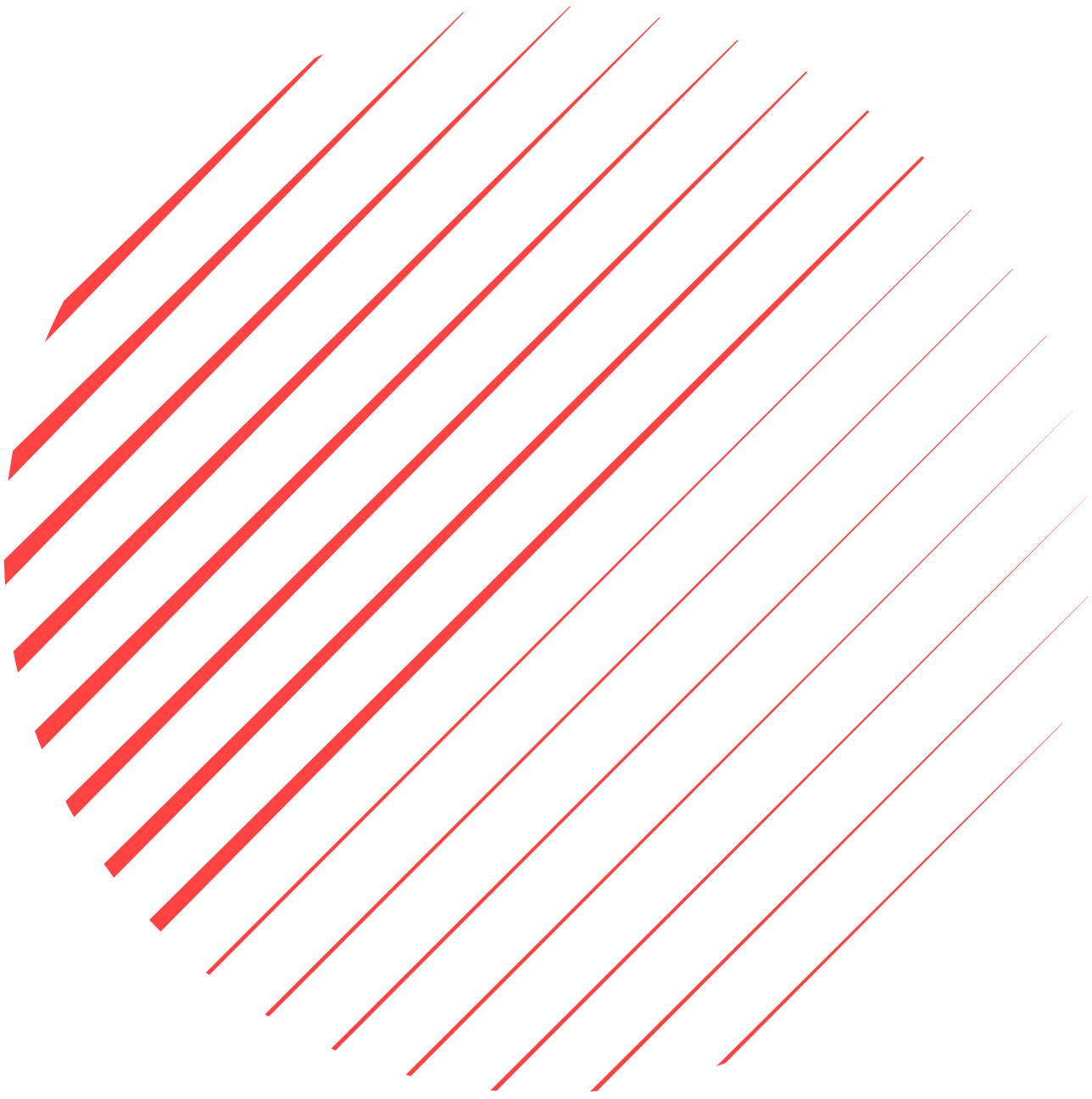
MINI PROJECT #3



Amir Mohammad Saffar
Dr. aliyarishorehdeli

FUNDAMENTAL OF INTELLIGENT SYSTEMS

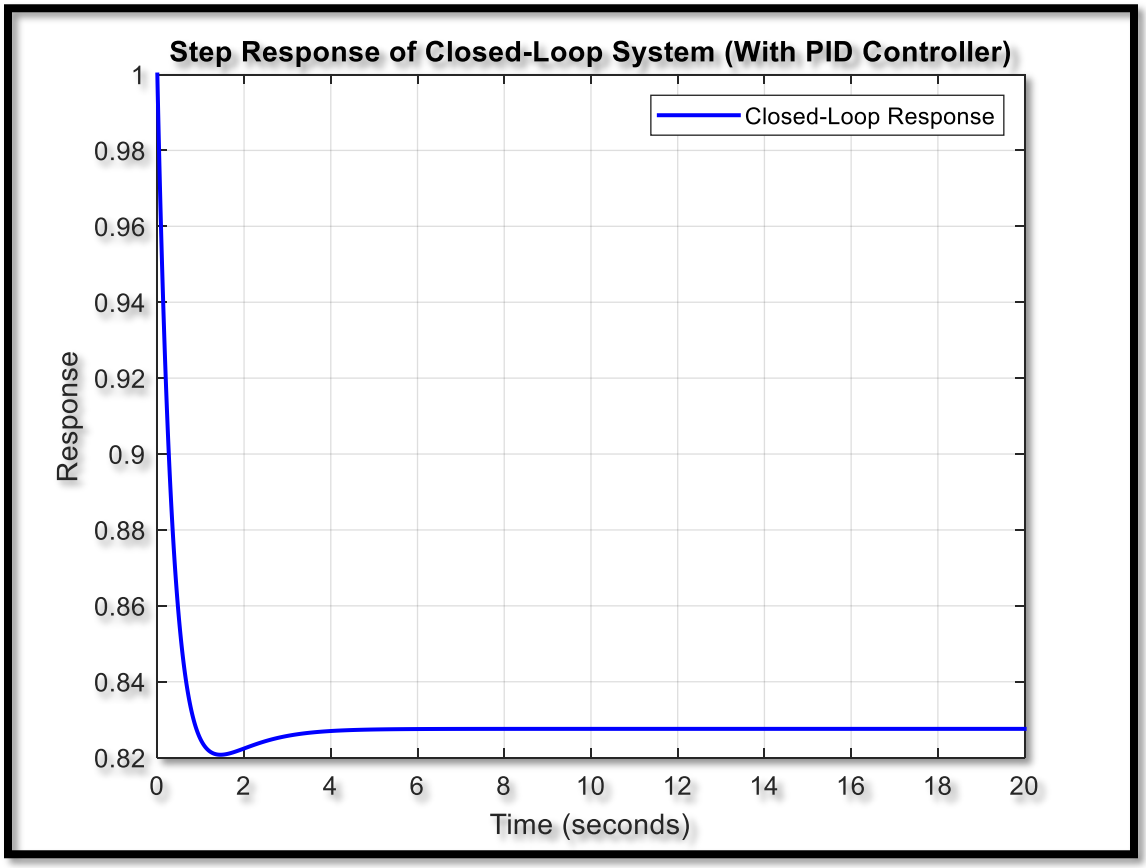
https://colab.research.google.com/drive/11JQ4800JZ0zHCydoxPbnyp-uaaFaRLPF?usp=sharing

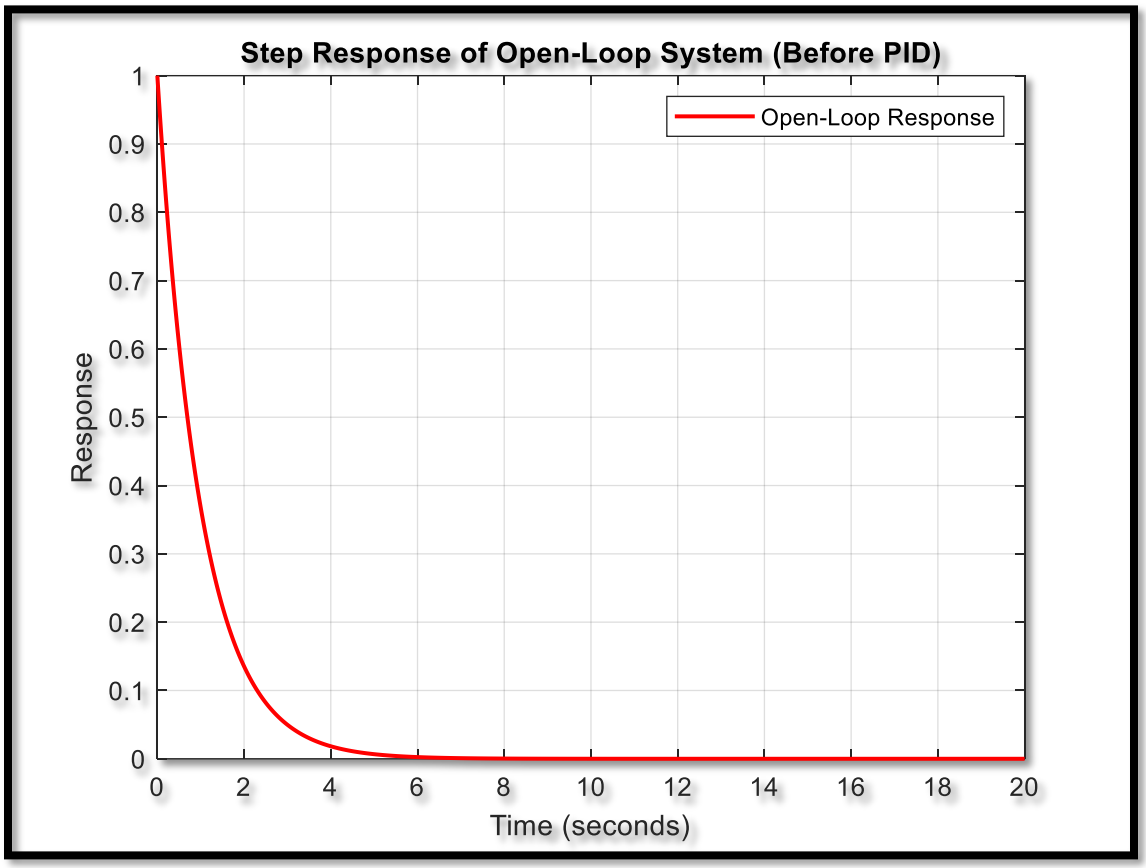
Question 1

در ایـن سـوال ابتـدا کـد متلبـی بـرای طراحـی کنتـرل کننـده PID مـی نویسـیم تـا بتـوانیم تغییـرات آن را قبـل و بعـد از اعمـال کنترل کننده بررسی کنیم.

question_one_design_ziegler_nicholes.m

Controller	K_p	T_i	T_d
P	$0.5\ K_{cr}$	Infinity	0
PI	$0.45\ K_{cr}$	$P_{cr}/1.2$	0
PID	$0.6\ K_{cr}$	$P_{cr}/2$	$0.125\ P_{cr}$

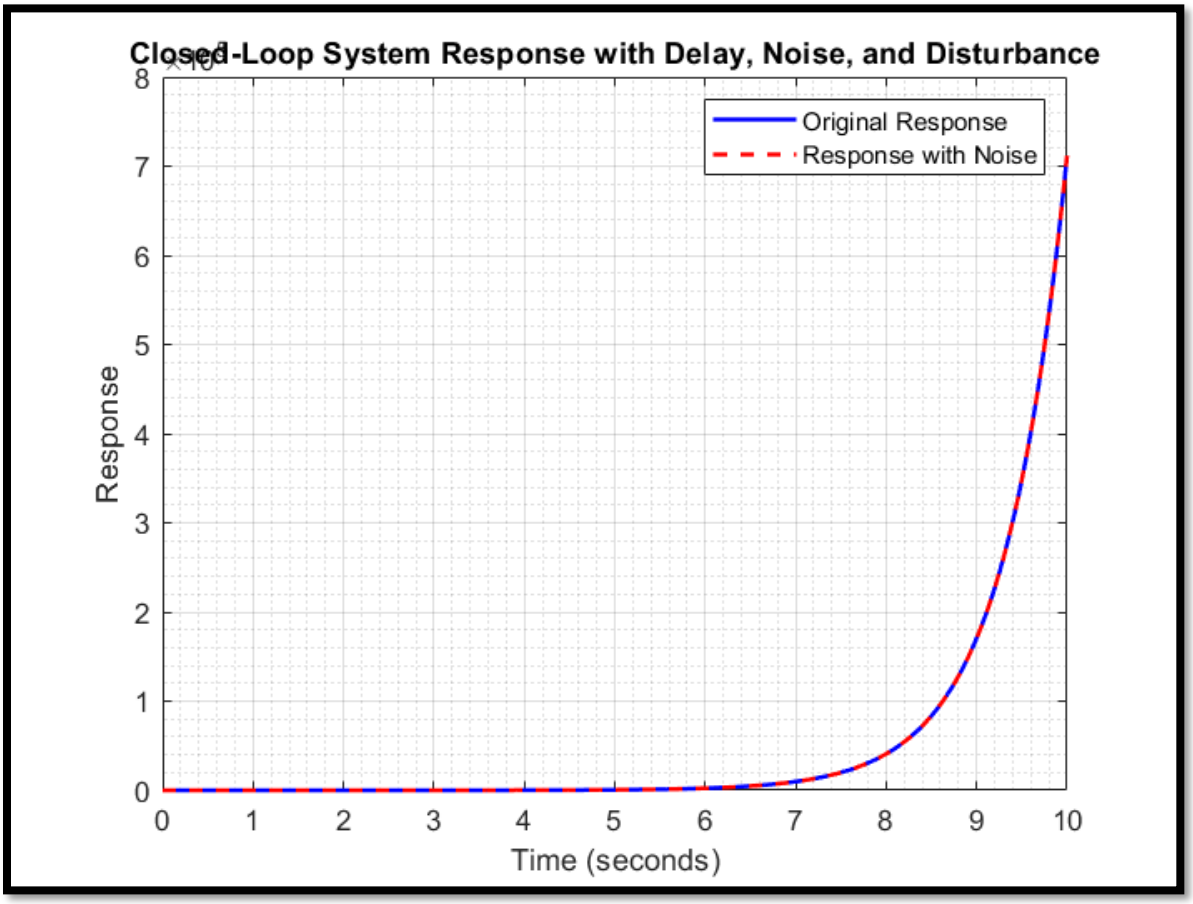





```
Open-Loop System Metrics:
  Rise Time: 0.0000 seconds
  Settling Time: NaN seconds
  Overshoot: Inf%
  Peak Time: 0.0000 seconds

Closed-Loop System Metrics (With PID Controller):
  Rise Time: 0.0000 seconds
  Settling Time: 0.6150 seconds
  Overshoot: 20.83%
  Peak Time: 0.0000 seconds
```

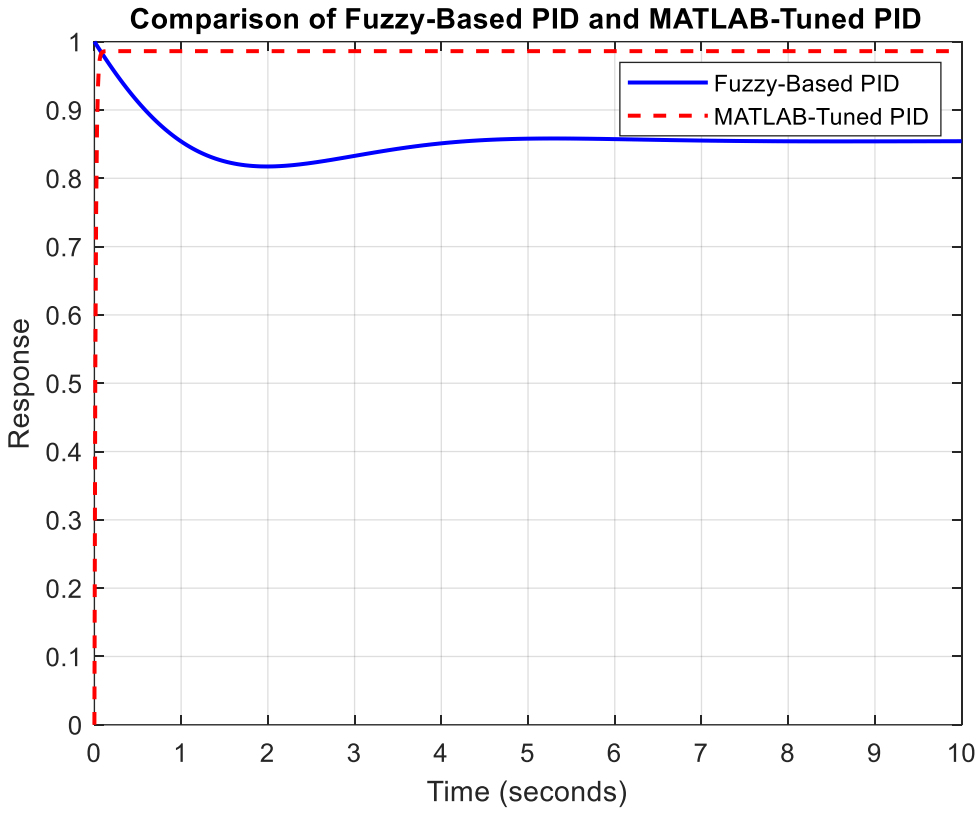
- در ایـن بخـش بـا اعمـال نـویز، اغتشـاش و تـاخیر بـه بررسـی مجـدد کنتـرل کننـده طراحـی شـده مـی پـردازیم و میبینـیم کـه خروجی مطلوبی به ما می دهد.

question_one_design_ziegler_nicholes_after_noise.m




```
Ku = 8.0;
Pu = 2.5;
Kp = 0.6 * Ku;
Ti = 0.5 * Pu;
Td = 0.125 * Pu;
C = pid(Kp, Kp/Ti, Kp*Td
```

در این بخش نیز با استفاده از کنترل فازی ضرایب را تعیین می کنیم و سپس با طراحی خود متلب مقایسه می کنیم.



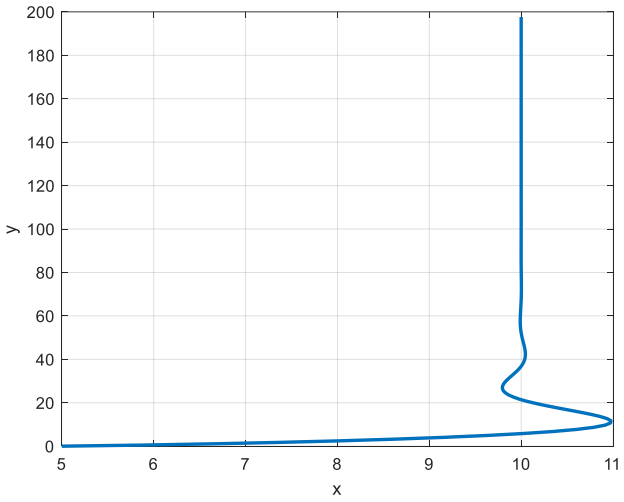

```
Fuzzy PID Coefficients:
  Kp: 5.8684
  Ki: 5.8684
  Kd: 5.0000

Performance Metrics for Fuzzy-Based PID:
         RiseTime: 0
    TransientTime: 6.0004
     SettlingTime: 3.2090
      SettlingMin: 0.8173
      SettlingMax: 1
        Overshoot: 17.0404
       Undershoot: 0
             Peak: 1
         PeakTime: 0
```

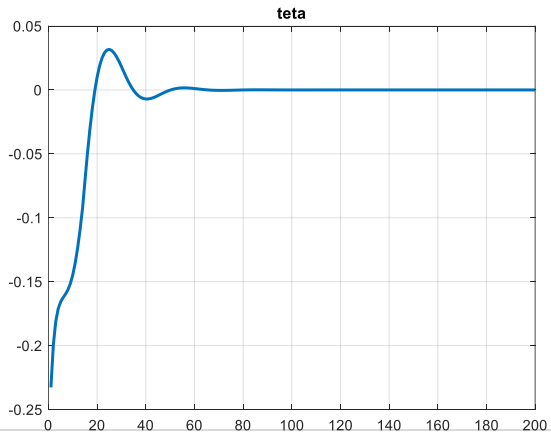
```
Performance Metrics for MATLAB-Tuned PID:
         RiseTime: 0.0306
    TransientTime: 0.0544
     SettlingTime: 0.0544
      SettlingMin: 0.8919
      SettlingMax: 0.9861
        Overshoot: 0
       Undershoot: 0
             Peak: 0.9861
         PeakTime: 0.1467
```

Question 2

این کد در متلب یک سیستم **کنترل فازی** برای **مدل‌سازی حرکت یک کامیون** را پیاده‌سازی می‌کند. در ادامه توضیح کلی کد در چند خط آورده شده است:

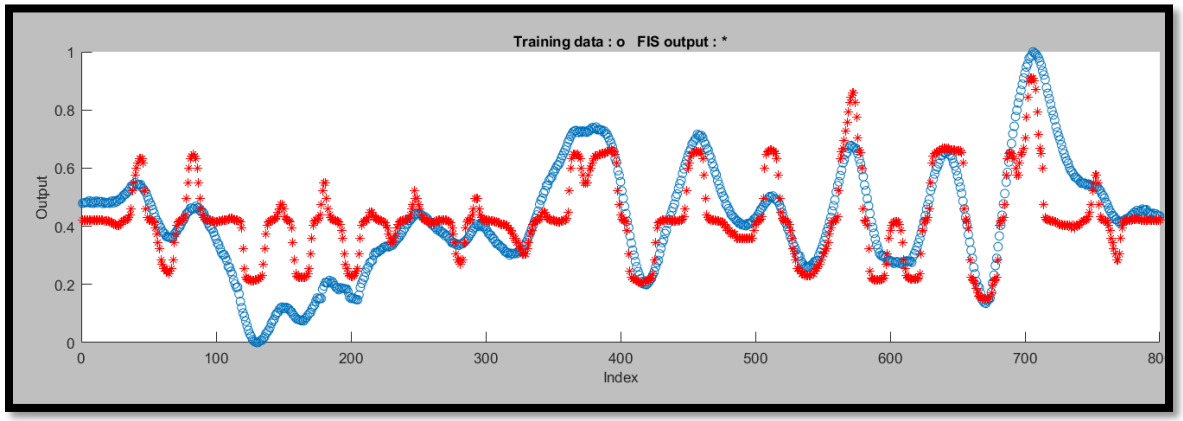
1. **تعریف توابع عضویت فازی :**دو تابع `mf1` و `mf2` برای توابع عضویت متغیرهای ورودی `x1` موقعیت کامیون و `x2` زاویه‌ی چرخش تعریف شده‌اند که در فواصل خاصی مقدار عضویت را تعیین می‌کنند.
2. **ایجاد مراکز قوانین فازی :**آرایه `cm` مراکز قوانین فازی را مشخص می‌کند که بر اساس ورودی‌ها، مقدار خروجی را تعیین خواهند کرد.
3. **پیاده‌سازی قوانین فازی :**حلقه‌ی `for` روی مجموعه‌ای از قوانین فازی اجرا شده و مقدار خروجی سیستم (`theta`) با استفاده از میانگین وزن‌دار محاسبه می‌شود.
4. **مدل‌سازی حرکت کامیون :**در هر گام زمانی، مقدار `phi` (زاویه حرکت) و مختصات `(x, y)` بر اساس فرمول‌های کتاب بروزرسانی می‌شوند.
5. **رسم نمودارها :**در پایان، دو نمودار شامل تغییرات زاویه `theta`و مسیر حرکت `(x, y)` رسم شده تا رفتار حرکت کامیون شبیه‌سازی شود.





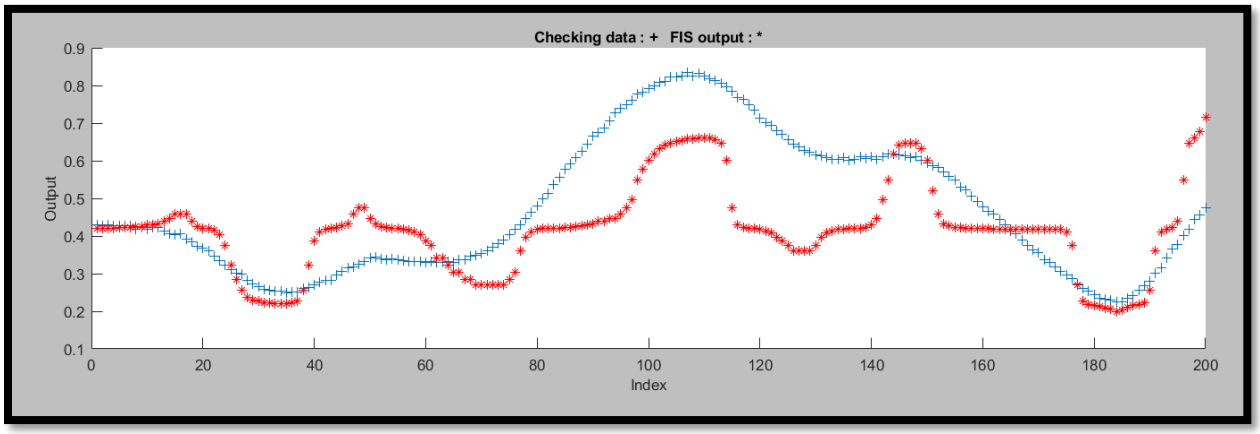
Question 3

- در ایـــن بخـــش ابتـــدا دیتـــای اول را در داخـــل پوشـــه متلـــب قـــرار مـــی دهـــیم. ســـپس وارد Neuro fuzzy designer مـــی شـــویم و فایـــل دیتـــا را کـــه بـــه دو بخـــش آمـــوزش و آزمـــون (20&80) تقســـیم شـــده را در دو بخـــش checking و training قـــرار مـــی دهـــیم. و ســـپس grid partition را انتخـــاب مـــی کنـــیم و تعـــداد membership function هـــا را مشـــخص مـــی نمـــاییم و مـــدل آن را بـــر روی گاوســـی تنظـــیم مـــی کنـــیم. تعـــداد ایپـــاک را هـــم بـــروی 25 قـــرار مـــی دهـــیم. همچنین مقدار حداقل ارور را هم 0 قرار می دهیم.



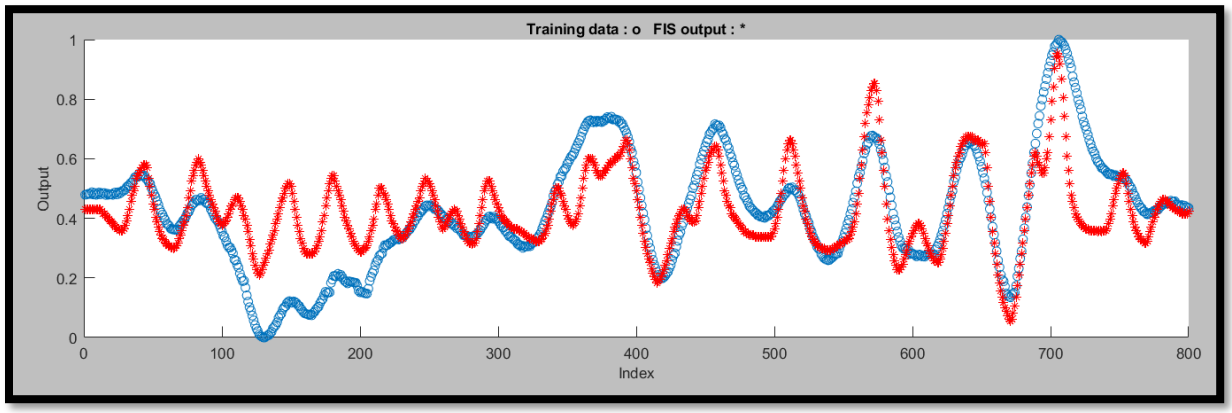
Average testing error: 0.13224

- حال با داده های آزمون آن را مجدد تست می کنیم.



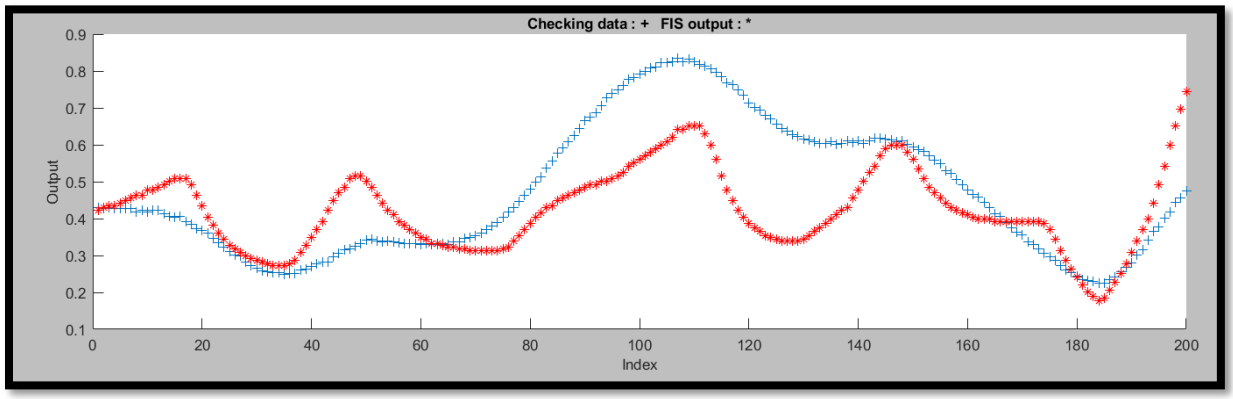
Average testing error: 0.14077

- در ایـن بخـش میتـوانیم همـان کـار را بـا یـک FIS دیگـر مجـدد انجـام دهـیم، مـدل sub.clustering را انتخـاب مـی کنـیم و مجدد آموزش را انجام می دهیم.
- مـــی تـــوانیم مقـــادیر reject_ratio accept_ratio squash_factor range_of_influence را بـــر اســـاس نیازمان تغییر دهیم تا مقدار خطلا کاهش یابد.



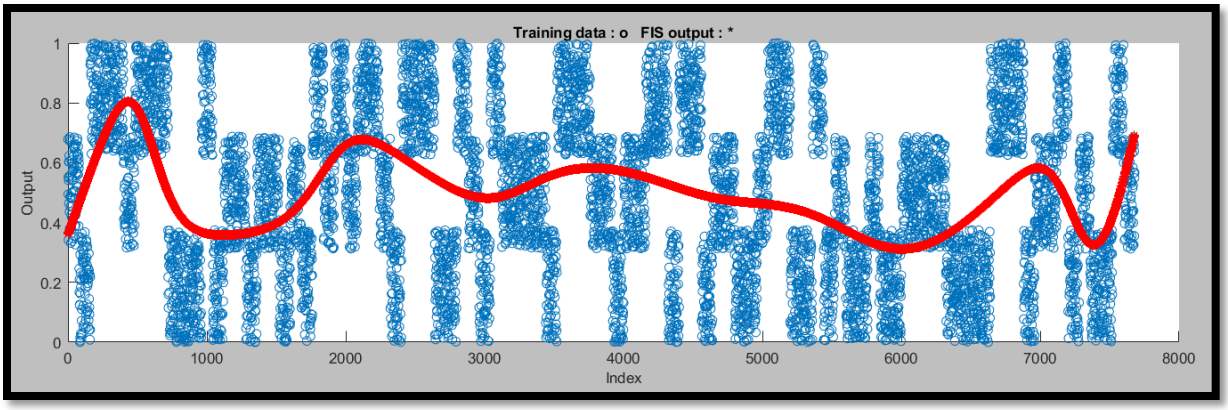
Average testing error: 0.13837

- حال با داده های آزمون آن را مجدد تست می کنیم.



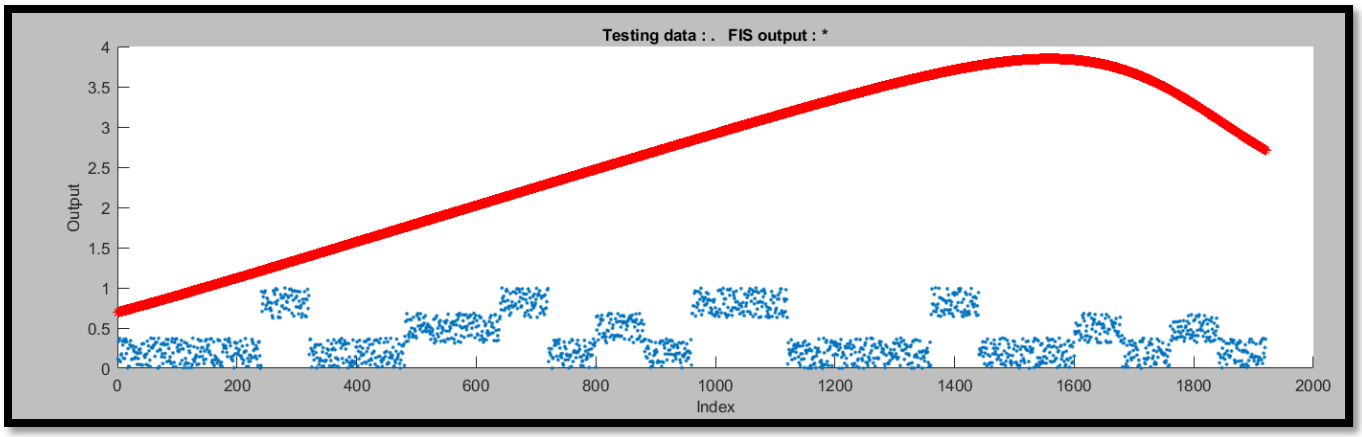
Average testing error: 0.14308

- برای بخش دوم نیز همانند بخش قبلی عمل می کنیم:



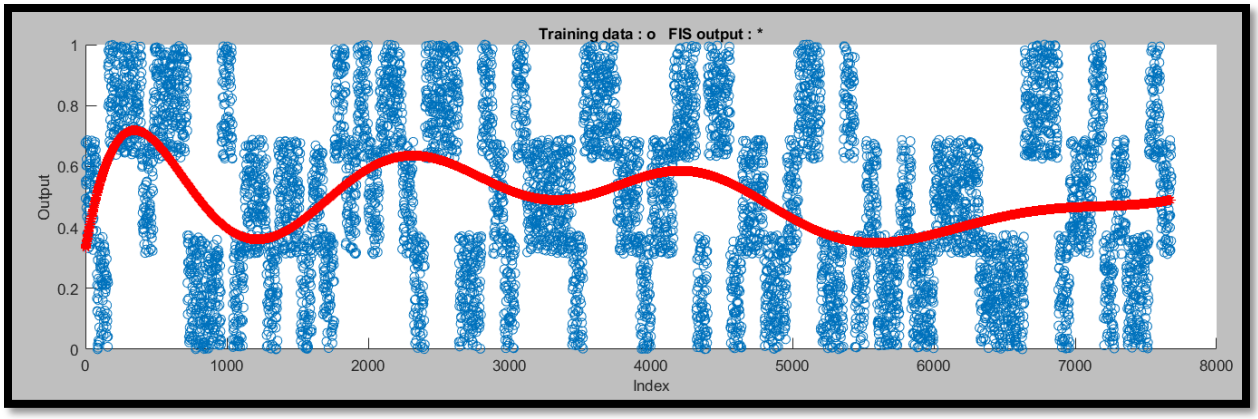
Average testing error: 0.25463

- حال با داده های آزمون آن را مجدد تست می کنیم.



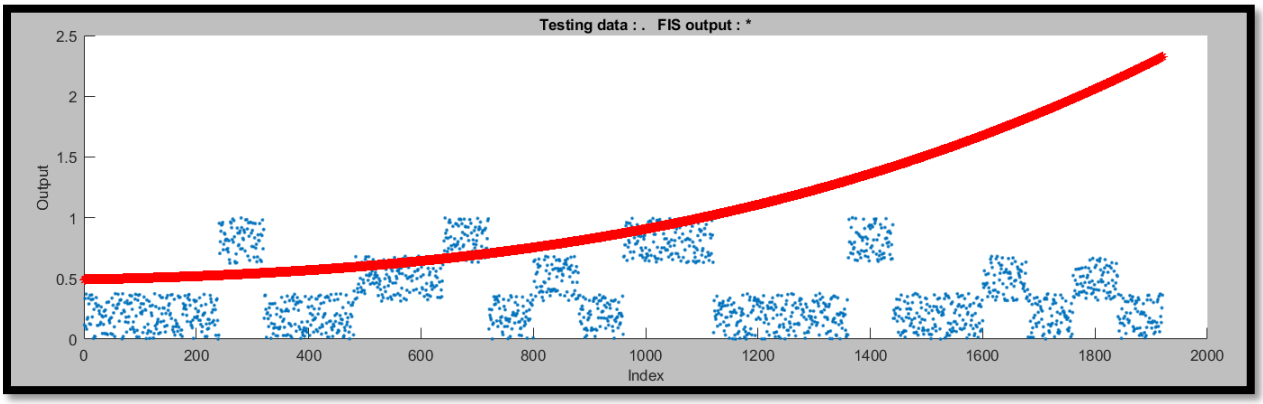
Average testing error: 2.4263

- در این بخش میتوانیم همان کار را با یک FIS دیگر مجدد انجام دهیم، مدل sub.clustering را انتخاب می کنیم و مجدد آموزش را انجام می دهیم.
- می توانیم مقادیر reject_ratio accept_ratio squash_factor range_of_influence را بر اساس نیازمان تغییر دهیم تا مقدار خطلا کاهش یابد.



Average testing error: 0.26144

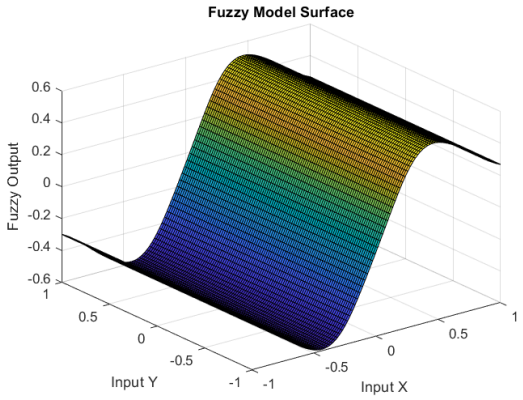
حال با داده های آزمون آن را مجدد تست می کنیم.

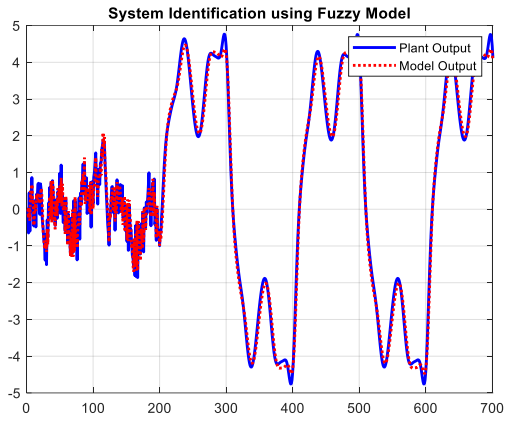


Average testing error: 0.90921

Question 4

این کد یک مدل فازی برای شناسایی سیستم را پیاده‌سازی می‌کند. ابتدا توابع عضویت و پارامترهای اولیه تنظیم می‌شوند. در مرحله آموزش، مدل با استفاده از نمونه‌های تصادفی ورودی و خروجی به‌روزرسانی می‌شود و مقادیر بهینه برای پارامترها محاسبه می‌گردد. در مرحله تست، مدل با ورودی‌های جدید ارزیابی شده و خروجی‌های پیش‌بینی‌شده با مقادیر واقعی مقایسه می‌شوند. در نهایت، میزان خطای RMSE محاسبه شده و نمودارهای نتایج و سطح فازی سیستم نمایش داده می‌شوند.



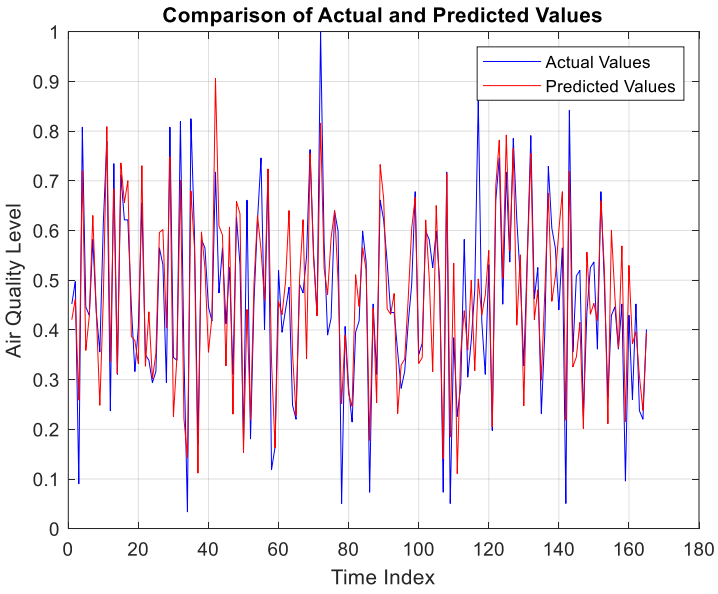


Question 5

مدل ANFIS در حل مسائل غیرخطی پیچیده معمولاً عملکرد بهتری دارد، زیرا توانایی ترکیب دانش انسانی و داده‌ها را دارد. در مقابل، RBFدر مسائل ساده‌تر با سرعت بیشتر برتری دارد .

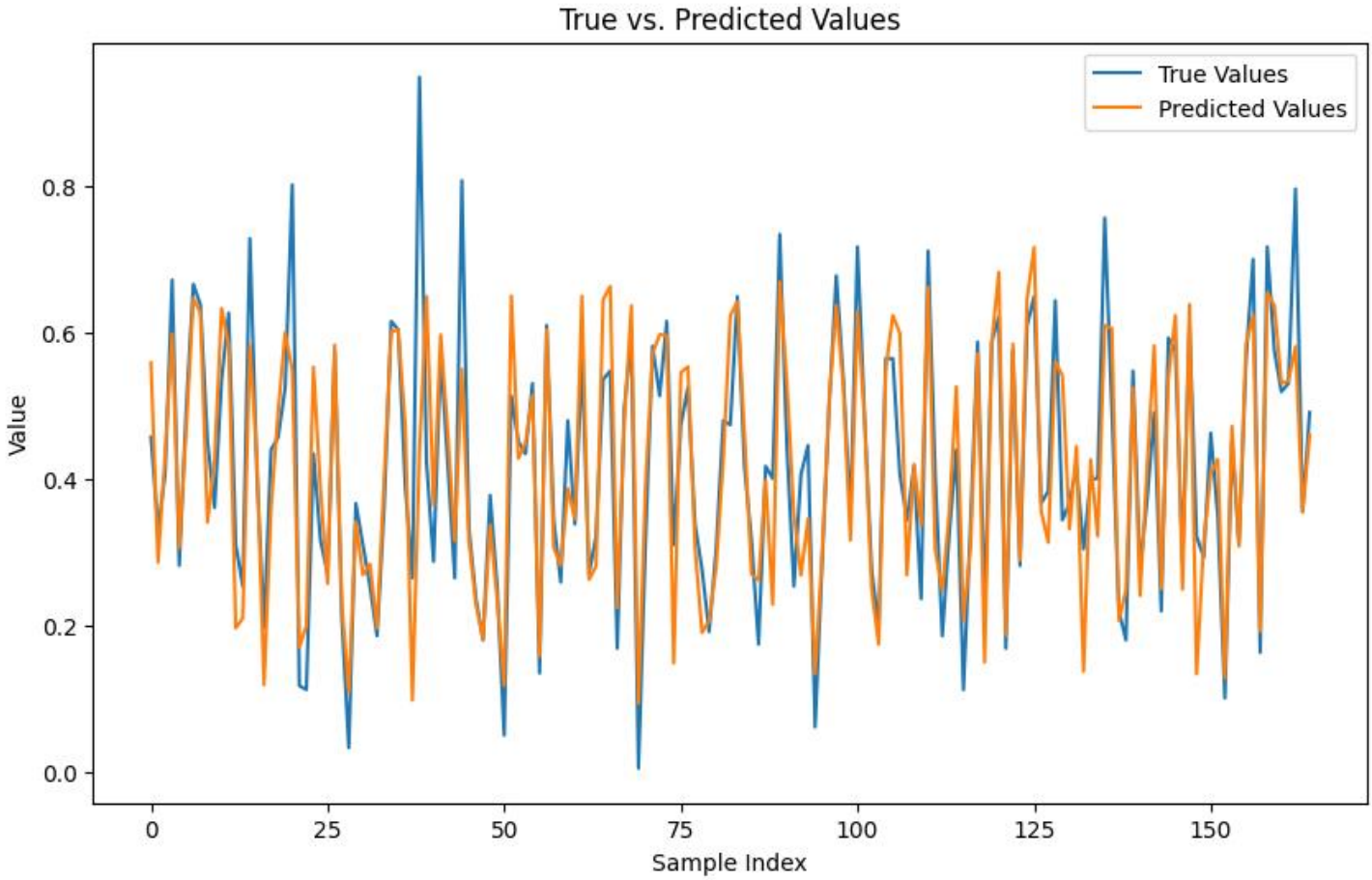
ANFIS

```
RMSE on Test Data: 0.094563
```



RBF

کد آن در نوت بوک کلب قابل مشاهده است:



Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 12)	0
rbf_kernel_layer_3 (RBFKernelLayer)	(None, 10)	120
dense_3 (Dense)	(None, 1)	11

Total params: 131 (524.00 B)
Trainable params: 131 (524.00 B)
Non-trainable params: 0 (0.00 B)