

**دانشکده مهندسی برق و الکترونیک**

**گزارش کار پروژه درس کنترل فازی**

طراحی سیستم فازی با روش های گرادیان نزولی و حداقل مربعات خطا

**دانشجو:**

نیما جهان بازفرد (400113020)

**استاد درس:**

جناب آقای دکتر مختار شاصادقی

**اردیبهشت 1404**



**فهرست مطالب**

**عنوان** **صفحه**

بخش 1: تعربف تابع اصلی و بازه مورد نظر جهت مدل سازی فازی

[1-1 تعربف تابع اصلی و بازه مورد نظر جهت مدل سازی فازی 2](#_Toc182860334)

[بخش 2: طراحی](#_Toc182860333) سیستم فازی با روش گرادیان نزولی

[1-2 روند طراحی 5](#_Toc182860334)

[2-2 خروجی ها 10](#_Toc182860334)

[بخش 3: طراحی](#_Toc182860346) سیستم فازی با روش حداقل مربعات خطا

[1-3 روند طراحی برای داده های pure با روش RLSو تحلیل خروجی ها 19](#_Toc182860347)

3-2 روند طراحی برای خروجی های آغشته به نویز سفید با روش RLS و تحلیل خروجی ها 25

3-3 روند طراحی برای خروجی های آغشته به نویز رنگی (قهوه ای) با روش RLS و تحلیل خروجی ها 30

3-4 روند طراحی برای خروجی های آغشته به نویز رنگی (قهوه ای) با روش ELS و تحلیل خروجی ها  [36](#_Toc182860347)

# بخش 1

# تعربف تابع اصلی و بازه مورد نظر جهت مدل سازی فازی

## عربف تابع اصلی و بازه مورد نظر جهت مدل سازی فازی

در این پروژه هدف مدلسازی فازی تابع  در بازه [-5 5] می باشد. اما با رسم این تابع در بازه گفته شده در متلب داریم:



شکل1 : تابع  در بازه [-5 5]

کامل طبق شکل 1 قابل مشاهده می باشد که تابع  در دو حد -5 و 5 به سمت بی نهایت میل می کند و این باعث می شود که مدل سازی فازی با روش الگوریتم گرادیان نزولی (به دلیل استفاده از مشتق)غیرممکن و نیازمند سیستم بسیار قوی برای تحلیل داده ها می باشد.( سیستم بنده توانایی تحلیل داده ها برای رسیدن به دقت مطلوب برای بازه های بزرگتر را ندارد چون گام حرکت باید کوچک باشد چندین ساعت باید سیستم در حال پردازش دیتا باشد و سیستم توانایی لازم را جهت انجام اینکار ندارد البته الگوریتم های ELS و RLS مشکلی از جهت بازه بهم نمیکنند.

پس با تغییر بازه به [-0.5 0.5] داریم:



شکل 2: تابع  در بازه [-0.5 0.5]

چون هدف از پروژه مدلسازی یک سیستم غیرخطی می باشد پس تغییر بازه تا زمانی که غیرخطی بودن سیستم همچنان برقرار باشد موردی ندارد و همانگونه که مشخص می باشد سیستم در بازه [-0.5 0.5] کاملا شکل غیرخطی خود را حفظ کرده و هدف پروژه را ارضا می کند.

در ادامه بخش ها به طراحی سیستم فازی f(x) برای تابع g(x)= در بازه [-0.5 0.5] می پردازیم.

# بخش 2 طراحی سیستم فازی با روش گرادیان نزولی

## 1-2 روند طراحی

در روند طراحی سیستم فازی با روش گرادیان نزولی با تعیین خطای مطلوب در ادامه و تولید تعداد قواعد فازی به گونه ای که با انتخاب تابع گوسین به عنوان تابع عضویت شدت های آتش را بر اساس تعیین تعداد قواعد بدست آورده و با بدست آوردن مراکز ثقل بهینه توسط خروجی سیستم فازی را بدست می آوریم.هدف از پیاده سازی این اگوریتم به حداقل رساندن خطا با بروزرسانی پارامتر های متغیری که در ادامه روند طراحی گفته می شود.

هدف ما در کل رسیدن به خروجی مطلوب فازی به ازا زوج ورودی و خروجی می باشد. یعنی حداقل کردن خطا بین خروجی فازی و خروجی سیستم اصلی.

خروجی سیستم فازی به این شکل محاسبه می شود:



در روند طراحی مقدار m ثابت در نظر گرفته می شود و پارامتر های  و  و  متغیر می باشند.  
هدف ما از این طراحی بکار گیری روش کرادیان نزولی برای بدست آوردن پارامتر های متغیر فوق می باشد.

سیستم فازی به این شکل عمل می کند:

ابتدای امر زاویه آتش تولید میشود:



که این زاویه آتش وابسته به پارامتر های متغیر  و  که باید در روند الگوریتم مدام محاسبه شوند.  
در مرحله بعد دو مقدار زیر محاسبه می شوند که خروجی سیستم فازی را باعث می شوند:







هدف ما از این پیاده سازی این می باشد که مقدار کمینه شود.  
روند بروزرسانی پارامتر های متغیر طبق فصل 13 کتاب وانگ در کد پیاده سازی شده است.

در ادامه :

با تغییر مقدارlearning rate و m الگوریتم پیاده سازی می شود:

حالت های مختلف:

1-با خطای مطلوب  و مقدار m=70 و learning rate=0.1 روند طراحی سیستم پیاده سازی می شود.

2- با خطای مطلوب  و مقدار m=100 و learning rate=0.1 روند طراحی سیستم پیاده سازی می شود.

3- با خطای مطلوب و مقدار m=40 و learning rate=0.1 روند طراحی سیستم پیاده سازی می شود.

4- با خطای مطلوب و مقدار m=70 و learning rate=1 روند طراحی سیستم پیاده سازی می شود.

5- با خطای مطلوب و مقدار m=70 و learning rate=0.01 روند طراحی سیستم پیاده سازی می شود.

Learning rate گام حرکت در مسیر بروزرسانی پارامتر ها روی سیستم اصلی می باشد که بین 0 تا 1 تعیین می شود.

function fuzzy\_system = fuzzy\_modeling\_3(input\_data, output\_data, M, alpha, max\_epochs, epsilon)

[num\_samples, n] = size(input\_data);

% Centers of input membership functions

x\_bar = zeros(n, M);

for i = 1:n

x\_bar(i,:) = linspace(min(input\_data(:,i)), max(input\_data(:,i)), M);

end

% Widths of input membership functions

sigma = zeros(n, M);

for i = 1:n

sigma(i,:) = (max(input\_data(:,i)) - min(input\_data(:,i))) / M \* 1.5; % Wider coverage

end

% Centers of output membership functions

y\_bar = linspace(min(output\_data), max(output\_data), M);

% Training loop max\_epoch=q according to wang book

for epoch = 1:max\_epochs

total\_error = 0;

for p = 1:num\_samples

x0 = input\_data(p,:)';

y0 = output\_data(p);

% Calculate firing strengths (z^l)

z = ones(1, M);

for l = 1:M

for i = 1:n

z(l) = z(l) \* exp(-((x0(i) - x\_bar(i,l)) / sigma(i,l))^2);

end

end

% calculating output

b = sum(z);

a = sum(y\_bar .\* z);

f = a / b; % Final output

% Error calculation

e = 0.5 \* (f - y0)^2;

total\_error = total\_error + e;

% Update y\_bar

for l = 1:M

y\_bar(l) = y\_bar(l) - alpha \* (f - y0) \* (1 / b) \* z(l);

end

% Update x\_bar and sigma

for l = 1:M

for i = 1:n

% Update x\_bar

x\_bar(i,l) = x\_bar(i,l) - alpha \* ...

((f - y0) \* (y\_bar(l) - f) \* z(l) / b) \* ...

2\*(x0(i) - x\_bar(i,l)) / (sigma(i,l)^2);

% Update sigma

sigma(i,l) = sigma(i,l) - alpha \*...

((f - y0) \* (y\_bar(l) - f) \* z(l) / b) \* ...

2\*(x0(i) - x\_bar(i,l))^2 / (sigma(i,l)^3);

end

end

end

fprintf('Epoch %d: Error = %.4f\n', epoch, total\_error);

% Early stopping if error is below threshold

if total\_error < epsilon

break;

end

end

% Store final parameters

fuzzy\_system = struct();

fuzzy\_system.M = M;

fuzzy\_system.x\_bar = x\_bar;

fuzzy\_system.sigma = sigma;

fuzzy\_system.y\_bar = y\_bar;

% Evaluation function

fuzzy\_system.evaluate = @(x) evaluate\_fuzzy\_system(x, x\_bar, sigma, y\_bar);

end

function y = evaluate\_fuzzy\_system(x, x\_bar, sigma, y\_bar)

% Evaluate fuzzy system for input x

[n, M] = size(x\_bar);

z = ones(1, M);

% Calculate firing strengths

for l = 1:M

for i = 1:n

z(l) = z(l) \* exp(-((x(i) - x\_bar(i,l)) / sigma(i,l))^2);

end

end

% Compute output

b = sum(z);

a = sum(y\_bar .\* z);

y = a / b;

end

clc

clear all

close all

%enter input threshold

xmin=input('enter down bound of input:');

xmax=input('enter up bound of input:');

% Generate sample data for y = 10x^4\*cosh(x)

x = linspace(xmin, xmax, 100)';

y = (10.\*(x.^4).\*((exp(x)+exp(-1.\*x))./2)) ;

% Training parameters

M = input('enter mont of fuzzy membership functions:');

alpha = input('enter learning rate:');

max\_epochs = input('enter mont of epochs:');

epsilon = input('enter desirable precision:');

% Train fuzzy system

fuzzy\_sys = fuzzy\_modeling\_3(x, y, M, alpha, max\_epochs, epsilon);

% Test the system

y\_pred = arrayfun(@(xi) fuzzy\_sys.evaluate(xi), x);

% Plot results

figure,plot(x, y, 'b-', 'LineWidth', 1.5, 'DisplayName', 'Actual Function');

grid on;

figure,plot(x, y\_pred, 'r--', 'LineWidth', 2, 'DisplayName', 'Fuzzy Approximation');

grid on;

title('Fuzzy System Approximation of y = 10x^4 cosh(x)');

figure,plot(x, y\_pred, 'r--',x, y, 'b-');

grid on;

xlabel('x');

ylabel('y');

legend();

figure,plot(x, y-y\_pred, 'g', 'LineWidth', 2, 'DisplayName', 'error');

grid on;

title(‘error’);

باید توجه کرد که max\_epoch تعریف شده در کد مفهوم زمانی دارد و چون در شبیه سازی با مفهوم زمان ارتباطی گرفته نمی شود آن را با epoch مدل می کنند. در روند شبیه سازی مقدار max\_epochs برابر با 20000 گرفته می شود.

در زیر بخش بعد به نمایش خروجی ها و تحلیل آنها می پردازیم.

## 2-2 خروجی ها

**حالت 1: در epoch = 2676 توانسته تابع هزینه را به مقدار 0.001 برساند**

خروجی سیستم فازی و سیستم اصلی برای حالت 1:



خروجی سیستم فازی:



خطای بین سیستم اصلی و سیستم فازی حالت 1:



**حالت 2: در epoch = 826 توانسته تابع هزینه را به مقدار 0.001 برساند**

خروجی سیستم فازی و سیستم اصلی برای حالت 2:



خطای بین سیستم اصلی و سیستم فازی حالت 2:



خروجی سیستم فازی حالت 2:



**حالت سوم: در epoch = 1840 توانسته تابع هزینه را به مقدار 0.001 برساند**

خروجی سیستم فازی و سیستم اصلی برای حالت 3:



خطای بین سیستم اصلی و سیستم فازی حالت 3:



خروجی سیستم فازی حالت 3:

****

**حالت 4: در این حالت بدلیل بزرگ بودن learning rate=1 نتوانست سیستم فازی را به درستی طراحی کند.**

خروجی سیستم فازی و سیستم اصلی برای حالت 4:



خطای بین سیستم اصلی و سیستم فازی حالت 4:



خروجی سیستم فازی حالت 4:



**حالت 5: در epoch = 1594 توانسته تابع هزینه را به مقدار 0.001 برساند**

خروجی سیستم فازی و سیستم اصلی برای حالت 5:



خطای بین سیستم اصلی و سیستم فازی حالت 5:



خروجی سیستم فازی حالت 5:



**تحلیل:**

در حالت های مختلف با مشخص می باشد مدل سازی که قرار است انجام بدهیم بر مقدار learning rate بسیار حساس می باشد و بزرگ تعیین کردن learning rate باعث مدل سازی اشتباه می شود اما قابل مشاهده است که در این سیستم غیر خطی هرچه مقدار learning rate کوچکتر بوده سیستم مدل شده فازی خطای بسیار کمتری داشته و تعداد توابع عضویت نیز در حد خود نیز تاثیر گذار بوده و هرچه مقدار آن بیشتر بوده در ابتدای امر خطای کمتری داشته ولی در گذر زمان خطا بیشتر می شود پس برای دست یابی به خطای کم در ابتدای مسیر و در گذر زمان باید trade off انجام داد و با آزمون و خطا مقدار آن را تعیین کرد .البته با الگوریتم های بهینه سازی نیز به این امر می توان دست یافت.(مانند الگوریتم ژنتیک)

# بخش 3

# طراحی سیستم فازی با روش حداقل مربعات خطا

## 1-3 روند طراحی برای داده های pure با روش RLS و تحلیل خروجی ها

در روند طراحی سیستم فازی با روش RLS به این گونه پیش می رویم که توابع عضویت ما گوسی می باشد . در روند طراحی باید ابتدای کار بردار رگرسور تشکیل داد که بردار رگرسور به شکل زیر می باشد:



در ادامه کار برای حل معادله زیر که خروجی مدل فازی می باشد بایذ مقدار اولیه ی  تعیین کرد که همان مراکز ٍثقل ما می باشند که با ضرب در بردار رگرسور ما خروجی فازی را برای ما تولید می کند. به ازای هر ورودی بروزرسانی شده تا اینکه مراکز ثقل ما به مقدار نهایی خود همگرا شده.

همانطور که مشخص است سیستم ما سیستم استاتیکی می باشد و بردار رگرسور در ابتدای امر شامل مقادیر خود می باشد. چالش اصلی ما بروزرسانی پارامترها  می باشد که به مقدار نهایی خود همگرا شوند.

تعیین خروجی:



مراحل بروزرسانی :



کد برای سیستم در بازه [-1 1] پیاده سازی می شود و مقدار اولیه ماتریس P برابر با  و مقدار اولیه مرکز ثقل ها پارامتر اول برابر 1 و بقیه پارامتر ها یزلبز صفر گرفته شده و تعداد داده های ورودی 10000 و تعداد قواعد هم 30 تا تعیین شده است.

clc

clear all

close all

% User Inputs

min\_input = input('Enter lower bound of input: ');

max\_input = input('Enter upper bound of input: ');

num\_rules = input('Enter number of rules: ');

learning\_rate = input('Enter learning rate for P matrix: ');

num\_samples = input('Enter number of input samples: ');

% Initialize Fuzzy System

centers = linspace(min\_input, max\_input, num\_rules);

width = (max\_input - min\_input) / num\_rules;

theta = eye(num\_rules, 1);

P\_matrix = learning\_rate \* eye(num\_rules);

% Storage

predictions = [];

errors = [];

true\_outputs = [];

inputs = linspace(min\_input, max\_input, num\_samples)';

trace\_p = [];

mean\_errors = [];

theta\_history = zeros(num\_samples, num\_rules);

for i = 1:num\_samples

x = inputs(i);

y = 10 \* (x^4) \* cosh(x);

true\_outputs = [true\_outputs; y];

% making regressor vector with fuzzy rules

mu = exp(-((x - centers)/width).^2)';

b = mu / sum(mu);

% Fuzzy system prediction

y\_pred = b' \* theta;

% Weight Update

K = P\_matrix \* b / (1 + b' \* P\_matrix \* b);

theta = theta + K \* (y - y\_pred);

P\_matrix = P\_matrix - K \* b' \* P\_matrix;

trace\_p = [trace\_p; trace(P\_matrix)];

% Store all theta values

theta\_history(i, :) = theta';

% Store results

predictions = [predictions; y\_pred];

errors = [errors; (y - y\_pred)];

current\_mean\_error = mean(errors(1:i));

mean\_errors = [mean\_errors; current\_mean\_error];

% Display progress

fprintf('Step %d: x=%.4f, y\_true=%.4f, y\_pred=%.4f, Mean Error=%.4f\n', i, x, y, y\_pred, current\_mean\_error);

end

% plots

% main System and fuzzy modeling system

figure;

plot(inputs, true\_outputs, inputs, predictions, 'r--');

legend('True','Fuzzy');

title('main System and fuzzy modeling system');

grid on;

% Error between true output and prediction

figure;

plot(true\_outputs - predictions);

title('Error between True Output and Prediction');

grid on;

% Trace of P matrix

figure;

plot(trace\_p);

title('Trace of P Matrix');

grid on;

% Plot all theta weights

figure;

hold on;

for k = 1:num\_rules

plot(theta\_history(:, k), 'DisplayName', sprintf('theta\_%d', k));

end

hold off;

title('Evolution of All Theta Weights');

xlabel('Step');

ylabel('Theta Value');

legend('show');

grid on;

% Mean Error plot

figure;

plot(mean\_errors, 'LineWidth', 1.5);

title('Mean Absolute Error (True Output vs Prediction)');

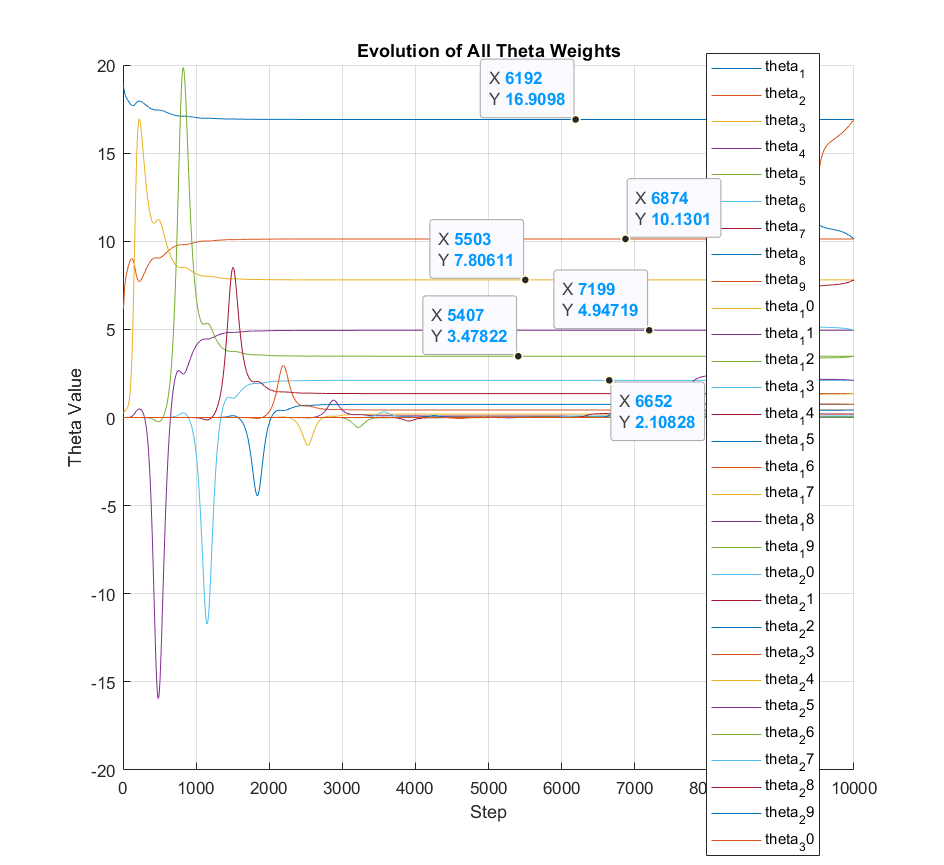
xlabel('Step');

ylabel('Mean Error');

grid on;

**خروجی ها:**

همگرایی پرامتر های بردار :



همگرایی میانگین خطا در طول فرآیند الگوریتم:



خطای بین سیستم اصلی و سیستم فازی:



سیستم اصلی و سیستم فازی:



همگرایی Trace ماتریس P :



**تحلیل:**

با توجه به trace ماتریس p کاملا مشخص است که به همگرایی لازم رسیدیم و مقدار آن تقریبا به صفر میل پیدا کرده پس می توان از روی پارامتر های تخمینی هم فهمید که به مقدار نهایی خور همگرا شده اند و این مورد ارتباط مستقیمی با همگرایی ماتریس P دارد. از روخی مقایسه ی خروجی اصلی و خروجی سیستم فازی کاملا قابل مشاهده می باشد به دلیل تعریف شرایط اولیه 1 برای تمام پارامتر های بردار  خطای بزرگی داریم ولی رفته رفته با بروز رسانی پارامتر ها و همگرایی آنها خطای میان سیستم اصلی و سیستم فازی کاهش یافته و به نزدیک شدن به ورودی صفر خطا ی میان این دو تقریبا صفر می باشد ولی به دلیل تپولوژی سیستم زمانی که از 0 به سمت 1 حرکت میکنیم خطا کمی زیاد می شود ولی در حدود 0.001 و دلیل آن تغیر سریع و ناگهانی و حالت صعودی سیستم اصلی می باشد ولی همچنان الگوریتم به خوبی عمل کرده و خطای بسیار کوچکی داریم و همچنین از روی همگرایی میانگین خطا که در حدود 0.004 می باشد نشان دهنده عملکرد فوق العاده در تخمین پارامتر ها و unbiased بودت تخمین پارامترها می باشد.

## 3-2 روند طراحی برای خروجی های آغشته به نویز سفید با روش RLS و تحلیل خروجی ها

در این زیر بخش خروجی هارو آغشته به نویز سفید با standard deviation =0.1 و تعیین سایر مقادیر همانند زیر بخش 1 داریم:

کد مربوطه:

clc

clear all

close all

% Input the parameters

min\_input = input('Enter lower bound of input: ');

max\_input = input('Enter upper bound of input: ');

num\_rules = input('Enter number of rules: ');

learning\_rate = input('Enter learning rate for P matrix: ');

num\_samples = input('Enter number of input samples: ');

noise\_level = input('Enter noise level (standard deviation): ');

% random number for fixing noise

rng(42);

% Initializing

centers = linspace(min\_input, max\_input, num\_rules);

width = (max\_input - min\_input) / num\_rules;

theta = eye(num\_rules, 1);

P\_matrix = learning\_rate \* eye(num\_rules);

%Storage of Results

predictions = [];

true\_outputs = [];

noisy\_outputs = [];

inputs = linspace(min\_input, max\_input, num\_samples)';

trace\_p = [];

mean\_errors = [];

theta\_history = zeros(num\_samples, num\_rules);

% Generate noise

noise = noise\_level \* randn(num\_samples, 1);

noise= noise(randperm(length(noise)));

% Training with Sequential Data

for i = 1:num\_samples

% making data

x = inputs(i);

y = 10 \* (x^4) \* cosh(x);

y\_noisy = y + noise(i);

true\_outputs = [true\_outputs; y];

noisy\_outputs = [noisy\_outputs; y\_noisy];

% making regressor vector with fuzzy rules

mu = exp(-((x - centers)/width).^2)';

b = mu/sum(mu);

% Calculate fuzzy output

y\_pred = b' \* theta;

% Update weights

K = P\_matrix \* b / (1 + b' \* P\_matrix \* b);

theta = theta + K \* (y\_noisy - y\_pred);

P\_matrix = P\_matrix - K \* b' \* P\_matrix;

trace\_p = [trace\_p; trace(P\_matrix)];

% Store all theta values

theta\_history(i, :) = theta';

% Store results

predictions = [predictions; y\_pred];

current\_error = mean((true\_outputs(1:i) - predictions(1:i)));

mean\_errors = [mean\_errors; current\_error];

% Display progress

fprintf('Step %d: x=%.4f, y\_true=%.4f, y\_noisy=%.4f, y\_pred=%.4f\n , mean\_errors=%.4f\n', i, x, y, y\_noisy, y\_pred, current\_error);

end

% plots

% main System and fuzzy modeling system

figure;

plot(inputs, true\_outputs, inputs, predictions, 'r--');

legend('True','Fuzzy');

title('main System and fuzzy modeling system');

grid on;

% Error between true output and prediction

figure;

plot(true\_outputs - predictions);

title('Error between True Output and Prediction');

grid on;

% Trace of P matrix

figure;

plot(trace\_p);

title('Trace of P Matrix');

grid on;

% Plot all theta

figure;

hold on;

for k = 1:num\_rules

plot(theta\_history(:, k), 'DisplayName', sprintf('theta\_%d', k));

end

hold off;

title('Evolution of All Theta Weights');

xlabel('Step');

ylabel('Theta Value');

legend('show');

grid on;

% Mean Error plot

figure;

plot(mean\_errors, 'LineWidth', 1.5);

title('Mean Error (True Output vs Prediction)');

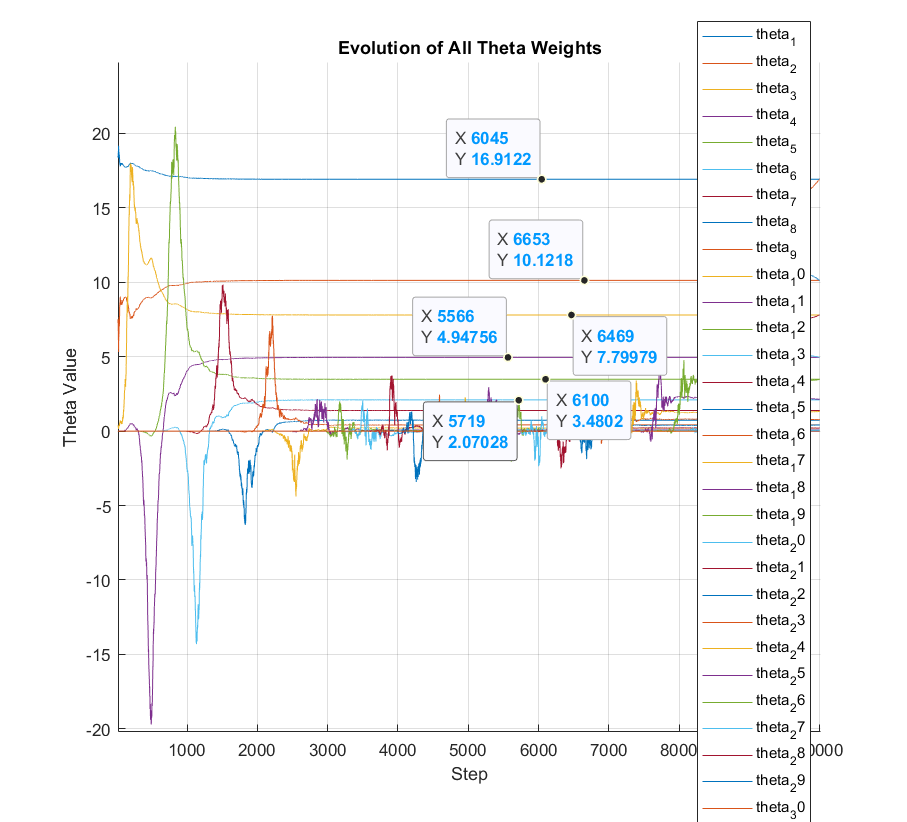
xlabel('Step');

ylabel('Mean Error');

grid on;

**خروجی ها:**

همگرایی پرامتر های بردار :



همگرایی میانگین خطا در طول فرآیند الگوریتم:



خطای بین سیستم اصلی و سیستم فازی:



سیستم اصلی و سیستم فازی:



همگرایی Trace ماتریس P :



**تحلیل:**

همانگونه که از روی trace ماتریس P مشخص می باشد بازهم هماننده زیربخش 1 trace ماتریس P به صفر میل پیدا کرده و همگرا شده است و از اینرو تخمین پارامترها در ابتدای امر نویز روی آن تاثیر اندکی گذاشته است ولی در نهایت همه ی پارامترها ی  با اختلاف بسیار اندکی در حد 0.01 یا حتی کمتر نسبت به زیر بخش 1 به همکرایی رسیده اند و نشان دهنده unbiased بودن پارامتر های تخمین زده شده می باشد و این نشان دهنده ی این است که الگوریتم RLS نسبت به داده هایی نویزی البته نویز سفید مقاوم بوده و توانایی خود برایی تخمین پارامتر هارا از دست نمیدهد و این مقاومت نیز از روی مینگین خطای خروجی نیز مشخص بوده که تقریبا برابر با میانگین خطای همگرا شده در زیر بخش 1 می باشد. و از روی خطای میان سیستم اصلی و سیستم مدل سشده فازی نیز می توان متجه این شد که نویز اعمالی باعث افزایش خطا میام این دو شده نسبت به زیربخش 1 ولی این زیاد شدن مقدار بزرگی ندارد و دلیل آن وجود نویز سفید می باشد و امری منطقی می باشد.

## 3-3 روند طراحی برای خروجی های آغشته به نویز رنگی (قهوه ای) با روش RLS و تحلیل خروجی ها

در ادامه در این زیر بخش نویز رنگی(قهوه ای) به خروجی سیستم اصلی افزوده شده با standard deviation=0.1 و سایر مقادیر هم مانند دو زیر بخش قبل می باشد.

در ادامه با کد مربوطه داریم:

clc

clear all

close all

% Input the parameters

min\_input = input('Enter lower bound of input: ');

max\_input = input('Enter upper bound of input: ');

num\_rules = input('Enter number of membership functions: ');

learning\_rate = input('Enter learning rate for P matrix: ');

num\_samples = input('Enter number of input samples: ');

noise\_level = input('Enter noise level (standard deviation): ');

rng(42);

% Initializing

centers = linspace(min\_input, max\_input, num\_rules);

width = (max\_input - min\_input) / num\_rules;

theta = eye(num\_rules, 1);

P\_matrix = learning\_rate \* eye(num\_rules);

%Storage of Results

predictions = [];

errors = [];

true\_outputs = [];

noisy\_outputs = [];

inputs = linspace(min\_input, max\_input, num\_samples)';

trace\_p = [];

mean\_errors = [];

theta\_history = zeros(num\_samples, num\_rules);

% Generate colored noise-brown

bnoise = zeros(num\_samples, 1);

white\_noise =noise\_level\* randn(num\_samples,1);

bnoise=cumsum(white\_noise);

bnoise=bnoise/max(abs(bnoise));

bnoise=bnoise(randperm(length(bnoise)));

for i = 1:num\_samples

% making data

x = inputs(i);

y = 10 \* (x^4) \* cosh(x);

y\_noisy = y + bnoise(i);

true\_outputs = [true\_outputs; y];

noisy\_outputs = [noisy\_outputs; y\_noisy];

% making regressor vector with fuzzy rules

mu = exp(-((x - centers)/width).^2)';

b = mu/sum(mu);

% Calculate fuzzy output

y\_pred = b' \* theta;

% Update weights

K = P\_matrix \* b / (1 + b' \* P\_matrix \* b);

theta = theta + K \* (y\_noisy - y\_pred);

P\_matrix = P\_matrix - K \* b' \* P\_matrix;

trace\_p = [trace\_p; trace(P\_matrix)];

% Store all theta values

theta\_history(i, :) = theta';

% Store results

predictions = [predictions; y\_pred];

current\_error = mean((true\_outputs(1:i) - predictions(1:i)));

mean\_errors = [mean\_errors; current\_error];

% Display progress

fprintf('Step %d: x=%.4f, y\_true=%.4f, y\_noisy=%.4f, y\_pred=%.4f\n , mean\_errors=%.4f\n', i, x, y, y\_noisy, y\_pred,current\_error);

end

% plots

% main System and fuzzy modeling system

figure;

plot(inputs, true\_outputs, 'b', inputs, predictions, 'r--');

legend('True','Fuzzy');

title('main System and fuzzy modeling system');

grid on;

% Error between true output and prediction

figure;

plot(true\_outputs - predictions);

title('Error between True Output and Prediction');

grid on;

% Trace of P matrix

figure;

plot(trace\_p);

title('Trace of P Matrix');

grid on;

% Plot power spectral density of noise

figure;

pwelch(bnoise, [], [], [], 1);

title('Power Spectral Density of Colored Noise');

grid on;

% Plot all theta weights

figure;

hold on;

for k = 1:num\_rules

plot(theta\_history(:, k), 'DisplayName', sprintf('theta\_%d', k));

end

hold off;

title('Evolution of All Theta Weights');

xlabel('Step');

ylabel('Theta Value');

legend('show');

grid on;

% mean error plot

figure;

plot(mean\_errors, 'LineWidth', 1.5);

title('Mean Error (True Output vs Prediction)');

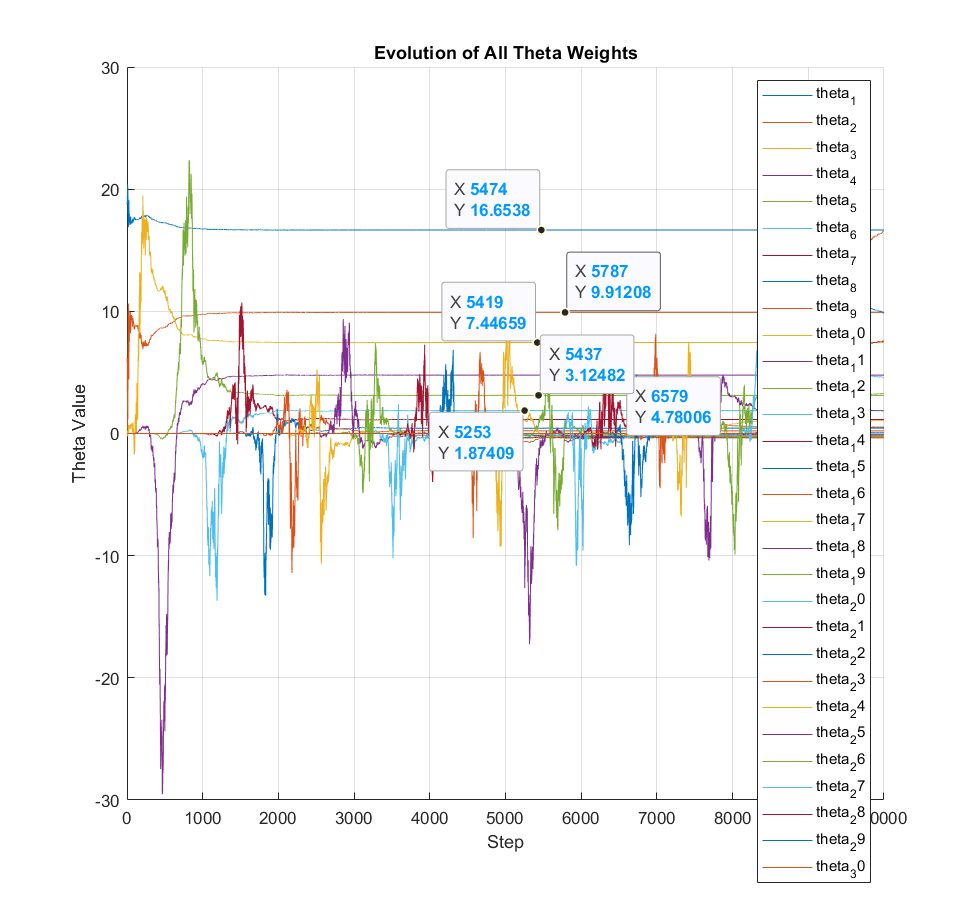
xlabel('Time Step');

ylabel('Mean Error');

grid on;

**خروجی ها:**

همگرایی پرامتر های بردار :



همگرایی میانگین خطا در طول فرآیند الگوریتم:



خطای بین سیستم اصلی و سیستم فازی:



سیستم اصلی و سیستم فازی:



همگرایی Trace ماتریس P :



Power spectral نویز رنگی:



**تحلیل:**

با توجه به نمودار همگرایی پارامترهای بردار  کاملا مشخص است که در ابتدای امر نویز رنگی تاثیر تقریبا زیادی روی تخمین پارامترها گذاشته است و در ادامه همگرا شده است اما این همگرایی با توجه به زیر بخش 1 همگرایی با bias می باشد یا می توان گفت که تخمین پارامتر ها biased می باشد و همچنین از روی ارتباط مستقیم میان همگرایی پارامتر ها و همگرایی trace ماتریس p به صفر نیز پی برد و از روی نمودار آن کاملا مشهود است که trace ماتریس P تقریبا به صفر میل پیدا کرده است. این bias بوجود آمده در تخمین پارامتر ها موجب ایجاد bias در نمودار همگرایی میانگین خطا به صفر هم می شود و کاملا مشخص است که ئر نمودار میانگین خطا نیز bias وجود دارد و این bias حدود 0.27 که این ععمر نشان دهنده وجود خطای مدل سازی فازی می باشد از روی نمودار خطای میان سیستم اصلی و همچنین نمایش دو سیستم روی هم قابل مشاهده می باشد که نشان دهنده عملکرد نسبتا نامقاوم الگوریتم RLS نسبت به وجود نویز رنگی می باشد. که در ادامه با الگوریتم ELS این امر را بهبود میبخشیم و الگوریتمی مقاوم تر نسبت به RLS طراحی میکنیم.

## 4-3 روند طراحی برای خروجی های آغشته به نویز رنگی (قهوه ای) با روش ELS و تحلیل خروجی ها

این الگوریتم تفاوت چندانی با الگوریتم RLS نداشته و عمل بروزرسانی آن همانند RLS می باشد . تنها تفاوت این الگوریتم بردار رگرسور آن می باشد که علی رغم وجود خود رگرسور b باید ترم خطا نیز به آن اضافه شود تا بتوان دینامیک نویز رنگی را در بردار  تخمین زد و به توان به گونه ای با آن مقابله کرد.در ادامه با Extend یا گسترش بردار رگرسور داریم:



و  جدید هم به شکل زیر در می آید که ترم تخمین دینامیک نویز رنگی به آن افزوده شده است:



خروجی سیستم فازی ما اینگونه تعیین میشود:



م همنطور که گفته شد بروزرسانی بردار تخمینی جدید هم مانند RLS صورت میگیرد:



در ادامه روند پیاده سازی این الگوریتم روی این سیستم ابتدا برای 3 داده اول خطا ها تعیین شده و این 10 داده در بردار رگرسور جدید قرار میگیرد و برای سایر داده ها با استفاده از این بردار رگرسور جدید الگوریتم ELS پیاده سازی می شود و کار دیگه ای در پیاده سازی انجام شده این است که به دلیل ترم خطا در بردار رگرسور مقدار اولیه ماتریس P باید تقریبا خیلی کوچکتر نسبت به زیربخش های قبلی تعیین شود که در اینجا ماتریس P را به دوبخش تقسیم کرده که بخش اول مربوط به بردار رگرسور  بوده و مقدار و بخش دوم که مربوط به ترم خطا بوده را  تعیین کرده که این مقادیر با آزمون خطاهای متعدد بدست آمده و البته با الگوریتم های بهینه سلزی متنوعی نیز قابل پیاده سازی می باشد. و در ادامه ماتریس P اصلی که تلفیقی از این دو ماتریس در روند پیاده سازی می باشد بدست آورده می شود که با دستور  قابل پیاده سازی می باشد.در الگوریتم ElS تعیین مقدار اولیه برای بردار  بسیار اهمیت دارد . در ادامه با تعیین شرایط اولیه 10 برای پارامترهای بخش اولیه که مربوط به  و 0 مربوط به بخش خطا و سپس این دو بردار را تلفیق کرده و بردار  جدید تشکیل می شود.

در ادامه با تعیین تعداد قواعد 30 و standard deviation=0.1و تعداد داده ها برابر با 10000 و همانطور که گفته شد تعداد خطای مورد استفاده در بردار رگرسور هم برابر با 3 می باشد.که در ادامه کد مربوطه آورده می شود و در انتها خروجی ها و تحلیل آنها آورده می شود:

کد مربوطه:

clc;

close all;

clear all;

% User-defined parameters

min\_input = input('Enter lower bound of input: ');

max\_input = input('Enter upper bound of input: ');

num\_rules = input('Enter number of membership functions: ');

learning\_rate\_fuzzy = input('Enter learning rate for fuzzy part of P matrix: ');

learning\_rate\_error = input('Enter learning rate for error part of P matrix: ');

num\_samples = input('Enter number of input samples: ');

noise\_level = input('Enter noise level (standard deviation): ');

rng(42);

% Fuzzy system parameters

centers = linspace(min\_input, max\_input, num\_rules);

width = (max\_input - min\_input) / num\_rules;

max\_error\_order = input('max\_error\_order: ');

% Initializing

theta =10\*ones(num\_rules, 1);

theta\_e=zeros(max\_error\_order,1);

theta=[theta;theta\_e];

% Create P\_matrix with different learning rates for fuzzy and error parts

P\_fuzzy = learning\_rate\_fuzzy \* eye(num\_rules);

P\_error = learning\_rate\_error \* eye(max\_error\_order);

P\_matrix = blkdiag(P\_fuzzy, P\_error);

%storage

predictions = zeros(num\_samples, 1);

true\_outputs = zeros(num\_samples, 1);

noisy\_outputs = zeros(num\_samples, 1);

inputs = linspace(min\_input, max\_input, num\_samples)';

trace\_p = zeros(num\_samples, 1);

mean\_errors = [];

theta\_history = zeros(num\_samples, num\_rules + max\_error\_order);

% Generate colored noise-brown

white\_noise = noise\_level \* randn(num\_samples, 1);

bnoise = cumsum(white\_noise);

bnoise = bnoise / max(abs(bnoise));

bnoise = bnoise(randperm(length(bnoise)));

% Error memory

error\_memory = zeros(max\_error\_order, 1);

% making errors

for i = 1:max\_error\_order

x = inputs(i);

y = 10 \* (x^4) \* cosh(x);

y\_noisy = y + bnoise(i);

true\_outputs(i) = y;

noisy\_outputs(i) = y\_noisy;

% making regressor vector with fuzzy rules

mu = exp(-((x - centers)/width).^2)';

b = mu / sum(mu);

% Regressor only contains membership vector

b = [b; zeros(max\_error\_order, 1)];

% Predict output

y\_pred = b' \* theta;

e = y\_noisy - y\_pred;

% Update parameters using ELS

K = (P\_matrix \* b) / (1 + b' \* P\_matrix \* b);

theta = theta + K \* e;

P\_matrix = P\_matrix - K \* b' \* P\_matrix;

% Save error into memory

error\_memory(i) = e;

% Store results

predictions(i) = y\_pred;

theta\_history(i, :) = theta';

trace\_p(i) = trace(P\_matrix);

mean\_errors = [mean\_errors; mean(true\_outputs(1:i) - predictions(1:i))];

fprintf('Init Step %d: x=%.4f, y=%.4f, y\_noisy=%.4f, y\_pred=%.4f, mean\_error=%.4f\n', ...

i, x, y, y\_noisy, y\_pred, mean\_errors(end));

end

for i = 1+max\_error\_order:num\_samples

x = inputs(i);

y = 10 \* (x^4) \* cosh(x);

y\_noisy = y + bnoise(i);

true\_outputs(i) = y;

noisy\_outputs(i) = y\_noisy;

% Compute fuzzy rule activations

mu = exp(-((x - centers)/width).^2)';

b = mu / sum(mu);

% Regressor

b = [b; -1\*error\_memory];

% fuzzy output

y\_pred = b' \* theta;

e = y\_noisy - y\_pred;

% Update parameters

K = (P\_matrix \* b) / (1 + b' \* P\_matrix \* b);

theta = theta + K \* e;

P\_matrix = P\_matrix - K \* b' \* P\_matrix;

% Update error memory (shift and add new error)

error\_memory = [e; error\_memory(1:end-1)];

% Store results

predictions(i) = y\_pred;

theta\_history(i, :) = theta';

trace\_p(i) = trace(P\_matrix);

mean\_errors = [mean\_errors; mean(true\_outputs(1:i) - predictions(1:i))];

fprintf('Train Step %d: x=%.4f, y=%.4f, y\_noisy=%.4f, y\_pred=%.4f, mean\_error=%.4f\n', ...

i, x, y, y\_noisy, y\_pred, mean\_errors(end));

end

% Plot

% main System and fuzzy modeling system

figure;

plot(inputs, true\_outputs, 'b', inputs, predictions, 'r--');

legend('True Output', 'Fuzzy Prediction');

title('main System and fuzzy modeling system');

grid on;

% Error between true output and prediction

figure;

plot(true\_outputs - predictions);

title('Prediction Error (True - Estimated)');

grid on;

%Trace p matrix

figure;

plot(trace\_p);

title('Trace of P Matrix (Uncertainty)');

grid on;

% Plot power spectral density of noise

figure;

pwelch(bnoise, [], [], [], 1);

title('Power Spectral Density of Colored Noise');

grid on;

% Plot all theta weights

figure;

hold on;

for k = 1:(num\_rules + max\_error\_order)

plot(theta\_history(:, k), 'DisplayName', sprintf('theta\_%d', k));

end

hold off;

title('Evolution of Parameters (Theta)');

xlabel('Step');

ylabel('Theta Value');

legend('show');

grid on;

% mean error plot

figure;

plot(mean\_errors, 'LineWidth', 1.5);

title('Mean Error over Time');

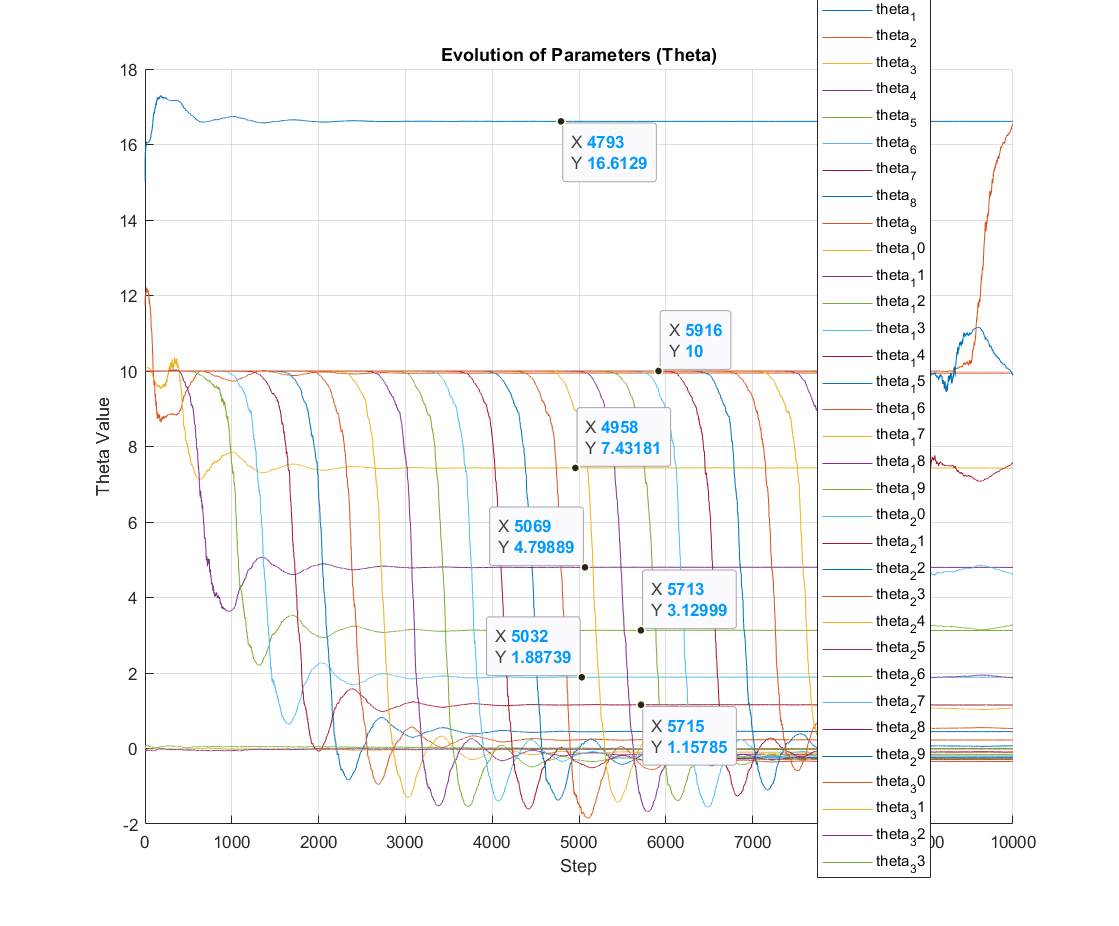
xlabel('Step');

ylabel('Mean Error');

grid on;

**خروجی ها:**

همگرایی پرامتر های بردار :



همگرایی میانگین خطا در طول فرآیند الگوریتم:



خطای بین سیستم اصلی و سیستم فازی:



سیستم اصلی و سیستم فازی:



همگرایی Trace ماتریس P :



Power spectral نویز رنگی:



**تحلیل:**همانطور که مشخص می باشد trace ماتریس p به همگرایی رسیده و تقریبا به صفر میل پیدا کرده است همانطور که مشخص می باشد پرامتر ها هم متقابلا به همگرایی رسیده اند و همانطور که مشخص است همچنان  تا  که مربوط به مرکز های ثقل می باشند نسبت به زیربخش 1 دارای biase می باشند و این بایاس به دلیل وجود ترم نویز رنگی می باشد و همانطور که مشخص است  تا  که در تلاش برای تخمین دینامیک نویز رنگی و کم کردن اثر آن می باشند تقریبا به خوبی عمل کرده و از روی میانگین خطا که به 0.08 میل کرده می توان به عملکرد بسیار خوب این الگوریتم پی برد زیرا در زیربخش قبل مشاهه شد که مینگین خطا به سمت 0.27 همگرا شده است و همچنین از روی مقایسه ی سیستم اصلی و سیستم فازی مدل شده کاملا مشخص است که این الگوریتم عملکرد به شدت هتری نسبت به زیربخش قبل داشته است.