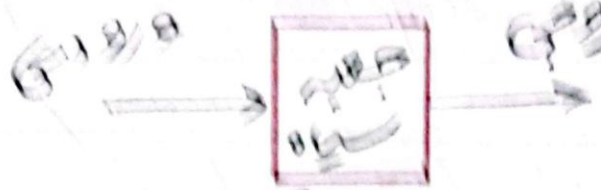
جلسه دهم منطق فازی

ما تا الان یک سیستم بر مبنای منطق فازی طراحی کرده ایم. یکی از منابع خیلی خوب برای طراحی سیستم های فازی Expert ها یا خبره ها هست که در آن سیستم ها وجود دارد حال فرض کنید این Expert ها را نداریم حال به مدلسازی آن سیستم که می خواهیم طراحی کنیم دسترسی داریم اما نه به محتویات آن جعبه آبی به محتویات دسترسی نیست در نگاه منطق فازی نیازی نیست که حتما مدلسازی عالمی یا فرمولی آورده شود.



طراحی سیستم های ترکیبی مبتنی بر داده های ورودی و خروجی.

۱- روش طراحی مبتنی بر جدول راجع: جستجو Look up table

بین اطلاعات را پیدا می کنیم و در نهایت تصمیم می گیریم بر اساس اطلاعات گذشته یا نداشته.

فرض بر این است که داده ورودی و خروجی به صورت زوج مرتب هایی داریم:

(ورودی، خروجی)

(x_p, y_p) $\quad p = 1, 2, \ldots, P$

$x_1 \longrightarrow y_1$ $\quad x_2 \longrightarrow y_2$.

نحوه عملکرد مبتنی بر Lookup table به صورت است:

در ابتدای کار باید x ها را یا ورودی ها را

به صورت membership ها تبدیل کنیم.

$$\begin{cases} x_i^{min} = \min_p x_{ip} \\ x_i^{max} = \max_p x_{ip} \end{cases} \rightarrow x_{ip} \in [x_i^{min}, x_i^{max}]$$

$$\begin{cases} y^{min} = \min_p y_{ip} \\ y^{max} = \max_p y_{p} \end{cases} \rightarrow y_p \in [y^{min}, y^{max}]$$

حال برای این سیستم داریم:

X ورودی ها: $[x_1^{min}, x_1^{max}] \times [x_2^{min} : x_2^{max}] \dots \times [x_n^{min} : x_n^{max}]$

Y: $Y = [y^{min} \quad y^{max}]$

KANDOO

حال هدف ما این است که یک نگاشت

داشته باشیم از ورودی به خروجی‌ها:

$$f: X \longrightarrow Y$$

یعنی این نگاشت به نحوی باشد که با اعمال اغتشاش

ماکزیمم‌ترین فاصله را ایجاد کند.

حال ما هر مجموعه از x_i های max و min می‌خواهیم

یک توزیع نازی به هر عضو را ایجاد کنیم مثلاً کاوس

با عنایتی یا هر روش دیگر

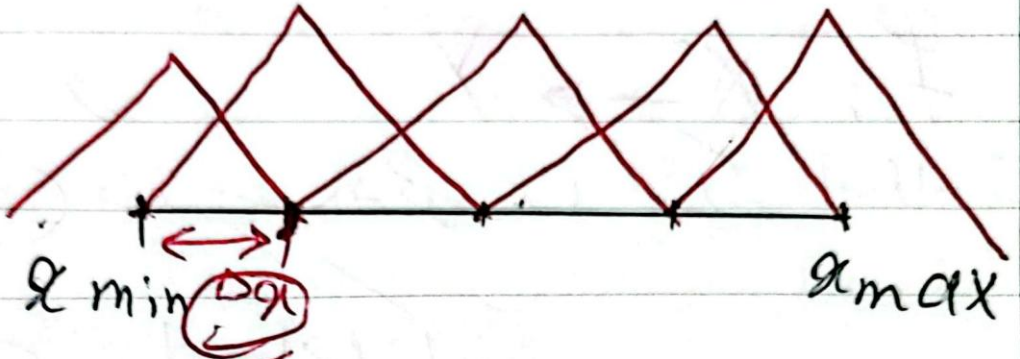
$$\forall x_i \in [x_i^{min}, x_i^{max}] \; ; \; \exists j : \mu_{A_{ij}}(x_i)$$

یعنی این که بین دو بازه حداقل و حداکثری

با درجه عضویت نسبتی می‌شود باید همه x_i ها بین

مثلا برای این ② ی خواهیم ④ فازیشن بندی

نیم. بین $x_{i\max}$ $x_{i\min}$



می‌خواهیم این ها را تعلیم کرد:

$$\Delta x = \frac{x_i^{max} - x_i^{min}}{n-1}$$

مثلا مطلب از کتاب Linspace

یک قاعده حسابی بین عضوهای

ماوی.

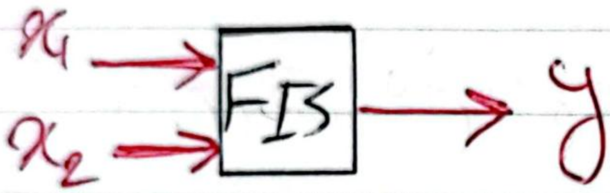
$$x_i \longrightarrow \underbrace{A_{i1}, A_{i2}, \dots A_{icmi}}_{c\,m_i}$$

$$y \longrightarrow \underbrace{B_1, B_2, \dots B_{my}}_{c\,m_y}$$

تعداد کل قواعد این سیستم :

تعداد کل قواعد : $m_1 \times m_2 \times \dots \times m_n \times m_y$

فرض کنیم می‌خواهیم یک سیستم ۲ ورودی و یک خروجی



Rule : If x_1 is A_{1,lr_1} and x_2 is A_{2,lr_2}

Then y is B_{lr_y}

(x_{1p}, x_{2p}, y_p)

می‌خواهیم ببینیم این رول یا قاعده چقدر

می‌تواند سیستم فازی مجموعه شرایط یا دیتا را تعمیم

دهد.

یعنی از بین این قاعده ها خواهیم تصمیم.

صحت Rule: $\mu_{A_1 B_1}(x_1) \wedge \mu_{A_2 B_2}(x_2) \wedge \cdots \wedge \mu_{B_1 B_2}(y)$

$\wedge \rightarrow$ ضابطین

این شد صحت Rule ما.

حال می توانیم برای هر چند عدد ای که داریم

درجه صحت جمشون تعلق دهیم به

$$S_{total}^{r} = \sum_{p=1}^{p} S_r(x_p, y_p)$$

یا

$$S_{total}^{r} = \max_{p} S_r(x_p, y_p)$$

بیشینه کردن

If x_1 is A_{11} and x_2 is A_{22} Then y is B_3

If x_1 is A_{11} and x_2 is A_{25} Then y is B_4

حال کدام رول بیشتر شدن را فعال می‌کند.

حال می‌آییم و به خاطر این در رابطه‌ی S_{total} هر کدام بیشتر شدن را فعال می‌کنیم.

برای سیستم منطق فازی Lookup جدولی

نوشته شده باید خودمان کد نویسی کنیم.

طراحی مبتنی بر جدول ارجاع :

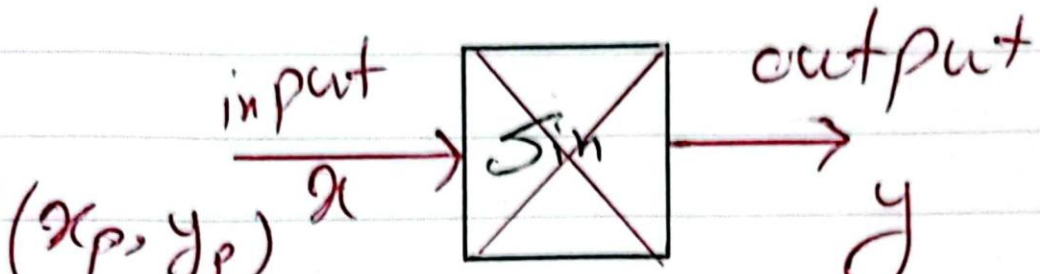
۱_ تقسیم بندی و تعریف مجموعه های فازی برای ورودی ها و خروجی ها

۲_ تشکیل پوشش ممکن تمام

۳_ محاسبه امتیاز هر کدام از قواعد

۴_ حذف قواعد متضاد و ضعیف تر

حال ما می خواهیم تابع (Sin) را در سیستم فازی به کمک جدول مبتنی بر ارجاع کار کنیم در منطق فازی.



فرض
می کنیم نمی دانیم
مثل یک جعبه سیاه Blackbox

یک سری زوج مرتب ها را داریم برمی داهیم تقریب
بزنیم تابع را به روش منطق فازی

در حقیقت مثلب بخواهیم از درجه member ship functions

کنیم ابتدا روش تولید تابع دوم که همین
کار را کرده ایم.

خب در این بخش از درس ما به تحلیل منطق فازی یا طراحی سیستم فازی به روش Look Up Table میپر دازیم .

خب همان طور پیشتر هم که گفته شد یکی از کاربرد های اصلی منطق فازی و یا طراحی سیستم های فازی تقریب توابع غیر خطی یا توابعی که هیچ اطلاعاتی از آن نداریم و فقط یک سری ورودی و خروجی از آنها داریم و مانند یک جعبه سیاه می‌باشد که باید با کمک سیستم های فازی که شامل یک سری قوانین منحصر به فرد خودش است تقریب بزنیم .

حال نرم افزار متلب تابع خاصی برای این کار طراحی نکرده اما ما میتوانیم با یک سری الگورتیم های خاصی که خودمان طراحی میکنیم تقریب بزنیم .

روش طراحی بر اساس جدول ارجاع یا همان Look Up Table میتوان به جندتا گام تقسیم بندی کرد تا این گام ها در برنامهه نویسی به صورت الگورتیمیک بنویسیم :

1- تقسیم بندی و تقریب مجموعه های فازی برای ورودی و خروجی ها
2- تشکیل تمام قوائد ممکن
3- محاسبه امتیاز هر کدام از قوائد ها
4- حذف قوائذد متضاد و ضعیف تر

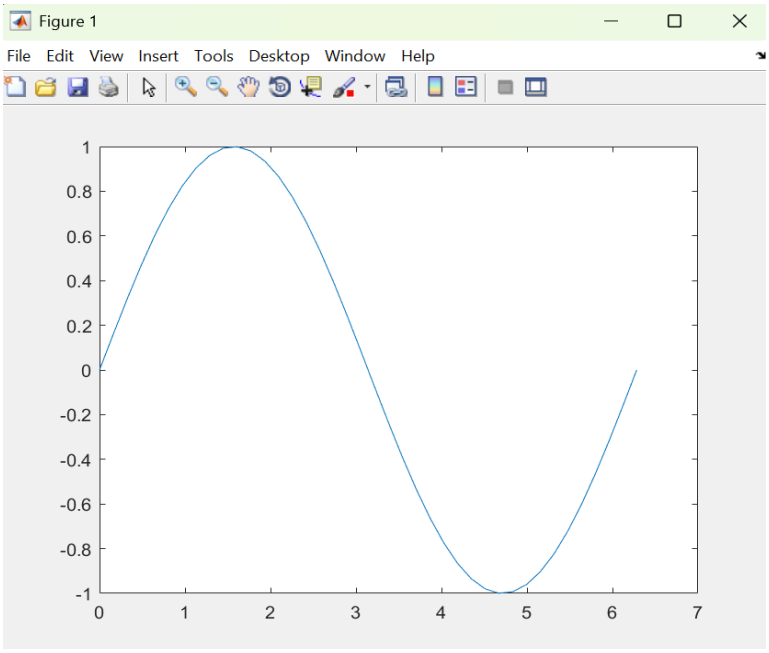
خب این مراحل که گفته شد برای جدول ارجاع یک سری گام هستند که خیلیل مهم و اساسی نیز میباشند .

حال ما برای یک تابع sin (x) میخواهیم تقریب را با این روش انجام دهیم در نرم افزار متلب خب برای این کار ابتدای امر خود تابع را در بین یک باز 0 تا 2p را در محیط متلب با خود تابع اصلی خودش رسم میکنیم و خروجی کار را در میاوریم .

که به شکل زیر است :

```
%% 0: Generate Data

f=@(x) sin(x);

xmin=0;
xmax=2*pi;
P=40;

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

y=f(x);
```

خب این کدی هست که در گام صفرم نوشته ایم و خروجی هم میشه این :



خب چیز خاصی هم نبود و قابل حدس زدن هم هست برای کسی که این متن رو میخونه خروجی تابع sin(x) بین 0تا 2p چه فرمی به خود میگیرد . حال بگذریم در ادامه میخواهیم روند منطق فازی رو شروع کنیم.

در ابتدای کار ما باید membership functions ها رو تشکیل بدیم که برای این کار میتونیم از چند روش بهره بگیریم که بهترین راه این میباشد به صورت یکگ تابع جداگانه نشوته شود در یک m فایل تابعی جداگانه و در این فایل استفاده شود مثل کاری که در اینجا ما انجام داده ایم :

```
%% 1: Create Membership Functions

nA=20;
A=CreateMembershipFunctions(x,nA,'gaussmf');

nB=10;
B=CreateMembershipFunctions(y,nB,'gaussmf');
```

خب توابع نوشته شده را اگر نگاهی کنید میفهمید که عضویت مقادیر ها به صورت گاووسی هستند که ما میتوانیم به فرم های دیگری مانده مثلثی یا ذوزنقه ای یا فرم های دیگر یکه بخواهیم توابع عضویت را در رسم کنیم .

حال تابع جداگانه ای که برای این کار نوشته ایم را شکل کاریش برای شما میاوریم :

```
function out=CreateMembershipFunctions(x,n,Type)

    if nargin<3
        Type='';
    end

    if isempty(Type)
        Type='trimf';
    end
```

همان طور در شکل دیده می شود تابع ما 3 تا وردی به خود میگیرد یکی برای مقادیر X های ما است N تعداد پاتیشن بندی نقاط در توابه عضویتی که تعین میکنیم که چند بخشی شوند و. آرگومان سوم برای این است که تایپ کاری ما چی باشد برای membership functions های ما یعنی از جنس توابع مثلثی کار کنیم که یک سری خواص را د ارد یا گاوسی که اونم برای خودش یک سری خواص دارد در بخش باید نام توابع را بیاوریم مثل gussmf or trimf این دو توابعی هستند که برای membership functionsهای ما تعریف شده اند .

```
a=linspace(xmin,xmax,n);

out=cell(n,2);
```

در ادامه این قسمت تابع برای ما محدوده متغیر های ما رو بر اساس تعداد n که وارد میشود بین حداقل و حداکثر بازه تقسیم بندی مساوری میکند که اینجا برای ما حداقل 0 تا 2p است که به تعداد دلخواد ه به آرگومان که میدهیم در فایل اصلی میتوانیم اینو تقسیم بندی کنیم .

خروجی هم به صورت سلولی در نظر گرفته ایم که بتوانیم هر چیزی در آن بتوانیم بگذاریم . خروجی n تا سطر داره و 2 تا ستون که برای خب n سطر برای تعداد خروجی های ما میباشد ولی 2 تاس ستون برای اینم هست خروجی ما یک قسمت درای عدد است و یک قسمت هم نوع تابع است که به صورت رشته تعریف شده به همین علت دارای nسطر 2 ستون است سلول ما .ادامه تابع CreateMembershipFunctions چیزی نیست جز تعریف تابع مثلثی و گوسی برای کار ما که بتونیم membership ها رو اضافه کنیم .

در گام بعدی تعداد قوانین را که وجود دارد را مینویسم که به تعداد ورودی و خروجی ما قوانین داریم که اینا به فرم زیر مینویسیم :

```
%% 2: Create Rules Matrix

S=zeros(nA,nB);
```

خب بعد از این کار قوانین حساب و کتاب کردن این قائده ها را پیاده سازی میکنیم که به فرم زیر است :

```
%% 3: Calculate Rank of Rules

for ai=1:nA
    amf=A{ai,1};
    aparam=A{ai,2};

    for bi=1:nB
        bmf=B{bi,1};
        bparam=B{bi,2};

        s=zeros(1,P);
        for p=1:P
            s(p)=feval(amf,x(p),aparam)*feval(bmf,y(p),bparam);
        end
```

خب حلقه for اول برای ورودی های ماست دومی برای خروجی و سومی برای میزان صحت یا همون درجه صحت کار است روند حلقه هم بر روی ارایه های سلولی کار میکند . feval برای این بتوانیم بر روی آرایه های سلولی خود کنترل داشته باشیم و هر یک از آرایه های سلولی نظیر به نظیر در هم ضرب شوند در ورودی و خروجی تا نتیجه S که صحت کار ما است برای این درجه صحت این مقدار چقدر میباشد به درستی انجام شود ما این کارا میکنیم.

در ادامه کار ما یک رون د داشتیم که از اول کار برای این متد از منطق فازی باید عمل کنیم حذف کردن یک سری از قوانین که درجه صحت آن ها کم تر از دیگری است در یک بازه معین خب برای این کار این کد نوشته شده زیر است :

```
%% 4: Delete Extra Rules

[~, ind]=max(S,[],2);

Rules=[(1:nA)' ind];

Rules(:,3)=1;
Rules(:,4)=1;
```

خب ما برای این کار به یک ماتریسی لازم داریم که صحت های با ارزش مکتان های آن ها را به ما دهد برای این کار این فرم هایلایت شده به کار میبریم که ماکسیمم S را میگیرد دوم خالی گذاشتیم که نمیخواهیم با چیزی قیاس شود حداکثر ممکن

و سوم به صورت ستونی میخواهیم یعنی عدد2 را وارد کرد ه ایم . اون تیلادایی که در سمت چپ است برای این میباشد که ما به مقدار max نیازی نداریم .

اگر یادتون باشه در جزوه جلسه قبلی هم نوشتم که قوانین برای منطق فازی به صورت یک ماتریس بود m+n+2 که m ستون برای ورود ی n ستون برای خروجی بود و همچنین دو ستون آخر این ماتریس 1 بودند چون weight آن یک است دوم برای عملگر and از عدد یک استفاده میشد که رنگ هایلات قرمز نشان دهدنده این موضوع است .

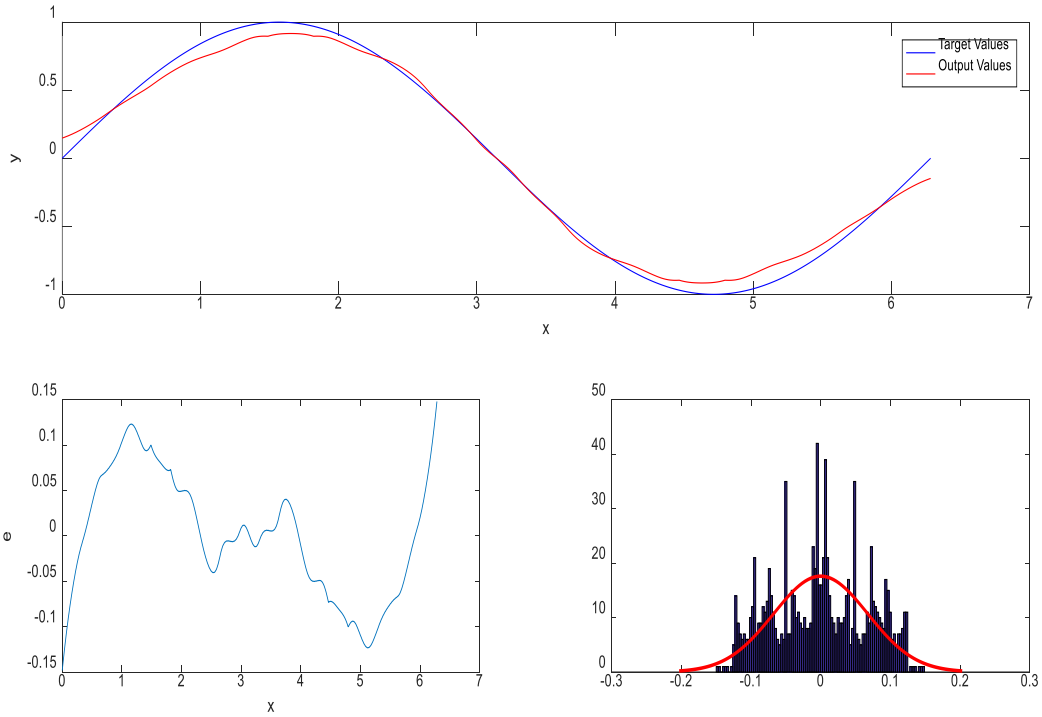
در ادامه ما تمامی این قوائد و کار هایی که انجام داده ایم را به سیستم فازی خود اضافاه میکنیم همانند جلسه های قبلی که از تا newfis استفاده میکردیم و یکسری قوانین و کار ها داشت انجام میدادیم . روند کاری تمامی همان روش قبلی است برای کار کردن با ماژول فازی متلب :

```
%% 5: Create FIS

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

fis=addvar(fis,'input','x',[min(x) max(x)]);
for ai=1:nA
    fis=addmf(fis,'input',1,['A' num2str(ai)],A{ai,1},A{ai,2});
end

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

fis=addrule(fis,Rules);
```

فقط برای این که نخواهیم Rule ها را ار دفعه دستی وارد کنیم از حلقه بکار برده ایم که این عملیات دستی انجام نشود . و در نهایت هم قائده mamdani برای سیستم فازی خود بکار برده ایم .

در ادامه کد نویسی برای پلات کردن و ترسیم است .

در ادامه میتوان با تغیر تعداد عدد nA , nB که تعداد ورودی و خروجی هالی سیستم ما است تقریب تابع خود را بیشتر به آن چیزی که وجود دارد سیستم اصلی ما نزدیک تر شود اما معمولا یک سری فواصل و خطا هایی وجود دارد که به علت این میباشد که قوانین ما درست نباشد یا تعداد ورودی و خروجی ما کم است ولی در کل تقریب زده میشود خروجی کار را برای این کد من اینجا میگذارم :



خب رفتار تابع تقریب زده و اصلی و همچنین گستره دیتا هارا سمت راست پایین میتوان دید و خطا ها را هم میتوان روی شکل اصلی دید.

من برای این گزراش دوتا کد را به اشتراک میگذارم که به اسم app1.m , CreateMembershipFunctions.m است که دومی تابع تعریف شده و باید در کنا دایرکتوری کدد app1 برای ران شدمن قرار بگیرد .