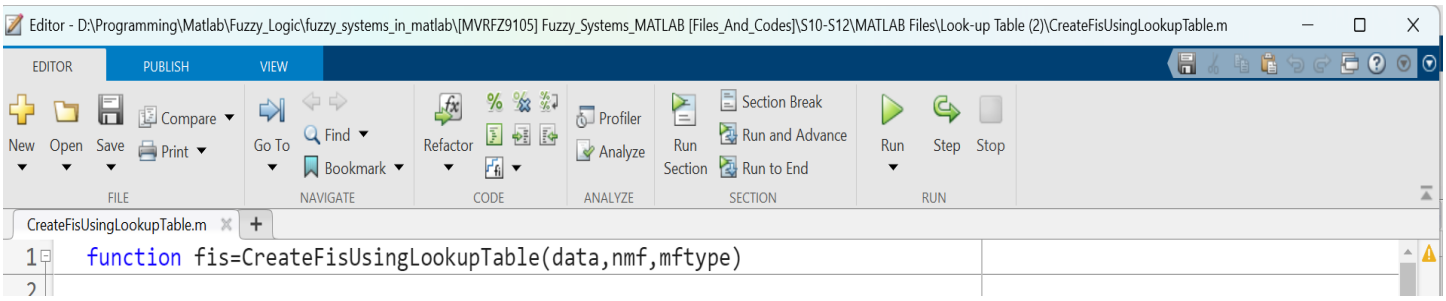
خب برای شروع کار این جلسه ما یک تابع را مینویسم که تمامی کار هایی که در جلسه قبلی به صورت جدا جدا انجام میشد در یک مجموعه انجام گیرد .

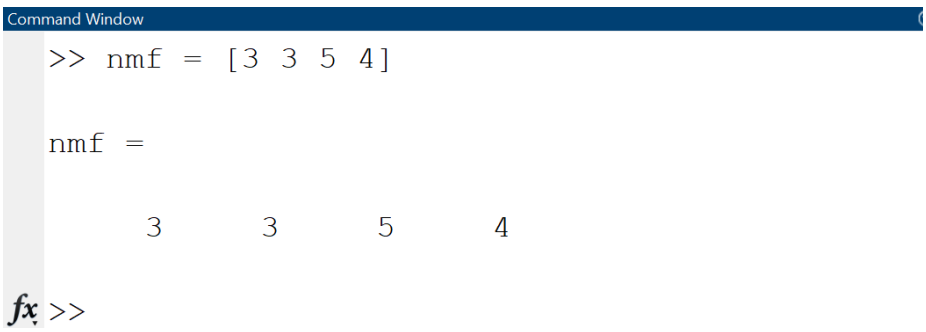
حال برای این کار یک فایل جدی را باز کرده و دستور function را تایپ کرده و مانند زیر عمل میکنیم :



خب میبنید که تابع را نوشته ایم یک اسم داده ایم به آن و خروجی آن هم fis و چند آرگومان برای آن داریم و دلیل این که آرگومان اول را دیتا گرفتیم این است ، عملکرد سیستم راحتر شود .

دوتای دیگر آرگومان اینجا nmf,mftype این دو یکی بر ای نوع تابع عضویت داده های ما است یعنی چه نوعی تابع ما باشد membership functions ها به چه نحوی توزیع بشند و nmf هم برای دیتای تعداد ورودی ها و خروجی های ما است مثلا 3 ورودی و یک خروجی داریم باید 4 بگذاریم .

مثلا بزارید برای شما شرح دهم که این عدد nmf را 4 داده اید حال چه چیزی رخ میدهد این یک ماتریس 4 ستونی و یک سطری میسازد فرض کنید به شکل زیر :



حال این چه میگوید یعنی 3تا ورودی داریم و یک خروجی اولا دوما اولیان و دومین ورودی ها را به 3 قسمت و سومین ورودی به 5

قسمت و آخرین هم که خروجی کار ما است را به 4 قسمت تقسیم بندی آن را انجام بده برای این تابع .

برای آرگومان آخر هم یک چنین چیزی داریم که نوع تایپ تابع عضویت داده ها را برای ما مشخص میکند که به شرح زیر است :

```
>> mftype = {'trimf' 'trimf' 'gaussmf' 'trimf'}

mftype =

  1×4 cell array

  Columns 1 through 3

    {'trimf'}    {'trimf'}    {'gaussmf'}
```

همان که ملاحضه میکنید دیگر ماتریسی تعریف نشده است و به فرم سلولی میباشد چون رشته داریم برای کاری که میخواغهیم انجام دهیم .

در قسمت قبل از اول یک سری تقسیم بندی ها را انجام داده ایم که بدین صورت میباشد که :

```
P=size(data,1);      % Total Number of Data Records

x=data(:,1:end-1);   % Input Data
nx=size(x,2);

y=data(:,end);       % Target Data
```

خب P تعداد دیتا های هر مرحله است خب رای این کار باید از تابع size روی دیتا بزنیم ولی تعداد سطر های آن را میخواهیم که عدد یک میشود آرگومان دوم size که به فرم بالا نوشته شده .

ستون آخر یعنی خروجی های ما دیگه .

ور برای y هم روی دیتا ها را تمامی سطر ها و ستون آخر را برای این کار میخحواهیم یعنی یک بردار n سطری و یک ستونی میشود که منطقی هم است برای تعداد یک خورجی .

در ادامه ورودی ها را تحت عنوان یک آرایه سلولی در نظر میگیریم که به فرم زیر است :

```
%% 1: Create MFs

A=cell(nx,1);
for i=1:nx
    A{i}=CreateMembershipFunctions(x(:,i),nmf(i),mftype{i});
end
```

خب در این شکل A برای آن ی سلول به تداد ورودی ها در نظر گرفته ایم که دارای یک ستون است .

حلقه برای این میباشد که با تابعی که در جلسه قبلی نوشته شده است به a مقدار دهی کنیم که آرگومان های آن عبارتند از x(:,i) این یعنی سطر های ورودی و i امین ستون که منطور اون ورودی مدنظر iام است دومین برای تعداد همان تقسیم بندی های ما برای تعداد ورودی و خروجی که داریم و سوکمین هم تایپ تابع عضویت ماست .

خب برای این که تابعی که ما در حال حاضر مینویسیم دارای این قدرت باشد که اگر کسی که آن را مورد استفاده قرار میدهد اگر یکی از آرگومان های تابع را به آن ندهد ما یکسری دستور نشوته ایم که به فرم زیر است :

```
%% Check Input Arguments

if nargin<2 || isempty(nmf)
    nmf=3*ones(1,nx+1);
end

if nargin<3 || isempty(mftype)
    mftype=cell(1,nx+1);
    for i=1:nx+1
        mftype{i}='gaussmf';
    end
end
```

خب در ابتدا nargin<2 یعنی اگر تعداد آرگومان های ورودی ما کم تر از 2 بود || این در متلب یعنی or (یا) isempty(nmf) این دستور هم یعنی میگوید خالی باشد مقدرا nmf بیا یک حداقل ماتریس یکه بساز یک سطر و به تعداد تقسیم های ورودی هم ستون بهش بده و حداقل هم سایی کن این یعنی 3 قسمت بکن اون ورودی که اگر نداد کاربرد به سیستم به صورت پیش فرضا خودمان برای آرگومان آخر همم ما هم چنین کاری را انجام داده ایم که دوتا شرط را گذاشته که همانند قبلی است ، اما تایپ چون برای رشته است باید سلول ایجاد کنیم به ابعاد همان یک سطر و به تعداد ستون های ورودی بعلاوه یک و حلقه برای این استت که برای هر یک از دیتا ها از اولین ورودی تا آخرین ورودی و همچنی برای خروجی نوع تابع مورد استفاده برای membership functions آن ها تعین شود .

در ادامه برای خروجی هم همان کاری که انجام دادیم که ساخت member ship است انجام میدهیم با این تفاوت که این بار نیاز به حلقه نیست چون تک خروجی است سیستم ما .

```
B=CreateMembershipFunctions(y,nmf(end),mftype{end});
```

خب در دامه کاری حود باید قوانین یا Rules های سیستم خود را پیاده سازی کنیمم چون این قوانین برروی ورودی های ما هستند و ورودی های ما حودش هر کدام به پارتیشن های مختلفی طبقه بندی میشود به همین علت درگیر یک ماتریس چند بعدی میشویم برای این کار ما از ماتریس استفاده نمیکنی و از آرایه ها بهره گیری میکنیم فرض کنید 3 ورودی داریم با پاترتیشن بندی های

(3 و 3 و5) و خروجی هم 4 قسمت بشه این یعنی یک ماتریس 1 سطری چهار ستونی داریم تا اینجای کار حال برای ایجاد قوانین باید بر روی وردی ها یک Cell Array بسازیم که به فرم زیر است :

```
S = cell(nmf(1:end-1))
```

یک آرایه سلولی تولید میکند که دارای 45 عضو است یعنی 5 تا ماتریس 3*3 ایجاد میکند یعنی 45 عضود داریم .

شکل زیر نمایش 2 تا از 5 تا ماتریس تشکیل داده شده است :

همان طور در تصویر مشاهده میکنید دارای 3 ردیف سه ستون است و 5 بار این کار تکرار کرده ایم به علت این که وردی کار

```
val(:,:,3) =

    {0×0 double}    {0×0 double}    {0×0 double}
    {0×0 double}    {0×0 double}    {0×0 double}
    {0×0 double}    {0×0 double}    {0×0 double}

val(:,:,4) =

    {0×0 double}    {0×0 double}    {0×0 double}
    {0×0 double}    {0×0 double}    {0×0 double}
    {0×0 double}    {0×0 double}    {0×0 double}
```

```
>>nmf = [4 5 3 3]
```

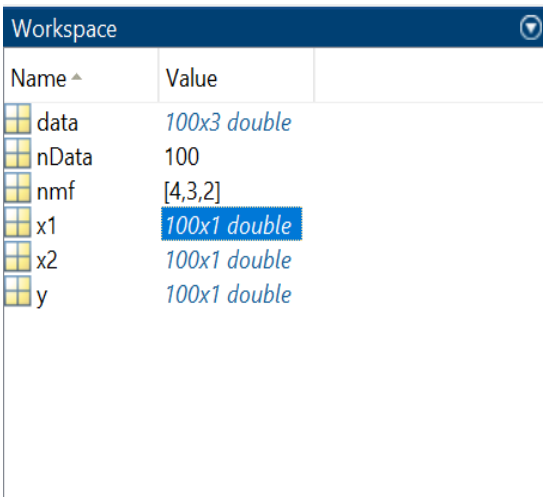
```
nmf =

     4     5     3     3
```

خب تا اینجای کار برای مقدار دهی به membership function ها را انجام داده ایم حال در ادامه میخواهیم به قوانین یا تولید قائده بپردازیم که برای این کارباید بر روی ورودی کار کرد بدین

صورت که ابتدای امر یک سلول از ورودی های خود تولید میکنیم سپس آن را عمودی میکنی یعنی یک بردار میکنیم مثلا اینجا prod ورودی را محاسبه کنیم 3*3*5 می‌شود 45 تا سطر و یک ستون را به ما میدهد در ادامه یک حلقه مینویسم که روی این کار کند بدین نحوه که متغیر nRule = numel(S) را تولید کرده که تمامی متغیر های ما را فارغ از این که چندتا است شمارش میکند سپس حلقه خود را تولید میکنیم از شمارمنده اول تا آخرین چیزی که در این ماتریس تک ستونی وجود دارد و به ازای S{i} های خود یک مقدار تابع zero صفر تعلق میدهیم که در هر از سلول های ما که به عنوان یک سطر شناخته میشه دارای یک ستون است.اگر تعداد ور.ودی ها بیشتر شود جدا از هم تولید یک بردار میکند .

یک مثال زیر ما از 100 تا دیتا استفاده کرده ایم 2 تا وردی دارد و یک خروجی و وردی ها هم یک به 4 قسمت یک به 3 قسمت تقسیم میشه سپس نوع type تابع مورد استفاده هم مشخص نمده ایم و درنهایت هم خروجی کار برای ورودی یک بردار شده از 100 تا ردیف و یک ستون و خروجی هم به همین نحو ولی تقسیم بندی دیتا ها 3،4و2 تایی است :




```
clc;
clear;
close all;

nData=100;
x1=rand(nData,1);
x2=rand(nData,1);
y=x1+x2;

data=[x1 x2 y];

nmf=[4 3 2];

mftype={'trimf','gaussmf','trimf'};

fis=CreateFisUsingLookupTable(data,nmf,mftype);
```

خب همان طور که میبنید روال کاری در حلقه و. ابتدای تابع را نمایش داه ایم .

یک دستور در متلب داریم به اسم sub2ind ,ind2sub این برای تبدیل کرد ن اندیسش های دوتایی یعنی subindex های مربوط به ماتریس ها به index تکی یا بلعکس که روند کاری اون به نحوه زیر است که میبینید :

همان طور که مشاهده میکنید یک ماتریس 3 سطری و2 ستونی داریم و اندیکس 2و2 میخواهیم و نحوه شمارش هم از بالا به

```
Command Window
>> sub2ind([3 2] ,2,2)

ans =

     5
```

پایین حساب کنیم میبینیم میشود عدد 5 .

حال برای بلعکس این موضوع داریم :

خب این موضوع چه کاربردی برای ما دارد حال در ادامه روند کد نویسی خود میبینییم که روند کاری به چه صورتی می‌شود .

```
>> [i j] = ind2sub([3 2],5)

i =

     2

j =

     2
```

خب وقتی که این دستور را در حالت عادی بکار ببریم فقط یک خروجی میدهد در حلقه نویسی حال اگر بخواهیم دوتا آرایه به ما بدهد باید به فرم سسلولی آن نوشته شود که به صورت زیر در میاید :

این ففرمت که نوشته شده برای ما در حلقه کد نویسی کاربیردی تر است چون وقتی به فرم قبلی بنویسم در حلقه فقط یک اندیس به ما

```
>> [xind{1:2}] = ind2sub([3 2],5)

xind =

  1×2 cell array

    {[2]}    {[2]}
```

میدهد که کارایی کامل را برای ما ندارد .

خب برای ورودی های خود میخواهیم ماتریس قوائد را درست کنیم که ابتدای امر برای ایندکس های خود تولید ماتریس صفر میکنیم که به تعداد قوائد که همان تعداد numel(s) است سطر و ستون های آن معادل تعداد ورودی های ما میباشد .

سپس یک حلقعه تولید میکنیم که r=1:nRule که تولید کردیم حرکت میکند سپس یک ایندکس تولید میکنیم که به روش xind = cell(1,nx) یعنی یک سطر و به تعداد ورودی ستون دارد این سلول ما سپس به نحوه ای که توضیح دادم اندیکس هر یک از قوائد را در میاوریم سپس بعد از آن تولید سلول میشود که ما در خط بعدیش سلول را به ماتریس تبدیل میکنیم.

بعد از این که ماتریس قوائد ساخته شد باید به محاسبه آن بپردازیم که برای محاسبه آن بدین صورت عمل میکنیم که :

```
%% 3: Calculate Score of Rules

XIND=zeros(nRules,nx);

for r=1:nRules

    xind=cell(1,nx);
    [xind{1:nx}]=ind2sub(SSize,r);
    xind=cell2mat(xind);

    XIND(r,:)=xind;

    for yind=1:nmf(end)

        bmf=B{yind,1};
        bparam=B{yind,2};
```

در این قسمت دیده می شود که پارمتر خروجی کار برایش یک حلقه تعریف شده تا تمامی قوائد بر روی آن بتوانیم پیاده سازی کنیم در شکل بعدی هم همین روال را برای پارامتر A گرفتیم به عنوان ورودی که بتوانیم ماتریس قوانین و محاسبه اونو داشته باشیم و در نهایت S که ارزش یا درجه صحت هر کدام از قوانین در هر یک از مراحل iام به حساب میاید را برایش یک حلقه تعریف کرده و محاسبات خود را بر روی آن انجام میدهیم .

حال برای این که بر اساس ارزش هر کدام از یک اون قائده ای که بیشتر است را بماند و اونی که درجه صحت کم تری دارد حذف شود باید این حلقه زیر را تولید کر بر روی nRule که داریم :

```
%% 4: Delete Extra Rules

R=zeros(size(S));
Keep=true(size(S));
for r=1:nRules
    [Smax, R(r)]=max(S{r});
    if Smax==0
        Keep(r)=false;
    end
end
```

این یک تیمه برای این میباشد دوتای ستون آخر مربوط به and /or و weight است را یک نگه داریم در کار خود.

```
Rules=[XIND R];
Rules(:,end+1)=1;
Rules(:,end+1)=1;

Rules=Rules(Keep,:);
```

حال تمامی این ها را انجام دادیم باید تبدیل بشود به یک سیستم فازی که برای تبدیل شدن سیتسم فازی کاری ساده در پیش داریم که برای این کار ابتدای امر newfis را فراخوانی میکنیمو آرگومان های آن را مثل قبل پر میکنیم .

```
%% 5: Create FIS

fis=newfis('Lookup Table FIS','mamdani');

for i=1:nx
    fis=addvar(fis,'input',['x' num2str(i)],[min(x(:,i)) max(x(:,i))]);
    for j=1:nmf(i)
        fis=addmf(fis,'input',i,...
            ['A(' num2str(i) ',' num2str(j) ')'],...
            A{i}{j,1},A{i}{j,2});
    end
end

fis=addvar(fis,'output','y',[min(y) max(y)]);
for j=1:nmf(end)
    fis=addmf(fis,'output',1,['B' num2str(j)],B{j,1},B{j,2});
end
```

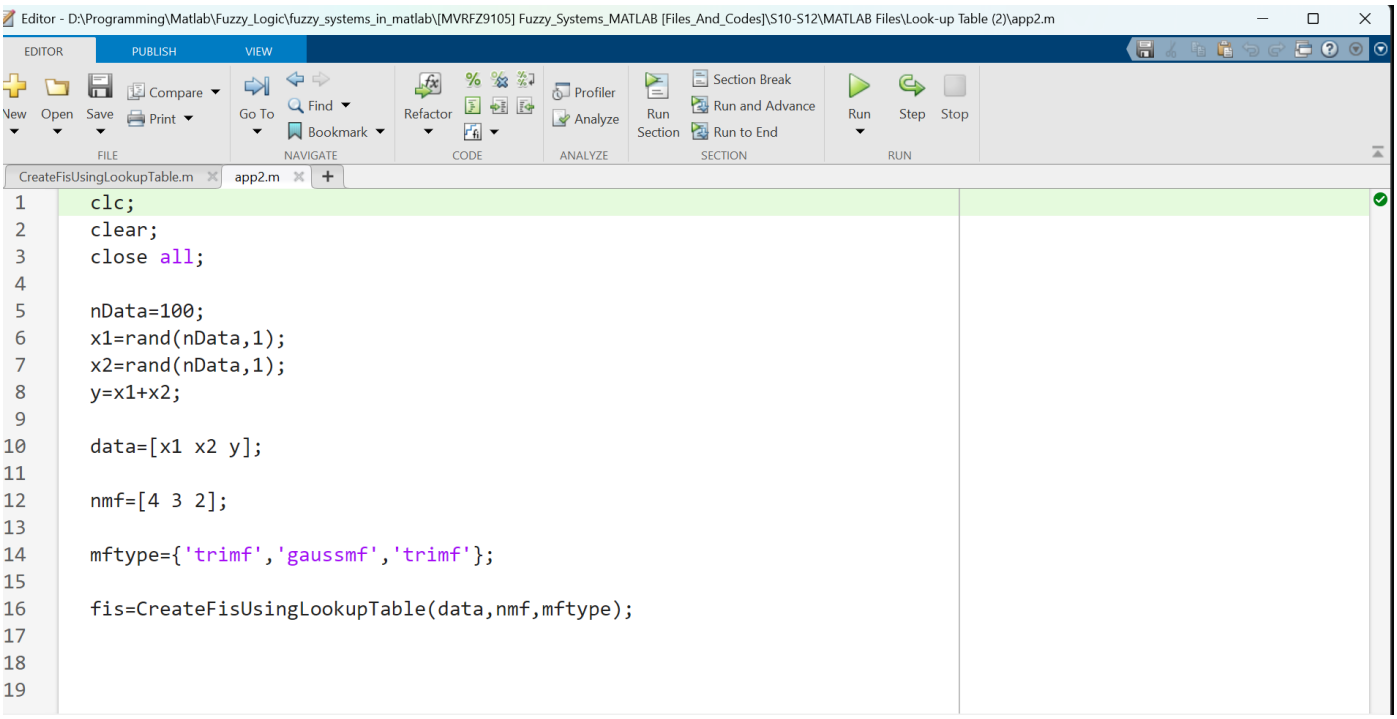
خب همان طور که میبیند روال کاری همان قبلی است تابع addvar برای دیتا دادن به سیستم فازی خود است addmf برای فهماندن تابع بهع سیستم فازی خود است حال برای جزیئات بیشتر کد ادامه مینویسم.

دوتا حلقه داریم یکی برای اضافه کردن متغیر هیا ورودی هست که میگوید از اولین متغیر تا nx را اضافه به سیستم فازی به صورت input و متغیری به اسم x هست که تبدیلش کن از عدد به رشته چون این متغیر در addvar باید رشته شود سپس میگوییم ،تمامی سطر ها و ستون i ام min ,max ورودی های خود .

حالا باید به متغیر هایی که وارد کردیم مقدار دهیم که پارامتر A به عنوان ورودی ما است برای این کار تبدیل عدد به رشته را انممجام میدهیم سپس تمام مقادیر ممکن را که در قبل حساب کرده بودیم را ارجاع به ورودی میدهیم.

برای خروجی هم همین رول کاری را داریم باید یک لوپ بنویسم و تمامی خروجی ها را اضافه کنی.

در نهایت قوانین را هم اضافه میکنیم و برنامه را برای یک اسکریپت که نوشته شده میتوانیم از این تابع بهره ببریم که فایل اسکریپتی به شکل زیر است :



در

ادامه برای این بخش کد های مربوطه آندر ریپازیتوری گیت هاب آپلود شده است .