به نام خدا

1928
K. N. Toosi University of Technology

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش مینی پروژه شماره سه

[ابوالفضل ولی زاده لاکه]

[40010273]

استاد : آقای دکتر مهدی علیاری

بهمن 1403

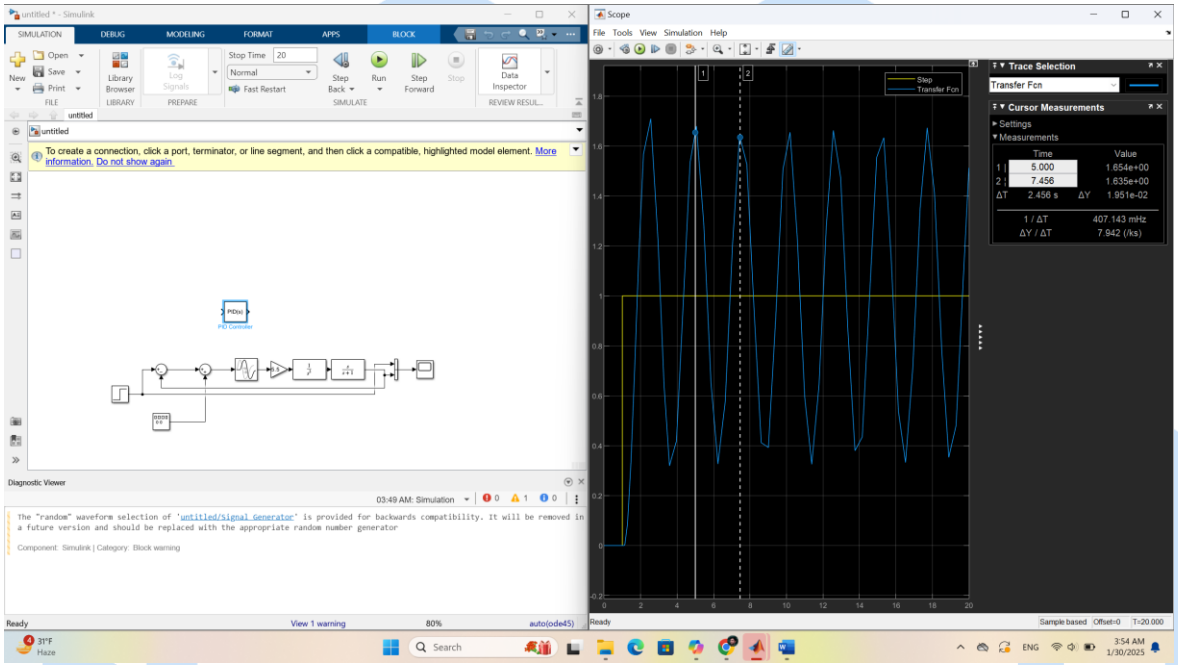
فهرست مطالب

عنوان شماره صفحه

سوال اول بخش 1

در این سوال طبق گفته سوال می بایست کنترلر PID به روش زیگلر نیکولز طراحی کنیم. برای اینکار ابتدا می بایست دو گین مربوط به I و D را صفر کنیم سپس K را تا جایی افزایش بدهیم که به نوسان پایدار برسیم. در این حالت باید دقت کنیم که تابع انتقالی حلقه باز باید از درجه دوم باشد تا نوسان کند پس سیستم را به کمک یک کنترلر 1/s^2 به سیستم مرتبه دو تبدیل می کنیم.

حال آنقدر K را افزایش می دهیم تا سیستم نوسان پایدار کند:



همانطور که مشاهده می شود در تاخیر 0.1 ثانیه ای و گین 5.5 سیستم نوسان پایدار می کند. دوره تناوب این نوسان برابر با 2.456 ثانیه حال به کمک متد زیگلر نیکلز کنتلر PID را طراحی می کنیم:

Control Type	K_p	T_i	T_d	K_i	K_d
classic PID[2]	$0.6K_u$	$0.5T_u$	$0.125T_u$	$1.2K_u/T_u$	$0.075K_uT_u$

Kp = 0.6 *5.5 = 3.3

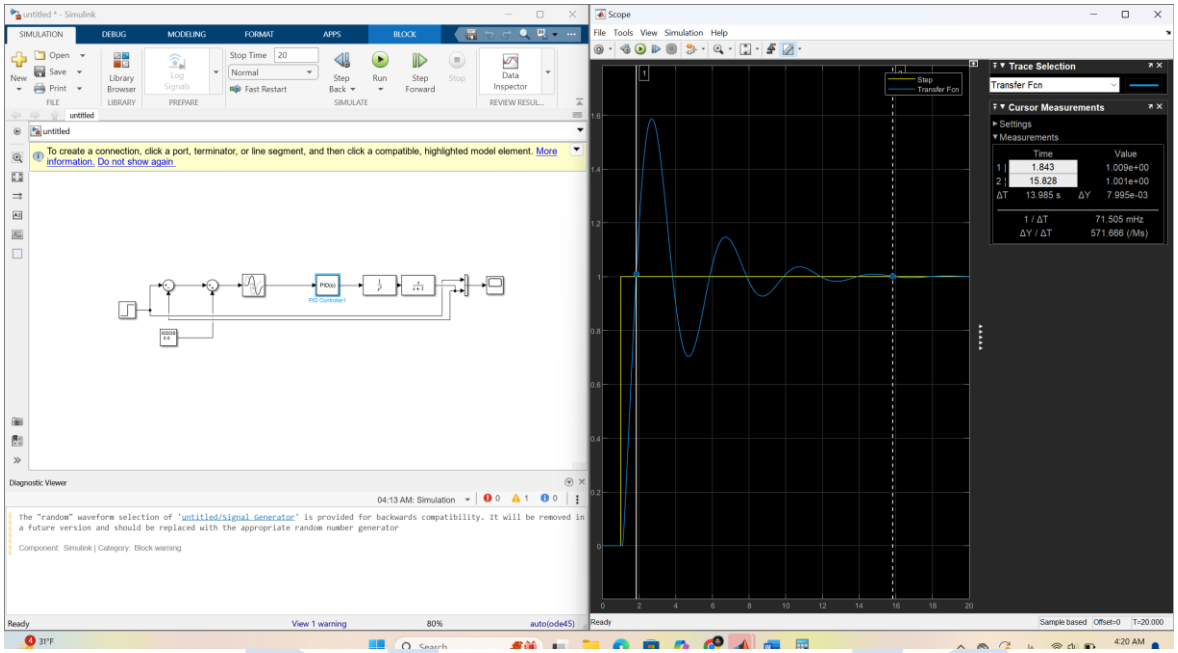
Ti = 0.5 * 2.456 = 1.228

Td = 0.125 * 2.456 = 0.307

Ki = 1.2 * 5.5 / 2.456 = 2.687

Kd = 0.075 * 5.5 * 2.456 = 1.0131

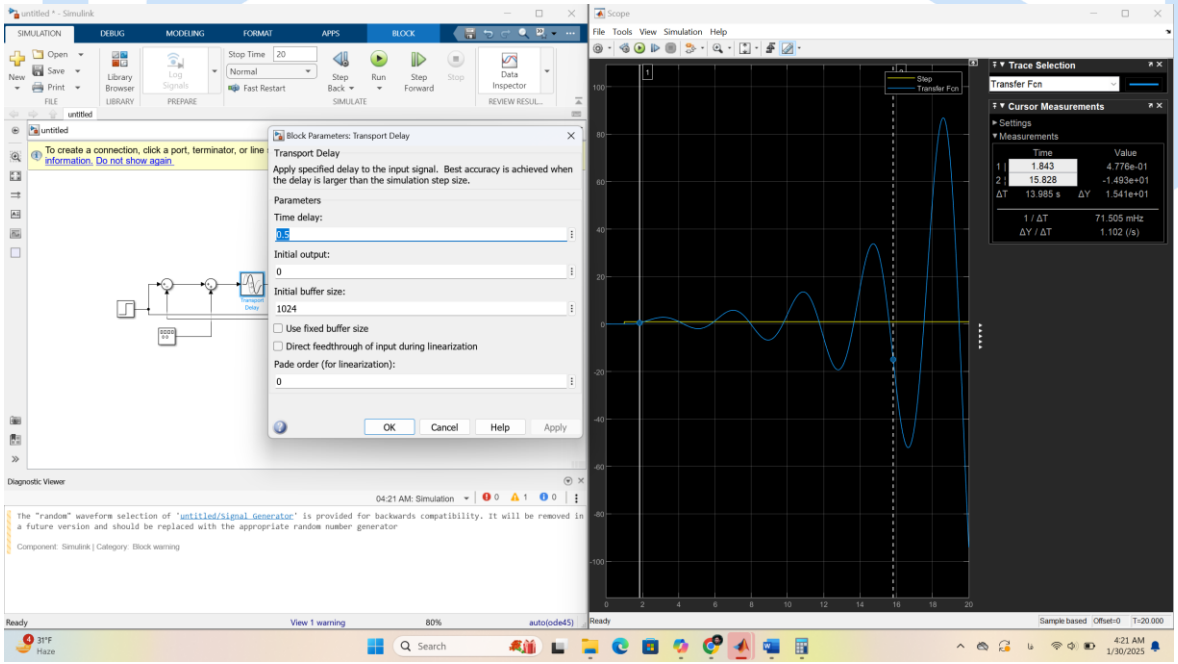
سپس با پارامتر های مذکور کنتلر کننده خود را تحت پارامتر های مشتق شده از روش زیگلر نیکلز تنظیم می کنیم:



همانطور که مشاهده می شود کنتلر طراحی شده دارای زمان خیز و نشست و اورشوت معقولیست.

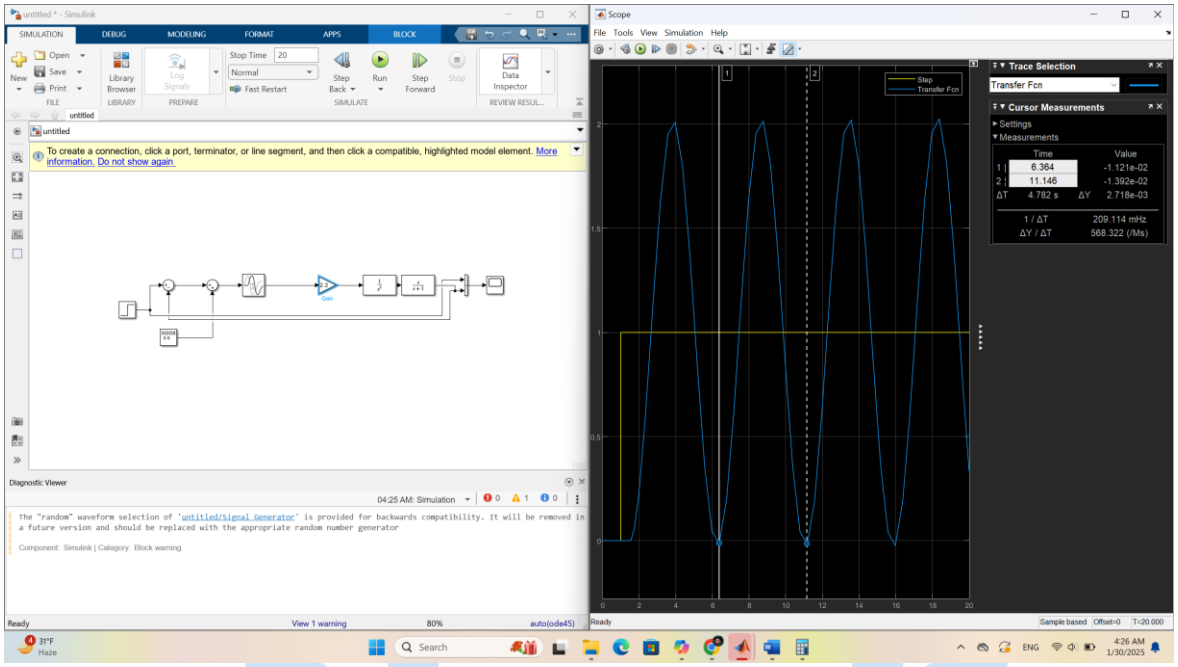
بخش 2

حال اثر تاخیر را بیشتر می کنیم:



همانطور که مشاهده می شود با افزایش تاخیر از 0.1 به 0.5 سیستم ناپایدار می شود.

مجددا برای کنتل از طریق روش زیگلر نیکلز اقدام می کنیم:



همانطور که مشاهده می شود به ازای گین برابر با 2.2 خروجی نوسان پایدار با دوره تناوب 4.782 ثانیه می کند:

مجددا به روش زیگلر نیکلز مقادیر PID را محاسبه می کنیم:

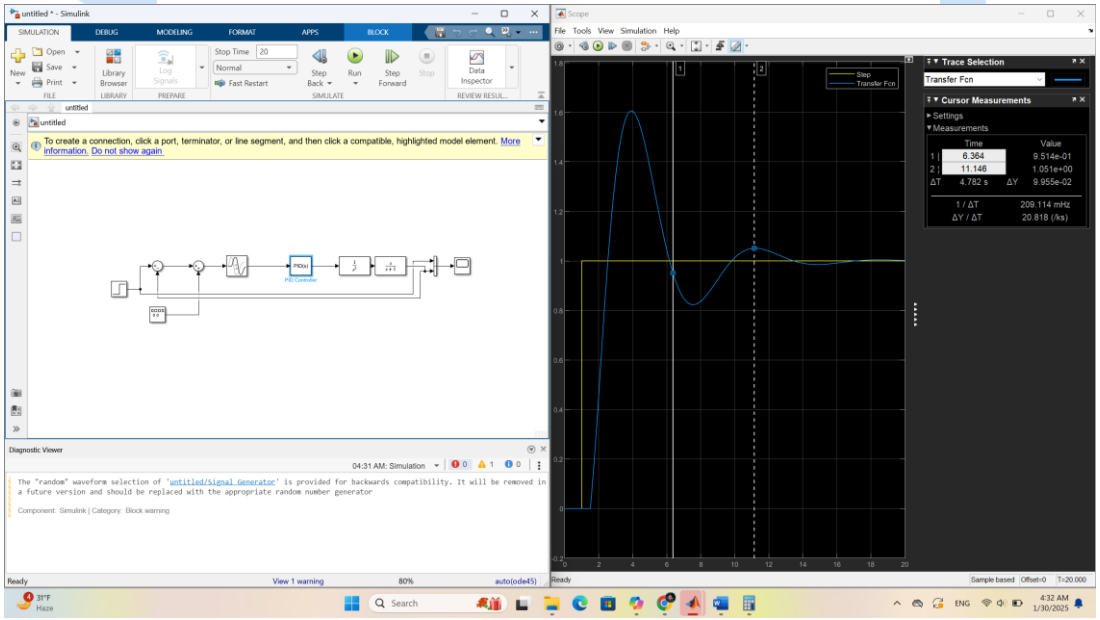
Kp = 0.6 *2.2 = 1.32

Ti = 0.5 * 4.782 = 2.391

Td = 0.125 * 4.782 = 0.59775

Ki = 1.2 * 2.2 / 4.782 = 0.552

Kd = 0.075 * 2.2 * 4.782 = 0.789



مشاهده می شود که کنترلر به خوبی عمل می کند.

سوال دوم

هدف ما طراحی یک کنترل‌کننده است که رفتار راننده را هنگام حرکت معکوس شبیه‌سازی کند. موقعیت مؤلفه با سه متغیر شامل موقعیت افقی، زاویه چرخش و زاویه نسبت به محور افقی مشخص می‌شود. کنترل سیستم از طریق زاویه فرمان انجام می‌شود که در حرکت ساعت‌گرد مقدار مثبت و در حرکت پادساعت‌گرد مقدار منفی دارد. برای ساده‌سازی، فرض می‌کنیم که فضای کافی بین مؤلفه و دیوار وجود دارد و نیازی به در نظر گرفتن آن نیست. ورودی‌های سیستم شامل موقعیت افقی و زاویه چرخش هستند و خروجی آن زاویه نسبت به محور افقی است. هدف نهایی این است که کنترل‌کننده‌ای طراحی کنیم که با دریافت موقعیت و زاویه چرخش، زاویه مناسب را برای تنظیم حرکت معکوس تعیین کند.

ابتدا با استفاده از روش آزمون و خطا، مجموعه‌ای از جفت‌های ورودی-خروجی را تولید می‌کنیم. در این فرآیند، از یک وضعیت اولیه شروع کرده و مقدار مناسب برای کنترل را بر اساس تجربه و درک شهودی تعیین می‌کنیم. پس از انجام چندین آزمایش، جفت‌هایی را انتخاب می‌کنیم که منجر به هموارترین و موفق‌ترین مسیر حرکت شوند. برای این کار، از چهارده وضعیت اولیه استفاده شده است تا داده‌های کافی برای طراحی کنترل‌کننده فراهم شود.

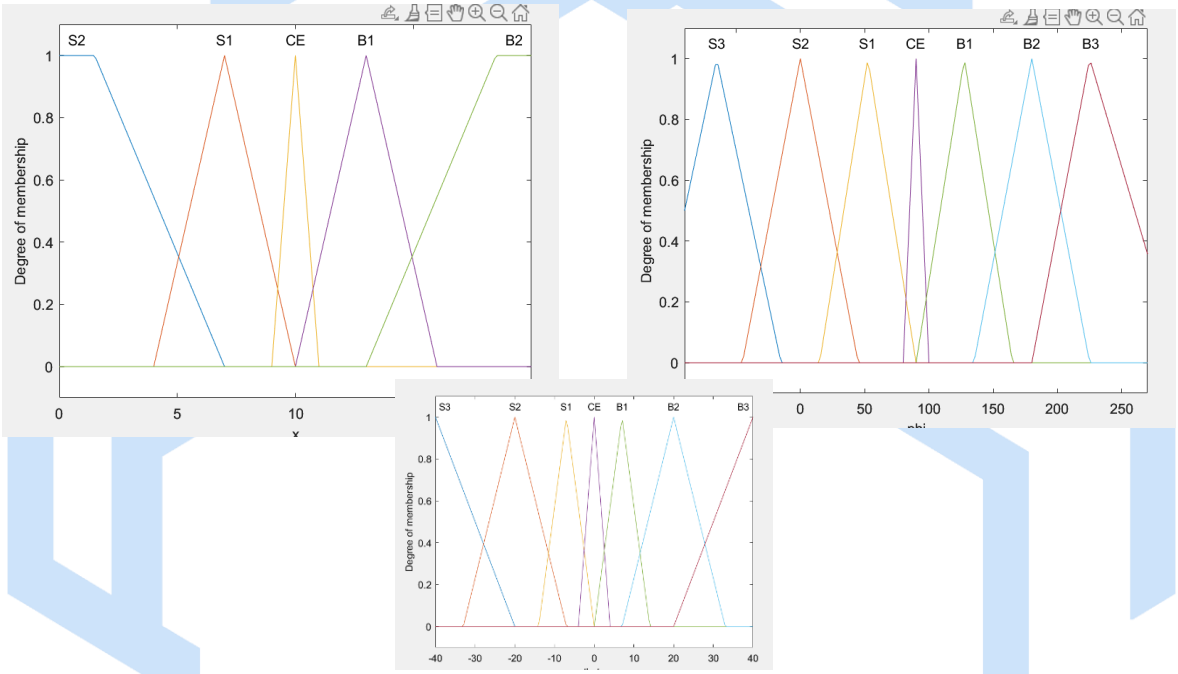
از چهارده وضعیت اولیه‌ای که در مراحل قبل انتخاب شده‌اند، در مجموع حدود ۲۵۰ جفت ورودی-خروجی به دست می‌آید. اکنون می‌توانیم با استفاده از این داده‌ها، یک سیستم فازی بر اساس روش جدول جستجو طراحی کنیم.

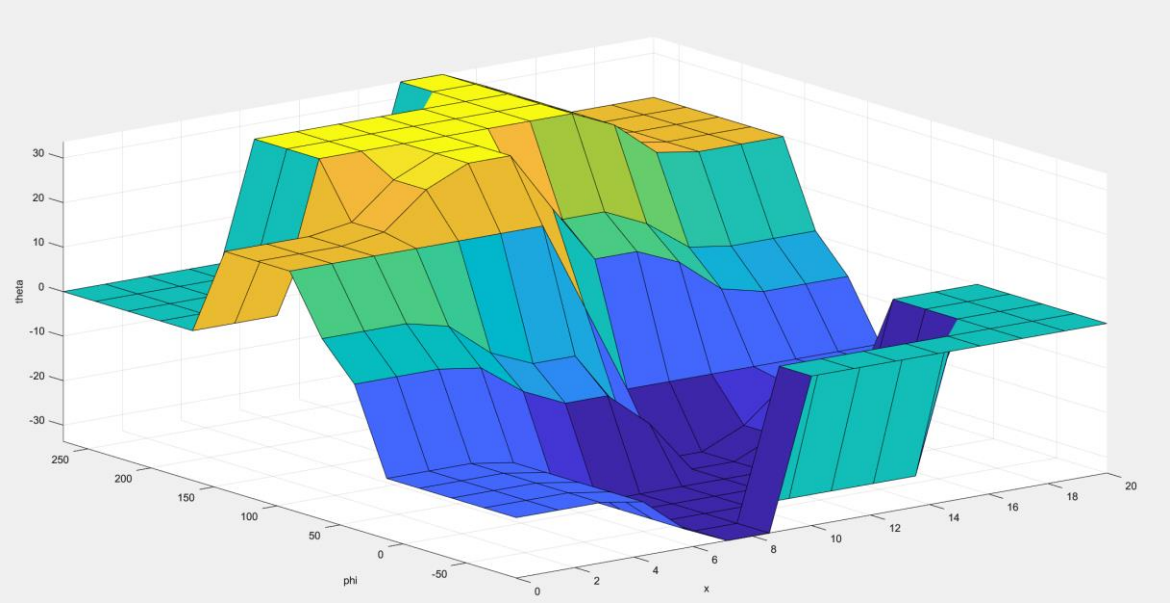
با توجه به داده‌های جمع‌آوری‌شده، یک سیستم فازی بر اساس طرح جدول جستجو توسعه داده‌ایم که در آن، مقدار خروجی موردنظر برای هر ترکیب از ورودی‌ها در یک جدول ذخیره شده است. این جدول به عنوان نگاشت ورودی به خروجی عمل کرده و کنترل سیستم را بر اساس مقادیر به‌دست‌آمده از آزمایش‌ها انجام می‌دهد. جدول جستجو به‌صورت زیر ارائه شده است:

ϕ \ X	S2	S1	CE	B1	B2
S3	S2	S3			
S2	S2	S3	S3	S3	
S1	B1	S1	S2	S3	S2
CE	B2	B2	CE	S2	S2
B1	B2	B3	B2	B1	S1
B2		B3	B3	B3	B2
B3				B3	B2

بررسی جدول جستجو نشان می‌دهد که برخی از خانه‌ها خالی هستند، به این معنی که جفت‌های ورودی–خروجی تمامی فضای حالت را پوشش نمی‌دهند. بااین‌حال، مشاهده خواهیم کرد که قوانین تعریف‌شده برای کنترل وسیله به موقعیت مطلوب کافی هستند، حتی اگر حرکت از طیف گسترده‌ای از شرایط اولیه آغاز شود.

برای طراحی سیستم فازی، هفت تابع عضویت برای زاویه چرخش، پنج تابع عضویت برای موقعیت افقی و هفت تابع عضویت برای زاویه خروجی تعریف کرده‌ایم. نوع توابع عضویت عمدتاً مثلثی هستند، اما در برخی موارد از ذوزنقه‌ای نیز استفاده شده است. در تصویر زیر، سیستم فازی طراحی‌شده بر اساس این قوانین نمایش داده شده است که شامل دو ورودی و یک خروجی است.





برای آزمایش سیستم استنتاج فازی، ابتدا باید یک وضعیت اولیه برای ورودی‌ها (موقعیت افقی و زاویه چرخش) تعیین کنیم. سپس یک حلقه طراحی می‌کنیم که در هر مرحله وضعیت زاویه خروجی را محاسبه کرده و بر اساس آن، موقعیت مرحله بعدی را با استفاده از معادلات کینماتیکی محاسبه کند.

$$x_{(k+1)} = x_{(k)} + \cos\cos\left[\Phi_{(k)} + \theta_{(k)}\right] + \sin\sin\left[\theta_{(k)}\right]\sin\sin\left[\Phi_{(k)}\right].$$

$$\mathbf{y}_{(k+1)} = y_{(k)} + \cos\cos\left[\Phi_{(k)} + \theta_{(k)}\right] - \sin\sin\left[\theta_{(k)}\right]\sin\sin\left[\Phi_{(k)}\right],$$

$$\Phi_{(k+1)} = \Phi_{(k)} - \left[\frac{2\sin\sin\left[\theta_{(k)}\right]}{b}\right],$$

$$b = 2$$

برای ادامه آزمایش سیستم استنتاج فازی، حلقه باید تا زمانی که **خطا** (تفاوت بین موقعیت کنونی و موقعیت هدف) به حد مشخصی مانند **0.01**کاهش یابد، ادامه پیدا کند. این فرآیند به شرح زیر خواهد بود:

- ✓ ابتدا وضعیت اولیه (موقعیت افقی و زاویه چرخش) و موقعیت هدف مشخص می‌شود.
- ✓ با استفاده از معادلات کینماتیکی، موقعیت جدید (موقعیت افقی و زاویه چرخش) به‌روز می‌شود.
- ✓ خطا (تفاوت بین موقعیت کنونی و موقعیت هدف) محاسبه می‌شود.
- ✓ حلقه ادامه می‌یابد تا زمانی که مقدار خطا کمتر از 0.01 شود.

همانطور که از شکل مشاهده می‌شود، سیستم استنتاج فازی (FIS) به‌طور موفقیت‌آمیز توانسته است کامیون را به موقعیت هدف هدایت کند و موقعیت نهایی در این حالت به شرح زیر است:

موقعیت افقی نهایی**9.9996**

زاویه چرخش نهایی **89.9907**

زاویه خروجی نهایی **29.6992**

این نتایج نشان‌دهنده عملکرد مطلوب سیستم فازی در هدایت کامیون به موقعیت هدف است.

سوال سوم دیتا ست Ball and Beam

بارگیری مجموعه داده

- فایل ballbeam.datاز حالت فشرده خارج می‌شود.
- مجموعه داده بارگذاری شده و در متغیرهای input_data (ورودی) و output_data (خروجی) ذخیره می‌شود.

```
!gunzip ballbeam.dat.Z
    load ballbeam.dat
input_data=ballbeam(:,1);
output_data=ballbeam(:,2);
```

تقسیم‌بندی داده‌ها

- تعداد کل نمونه‌های داده تعیین می‌شود.
- داده‌ها به‌صورت تصادفی مرتب شده و به دو مجموعه تقسیم می‌شوند:
  - 85% برای آموزش
  - 15% برای آزمون
- داده‌های آموزشی و آزمونی جداگانه ذخیره می‌شوند.

```
num_samples = length(input_data);

random_indices = randperm(num_samples);

train_ratio = 0.85;
num_train = round(train_ratio * num_samples);

train_indices = random_indices(1:num_train);
test_indices = random_indices(num_train+1:end);

input_train = input_data(train_indices);
output_train = output_data(train_indices);

input_test = input_data(test_indices);
output_test = output_data(test_indices);

training data = [input train, output train];
```

ایجاد مدل فازی اولیه

- مدل اولیه فازی با استفاده از genfis2 ایجاد می‌شود.
- مقدار 0.15 به عنوان پارامتر تأثیرگذاری نقاط داده در ایجاد مدل در نظر گرفته می‌شود.

```
fis = genfis2(input_train, output_train, 0.15);
```

آموزش مدل ANFIS

- تنظیمات مربوط به آموزش مشخص می‌شود:
  - تعداد تکرارها :250 دوره آموزشی
  - هدف خطا :صفر
  - نرخ کاهش گام یادگیری: 0.9
  - نرخ افزایش گام یادگیری : 1.1
- مدل با استفاده از داده‌های آموزشی تمرین داده می‌شود.
- نمایش اطلاعات مربوط به مراحل آموزش مانند میزان خطا، مقدار گام یادگیری و نتایج نهایی فعال است.

```
MaxEpoch = 250;
ErrorGoal = 0;
InitialStepSize = 0.01;
StepSizeDecreaseRate = 0.9;
StepSizeIncreaseRate = 1.1;
TrainOptions = [MaxEpoch, ErrorGoal, InitialStepSize, StepSizeDecreaseRate, StepSizeIncreaseRate];

DisplayInfo = true;
DisplayError = true;
DisplayStepSize = true;
DisplayFinalResult = true;
DisplayOptions = [DisplayInfo, DisplayError, DisplayStepSize, DisplayFinalResult];

OptimizationMethod = 1;

[fis, trainError] = anfis(training_data, fis, TrainOptions, DisplayOptions, [], OptimizationMethod);

TrainOutputs = evalfis(input_train, fis);
TrainErrors = output_train - TrainOutputs;
TrainMSE = mean(TrainErrors(:).^2);
TrainRMSE = sqrt(TrainMSE);

PlotResults(output_train, TrainOutputs, 'Train Data');
```

ارزیابی مدل

- خروجی مدل برای داده‌های آموزشی و آزمونی محاسبه می‌شود.
- میزان خطای مدل با استفاده از معیارهای MSE و RMSE ارزیابی می‌شود.
- نمودارهای مربوط به خطاهای آموزشی و آزمونی رسم می‌شود.

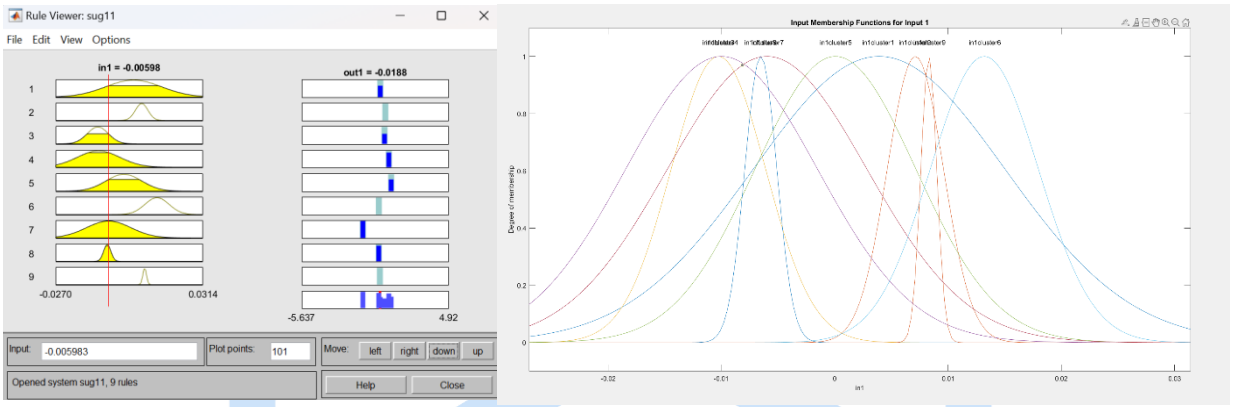
```
TestOutputs = evalfis(input_test, fis);
TestErrors = output_test - TestOutputs;
TestMSE = mean(TestErrors(:).^2);
TestRMSE = sqrt(TestMSE);

PlotResults(output_test, TestOutputs, 'Test Data');

disp(['Train RMSE: ', num2str(TrainRMSE)]);
disp(['Test RMSE: ', num2str(TestRMSE)]);
```

ترسیم نمودارها و تحلیل نتایج

- نمودارهای تطبیق :نمایش خروجی‌های واقعی و پیش‌بینی‌شده مدل.
- نمودار توزیع خطا :بررسی توزیع خطاهای خروجی مدل.
- نمودار توابع عضویت :نمایش توابع عضویت ورودی مدل فازی.
- نمودار سطحی :نمایش رابطه بین ورودی و خروجی مدل فازی.
- نمودار همگرایی خطا :بررسی کاهش خطا در طول دوره‌های آموزشی.

```
plotmf(fis, 'input', 1);
title('Input Membership Functions for Input 1');

figure;
[X, Y] = meshgrid(linspace(min(input_train), max(input_train), 100));
Z = evalfis([X(:)], fis);
Z = reshape(Z, size(X));
surf(X, Z);
xlabel('Input');
ylabel('Fuzzy Output');
zlabel('Output');
title('Surface Plot of Fuzzy Inference System');

ruleview(fis);

figure;
plot(1:MaxEpoch, trainError);
title('Convergence Plot (Training Error over Epochs)');
xlabel('Epoch');
ylabel('Training Error');

figure;
bar([TrainRMSE, TestRMSE]);
set(gca, 'xticklabel', {'Train RMSE', 'Test RMSE'});
title('RMSE Comparison (Train vs Test)');
ylabel('RMSE');
```

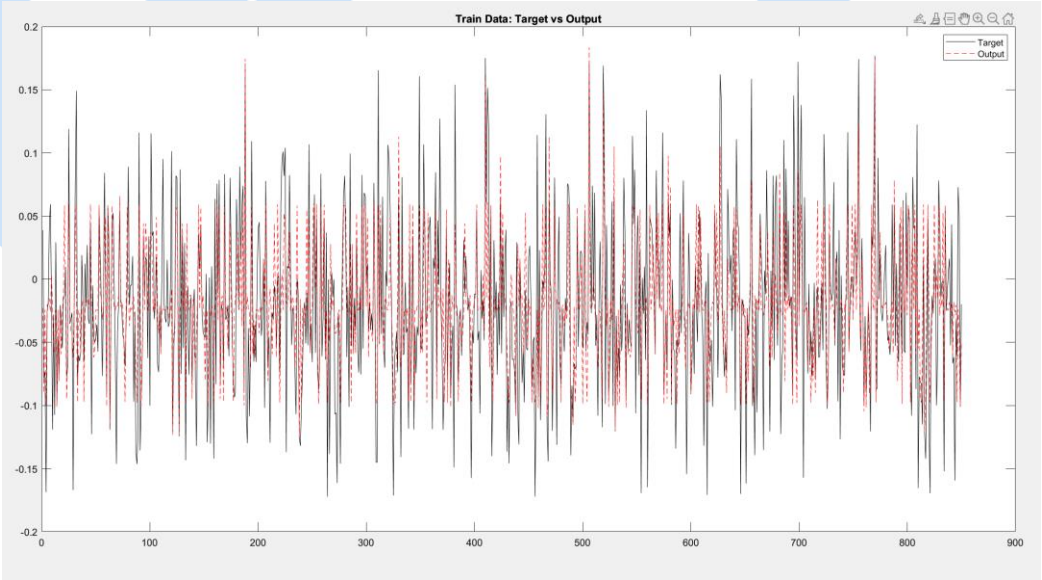
تحلیل نتایج:

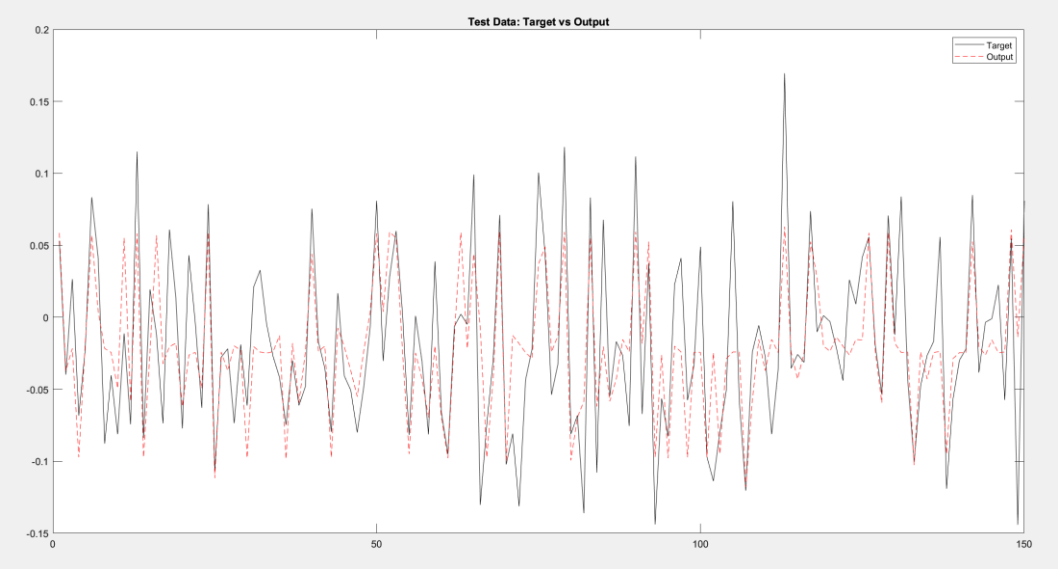


همانطور که مشاهده می شود مدل شامل 9 تابع تعلق می باشد که به صورت بالا به طور نامتقارن با پهنا ها (واریانس های مختلف) بخش شده اند.

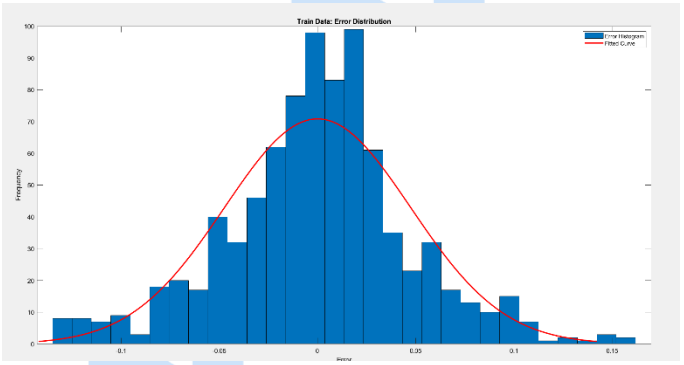
خروجی مدل تا حد قابل قبولی به خروجی واقعی نزدیک است. البته باید مدنظر داشت که تابع دارای پیچیدگی های زیادیست و برای کاهش خطا می بایست تعداد توابع تعلق را بیشتر کرد.

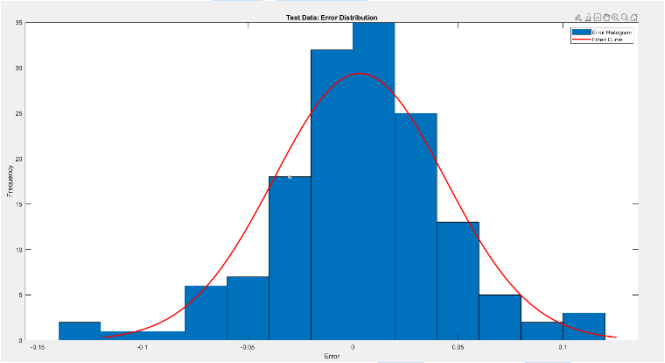
خروجی مدل و خروجی واقعی به شرح نمودار زیر برای داده های آموزش و آزمون می باشد:

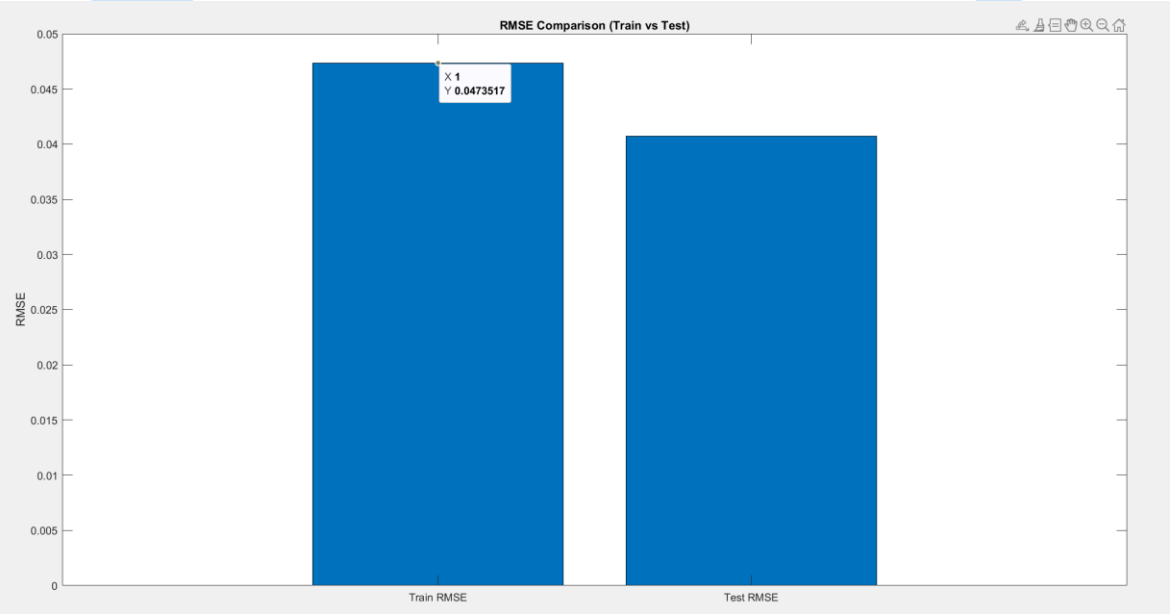




خروجی نمودار های هیستوگرام خطای داده های آموزش و تست نیز به صورت زیر می باشد که مطلوب هستند.

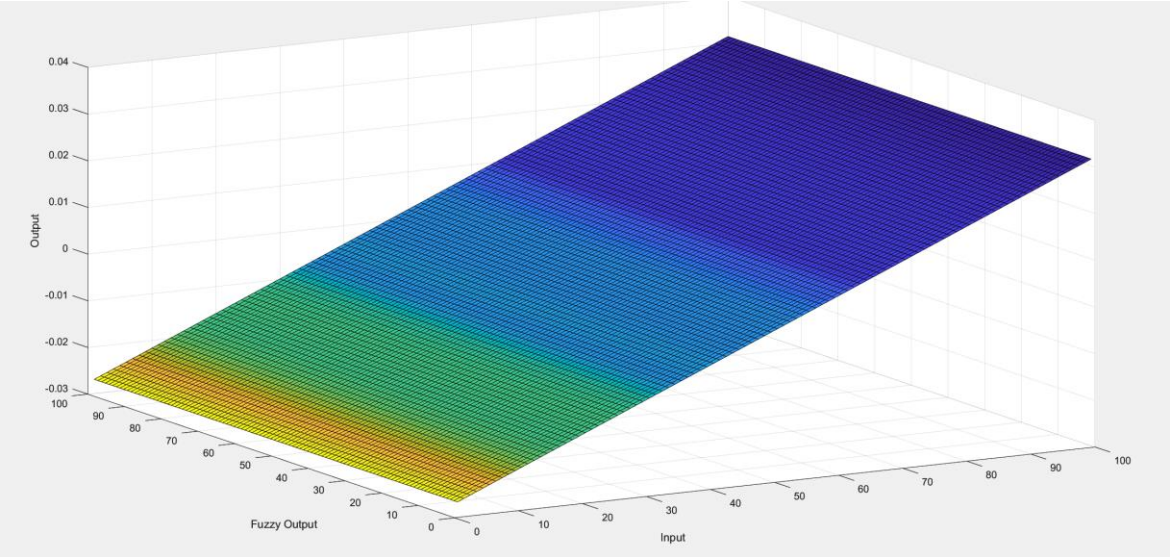






خطای RMSE ریشه مجموع مربعات به شرح نمودار بالا برای داده های آموزش و آزمون رسم شده که زیر 0.05 می باشد که می توان مدل Anfis طراحی شده را مطلوب ارزیابی کرد.

همچنین رویه مدل فازی نیز رسم شده که به ما بر حسب رنگبندی مرز های تصمیم گیری و برخورد توابع تعلق را نشان می دهد.



سوال چهارم

در این سوال قصد داریم بر حسب تابع بازگشتی مذکور را که به کمک تابع متناوب دیگری تعریف می شود برازش دهیم. تابع دوم نیز به دلیل اینکه از مجموع سه تا تابع سینوسی تشکیل شده دارای دوره تناوب برابر 2 می باشد. در این سوال بر حسب مراحل مذکور درکتاب اشاره شده در صورت سوال ابتدا مقادیر اولیه برای حل سوال از جمله تعداد داده ها، تعداد توابع تعلق، تعداد داده های آموزش و نرخ یادگیری لحاظ شده است. سپس توابع مهم را تعریف می کنیم و سپس به کمک الگوریتم گرادیان نزولی مراکز و واریانس(پهنای توابع گوسی) و همچنین مراکز توابع خروجی را بروزرسانی و بهینه می کنیم:

بروزرسانی پهنای توابع تعلق:

$$\sigma_i^l(q+1) = \sigma_i^l(q) - \alpha \frac{\partial e}{\partial \sigma_i^l}|_q$$

$$= \sigma_i^l(q) - \alpha \frac{f-y}{b}(\bar{y}^l(q) - f) z^l \frac{2(x_{0i}^p - \bar{x}_i^l(q))^2}{\sigma_i^{l3}(q)}$$

بروزرسانی پهنای مراکز توابع تعلق:

$$\bar{x}_i^l(q+1) = \bar{x}_i^l(q) - \alpha \frac{f-y}{b}(\bar{y}^l(q) - f) z^l \frac{2(x_{0i}^p - \bar{x}_i^l(q))}{\sigma_i^{l2}(q)}$$

بروزرسانی مراکز توابع خروجی:

$$\bar{y}^l(q+1) = \bar{y}^l(q) - \alpha \frac{f-y}{b} z^l$$

مقداردهی اولیه پارامترها

در ابتدای کد، تعدادی پارامتر اولیه تنظیم می‌شوند:

- M = 10 تعداد توابع عضویت که برای طراحی سیستم فازی استفاده می‌شود.
- num_training = 800 تعداد نمونه‌های آموزشی.
- total_num = 1000 تعداد کل داده‌ها.
- landa = 0.1 مقدار ثابت گام یادگیری.

تخصیص حافظه برای متغیرها

در این بخش، متغیرهای مورد نیاز برای ذخیره داده‌ها از قبل مقداردهی می‌شوند:

- متغیرهای x_bar، g_bar، sigmaبرای ذخیره مقادیر مرتبط با توابع عضویت.
- yو uبرای مقادیر خروجی و ورودی سیستم.
- x، y_hat، f_hat، z و g_u برای مراحل یادگیری و مدل‌سازی.

```
% Parameters
M = 10; % Number of membership functions
num_training = 800; % Number of training samples
total_num = 1000;
landa = 0.1; % Stepsize constant

% Preallocation
x_bar = zeros(num_training, M);
g_bar = zeros(num_training, M);
sigma = zeros(num_training, M);
y = zeros(total_num, 1);
u = zeros(total_num, 1);
x = zeros(total_num, 1);
y_hat = zeros(total_num, 1);
f_hat = zeros(total_num, 1);
z = zeros(total_num, 1);
g_u = zeros(total_num, 1);
```

مقداردهی اولیه اولین مقدار داده‌ها

در این قسمت:

- مقدار اولیه u(1) به‌صورت تصادفی مقداردهی می‌شود.
- تابع g_u(1) مقدار اولیه خود را با استفاده از یک ترکیب توابع سینوسی دریافت می‌کند.
- f_hat(1) برابر مقدار g_u(1) تنظیم می‌شود.

تنظیم محدوده ورودی‌ها و مقداردهی اولیه توابع عضویت

- محدوده ورودی از 1- تا 1 تنظیم شده و hمقدار گام‌بندی برای توابع عضویت تعیین می‌شود.
- درون یک حلقه، نقاط مرکزی توابع عضویت مقداردهی می‌شوند.
- مقدار g_barبر اساس ترکیب توابع سینوسی محاسبه می‌شود.

```
y(1) = 0;
u(1) = rand;
g_u(1) = 0.6 * sin(pi * u(1)) + 0.3 * sin(3 * pi * u(1)) + 0.1 * sin(5 * pi * u(1));
f_hat(1) = g_u(1);

u_min = -1;
u_max = 1;
h = (u_max - u_min) / (M - 1);
```

مقداردهی اولیه سیگما و ذخیره مقادیر اولیه

- مقدار sigmaبا در نظر گرفتن حداقل و حداکثر مقدار uمقداردهی می‌شود.
- مقادیر اولیه x_bar، sigma، و g_barذخیره می‌شوند.

```
sigma(1, 1:M) = (max(u(1, :)) - min(u(1, :))) / M;
x_bar(2, :) = x_bar(1, :);
g_bar(2, :) = g_bar(1, :);
sigma(2, :) = sigma(1, :);

x_bar_initial = x_bar(1, :);
sigma_initial = sigma(1, :);
y_bar_initial = g_bar(1, :);
```

-

فاز آموزش با استفاده از خطا

- حلقه‌ای برای آموزش مدل اجرا می‌شود.
- مقدار x(q) و u(q) به‌صورت تصادفی مقداردهی می‌شوند.
- مقدار g_u(q) بر اساس ترکیب توابع سینوسی به‌دست می‌آید.
- مقدار تابع عضویت برای هر r محاسبه می‌شود و مقدار f_hat(q) به عنوان تقریب خروجی تنظیم می‌شود.
- به‌روزرسانی پارامترهای توابع عضویت g_bar ، sigma، و x_bar با استفاده از پس‌انتشار خطا انجام می‌شود.
- مقدار y(q+1) و y_hat(q+1) محاسبه می‌شوند.

```
for q = 2:num_training
    b = 0; a = 0;
    x(q) = -1 + 2 * rand; % Random input
    u(q) = x(q);
    g_u(q) = 0.6 * sin(pi * u(q)) + 0.3 * sin(3 * pi * u(q)) + 0.1 * sin(5 * pi * u(q));

    Z = zeros(1, M);
    for r = 1:M
        z(r) = exp(-((x(q) - x_bar(q, r)) / sigma(q, r))^2);
        Z(r) = z(r);
        b = b + z(r);
        a = a + g_bar(q, r) * z(r);
    end

    f_hat(q) = a / b; % Output approximation
```

ذخیره مقادیر نهایی توابع عضویت

- مقادیر نهایی x_bar، sigma، و g_bar پس از آموزش ذخیره می‌شوند.

```
        error = g_u(q) - f_hat(q);
        for r = 1:M
            g_bar(q + 1, r) = g_bar(q, r) + landa * Z(r) * error;
            sigma(q + 1, r) = sigma(q, r) + landa * Z(r) * error * ((x(q) - x_bar(q, r))^2 / sigma(q, r)^3);
            x_bar(q + 1, r) = x_bar(q, r) + landa * Z(r) * error * ((x(q) - x_bar(q, r)) / sigma(q, r)^2);
        end

        x_bar(q + 1, :) = x_bar(q, :);
        sigma(q + 1, :) = sigma(q, :);

        y(q + 1) = 0.3 * y(q) + 0.6 * y(q - 1) + g_u(q);
        y_hat(q + 1) = 0.3 * y(q) + 0.6 * y(q - 1) + f_hat(q);
    end

    x_bar_final = x_bar(num_training, :);
    sigma_final = sigma(num_training, :);
    g_bar_final = g_bar(num_training, :);
```

فاز آزمایش

- در این مرحله، مدل با داده‌های جدید مورد آزمایش قرار می‌گیرد.
- مقادیر ورودی از تابع sin(2 * q * pi / 200)تولید می‌شوند.
- مقدار g_u(q)بر اساس ترکیب توابع سینوسی محاسبه شده و خروجی مدل (f_hat(q)) محاسبه می‌شود.
- مقادیر y(q+1)و y_hat(q+1)بر اساس مدل به‌روز می‌شوند.

```
% Test phase
for q = num_training:1000
    b = 0; a = 0;
    x(q) = sin(2 * q * pi / 200);
    u(q) = x(q);
    g_u(q) = 0.6 * sin(pi * u(q)) + 0.3 * sin(3 * pi * u(q)) + 0.1 * sin(5 * pi * u(q));

    Z = zeros(1, M);
    for r = 1:M
        z(r) = exp(-((x(q) - x_bar(num_training, r)) / sigma(num_training, r))^2);
        Z(r) = z(r);
        b = b + z(r);
        a = a + g_bar(num_training, r) * z(r);
    end

    f_hat(q) = a / b; % Output approximation
    y(q + 1) = 0.3 * y(q) + 0.6 * y(q - 1) + g_u(q);
    y_hat(q + 1) = 0.3 * y(q) + 0.6 * y(q - 1) + f_hat(q);
end
```

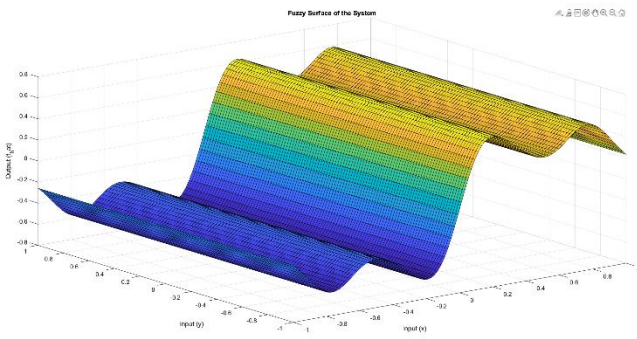
محاسبه خطای RMSE

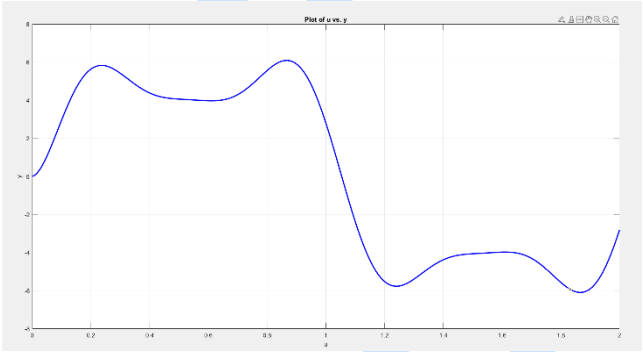
- مقدار **خطای میانگین مربعات ریشه‌ای (RMSE)** برای داده‌های آزمایشی محاسبه می‌شود و در خروجی نمایش داده می‌شود.
- به خطای در حدود 0.1 دست پیدا می کنیم که بسیار مطلوب می باشد.

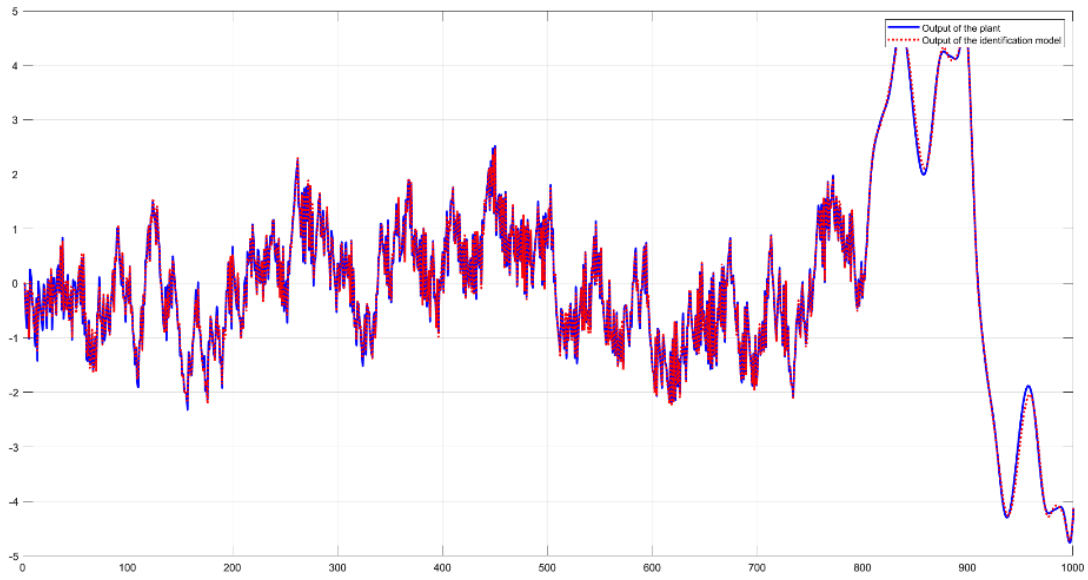
```
% RMSE Calculation
test_data = num_training+1:total_num;
RMSE = sqrt(mean((y(test_data) - y_hat(test_data)).^2));
fprintf('RMSE for test data: %.4f\n', RMSE);
```

```
>> Quest4_Obtimized
RMSE for test data: 0.1037
```

و در نهایت نتایج به صورت دو نمودار سطح فازی و نمودار خروجی واقعی و خروجی برازش شده به نمایش می گذاریم.







همانطور که مشاهده می شود خط نقطه چین که خروجی مدل ما می باشد با تقریب بسیار خوبی به تابع مذکور برازش شده است.

سوال چهارم بخش5 و 6

سوال پنجم

طراحی RBF:

داده‌های موجود در فایل AirQualityUCI.xlsxدر متغیر data به عنوان یک جدول (table) خوانده می‌شوند.

```
data = readtable('AirQualityUCI.xlsx');
```

outputData مقدار NO2_GT_(هدف مدل) را استخراج می‌کند.

inputData شامل تمامی متغیرهای به جز NO2_GT_، تاریخ (Date) و زمان (Time) است که به عنوان ویژگی‌های مدل در نظر گرفته نمی‌شوند. سپس نمونه های نامعتبر Nan را از مجموعه داده ها حذف می کنیم.

```
outputData = data.NO2_GT_;
inputData = data{:, setdiff(data.Properties.VariableNames, {'NO2_GT_', 'Date', 'Time'})};
```

مقادیر inputData نرمال‌سازی می‌شوند تا تمامی ویژگی‌ها در یک بازه مشخص (مثلاً [0,1]یا [-1,1])قرار بگیرند .این کار باعث بهبود عملکرد مدل‌های یادگیری ماشین می‌شود.

```
inputData = normalize(inputData);
```

مقدار ;rng(73) برای ثابت نگه داشتن مقدار تصادفی‌سازی تنظیم شده است.

داده‌ها به صورت تصادفی دسته‌بندی می‌شوند:

- 60%برای آموزش (train)
- 20%برای اعتبارسنجی (validation)
- 20%برای آزمایش (test)

```
rng(73);
n = size(inputData, 1);
idx = randperm(n);
trainIdx = idx(1:round(0.6*n));
valIdx = idx(round(0.6*n)+1:round(0.8*n));
testIdx = idx(round(0.8*n)+1:end);
```

سپس ویژگی‌های ورودی (X_train, X_val, X_test) و خروجی‌ها (Y_train, Y_val, Y_test) جدا می‌شوند.

```
X_train = inputData(trainIdx, :);
Y_train = outputData(trainIdx);

X_val = inputData(valIdx, :);
Y_val = outputData(valIdx);

X_test = inputData(testIdx, :);
Y_test = outputData(testIdx);
```

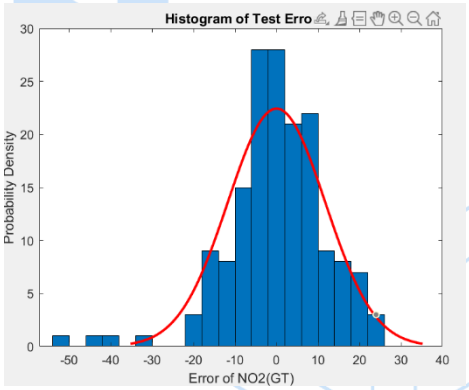
newrb یک شبکه RBF (Radial Basis Function)ایجاد می‌کند. تعداد نورون‌های لایه میانی برابر با 15است.

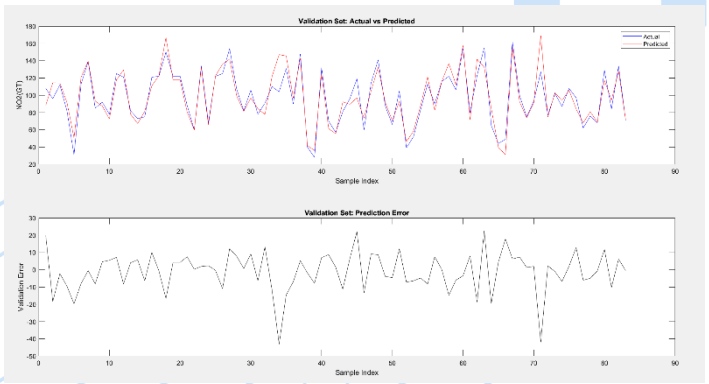
```
numRBFNeurons = 15;
net = newrb(X_train', Y_train', 0, 5, numRBFNeurons, 1);
Y_val_pred = sim(net, X_val');
mse_val = mean((Y_val' - Y_val_pred).^2);
fprintf('Validation MSE: %f\n', mse_val);
```

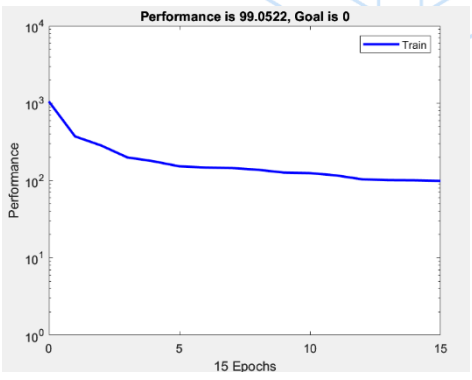
پس از آموزش مدل، پیش‌بینی‌ها روی مجموعه اعتبارسنجی انجام شده و مقدار MSE میانگین مربعات خطا محاسبه می‌شود سپس، مدل روی داده‌های آزمایشی تست شده و مقدار MSEمحاسبه و نمایش داده می‌شود.

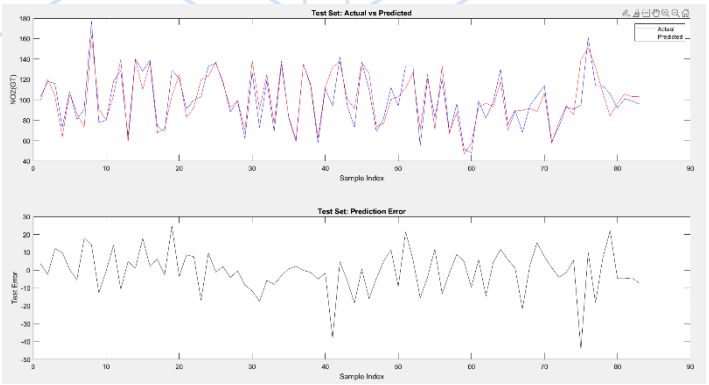
```
Y_val_pred = sim(net, X_val');
mse_val = mean((Y_val' - Y_val_pred).^2);
fprintf('Validation MSE: %f\n', mse_val);
```

برای تحلیل عملکرد مدل، نمودار مقایسه مقادیر واقعی و پیش‌بینی‌شده در مجموعه تست و اعتبارسنجی و همچنین نمودار هیستوگرام و توزیع نرمال خطا رسم شده و خطای پیش‌بینی نمایش داده می‌شود:









همانطور که مشاهده می شود مدل RBF دارای خطای نسبتا معقولی می باشد حال به مدل Anfis می پردازیم و در آخر این دو را با هم مقایسه می کنیم.

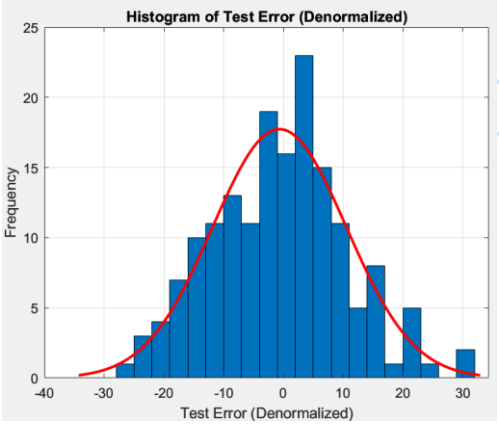
```
Validation MSE: 120.975627
Test MSE: 136.913188
```

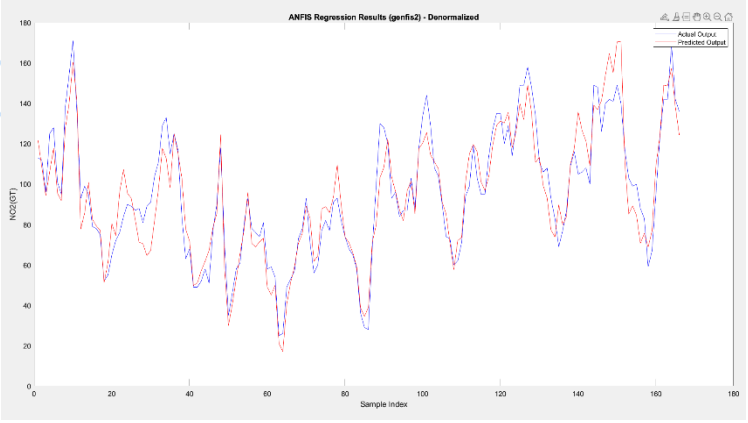
حال مدل ANFIS را برای پیش‌بینی مقدار NO2(GT) آموزش می‌دهیم. از آنجایی که بسیار کد با بخش RBF بخش مشترک زیادی دارد از آوردن مجدد کد ها خودداری شده. این بخش از کد ها به تفضیل در بخش RBF توضیح داده شده اند. ابتدا، داده‌های ورودی و خروجی را از مجموعه داده استخراج کرده و مقادیر 200- که نشان‌دهنده داده‌های نامعتبر هستند، با NaN جایگزین می‌کنیم. سپس، تمام سطرهایی که شامل مقادیر گمشده هستند، حذف می‌کنیم. در ادامه، میانگین و انحراف معیار داده‌های ورودی و خروجی را محاسبه کرده و داده‌ها را با استفاده از این مقادیر نرمال‌سازی می‌کنیم. مجموعه داده را به سه بخش آموزشی(60%) ، اعتبارسنجی (20%) و آزمایشی (20%) تقسیم کرده و مدل اولیه FIS را با استفاده از تابع genfis2 ایجاد می‌کنیم.

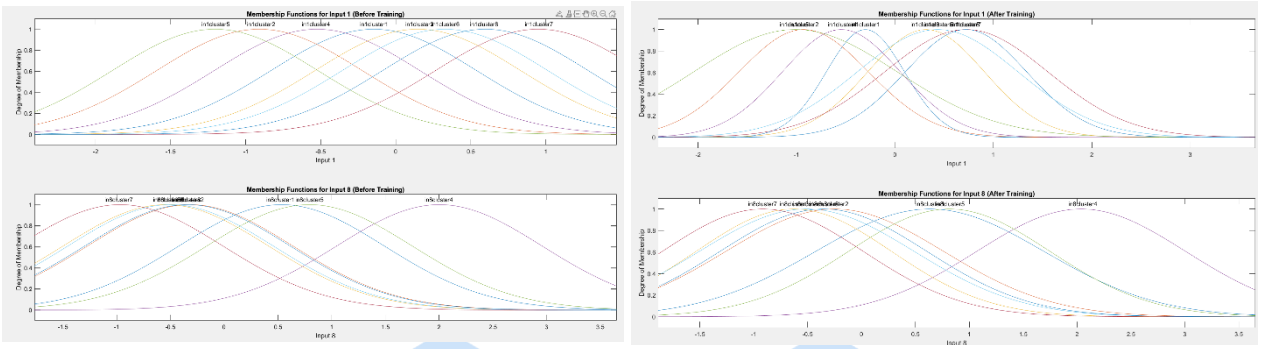
```
radius = 0.5;  % Adjust this parameter as needed
in_fis = genfis2(trainInput, trainOutput, radius);
```

قبل از آموزش مدل، توابع عضویت مربوط به دو ویژگی ورودی 1 و 8 را برای بررسی وضعیت اولیه رسم می‌کنیم. سپس، مدل را با استفاده از داده‌های آموزشی و اعتبارسنجی برای 100 دوره آموزش داده و میزان خطای آموزش و اعتبارسنجی را ذخیره می‌کنیم. پس از آموزش، توابع عضویت را مجدداً برای بررسی تغییرات پس از یادگیری مدل نمایش می‌دهیم. مدل نهایی را برای داده‌های آموزشی، اعتبارسنجی و آزمایشی ارزیابی کرده و مقادیر پیش‌بینی‌شده را از حالت نرمال خارج می‌کنیم. سپس، RMSE و MSE را برای هر مجموعه داده محاسبه کرده و نمایش می‌دهیم.

در نهایت، نمودارهایی برای نمایش روند خطای آموزش و اعتبارسنجی در طول دوران آموزش رسم می‌کنیم. همچنین، خروجی واقعی و پیش‌بینی‌شده را برای مجموعه آزمایشی مقایسه کرده و توزیع خطای مدل را در قالب هیستوگرام نمایش می‌دهیم. این تحلیل‌ها به ما کمک می‌کنند تا عملکرد مدل را ارزیابی کرده و دقت آن را در پیش‌بینی مقدار NO2(GT)بررسی کنیم.







همانطور که مشاهده می شود مدل انفیس نیز مشابه RBF قادر به پیشبینی خروجی با خطای مطلوبی می باشد.

```
Validation Set:
  Denormalized RMSE: 16.3167
  Denormalized MSE: 266.2363
Test Set:
  Denormalized RMSE: 11.1818
  Denormalized MSE: 125.0327
```

اما همانطور که مشاهده می شود خطای مدل RBF از ANFIS کمتر می باشد. شبکه‌های RBF به دلیل استفاده از توابع پایه شعاعی در لایه پنهان، در شناسایی الگوهای محلی در داده‌ها عملکرد بسیار خوبی دارند. این ویژگی به‌ویژه در مدل‌سازی روابط غیرخطی مفید است، زیرا توابع فعال‌سازی گاوسی در لایه پنهان باعث می‌شوند مدل روی نواحی خاصی از فضای ورودی تمرکز کند و به تغییرات پیچیده و موضعی در داده‌ها بهتر پاسخ دهد. در مقابل، ANFIS با ترکیب منطق فازی و شبکه‌های عصبی بیشتر برای مدل‌سازی روابط کلی و سراسری طراحی شده است. این ویژگی در حالتی که داده‌ها دارای غیرخطیت‌های موضعی قوی باشند، ممکن است باعث کاهش کارایی مدل شود. همچنین، شبکه RBF با استفاده از تعداد پارامترهای کمتر و فرآیند آموزشی ساده‌تر، که شامل تعیین مراکز و پهنای توابع پایه و سپس بهینه‌سازی خطی وزن‌های خروجی است، اغلب به همگرایی سریع‌تر و تعمیم بهتر در مجموعه داده‌هایی با الگوهای پیچیده منجر می‌شود.