Cvičení 7

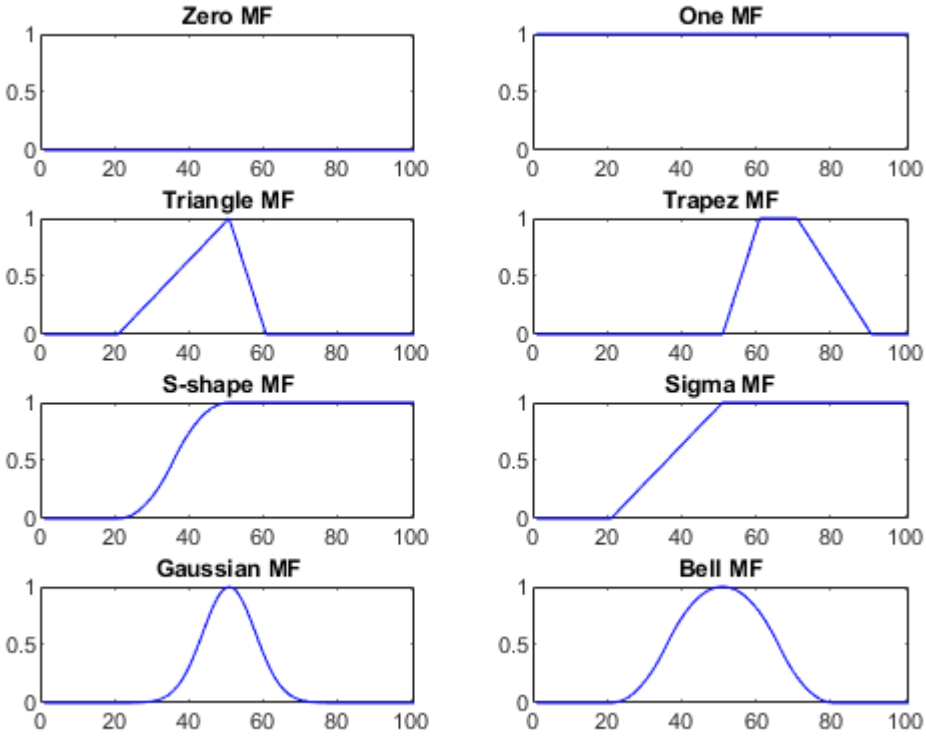
Fuzzy množiny

membership funkce:

`zeromf()`, `triangmf()`, `trapezmf()`, `sshapemf()`, `sigmamf()`, `gaussmf()`, `onemf()`, `bellmf()`

Funkce jsou jen přímou implementací definice těchto membership funkcí.

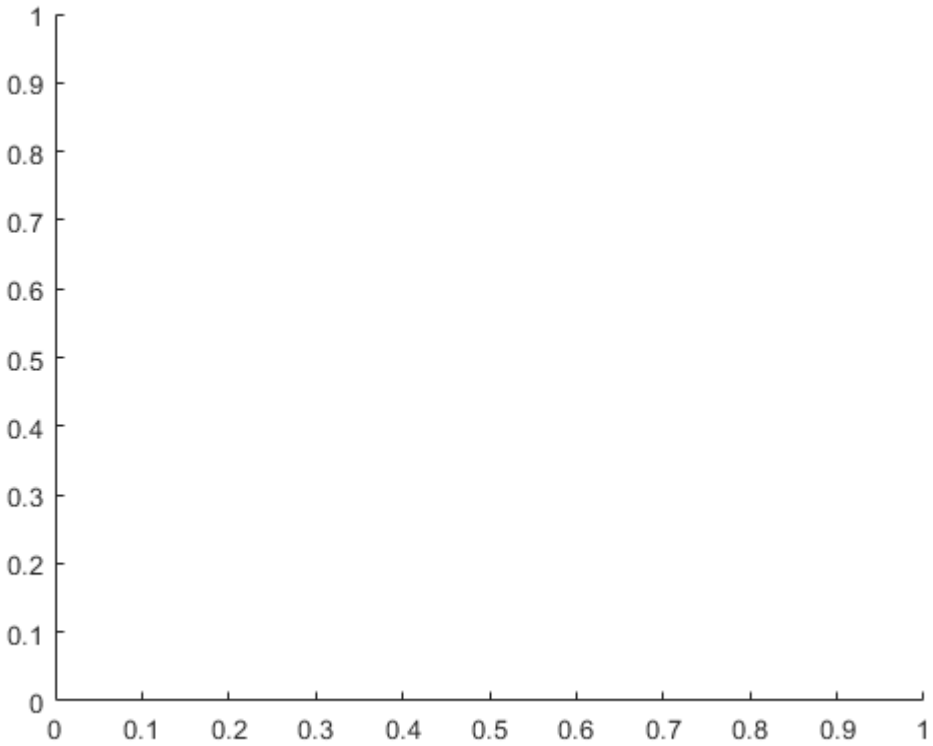
```
% Univerzum
z = 0:100;

% Membership funkce
uA = zeromf(z);
uB = onemf(z);
uC = triangmf(z,20,50,60);
uD = trapezmf(z,50,60,70,90);
uE = sshapemf(z,20,50);
uF = sigmamf(z,20,50);
uG = gaussmf(z,20,50,10);
uH = bellmf(z,20,50);

figure,
subplot(4,2,1), plot(uA, 'b');
ylim([0 1])
title('Zero MF')
subplot(4,2,2),  plot(uB, 'b');
ylim([0 1])
title('One MF')
subplot(4,2,3),  plot(uC, 'b');
ylim([0 1])
title('Triangle MF')
subplot(4,2,4),  plot(uD, 'b');
ylim([0 1])
title('Trapez MF')
subplot(4,2,5),  plot(uE, 'b');
ylim([0 1])
title('S-shape MF')
subplot(4,2,6),  plot(uF, 'b');
ylim([0 1])
title('Sigma MF')
subplot(4,2,7),  plot(uG, 'b');
ylim([0 1])
title('Gaussian MF')
subplot(4,2,8),  plot(uH, 'b');
ylim([0 1])
title('Bell MF')
```



Operace

Komplement

```
figure,
hold on
```



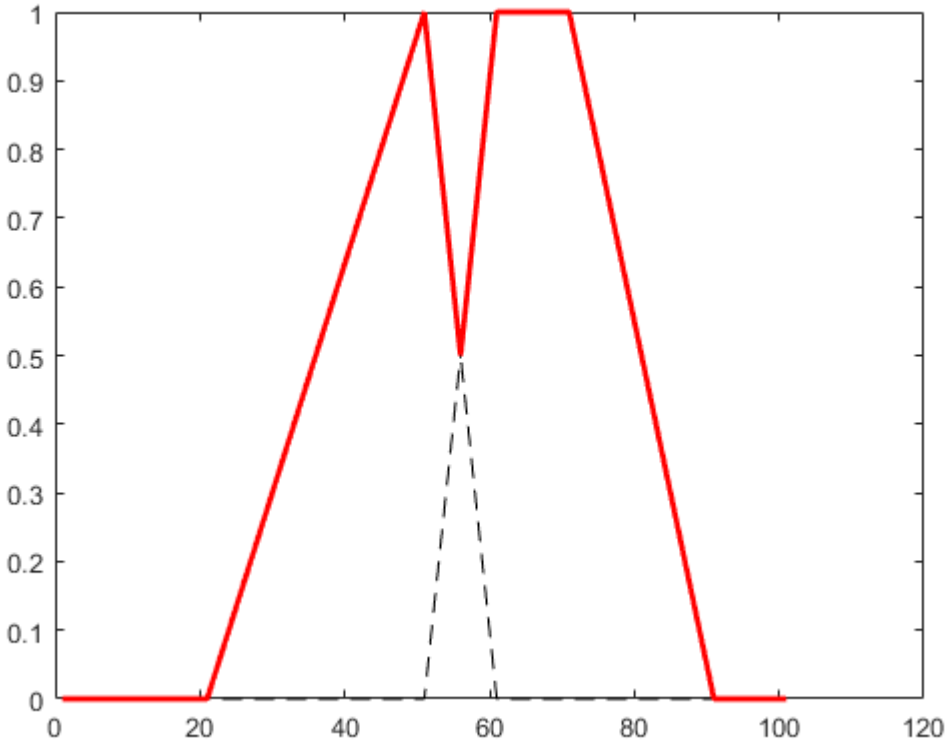

```
plot(uD, 'k--');
```

```
Unrecognized function or variable 'uD'.
```

```
p = plot(1-uD, 'r');
p.LineWidth = 2;
```

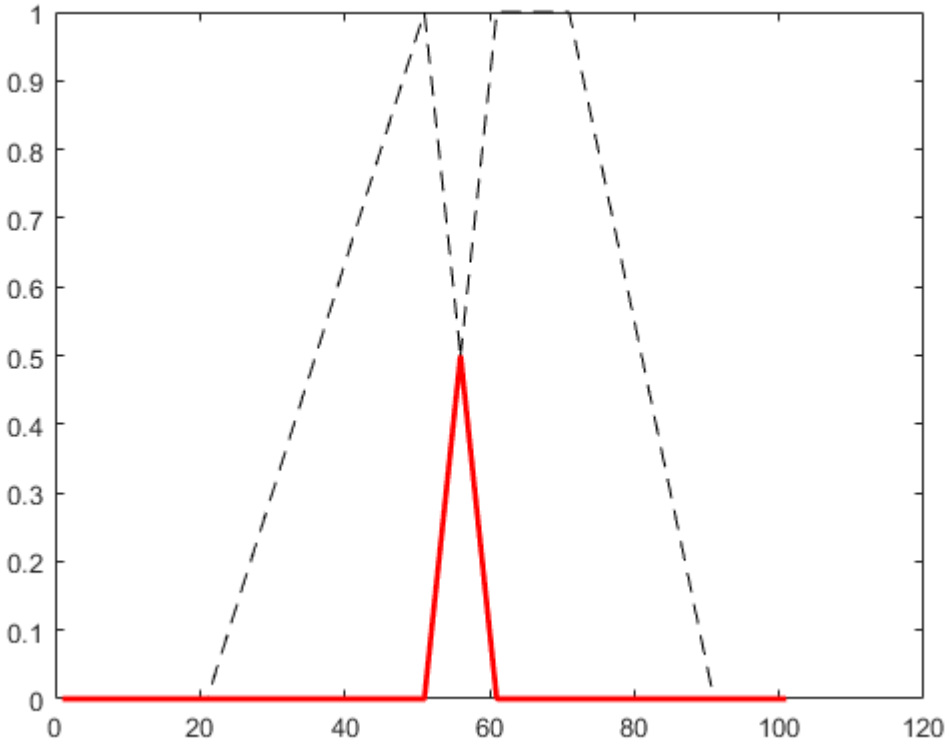
Spojení

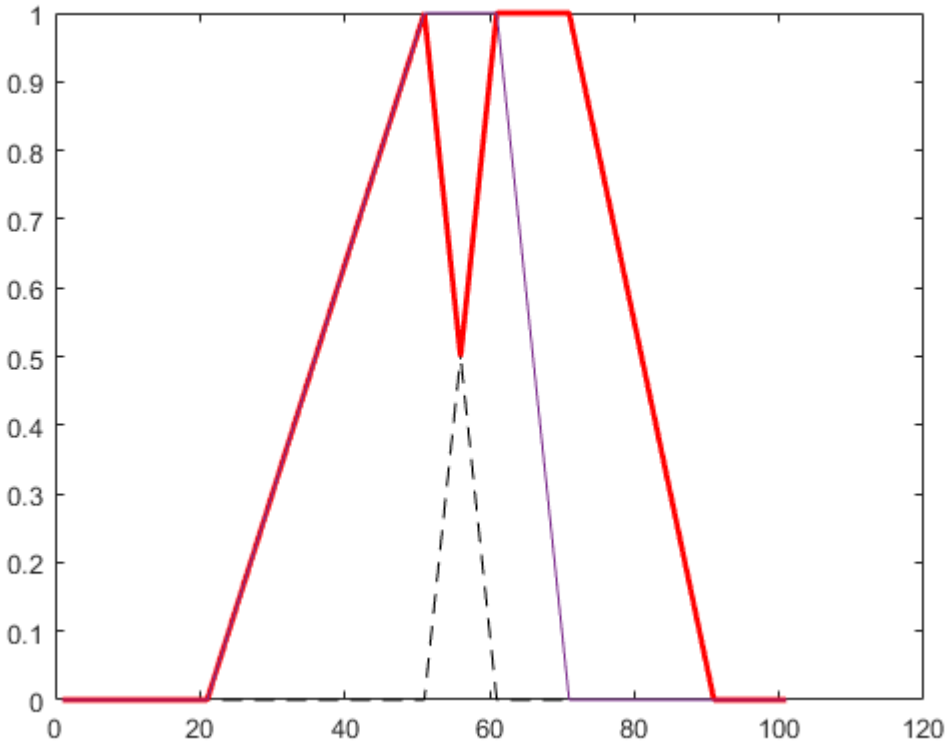
```
figure,
plot(uC,'k--');
hold on
plot(uD,'k--');
p = plot(max(uC,uD),'r');
p.LineWidth = 2;
```



Průnik

```
figure,
plot(uC,'k--');
hold on
plot(uD,'k--');
p = plot(min(uC,uD),'r');
p.LineWidth = 2;
```





Fuzzy systémy

Motivace: Předpokládejme, že máme systém, který měří (vyhodnocuje stav) motoru napájecí stanice. Stav se vyhodnocuje jako množství vibrací, které produkuje.

Z ... průměrná frekvence vibrací

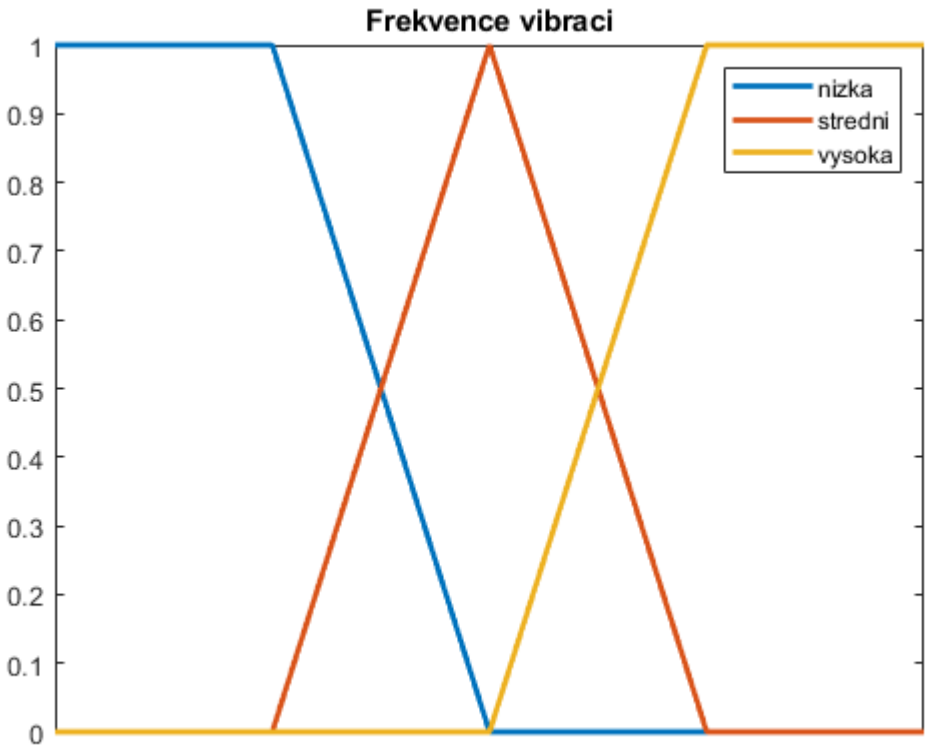
(3 rozsahy frekvencí: - nízká - střední - vysoká)

Řekneme, že motor pracuje normálně, pokud jsou vibrace v nízkém rozsahu, částečně, pokud jsou vibrace ve středním rozsahu, špatně (je blízko selhání), pokud jsou vibrace ve vysokém rozsahu.

Tyto pojmy budeme fuzzyfikovat - ze slovního popisu (linguistic variable) uděláme pravidla

Naše vědění (knowledge) převedeme na if-then pravidla

```
nizka = [ 1 1 1 0.5 0 0 0 0 0]; % u_nizka
stredni = [0 0 0 0.5 1 0.5 0 0 0]; % u_stredni
vysoka = [0 0 0 0 0 0.5 1 1 1]; % u_vysoka

figure,
p = plot(1:9,nizka,1:9, stredni, 1:9, vysoka); % vykresleni funkci
p(1).LineWidth = 2; % nastaveni sirky linie
p(2).LineWidth = 2;
p(3).LineWidth = 2;
title('Frekvence vibraci');
legend('nizka','stredni','vysoka');

set(gca,'xtick',[]);
```



Pravidla:

Pokud je frekvence nízká, pak funkce motoru je normální.

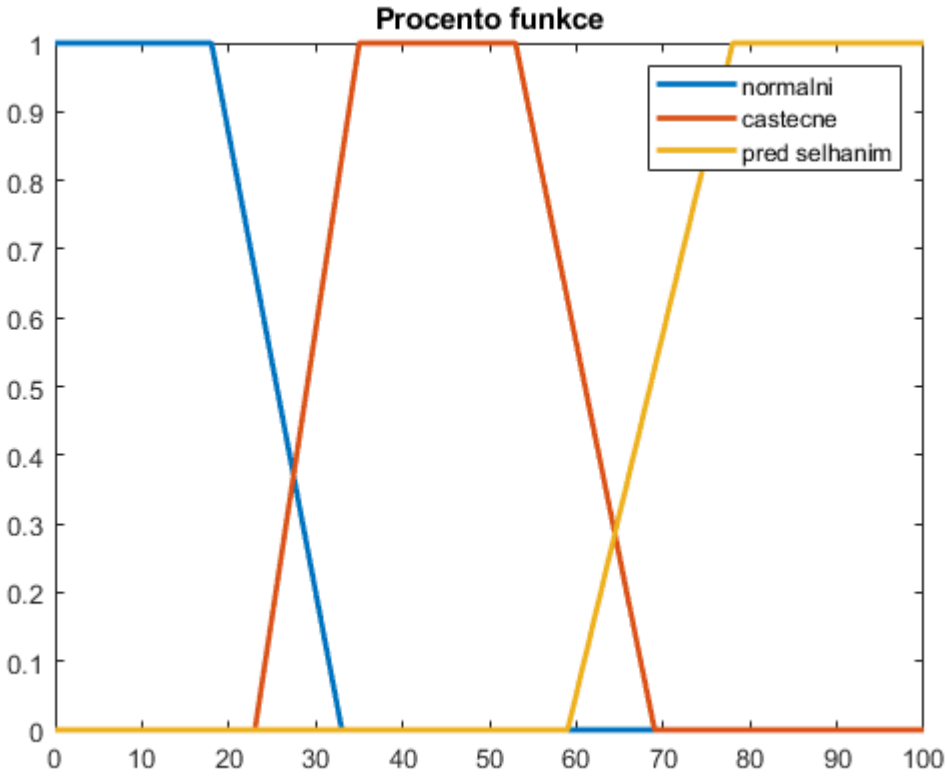
Pokud je frekvence střední, pak je funkce motoru částečná.

Pokud je frekvence vysoká, pak je funkce motoru před selháním.

Dalším krokem je najít způsob, jak z konkrétních hodnot a knowledge vypočítat výsledek systému (implikace / inference).

Vstupy jsou fuzzy -> výstupy také (musíme i výstupy definovat jako fuzzy množiny) procento nenormální funkce, čím nižší, tím lépe.

```
procento = [0 : 1 : 100];
z=0:0.01:1;

normalni = 1 - sigmamf(z, 0.18, 0.33);
castecne = trapezmf(z,0.23,0.35,0.53,0.69);
pred_selhanim = sigmamf(z, 0.59, 0.78);

p = plot(procento,normalni,procento, castecne, procento, pred_selhanim); %
vykresleni funkci
p(1).LineWidth = 2; % nastaveni sirky linie
p(2).LineWidth = 2;
p(3).LineWidth = 2;
title('Procento funkce');
legend('normalni','castecne','pred selhanim');
```



Předpokládejme, že pro konkrétní z0 máme následující příslušnosti k jednotlivým funkcím:

```
u_nizka_z0 = 0.8;
u_stredni_z0 = 0.2;
u_vysoka_z0 = 0;
```

Výstup korespondující s pravidlem 1:

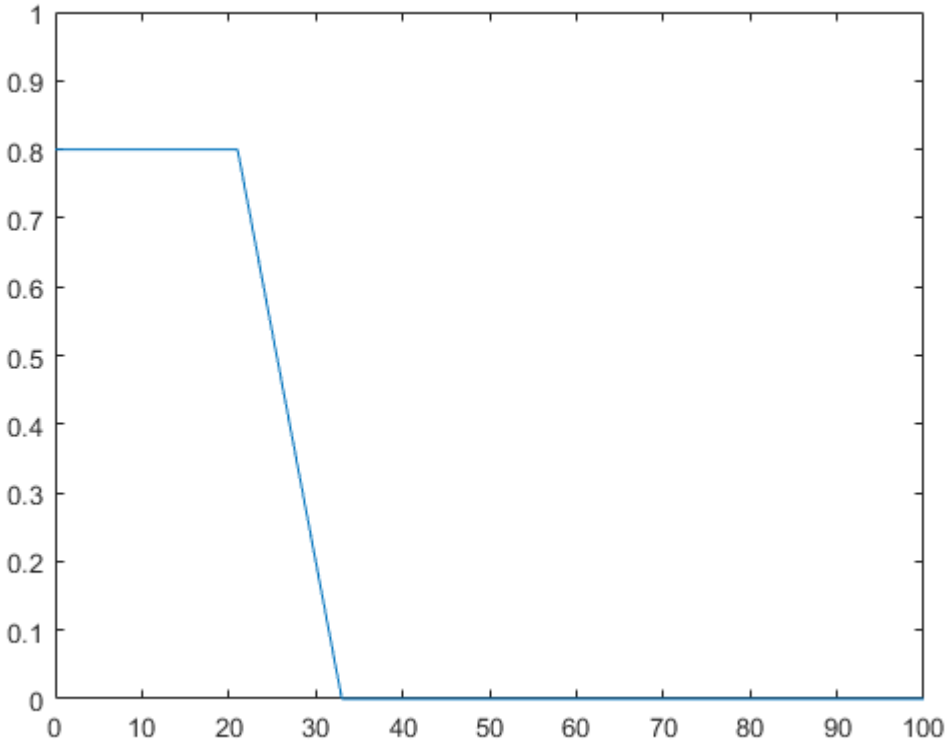
Pokud je frekvence nízká, pak funkce motoru je normální.

Q1(v) = min(u_nizka(z0), u_1(z0,v)) = min(u_nizka(z0), min(u_nizka(z0),u_normalni(v))) = min(u_nizka(z0),u_normalni(v));

Q1 ... fuzzy výstup pro pravdlo 1 a konkretní hodnotu vibrací

u_1(z0,v) ... pravidlo 1 a konkretni hodnota z0

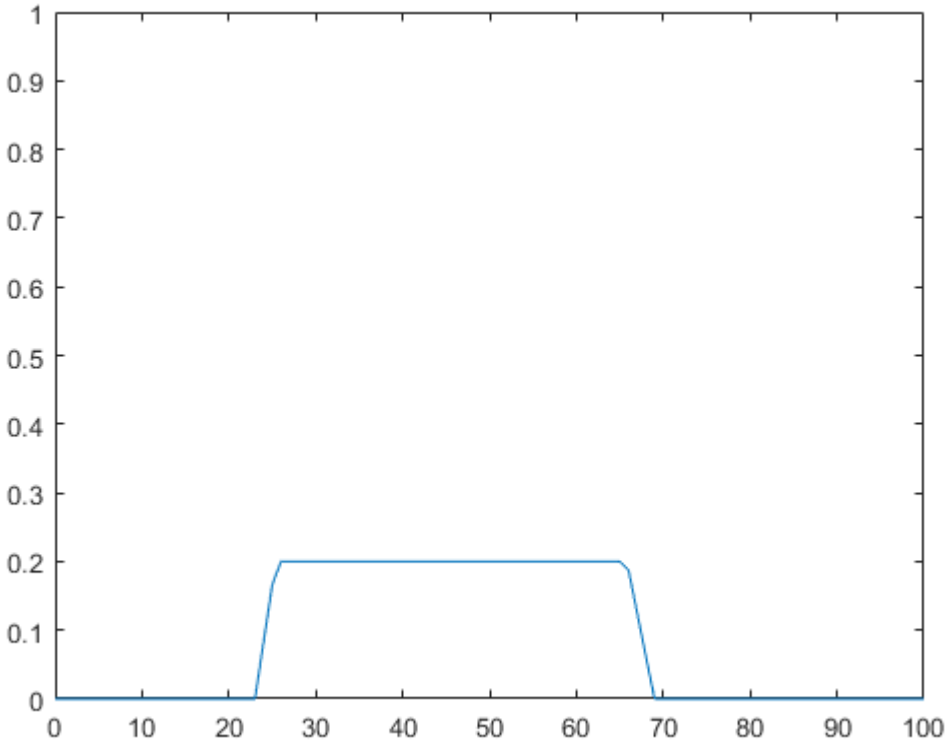
```
figure,
p1 = plot(procento,min(normalni,ones(1,101)*u_nizka_z0));
ylim([0 1])
```



Obdobně pravidlo 2:

Q2(v) = min(u_stredni(z0),u_castecne(v));

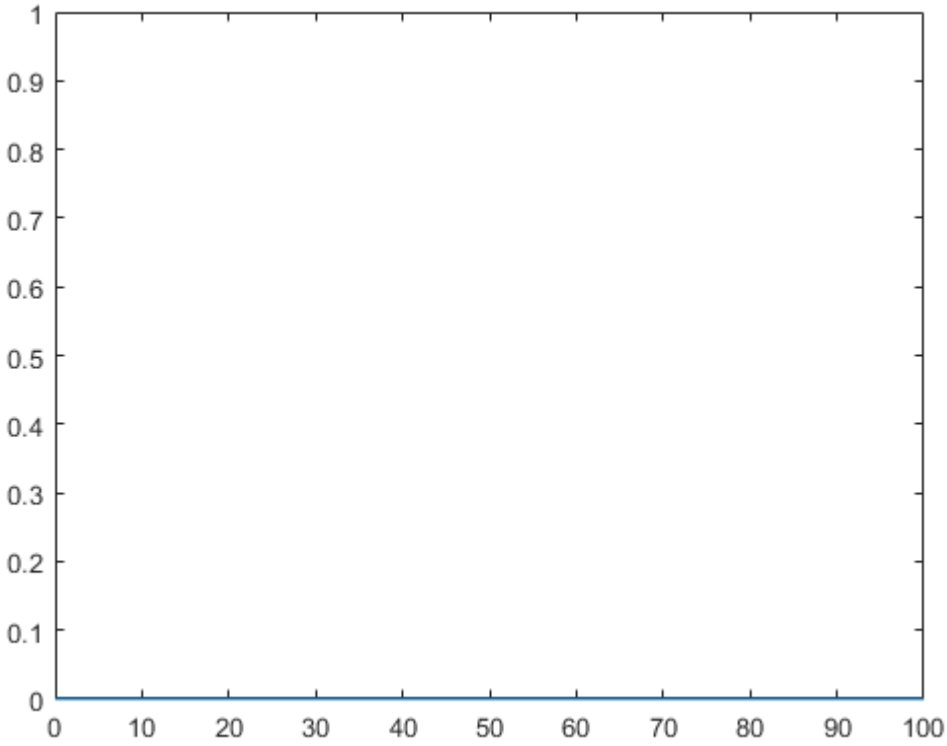
```
figure,
p2 = plot(procento,min(castecne,ones(1,101)*u_stredni_z0));
ylim([0 1])
```



Pravidlo 3:

Q3(v) = min(u_vysoka(z0),u_selhani(v));

```
p3 = plot(procento,min(pred_selhanim,ones(1,101)*u_vysoka_z0));
ylim([0 1])
```

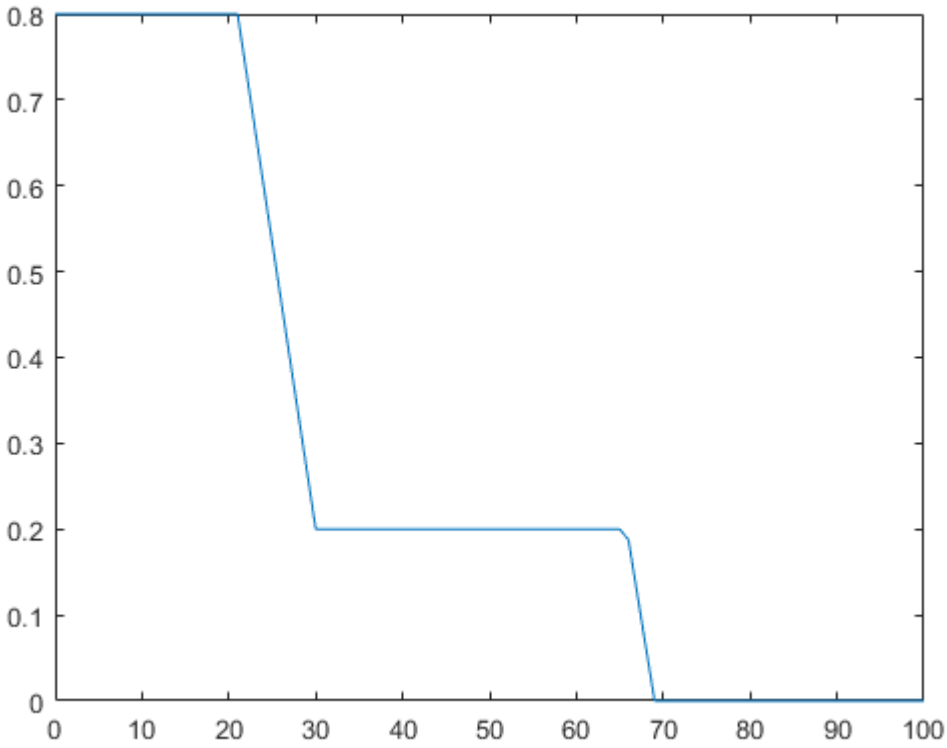


Abychom získali 1 výsledek, musíme použít agregaci 3 výstupů.

Asociujeme je buď pomocí OR nebo AND operace

Q(v) = Q1(v) or Q2(v) or Q3(v)

```
Qv =
max([min(normalni,ones(1,101)*u_nizka_z0);min(castecne,ones(1,101)*u_stredni_z0);mi
n(pred_selhanim,ones(1,101)*u_vysoka_z0)]);
figure,
plot(procento,Qv);
```



Nakonec musíme výslednou fuzzy množinu Q(v) defuzzyfikovat, abychom obdrželi 1 hodnotu např. pomocí center of gravity:

v0 = integral(v Q(v) dv) / integral(Q(v) dv)

Shrnutí: - fuzzyfikace vstupů - fuzzy logické operace - aplikování implikace - agregace - defuzzyfikace

Funkce potřebné k vytvoření fuzzy systému

V následující části je postup, jak vytvořit fuzzy systém v matlabu.

Nested functions

pro práci s fuzzy množinama budeme potřebovat znát tzv. nested funkce Ve stručnosti (co nám stačí znát o nested funkcích pro potřeby pochopení následujícího kódu) nested funkce je funkce, která je defnovaná v těle jiné fukce:

Příklad:

```
function [outputs1] = outer_function(arguments1)

...

function [outputs2] = inner_function(arguments2)

...
```

```
end

...

end
```

```
% Podivejte se na prikad na definici funkce parent
parent();
```

Proměnné definované ve vnější funkci zůstávají ve workspace nested funkce, takže k nim může přistupovati nested funkce. Podívejte se na funkci `main1()`.

```
main1();
```

Proměnná musí být definovaná ve vnější funkci, pak nested funkce mohou s touto proměnnou pracovat (měnit jí), pokud je definovaná v nested funKci, druhá nested funce s ní nemůže pracovat. Zkuste `main2()`. Definujte `x` ve vnější funkci a podívejte se na rozdíl.

```
main2();
```

Funkce pro vytvoření Fuzzy systému

Fuzzyfikace vstupů

Vytvoření MF viz výše.

Dalším krokem je vyhodnocení vstupu -> rule strengths (lambda funkce). Vytvoření funkcí, které představují fuzzyfikované vstupy.

viz. `lambdafcns.m`

tím, že v nich použijeme nested funkci, bude výstupem místo čísla množina funkcí.

Poznámky ke kódu `lambdafcns.m`:

- vytváří množinu lambda funkcí odpovídající množině fuzzy pravidel

- L je pole buněk (cell) ukazatelů na funkce

- `inmf` je MxN matice ukazatelů na membership funkce (M - počet pravidel, N počet fuzzy systémových vstupů)

- `inmf(i,j)` je vstupní membership funkce aplikovaná na i-té pravidlo pro j-tý vstu

- op ukazatel na funkci, kterou se budou kombinovat pravidla `@min / @max` (defaultně `min`)

- každá funkce (output funkce) volá nested funkci `ruleStrength()` s indexem na řádek matice pravidel `i`, následovaná všemi z argumenty (`varargin` označuje všechny další nepojmenované argumenty)

Implikace

Další, co je potřeba definovat je funkce provádějící implikaci.

Viz `implfcns.m`

Funkce potřebuje jako vstup odpověď každého pravidla a množinu korespondujících membership funkcí `lambdafcns` vrací obecné funkce pro pravidla, zde potřebujeme konkrétní vstupy opět zde používáme nested funkce.

Poznámky ke kódu `implfcns.m` :

- vrací množinu implikačních funkcí z množiny lambda funkcí `L`, množiny output membership funkcí (`outmf`) a množiny vstupů `z1, ..., zn`

- `L` je pole buněk rule-strength funkcí

- `outmf` je pole buněk output membership funkcí

- počet prvků `outmf` může být buď shodný s počtem funkcí `L` nebo o 1 více

- pokud je o 1 více - odpovídá to else pravidlu

- každá output funkce volá funkci `implication()` (s identifikátorem `i` odkazující na to, jaká lambda hodnota se má použít)

Agregace funkcí z implikací

viz `aggfcn.m`

Funkce vytváří agregační funkci `QA` z množiny implikací `Q` výsledkem je ukazatel na funkci, která může být volána s konkrétní hodnotou `q = QA(v)`

Defuzzyfikace

viz `defuzzyfy.m`

Výstupem předchozí funkce je fuzzy funkce, tu potřebujeme defuzzyfikovat funkce transformuje agregační fci `QA` na fuzzy výsledek s použitím center of gravity metody `vrange` je 2 rozměrný vektor specifikující rozsah vstupních hodnot po `QA` pokud jsou všude hodnoty rovny `0`, pak `out` je `NaN`, automaticky se vezme střed `vrange`.

Fuzzy systém

Spojíme-li vše dohromady do jedné funkce dostaneme celý systém.

viz `fuzzysysfcn.m`

Funkce bere jako vstup množinu vstupních a výstupních membership funkcí a vrací fuzzy systém, který může být vyhodnocen pro libovolnou množinu vstupů.

`inmf` - MxN matice ukazatelů na membership funkce (M počet pravidel, N počet vstupů)

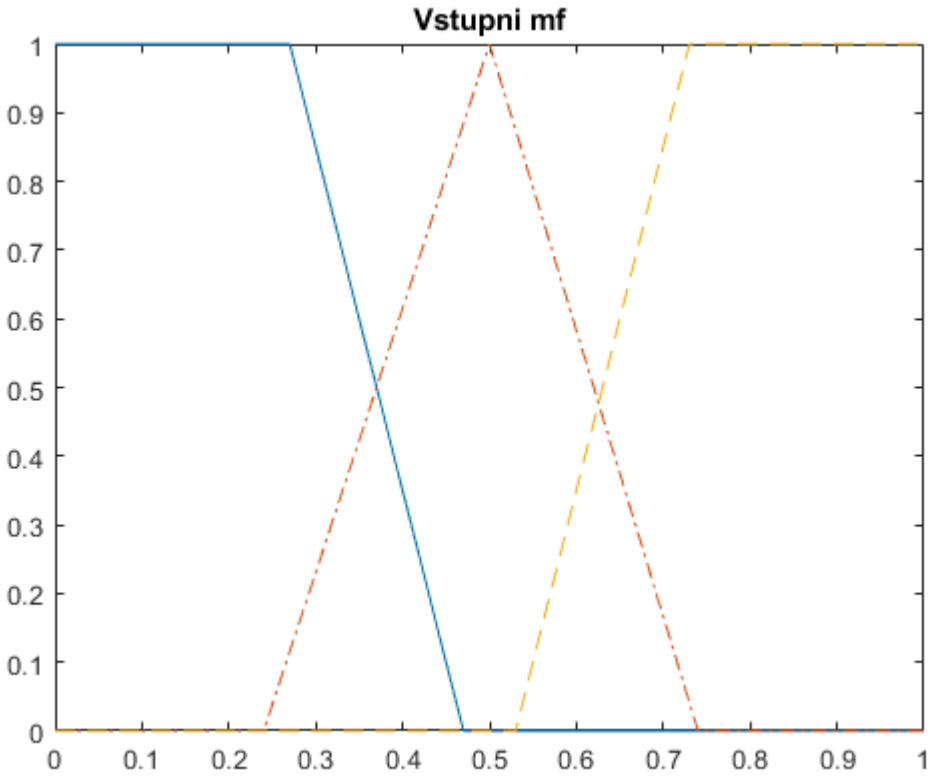
`vrange` - 2 rozměrný vektor specifikující validní rozsah vstupních hodnot pro vstupní membership funkce op je ukazatel na funkci, kterou se budou pravidla spojovat (defaultně `min`) výstup je funkce `F` počítající výstup z fuzzy systému pro danou množinu vstupů.

`F(z1,...,zn)`

Fuzzy system - motor

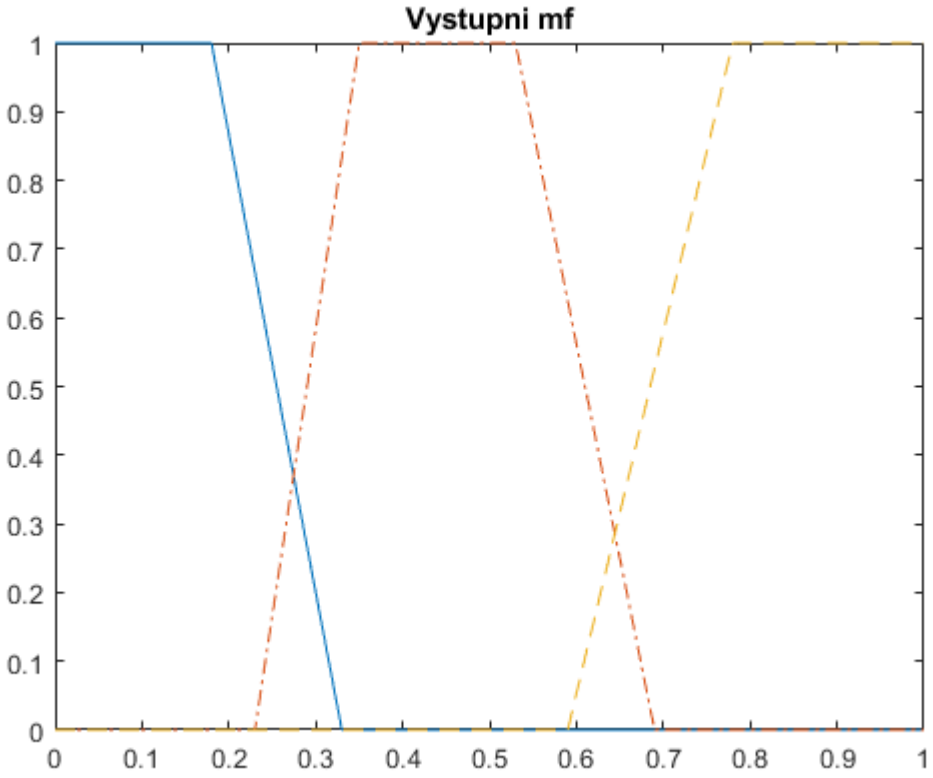
vstupní mf

```
u_nizka = @(z) 1 - sigmamf(z, 0.27, 0.47);
u_stredni = @(z) triangmf(z,0.24,0.5,0.74);
u_vysoka = @(z) sigmamf(z, 0.53, 0.73);

% zobrazeni funkci (fplot vykreslí funkci)

figure,
fplot(u_nizka,[0 1], 20);
hold on
fplot(u_stredni,[0 1],'-.', 20);
fplot(u_vysoka,[0 1],'--', 20);
hold off
title('Vstupni mf');
```



výstupní mf

```
u_normalni = @(z) 1 - sigmamf(z, 0.18, 0.33);
u_castecne = @(z) trapezmf(z,0.23,0.35,0.53,0.69);
u_selhani = @(z) sigmamf(z, 0.59, 0.78);

%zobrazeni funkci
figure,
fplot(u_normalni,[0 1], 20);
hold on
fplot(u_castecne,[0 1],'-.', 20);
fplot(u_selhani,[0 1],'--', 20);
hold off
title('Vystupni mf');
```



vytvoření systému

```
%F = fuzzysysfcn(inmf,outmf,vrange,op)
inmf = {u_nizka; u_stredni; u_vysoka};
outmf = {u_normalni,u_castecne, u_selhani};
vrange = [0 1];

F = fuzzysysfcn(inmf,outmf,vrange);
```

použití

```
z = 0.5;
```

```
F(z)
```

```
ans = 0.4515
```

Roztažení kontrastu

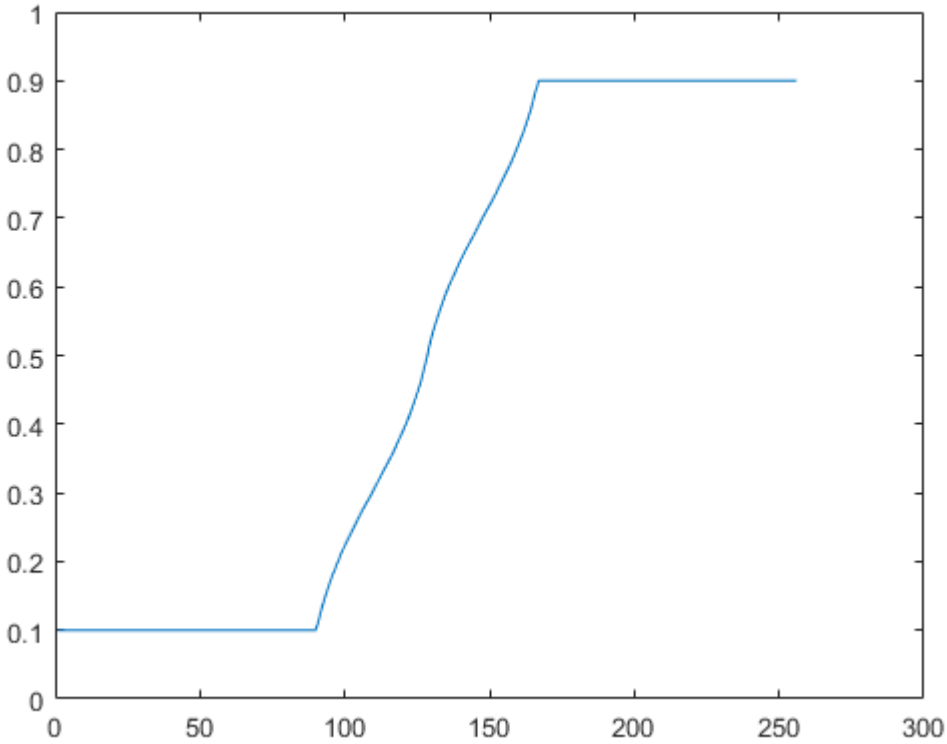
pravidla:

- pokud je pixel tmavý, uděléj ho tmavší

- pokud je pixel šedý, nech ho šedý

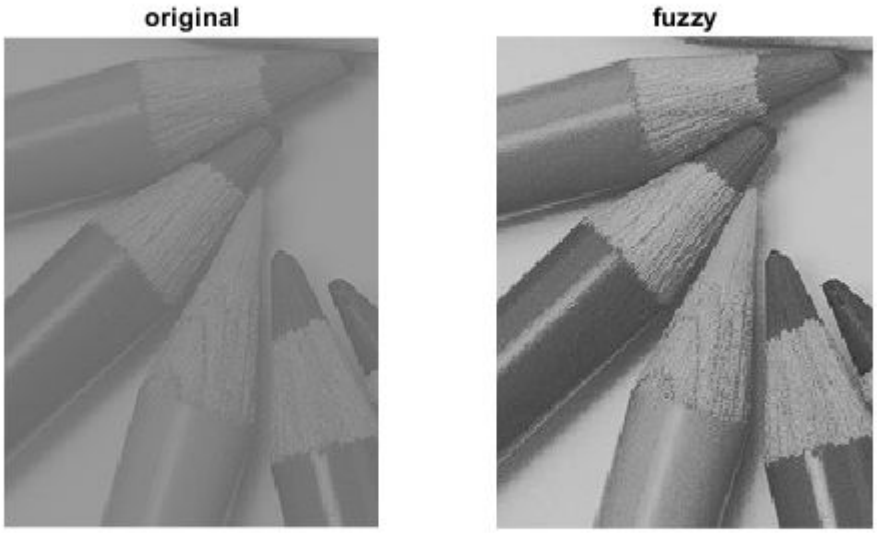
- pokud je pixel světlý, udělej ho světlejší

```
% vstupni hodnoty
% definujeme si, co znamená, že je pixel tmavý/šedý/světlý (bereme jasové
% hodnoty 0 ... 1)
utmavy = @(z) 1 - sigmamf(z, 0.35, 0.5);
usedy = @(z) triangmf(z,0.35,0.5,0.65);
usvetly = @(z) sigmamf(z, 0.5, 0.65);

% vystupni mf
utmavsi = @(z) bellmf(z, 0, 0.1);
ustrednesedy = @(z) bellmf(z, 0.4, 0.5);
usvetlejsi = @(z) bellmf(z, 0.8, 0.9);

% vytvoreni systemu
inmf = {utmavy; usedy; usvetly};
outmf = {utmavsi,ustrednesedy, usvetlejsi};
vrange = [0 1];

F = fuzzysysfcn(inmf,outmf,vrange);

%% Vytvoreni transformacni funkce
z = linspace(0,1,256); % pro obrazek uint8 = vytvoří vektor s 256 hodnotama
rovnoměrně rozloženýma na intervalu <0, 1>
T = F(z); % Aplikace transformační funkce definované fuzzy systémem.

figure,
plot(T); % vykreslení transformační funkce
```



Aplikace na obrázek s malým kontrastem

```
f = imread('p3.png');

g = T(f+1);

subplot(1,2,1), imshow(f);
title('original');
subplot(1,2,2), imshow(g);
title('fuzzy');
```



ÚKOL 1

Porovnejte výsledný obrázek pro fuzzy roztažení kontrastu s klasickou funkcí roztažení kontrastu a s výsledným obrázkem po aplikaci vyvážení histogramu.

ÚKOL 2

Zkuste vytvořit systém pro jinou jasovou transformaci.

Filtrování - hledání hran

Definujeme fuzzy okolí - pokud pixel patří do uniformního okolí nech ho bílý, jinak černý

Bílá a černá jsou fuzzy proměnné.

Pro definici uniformího okolí uvažujeme pixel ve středu filtru a jeho rozdíl intenzit mezi sousedy

okolí 3x3:

intenzity pixelů:

```
z1  z2  z3
z4  z5  z6
z7  z8  z9
```

rozdíly intenzit (`di = zi - z5`):

d1 d2 d3

d4 0 d6

d7 d8 d9

Pravidla:

- pokud d2 je nula and d6 je nula pak z5 bílý

- pokud d6 je nula and d8 je nula pak z5 bílý

- pokud d8 je nula and d4 je nula pak z5 bílý

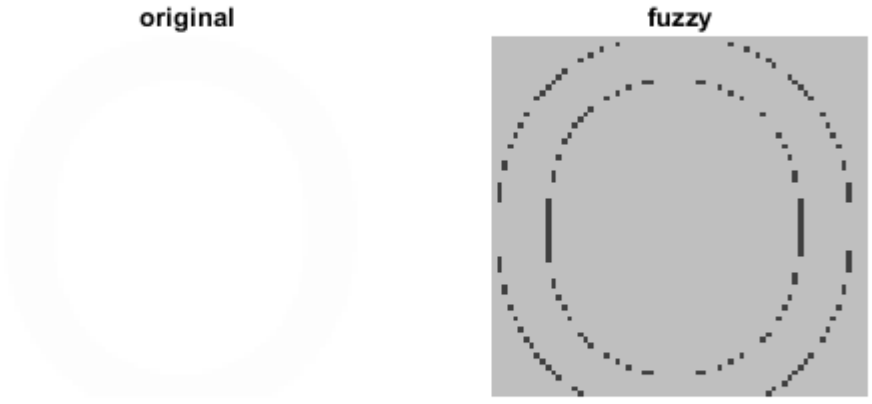
- pokud d4 je nula and d2 je nula pak z5 bílý

- jinak z5 černý

```
%nula = @(z) bellmf(z,-0.3,0);  % rozdíl intenzit = 0 (okolí 0)
nula = @(z) bellmf(z,-0.001,0);  % rozdíl intenzit = 0 (okolí 0)
not_used = @(z) onemf(z); % nepoužité pravidlo můžeme bez obav nastavit na 1,
protože používáme operaci min pro spojování. Tedy 1 nám výsledek neovlivní

cerny = @(z) triangmf(z, 0, 0,0.75);
bily = @(z) triangmf(z,0.25, 1,1);

% každý řádek matice odpovídá jednomu pravidlu
inmf = {nula, not_used, nula, not_used;  %d2,d4,d6,d8
        not_used,not_used, nula,nula;
        not_used, nula, not_used, nula;
        nula, nula, not_used, not_used};
outmf = {bily,bily,bily,bily,cerny};

vrange = [0,1];

F = fuzzysysfcn(inmf,outmf,vrange);
```

výpočet je náročný, nezkoušejte na velkých obrázcích

```
img = imread('cv6-img.png');
img = img(361:427,788:857);
%img = rgb2gray(imread('obr3.png'));

% Funkce aplikující fuzzy systém jako filtr fuzzyFilt()
% (výpočet hodnot d2,d4,d6,d8 a aplikace systému na tyto hodnoty)
g = fuzzyFilt(img, F);

figure,
subplot(1,2,1), imshow(img);
title('original');
subplot(1,2,2), imshow(g);
title('fuzzy');
```



ÚKOL 3

Jak by se změnila pravidla pro detekci svislých a vodorovných hran?