیش از حد با داده‌های آموزش، قادر به ارائه نتایج مشابه با داده‌های تست نبوده و دچار overfitting شده است.