به نام خدا

توجه: شبیه سازی این پروژه با Matlab R2012a انجام شده. پس اگر مشکلی در اجرای برنامه داشتید اول از صیح بودن ورژن Matlab خود اطمینان حاصل کنید.

1 - 1 - مقدمه

در دنیای امروز روباتها به بخش مهمی در صنعت تبدیل شده‌اند. بنابراین کنترل روبات‌ها به یک موضوع مهم و اساسی تبدیل شده‌است.

کنترل کننده PID به عنوان محبوبترین کنترل کننده به علت سادگی ساختار و طراحی شناخته می‌شود. 90% کاربردهای کنترلی به یکی از فرمهای PID مانند PD و PI محدود می‌شود. روش‌های مختلفی برای تنظیم پارامترهای pid وجود دارد. مانند روش زیگلر نیکولز یا کوهن کُن. می‌توان از منطق فازی برای کنترل با نظارت PID استفاده کرد.

در روشهای سنتی طراحی PID پارامترهای کنترل کننده در طول زمان ثابت هستند و تغییر نمی‌کنند. در مواجه با سیستمهای غیر خطی و پیچیده ثابت بودن پارامترهای کنترل کننده موجب غیر مطلوب شدن پاسخ سیستم می‌شود. در اینجا از قوانینی که توسط یک فرد خبره در قالب یک سیستم فازی ارائه می‌شود برای تغییر دادن پارامترها در طول زمان استفاده می‌شود.

در این پروژه از کنترل کننده فازی با نظارت در کنترل روبات با پنج درجه آزادی استفاده می‌شود.

1 - 2 - کنترل کننده PID

تابع تبدیل این کنترل کننده به یکی از دو صورت زیر بیان می‌شود:

$$G_{PID}(s) = K_P + K_I/s + K_D s \qquad (1)$$

$$G_{PID}(s) = K_P(1 + 1/(T_I s) + T_D s) \qquad (2)$$

که در آن KP و KI و KD بهره‌های تناسبی و انتگرالی و مشتقی هستند و TI و TD ثابت‌های زمانی انتگرالی و مشتقی هستند.

$$T_I = K_P/K_I \quad , \quad \text{and} \quad T_D = K_D/K_P$$

قواعد سر انگشتی برای تنظیم پارامترهای PiD وجود دارد. از جمله:

1. اگر ورودی بزرگ و مثبت است، KP باید بزرگ، KI کوچک و KD نیز کوچک انتخاب می‌شود تا خروجی سرعت داده شود.

2. اگر ورودی خیلی کوچک باشد، KP باید کوچکتر، Ki بزرگتر و KD بزرگتر شودتا فراجهش پاسخ کمتر و پاسخ سریعتر شود.

1 - 3 - کنترل کننده منطق فازی

این کنترل کننده چهار بخش عمده دارد: فازی ساز، پایگاه قواعد، موتور استنتاج فازی و غیرفازی ساز. فازی ساز ورودی واقعی را به ورودی فازی سازی شده تبدیل می‌کند. پایگاه قواعد حاوی قوانینی است که در تولید خروجی باید در نظر گرفته شوند. موتور استنتج از روی خروجی قوانین تصمیم می‌گیرد که مقدار خروجی چقدر باشد. غیرفازی ساز خروجی موتور استنتاج را به سیگنال کنترلی تبدیل می‌کند.

کنترل کننده های فازی-PID به دو دسته تقسیم می‌شوند. در دسته اول کنترل کننده فازی جایPID را می‌گیرد و سیگنال کنترلی مستقیما از روی قواعد نتیجه می‌شود. در روش دوم که روش با نظارت است از سیستم فازی برای تنظیم پارامترهای PID استفاده می‌شود.

مراحل طراحی کنترل کننده فازی به صورت زیر است:

1- متغیرهای ورودی-خروجی کنترل کننده فازی را تعیین کنید. در این پروژه ورودیهای سیستم فازی $e(t)$ و $\Delta e(t)$ و خروجیهای سیستم KP و KI و KD هستد.

2- برای هر متغیر ورودی هفت تابع عضویت در نظر می‌گیریم که از منفی بزرگ (NB) تا مثبت بزرگ (PB) می‌باشد. خروجیهای KP و KD دو تابع عضویت و متغیر KI دارای یک تابع عضویت می‌باشد.

3- در این پروژه از سیستم فازی ممدانی (مینیمم ماکزیمم) برای موتور استنتاج استفاده می‌شود.

4- غیرفازی سازی در این پروژه با روش مرکز ثقل با رابطه زیر انجام می‌شود:

$$u = \frac{\sum_{i=1}^{m} \mu(x_i).x_i}{\sum_{i=1}^{m} \mu(x_i)} \qquad (3)$$

1 - 4 - کنترل کننده فازی با نظارت

در این کنترل کننده از خطا و تغییرات خطا برای تنظیم پارامترهای PID استفاده میشود. این وروردیها به صورت زیر هستند:

$$e(t) = r(t) - y(t) \qquad (4)$$

$$\Delta e(t) = e(t) - e(t-1) \qquad (5)$$

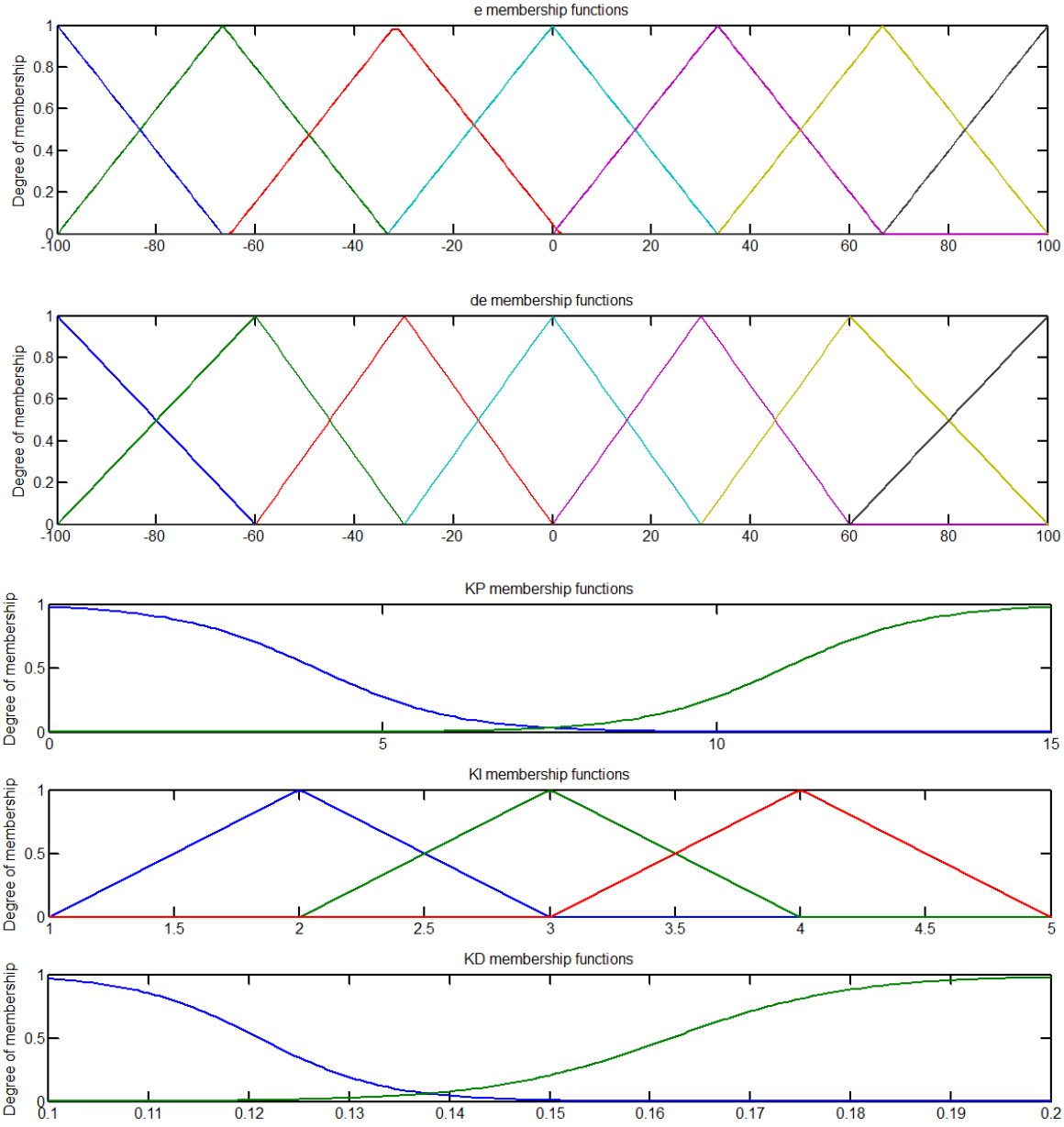
محدوده خروجیهای سیستم فازی به صورت زیر است:

$$K_P \in [0, 15]$$

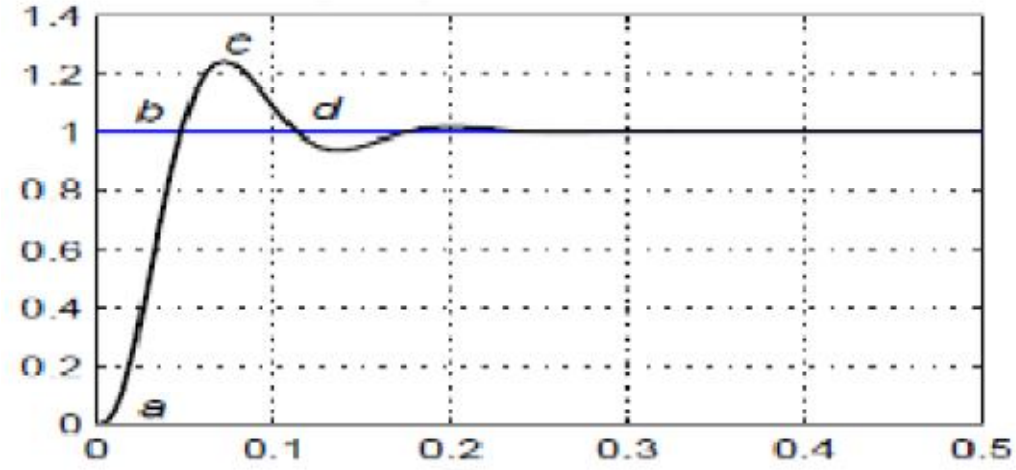
$$K_I \in [0.001, 0.005]$$

$$K_D \in [0.1, 0.2]$$

تابع عضویت برای متغیرهای ورودی و خروجی در شکلهای زیر آمده است:



قوانین سیستم فازی باید به گونه‌ای انتخاب شوند که پاسخ سیستم دارای زمان خیز کوتاه، فراجهش کم و خطای حالت ماندگار کم باشد. شکل زیر پاسخ پله سیستم را نشان می‌دهد. این پاسخ به چهار قسمت تبدیل می‌شود.



برای نواحی نزدیک (a) برای داشتن زمان خیز سریع باید یک سیگنال کنترلی بزرگ داشته باشیم. برای حذف خطا بهره انتگرالی مورد نیاز است و برای سرعت بخشیدن به پاسخ باید بهره مشتقی وجود داشته باشد. قانون این ناحیه به صورت زیر بیان می‌شود

$$If\ e\ is\ PB\ and\ \Delta e\ is\ Z\ then\ K_P'\ is\ B, K_D'\ is\ S\ and\ K_I'\ is\ S \qquad (11)$$

زمانی که در نواحی نقطه (b) خطا منفی است سیستم باید آرام باشد تا فراجهش کم باشد. این کار با کم کردن ضریب تناسبی و بهره انتگرالی کم و بهره مشتقی بزرگ انجام می‌شود. قانون مربوط به این ناحیه به صورت زیر بیان می‌شود:

$$If\ e\ is\ Z\ and\ \Delta e\ is\ NB\ then\ K_P'\ is\ S_1, K_D'\ is\ B\ and\ K_I'\ is\ S \qquad (12)$$

قوانین دیگر نیز به طریق مشابه بدست می‌آیند. در جداول زیر قوانین سیستم فازی آمده است:

TABLE I. FUZZY CONTROL RULE OF KP

KP		ERROR						
		NB	NM	NS	Z	PS	PM	PB
CANGE OF ERROR	NB	B	S	S	S	S	S	B
	NM	B	B	S	S	S	B	B
	NS	B	B	B	S	B	B	B
	Z	B	B	B	B	B	B	B
	PS	B	B	B	S	B	B	B
	PM	B	B	S	S	S	B	B
	PB	B	S	S	S	S	S	B

TABLE II. FUZZY CONTROL RULE OF KD

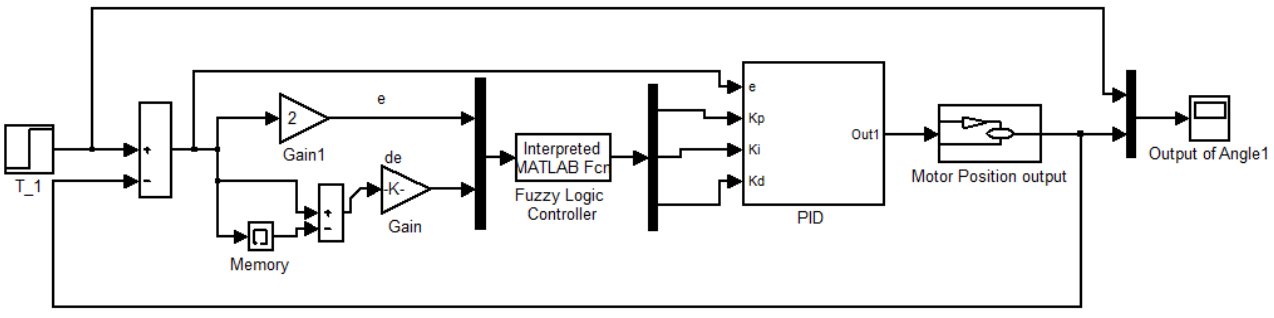
KD		ERROR						
		NB	NM	NS	Z	PS	PM	PB
CANGE OF ERROR	NB	S	B	B	B	B	B	S
	NM	S	B	B	B	B	B	S
	NS	S	S	B	B	B	S	S
	Z	S	S	S	B	S	S	S
	PS	S	S	B	B	B	S	S
	PM	S	B	B	B	B	B	S
	PB	S	B	B	B	B	B	S

TABLE III. FUZZY CONTROL RULE OF KI

KI		ERROR						
		NB	NM	NS	Z	PS	PM	PB
CANGE OF ERROR	NB	S	M	B	B	B	M	S
	NM	S	M	M	B	M	M	S
	NS	S	S	M	M	M	S	S
	Z	S	S	S	M	S	S	S
	PS	S	S	M	M	M	S	S
	PM	S	M	M	B	M	M	S
	PB	S	M	B	B	B	M	S

1 - 5 - پیاده سازی روش در Matlab

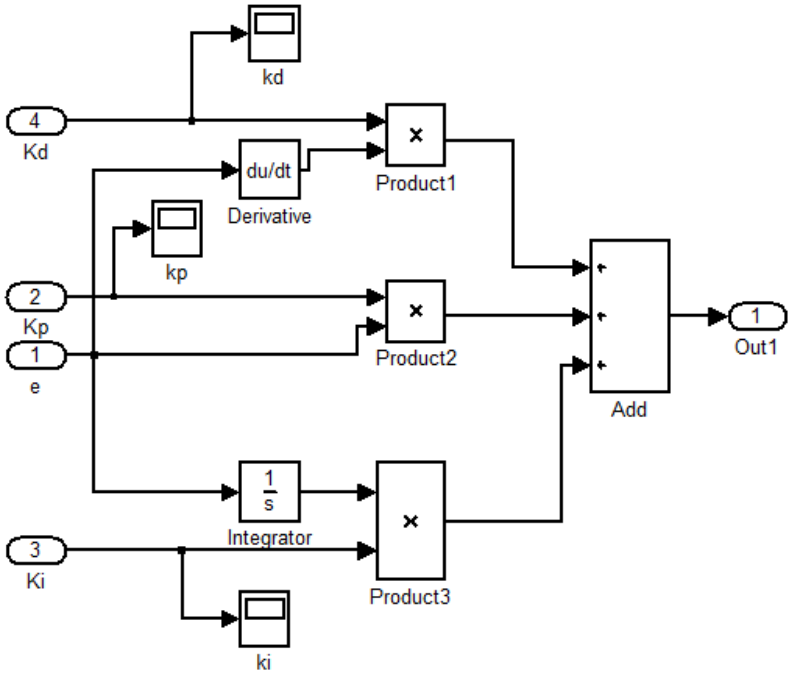
کنترل کننده فازی ارائه شده در محیط سیمولینک به صورت زیر شبیه سازی شده.



در این قسمت عملکرد کنترل کننده معرفی شده را روی یکی از بازوهای روبات بررسی می‌کنیم. هر بازوی روبات دارای یک موتور DC است که تابع تبدیل آن به صورت زیر است:

$$G(s) = \frac{19649}{s^3 + 201s^2 + 6290s} \qquad (13)$$

بلوک PID در شکل بالا به صورت زیر است:



در بلوک fuzzy logic controller تابع زیر برای ورودی دادن به سیستم فازی استفاده شده است:

```
function y = ccontrol( x )
load controller
y=evalfis(x,controller);
end
```

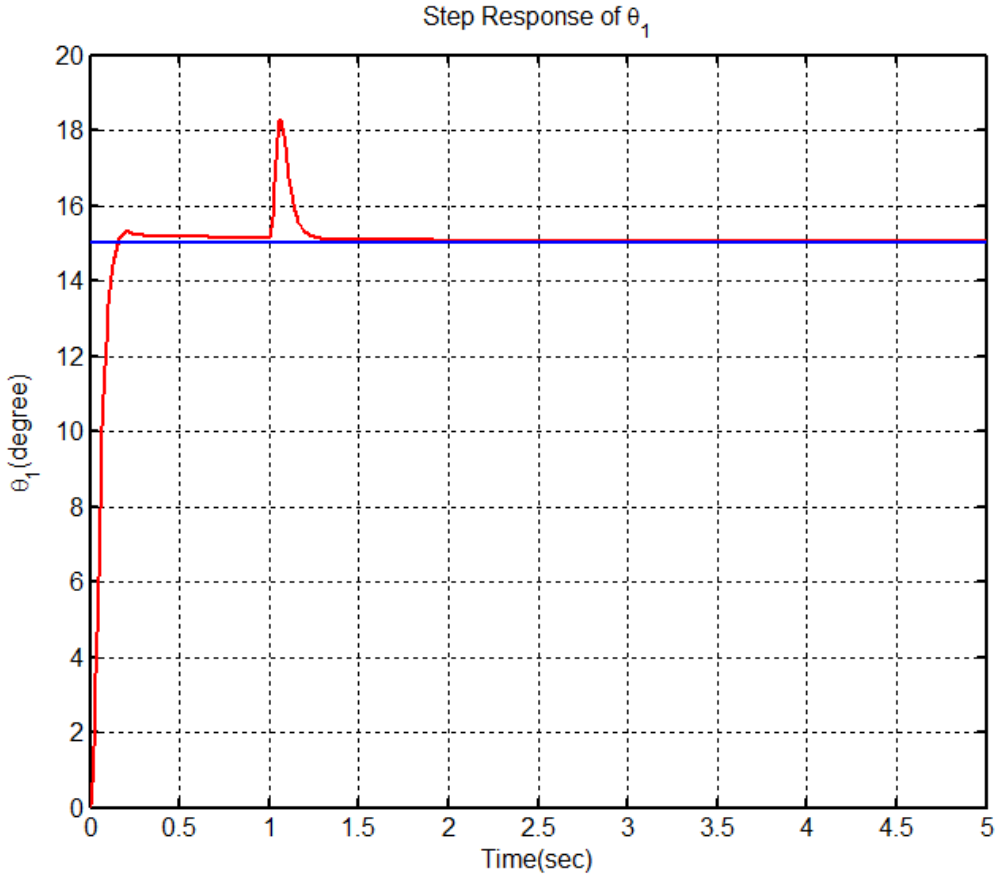
در این تابع ابتدا سیستم فازی با قوانین و توابع عضویت معرفی شده که قبلا با جعبه ابزار منطق فازی ساخته شده و در فایل controller.m ذخیره شده لود می‌شود. سپس با تابع evalfis به این سیستم ورودی داده می‌شود و نتیجه آن در خروجی تابع قرار می‌گیرد.

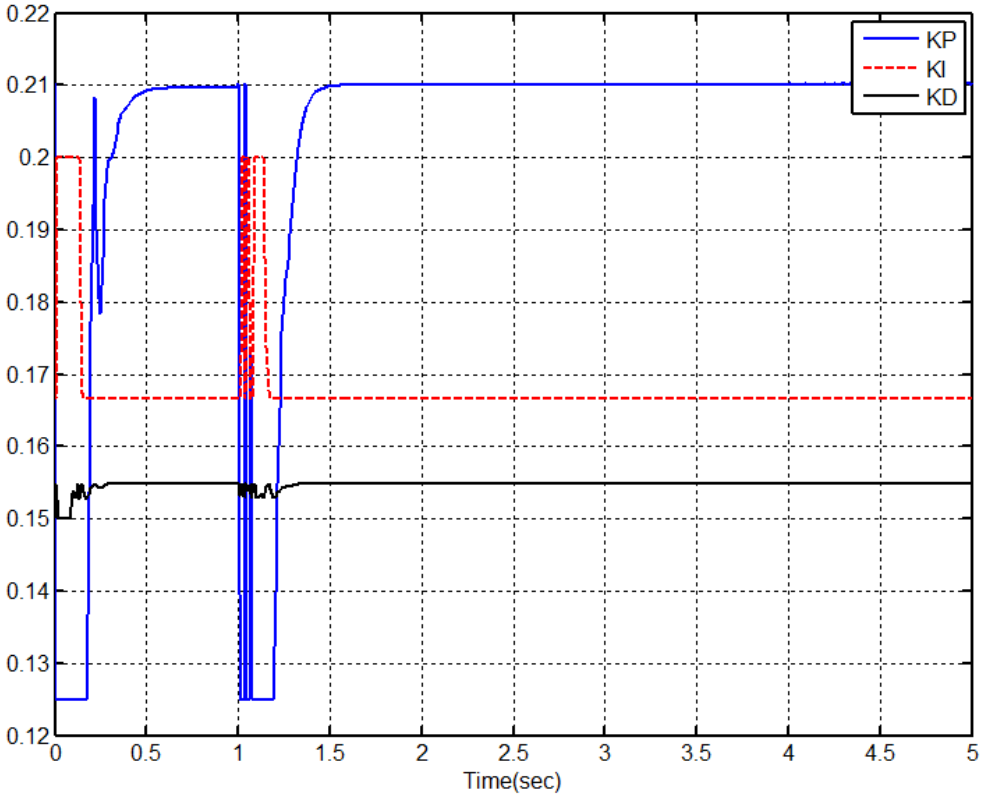
برای اجرای شبیه سازی و رسم نتایج کدهای فایل FSC.m نوشته شده. این کدها به صورت زیر هستند:

```
clc;clear all;close all;

T_1=15;
D_1=5;

load controller;warning('off');
sim ('FFS2');
```

```
figure
plot(th1(:,1),th1(:,3),'r','linewidth',2);hold on
plot(th1(:,1),th1(:,2),'linewidth',2)
ylabel '\theta_1(degree)';xlabel 'Time(sec)';grid
title 'Step Response of \theta_1'

figure
plot(kp(:,1),kp(:,2)/60,'linewidth',2);hold on
plot(ki(:,1),ki(:,2)/15,'r--','linewidth',2)
plot(kd(:,1),kd(:,2),'k','linewidth',2);grid;
xlabel Time(sec)
legend('KP','KI','KD')

t=th1(:,1);t=t(t<.99);
y=th1(1:length(t),3);
S=stepinfo(y,t);
display(['OS(%) = ' num2str(S.Overshoot)])
display(['RiseTime(s) = ' num2str(S.RiseTime)])
display(['SSE(%) = ' num2str(abs(T_1-y(end)))])

% Extract and plot MFs of Inputs
figure
[x,mf] = plotmf(controller,'input',1);
subplot(2,1,1), plot(x,mf);
title('e membership functions')
ylabel('Degree of membership')
[x,mf] = plotmf(controller,'input',2);
subplot(2,1,2), plot(x,mf);
title('de membership functions')
ylabel('Degree of membership')

% Extract and plot MFs of outputs
figure
[x,mf] = plotmf(controller,'output',1);
subplot(3,1,1), plot(x,mf);
title('KP membership functions')
ylabel('Degree of membership')
[x,mf] = plotmf(controller,'output',2);
subplot(3,1,2), plot(x,mf);
title('KI membership functions')
ylabel('Degree of membership')
[x,mf] = plotmf(controller,'output',3);
subplot(3,1,3), plot(x,mf);
title('KD membership functions')
ylabel('Degree of membership')

warning('on');
```

در این کدها ابتدا دامنه ورودی و اغتشاش توسط متغیرهای T_1 و D_1 توسط کاربر تعیین می‌شود. سپس فایل سیمولینک اجرا می‌شود و نتایج خروجی به workspace فرستاده می‌شوند. سپس نتایج بدست آمده و همچنین تغییرات پارامترها در طول زمان رسم می‌شوند و برای داشتن یک معیار کمی، توسط تابع stepinfo مشخصات پاسخ پله مانند فراجهش و زمان خیز را بدست می‌آوریم و در command window نمایش می‌دهیم. سپس توسط تابع plotmf شکل توابع عضویت را برای ورودی و خروجی‌ها استخراج و نمایش می‌دهیم. همچنین برای بررسی اثر اغتشاش، یک اغتشاش در زمان t=1s به سیستم وارد شده است. شکل‌های زیر نتیجه اجرای کدهای بالا را نشان می‌دهند:





Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
OS(%) = 0.9875
RiseTime(s) = 0.089294
SSE(%) = 0.14827
>>
```