ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA TECHNOLOGIÍ A MĚŘENÍ

BAKALÁŘSKÁ PRÁCE

Fuzzy regulátor v řízení robota

Martin Partingl **2014**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin PARTINGL**
Osobní číslo:	**E11B0146P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komerční elektrotechnika**
Název tématu:	**Fuzzy regulátor v řízení robota**
Zadávající katedra:	**Katedra technologií a měření**

Zásady pro vypracování:

Navrhněte a realizujte vzorové příklady využití fuzzy řízení.

1. Uvažujte malého 2-kolového robota, příp. další typy robotů.
2. Najděte a použijte vhodné SW nástroje a knihovny.
3. Uveďte možnosti modelování na PC, příp. řízení robota z PC.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce: **Ing. Petr Weissar, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **14. října 2013**

Termín odevzdání bakalářské práce: **9. června 2014**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

Doc. Ing. Vlastimil Skočil, CSc.
vedoucí katedry

V Plzni dne 14. října 2013

Abstrakt

Tato bakalářská práce popisuje fuzzy množiny, proces fuzzifikace, defuzzifikace a fuzzy systémy. Práce uvádí možnosti návrhu fuzzy systému a popisuje software, který jej umožňuje. Dále popisuje knihovnu pro použití fuzzy logiky v embedded systémech. V rámci této práce byl vytvořen program s fuzzy regulátorem pro malého dvoukolového robota. Program pro PC s fuzzy regulátorem, který ovládá robota přes bluetooth a program, který generuje fuzzy systém v C kódu.

Klíčová slova

Fuzzy, fuzzy množiny, fuzzy logika, fuzzy regulátor, fuzzy regulace, fuzzy systémy, fuzzy knihovny, fuzzifikace, defuzzifikace, fuzzy logic toolbox.

Abstract

This bachelor's thesis describes fuzzy sets, the process of fuzzification, defuzzification and fuzzy systems. Thesis mentions possibilities design of fuzzy systems and describes software which allows it. Also it describes library for the use of fuzzy logic in embedded systems. In this thesis was created program with fuzzy controller for the small two-wheel robot. Program for PC with fuzzy controller which controls a robot via bluetooth and program which generates fuzzy system in C code.

Key words

Fuzzy, fuzzy sets, fuzzy logic, fuzzy controller, fuzzy control, fuzzy systems, fuzzy libraries, fuzzification, defuzzification, fuzzy logic toolbox

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 3.6.2014 Martin Partingl

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Petru Weissarovi, Ph.D. za cenné profesionální rady, připomínky a metodické vedení práce.

Obsah

Seznam symbolů a zkratek

A, B, C.....fuzzy množiny

M .....klasická množina

$X_A(x)$.....charakteristická funkce

μ_A .....funkce příslušnosti

v.....lingvistická proměnná

e(k).....odchylka

$\Delta e(k)$ .....změna odchylky

u(k).....akční zásah

eFLL .....embedded Fuzzy Logic Library

FIS .....Fuzzy Inference System

FLT.....Fuzzy Logic Toolbox

LFLC 2000.....Linguistic Fuzzy Logic Controller

Úvod

V roce 1965 publikoval Lofti A. Zadeh referát s názvem ***Fuzzy Sets*** [1]. V tomto referátu byly definovány fuzzy množiny, které posloužily jako základ pro fuzzy logiku. Anglické slovo „Fuzzy“ se dá přeložit jako mlhavý (neurčitý, nejasný, neostrý). S tímto významem souvisí celá logika. Ta na rozdíl od booleovské logiky nerozeznává pouze dva stavy 1 a 0 (pravda, nepravda), ale je dána stupněm příslušnosti, který nabývá hodnot z intervalu $\langle 0,1 \rangle$.

Fuzzy logika je výborný nástroj pro regulaci. Její výhodou je, že může pracovat se slovním popisem systému. Fuzzy také umožňuje použití neurčitých výrazů jako téměř, skoro, částečně.

S fuzzy regulací se setkáváme stále více například u domácích elektrospotřebičů nebo automobilů.

1 Klasické množiny

Mezi základní prvky matematiky patří množiny. Množina je soubor prvků. Počet prvků v množině může být konečný, nekonečný nebo nulový, v takovém případě jde o prázdnou množinu.

Obvykle se množiny značí velkými písmeny a její prvky malými. Když se definuje množina jako M a prvek jako x, pak patří-li prvek x do množiny M zapisuje se jako $x \in M$. Jestliže do množiny M nepatří, tak se zapisuje jako $x \notin M$.

1.1 Definice množin

Množiny se mohou definovat dvěma způsoby.

- Výčtem prvků

$$M = \{0; 1; 2; 3; 4\} \tag{1.1}$$

Tato množina obsahuje pět prvků: 0, 1, 2, 3, 4.

- Charakteristickými vlastnostmi prvků.

$$M = \{x \in \mathbb{N}_0 \mid x < 5\} \tag{1.2}$$

Tato množina obsahuje stejné prvky jako předchozí.

Klasické množiny jednoznačně určují, zda prvek do množiny patří nebo nepatří. Tuto vlastnost množiny M určuje charakteristická funkce $X_M(x)$.[2]

$$X_M(x) = \begin{cases} 1 & pro \quad x \in M \\ 0 & pro \quad x \notin M \end{cases} \tag{1.3}$$

Charakteristická funkce $X_M(x)$ může nabývat pouze hodnot 1 (pravda) nebo 0 (nepravda), je tedy přesně určena hranice mezi prvky. Takové množiny se nazývají ostré.[2]

2 Fuzzy množiny

Klasická množina je speciálním případem fuzzy množiny, protože nabývá pouze dvou stavů. V mnoha případech, je ale těžké určit, do jaké množiny by měl prvek patřit, protože dva stavy nemusí stačit. Můžeme tedy přidat třetí stav a tím rozšířit dvoustavovou Booleovu logiku na třístavovou.[2]

2.1 Základní operace pro vícehodnotovou logiku

$$\bar{a} = 1 - a \tag{2.1}$$

$$a \wedge b = min(a, b) \tag{2.2}$$

$$a \vee b = max(a, b) \tag{2.3}$$

$$a \Rightarrow b = min(1{,}1 + b - a) \tag{2.4}$$

$$a \Leftrightarrow b = 1 - |a - b| \tag{2.5}$$

Rovnice (2.1) - (2.5) jsou základní operace pro vícehodnotovou logiku.[2]

Tab. 2.1 Pravdivostní hodnoty pro třístavovou logiku (převzato z [2])

a	**b**	**Λ**	**V**	**⇒**	**⇔**
0	0	0	0	1	1
0	0,5	0	0,5	1	0,5
0	1	0	1	1	0
0,5	0	0	0,5	0,5	0,5
0,5	0,5	0,5	0,5	1	1
0,5	1	0,5	1	1	0,5
1	0	0	1	0	0
1	0,5	0,5	1	0,5	0,5
1	1	1	1	1	1

Takto můžeme postupně zvyšovat počet stavů až na n a v případě, kdy $n \to \infty$, získáme nekonečně stavovou logiku. Hodnoty této logiky jsou tedy reálná čísla z intervalu $\langle 0{,}1 \rangle$. Taková charakteristická funkce se nazývá funkcí příslušnosti a udává stupeň, který vyjadřuje příslušnost prvku k množině.[2]

2.2 Pojmy týkající se fuzzy množin

Funkce příslušnosti

$$\mu_A: X \rightarrow \langle 0,1 \rangle \tag{2.6}$$

Fuzzy množina

$$A = \{x, \mu_A(x)\} \tag{2.7}$$

Nosič (angl. Support)

Nosič fuzzy množiny (2.8) je ostrá množina, která obsahuje všechny prvky, které mají nenulový stupeň příslušnosti. Tyto prvky jsou důležité, protože prvky s nulovým stupněm příslušnosti mohou být libovolné.[3]

$$Supp(A) = \{x \mid \mu_A(x) > 0\} \tag{2.8}$$

α-řez (angl. α-cut)

Ostrá množina α-řez (2.9) obsahuje pouze prvky, které mají stupeň příslušnosti větší nebo roven α. Prvky s menším stupněm příslušnosti jsou odříznuty.[3]

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\} \tag{2.9}$$

Jádro (angl. Kernel)

Jádro fuzzy množiny (2.10) je ostrá množina, která obsahuje všechny prvky se stupněm příslušnosti 1.[3]

$$Ker(A) = \{x \mid \mu_A(x) = 1\} \tag{2.10}$$

Příklad

Fuzzy množina A

$$A = \left\{ {}^{0,2}/_{1}; {}^{0,7}/_{2}; {}^{0,1}/_{3}; {}^{0,5}/_{4}; {}^{1}/_{5} \right\} \tag{2.11}$$

Nosič

$$Supp(A) = \{1,2,3,4,5\} \tag{2.12}$$

α-řez

$$A_{0,5} = \{2,4,5\} \tag{2.13}$$

Jádro

$$Ker(A) = \{5\} \tag{2.14}$$

2.3 Operace s fuzzy množinami

Rovnost dvou fuzzy množin

Fuzzy množiny *A* a *B* jsou si rovny *(A = B)*.

$$\mu_A(x) = \mu_B(x), x \in X \tag{2.15}$$

Komplement fuzzy množiny

Komplement $\bar{A}$ fuzzy množiny *A*.

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{2.16}$$

Fuzzy podmnožina

Množina *A* je podmnožinou množiny *B (A ⊂ B)*.

$$\mu_A(x) \leq \mu_B(x) \tag{2.17}$$

Sjednoceni fuzzy množin

Výsledkem sjednocení fuzzy množin *A* a *B* je množina *C*, zapisuje se jako $C = A \cup B$.

$$\mu_C(x) = Max\{\mu_A(x), \mu_B(x)\}, x \in X \tag{2.18}$$

Průnik fuzzy množin

Výsledkem průniku fuzzy množin *A* a *B* je množina *C*, zapisuje se jako $C = A \cap B$.

$$\mu_C(x) = Min\{\mu_A(x), \mu_B(x)\}, x \in X \tag{2.19}$$

Algebraický součin fuzzy množin

Výsledkem součinu fuzzy množin *A* a *B* je *C*, zapisuje se jako $C = A \cdot B$.

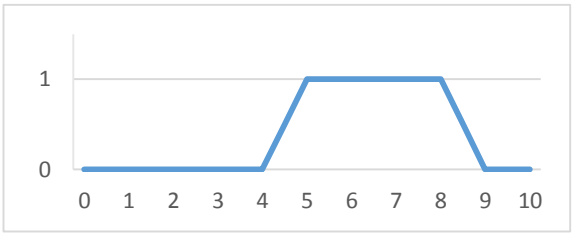
$$\mu_C = \mu_A \mu_B \tag{2.20}$$

Algebraický součet fuzzy množin

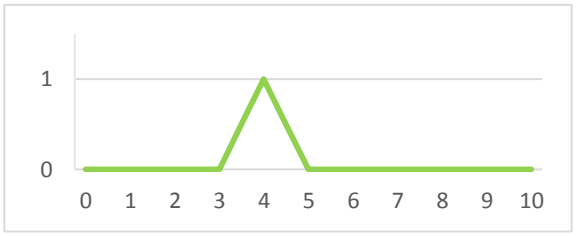
Výsledkem součtu fuzzy množin *A* a *B* je *C*, zapisuje se jako $C = A + B$.

$$\mu_C = \mu_A + \mu_B - \mu_A \cdot \mu_B \tag{2.21}$$

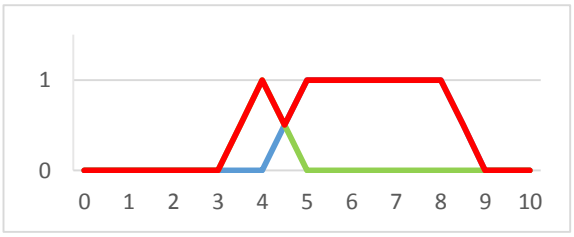
Tyto rovnice (2.15) – (2.21) pro operace s fuzzy množinami definoval L. A. Zadeh v [1].



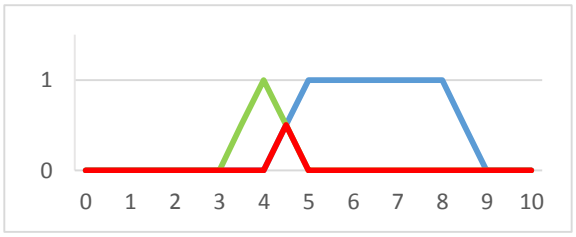
Obr. 2.1 Fuzzy množina A



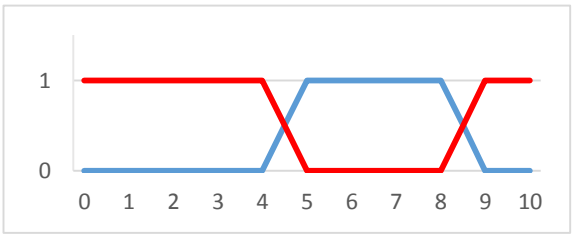
Obr. 2.2 Fuzzy množina B



Obr. 2.3 Sjednocení fuzzy množin A a B



Obr. 2.4 Průnik množin A a B



Obr. 2.5 Komplement fuzzy množiny A

Příklad

Fuzzy množiny A a B

$$A = \left\{ {}^{0,2}/_{1}; {}^{0,7}/_{2}; {}^{0,1}/_{3}; {}^{0,5}/_{4}; {}^{1}/_{5} \right\} \tag{2.22}$$

$$B = \left\{ {}^{1}/_{1}; {}^{0,4}/_{2}; {}^{0,3}/_{3}; {}^{0,6}/_{4} \right\} \tag{2.23}$$

Komplement fuzzy množiny

$$\bar{A} = \left\{ {}^{0,8}/_{1}; {}^{0,3}/_{2}; {}^{0,9}/_{3}; {}^{0,5}/_{4}; {}^{0}/_{5} \right\} \tag{2.24}$$

Sjednocení fuzzy množin

$$A \cup B = \left\{ {}^{1}/_{1}; {}^{0,7}/_{2}; {}^{0,3}/_{3}; {}^{0,5}/_{4}; {}^{1}/_{5} \right\} \tag{2.25}$$

Průnik fuzzy množin

$$A \cap B = \left\{ {}^{0,2}/_{1}; {}^{0,4}/_{2}; {}^{0,1}/_{3}; {}^{0,5}/_{4} \right\} \tag{2.26}$$

Algebraický součin fuzzy množin

$$A \cdot B = \left\{ {}^{0,2}/_{1}; {}^{0,28}/_{2}; {}^{0,03}/_{3}; {}^{0,3}/_{4} \right\} \tag{2.27}$$

Algebraický součet fuzzy množin

$$A + B = \left\{ {}^{1}/_{1}; {}^{0,82}/_{2}; {}^{0,37}/_{3}; {}^{0,8}/_{4}; {}^{1}/_{5} \right\} \tag{2.28}$$

3 Lingvistické proměnné

Využití fuzzy množin je zejména v regulační technice a je založeno na:

- **Aproximaci libovolné spojité funkce**

Podobá se využití aproximátorů, které jsou založeny na metodách nejmenších čtverců.[4]

- **Formalizaci znalostí jazykové vágní formy**

V případě složitých systémů, může být popis diferenciálními rovnicemi velmi složitý. V takovém případě je vhodné zavedení slovního popisu tzv. lingvistických proměnných.[4]

L. A. Zadeh formuloval **princip inkompatibility**.

Vzhledem k rostoucí složitosti systému, klesá naše schopnost dělat přesná a přesto významná prohlášení o jeho chování, dokud není dosaženo hranice, za níž přesnost a relevance jsou vzájemně téměř vylučující se charakteristiky.[5]

Lingvistická proměnná má schopnost vyjadřovat hodnoty pomocí běžného jazyka.

Tvar lingvistické proměnné.[2, 6]

$$v = (X, T, U, G, A) \tag{3.1}$$

X – název lingvistické proměnné

T – množina lingvistických hodnot

U – univerzum

G – syntaktické pravidlo, pro tvorbu jazykových výrazů z množiny T

A – sémantické pravidlo, pro přiřazení významu

4 Fuzzifikace

K fuzzifikaci je třeba nejdříve získat základní (ostré) hodnoty, se kterými se bude dále pracovat. Tyto hodnoty se obvykle získají měřením. Dále je třeba provést tzv. normalizaci. To znamená, že se tyto změřené hodnoty převedou na normalizované univerzum např. interval $\langle 0,1 \rangle$. Potom se hodnotám přiřadí stupeň příslušnosti minimálně k jedné nebo více fuzzy množinám.[1, 4]

K tomu aby mohlo přiřazení proběhnout, je zapotřebí pokrýt celé univerzum. Pokrytí musí být v celém rozsahu, aby nedošlo k tomu, že by nějaká hodnota měla stupeň příslušnosti 0 ke všem fuzzy množinám, proto se funkce příslušnosti musí překrývat. Každá hodnota získává minimální stupeň příslušnosti ε (velikost ε se obvykle volí nejméně 0,5). Protnutí funkcí musí být tedy na stupni $\geq \varepsilon$. Tímto je zajištěna existence jednoho dominantního pravidla. V případě, že nastane situace kdy $\mu = \varepsilon$, aktivují se dvě pravidla. Fuzzy množiny, které odpovídají termům vstupních veličin, se nazývají primární fuzzy množiny.[1, 4]

Otázkou je, kolik fuzzy množin volit k pokrytí univerza. Z různých studií řešících tento problém vyplývá obvyklé použití lichého počtu a to 3, 5 nebo 7 primárních fuzzy množin občas 9 nebo více. Rozložení pokrytí může být různé, stejně tak se může lišit tvar funkcí příslušnosti.[1, 4]

4.1 Přehled základních funkcí příslušnosti

L – funkce

$$L(x,\alpha,\beta)=\begin{cases}1 & pro & x<\alpha\\ (\beta-x)/(\beta-\alpha) & pro & \alpha\le x\le\beta\\ 0 & pro & x>\beta\end{cases} \tag{4.1}$$

Λ – funkce

$$\Lambda(x,\alpha,\beta,\gamma)=\begin{cases}0 & pro & x<\alpha\\ (x-\alpha)/(\beta-\alpha) & pro & \alpha\le x\le\beta\\ (\gamma-x)/(\gamma-\beta) & pro & \beta\le x\le\gamma\\ 0 & pro & x>\gamma\end{cases} \tag{4.2}$$

Π – funkce

$$\Pi(x,\alpha,\beta,\gamma,\delta)=\begin{cases}0 & pro & x<\alpha\\ (x-\alpha)/(\beta-\alpha) & pro & \alpha\le x<\beta\\ 1 & pro & \beta\le x\le\gamma\\ (\delta-x)/(\delta-\gamma) & pro & \gamma<x\le\delta\\ 0 & pro & x>\delta\end{cases} \tag{4.3}$$

Γ – funkce

$$\Gamma(x,\alpha,\beta)=\begin{cases}0 & pro & x<\alpha\\ (x-\alpha)/(\beta-\alpha) & pro & \alpha\le x\le\beta\\ 1 & pro & x>\beta\end{cases} \tag{4.4}$$

S – funkce

$$S(x,\alpha,\beta,\gamma)=\begin{cases}0 & pro & x\le\alpha\\ 2((x-\alpha)/\gamma-\alpha))^2 & pro & \alpha<x\le\beta\\ 1-2((x-\gamma)/(\gamma-\alpha))^2 & pro & \beta<x\le\gamma\\ 1 & pro & x>\gamma\end{cases} \tag{4.5}$$
$$\beta=(\alpha+\gamma)/2$$

5 Defuzzifikace

Výstup získaný použitím slovních pravidel je množina. K tomu aby bylo možno dále používat výsledek, je třeba získat ostrou hodnotu. Tato hodnota se získá procesem, který se nazývá defuzzifikace. K realizaci tohoto procesu existují různé metody.

5.1 Přehled metod defuzzifikace

Metoda COG (Center of Gravity/Area)

U této metody je nejdříve vypočtena plocha funkcí příslušnosti v rozsahu výstupní proměnné a souřadnice těžiště této plochy je výsledná hodnota. Tato metoda patří mezi nejpoužívanější. Nevýhodou této metody je vysoká výpočetní složitost.[3]

$$COG(A) = \frac{\sum_{i=1}^{n} A(u_i) \cdot u_i}{\sum_{i=1}^{n} A(u_i)} \tag{5.1}$$

Metoda MOM (Mean of Maxima)

Výsledkem této metody je střed prvků s maximálním stupněm příslušnosti. V případě, kdy je fuzzy množina symetrická s jedním maximem, je výsledek stejný jako u metody COG. Oproti COG má tato metoda menší výpočetní náročnost.[3]

$$MOM(A) = \frac{1}{n_{max}} \sum_{j=1}^{n_{max}} u_j^{max} \tag{5.2}$$

Metody FOM (First of Maxima) a LOM (Last of Maxima)

FOM a LOM patří mezi nejjednodušší metody defuzzifikace, jejich použití je však minimální. Princip je takový, že se vybere první (FOM) nebo poslední (LOM) prvek s nejvyšším stupněm příslušnosti. Tento prvek je výslednou hodnotou.[3]

Metoda COS (Center of Sums)

Tato metoda je podobná metodě COG. Průniky množin jsou u této metody počítány vícekrát. Výhodou je jednodušší algoritmus.[3]

Výsledná množina A je získána sjednocením fuzzy množin. Fuzzy množiny B_i jsou získány přibližným odhadem z užitých slovních pravidel.[3]

$$A = B_1 \cup \ldots \cup B_k \tag{5.3}$$

$$COS(A) = \frac{\sum_{i=1}^{n}(u_i \cdot \sum_{j}^{k} B_j(u_i))}{\sum_{i=1}^{n}\sum_{i=1}^{n} B_j(u_i)} \tag{5.4}$$

6 Fuzzy systémy

Logické řízení je založeno na obecném pravidlu

$$JESTLIŽE - PAK$$

Pro fuzzy regulaci jde o implikaci vyjádřenou pomocí dvou fuzzy výroků

$$JESTLIŽE \ \langle \ fuzzy \ výrok \ \rangle \ PAK \ \langle \ fuzzy \ výrok \ \rangle$$

Případně v anglické verzi

$$IF \ \langle \ fuzzy \ proposition \ \rangle \ THEN \ \langle \ fuzzy \ proposition \ \rangle$$

Tato podmínka je tzv. „produkční pravidlo“. První fuzzy výrok se nazývá antecedent a často jde o složený výrok pomocí logických spojek. Druhý fuzzy výrok se nazývá konsekvent.[7]

Tyto pravidla jsou součástí inferenčního mechanismu fuzzy systémů. V systému Mamdani jsou využita v uvedeném tvaru, zatímco systém Takagi–Sugeno-Kang používá pravidla s jiným konsekventem.

6.1 Mamdani

Nejčastěji používaný je fuzzy systém Mamdani. Tento systém je založen na teorii fuzzy množin. S návrhem přišli v roce 1975 E. H. Mamdani a S. Assilian. Jejich snahou byl pokus řídit parní stroj syntézou slovních pravidel, získaných od zkušených operátorů.[8]

Tento systém je postaven na bázi pravidel IF-THEN, kde antecedent i konsekvent jsou fuzzy výroky.[9]

$$IF \ \ x = A \ \ AND \ \ y = B \ \ THEN \ \ z = C \tag{6.1}$$

A, B, C jsou fuzzy množiny; *x, y* jsou vstupní proměnné, *z* je výstupní proměnná.

Jako první proběhne fuzzifikace, potom se pomocí báze pravidel určí výstupní fuzzy množina a z té se deffuzifikací získá ostrá hodnota.[9]

6.2 Takagi-Sugeno-Kang

Tento systém vytvořili T. Takagi a M. Sugeno v roce 1985 a M. Sugeno a G. Kang v roce 1988. Rozdílem oproti systému Mamdani jsou pravidla, která mají v konsekventu funkci.[9]

$$IF \quad x = A \quad AND \quad y = B \quad THEN \quad z = f(x, y) \tag{6.2}$$

A, B jsou fuzzy množiny; *x, y* jsou vstupní proměnné, *z* je výstupní proměnná; *f(x,y)* je výstupní funkce.

Ostrá hodnota se u tohoto systému nezíská defuzzifikací jako u systému Mamdani, ale výpočtem váženého průměru.[9]

$$z = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \tag{6.3}$$

7 Software pro návrh fuzzy systému

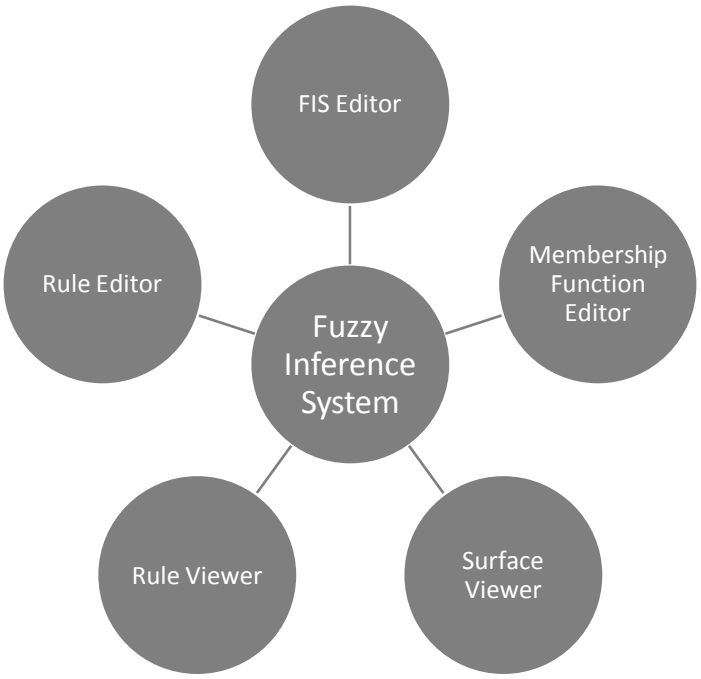
Pro návrh fuzzy systému existují různé nástroje. Po vyzkoušení některých uvádím popis dvou. Jako první Fuzzy Logic Toolbox. Ten je součástí MATLABu a jeví se mi jako nejlepší. Dalším zajímavým programem je FLFC 2000.

7.1 Fuzzy Logic Toolbox

Pro návrh fuzzy regulátoru je možné využít toolboxu, který je součástí programu MATLAB. Tento toolbox umožňuje využití grafického rozhrání.

Základem toolboxu je pět nástrojů.

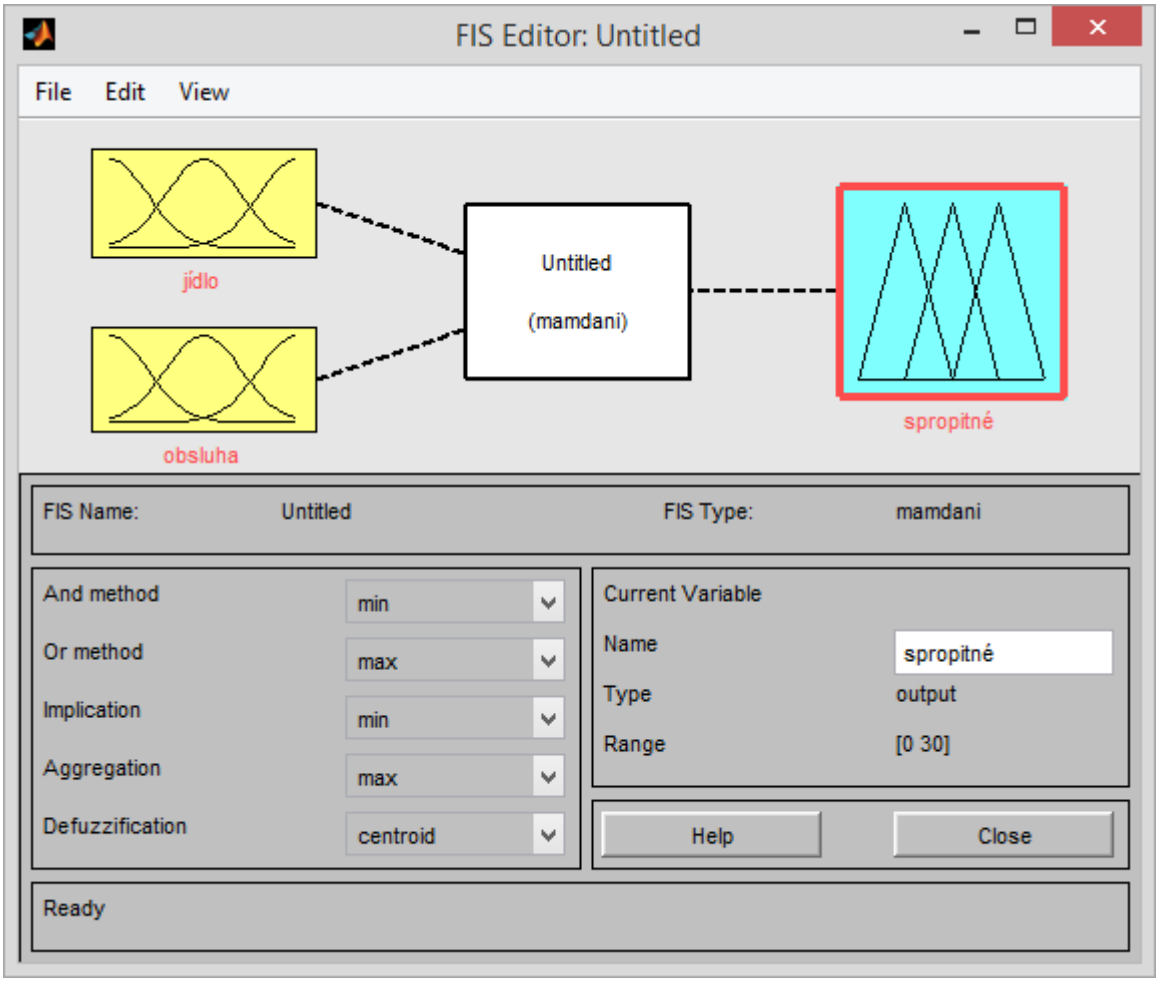
- Fuzzy Inference System (FIS) Editor
- Membership Function Editor
- Rule Editor
- Rule Viewer
- Surface Viewer



Obr. 7.1 Schéma Fuzzy Logic Toolboxu[10]

7.1.1 Fuzzy Inference System (FIS) Editor

FIS editor zobrazuje základní informace o fuzzy systému. K jeho spuštění je zapotřebí napsat do příkazového řádku MATLABu příkaz *fuzzy*. V tomto editoru lze nastavit počet vstupů, výstupů a další parametry systému.[10]



Obr. 7.2 FIS Editor

Menu tohoto editoru obsahuje tři položky.

- File – umožnuje vytvoření nového FIS, import, export, tisk a uzavření okna.
- Edit – umožňuje vrácení posledního kroku, přidání nebo odstranění vstupu/výstupu, spustit Membership Function Editor a Rule editor.
- View – umožňuje spustit Rule Viewer a Surface Viewer.

Zobrazené schéma systému se skládá ze tří částí.

- vlevo – vstupy
- uprostřed – typ systému

- vpravo – výstupy

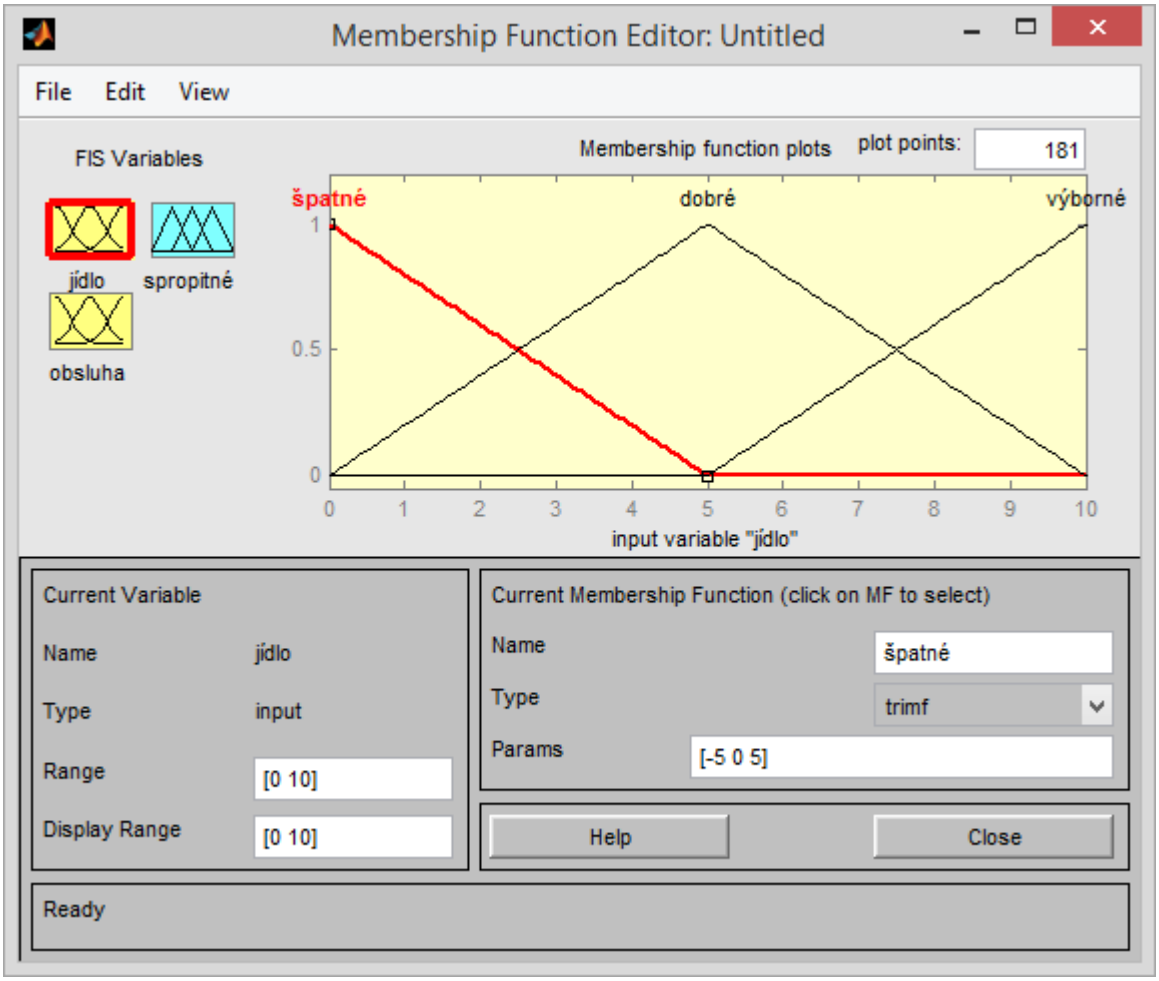
Ve spodní části se nastavuje, jak budou prováděny metody And, Or, implikace, agregace a defuzzifikace. Dále se zobrazují informace o označené proměnné a celém systému.

V pravém dolním rohu jsou tlačítka *Help* pro vyvolání nápovědy a *Close* pro uzavření, tyto tlačítka se vyskytují u všech editorů tohoto toolboxu.

7.1.2 Membership Function Editor

Membership Function Editor se spustí pomocí *Edit* > *MemberShip Function…* nebo dvojklikem na libovolnou proměnnou ve FIS editoru.

Membership Function Editor umožňuje zobrazovat a upravovat funkce příslušnosti jednotlivých vstupů a výstupů.



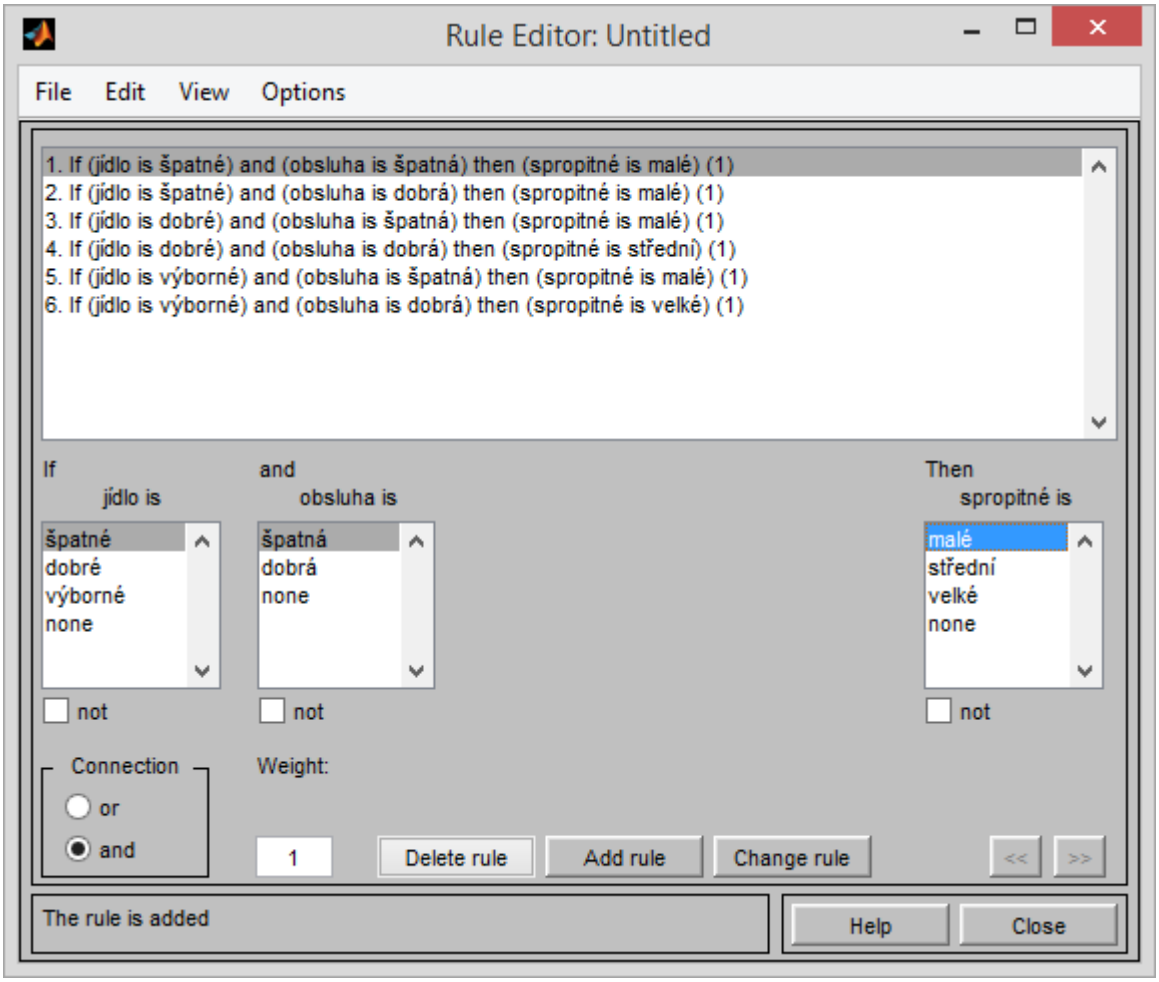
Obr. 7.3 Membership Function Editor

V levém horním rohu jsou na výběr proměnné FIS. Vedle tohoto výběru vpravo je graf, který zobrazuje funkce příslušnosti a zároveň umožňuje jejich úpravu myší. V levém dolním rohu se nastavuje rozsah vybrané proměnné a rozsah zobrazení. Vedle se mění název funkce, její typ a parametry.

Další funkce příslušnosti se vkládá pomocí *Edit* > *Add MFs…* nebo *Edit* > *Add Custom MF…*. Odstranění funkce příslušnosti se provádí pomocí *Edit* > *Remove Selected MF* (odstraní se pouze označená funkce příslušnosti) nebo *Edit* > *Remove All MFs* (odstraní se všechny funkce příslušnosti).

7.1.3 Rule Editor

Rule Editor se spustí pomocí *Edit* > *Rules…* nebo dvojklikem na prostřední blok ve FIS Editoru. Rule Editor je nástroj pro tvorbu pravidel, která jsou uplatňována.



Obr. 7.4 Rule Editor

Hlavní částí tohoto editoru je seznam, kam se zapisují vytvořená pravidla. Pod ním jsou vstupy a výstupy se svými funkcemi příslušnosti. V každém tomto seznamu je navíc položka *none*, která se vybere v případě, kdy není žádoucí aby vstup/výstup nějak ovlivňoval pravidlo.

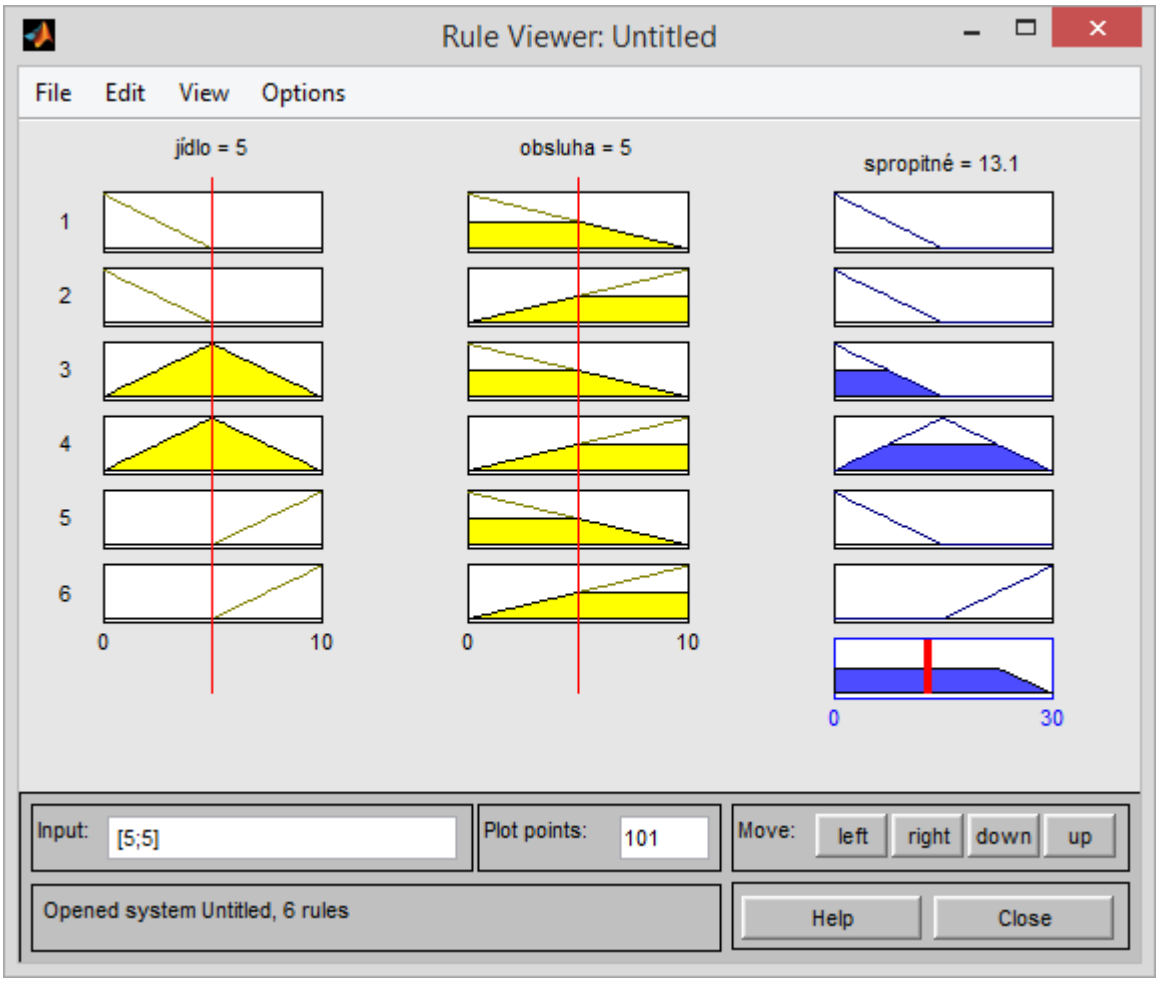
- Funkce příslušnosti se může negovat zaškrtávacím políčkem *not*.
- Pro spojení vstupů je na výběr funkce *or* nebo *and*.
- V poli *Weight* se nastavuje váha pravidla (0 až 1). Defaultně je nastavena na 1.

Tvorba, mazání, úprava pravidel

- Vytvoření se provádí výběrem funkcí příslušností jednotlivých proměnných a stisknutím tlačítka *Add Rule*.
- Pro odstranění pravidla se vybere hotové pravidlo a stiskne se tlačítko *Delete Rule*.
- Pro změnu pravidla se vybere hotové pravidlo, provede se výběr jako při vytvoření nového pravidla a stiskne tlačítko *Change Rule*.

7.1.4 Rule Viewer

Rule Viewer se spustí pomocí *View* > *Rules* (v menu okna Rule Editor). Rule Editor poskytuje grafické zobrazení nastavených pravidel a výsledku pro konkrétně zadané hodnoty.



Obr. 7.5 Rule Viewer

V levé části okna jsou žluté grafy, ty zobrazují funkce příslušnosti, které jsou nastaveny jako antecedenty. V pravé části okna jsou modré grafy, ty zobrazují funkce příslušnosti, které jsou nastaveny jako konsekventy.

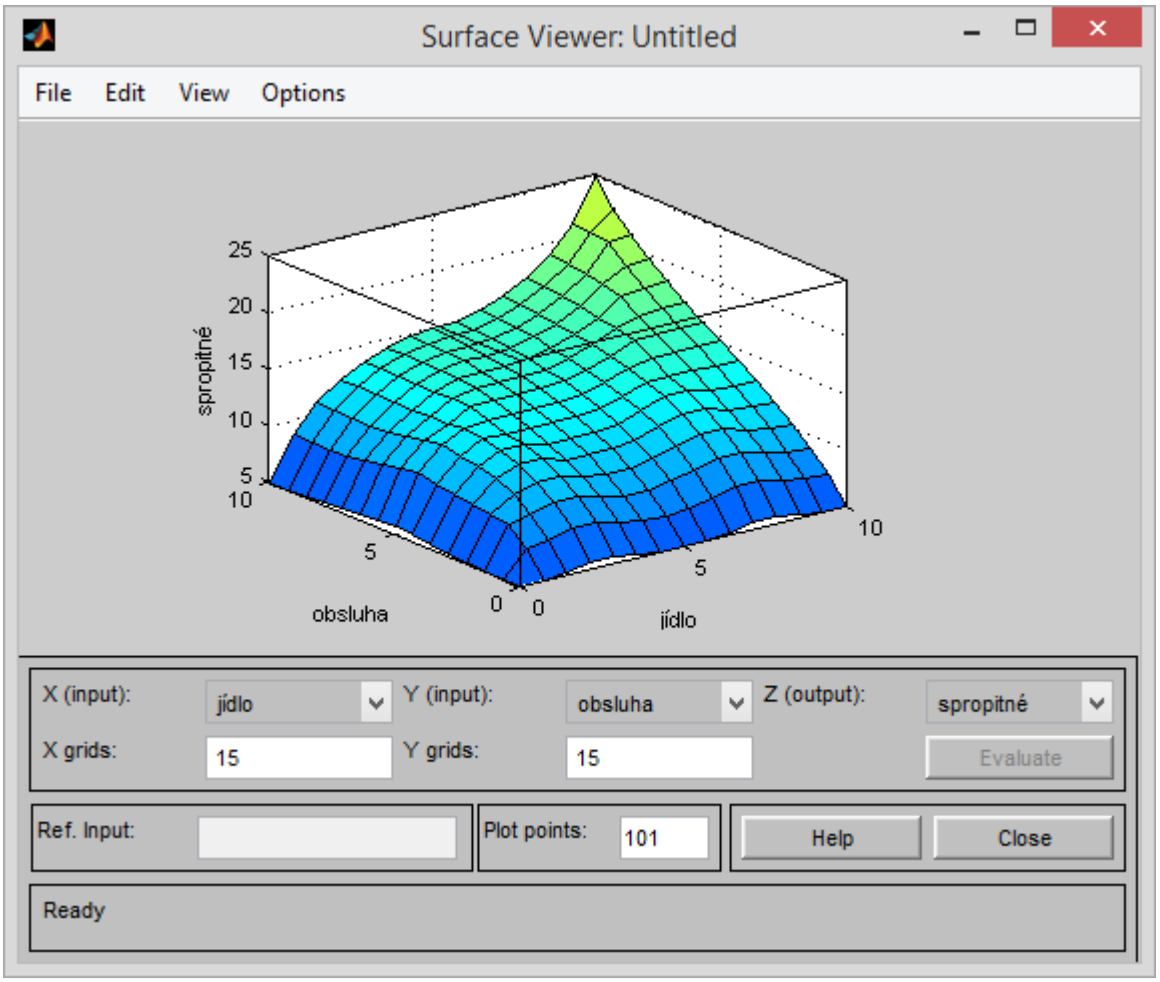
Aktuální hodnoty vstupů a výstupů jsou zobrazeny nad sloupci u názvů proměnných. Vstupní hodnoty lze zadávat napsáním hodnoty do pole *Input* v levém dolním rohu okna (v případě více vstupů se pro zápis použije vektor např. [5;5] pro dva vstupy s hodnotami 5 a 5) nebo posunutím svislé červené čáry myší.

7.1.5 Surface Viewer

Surface Viewer se spustí pomocí *View* > *Surface* (v menu okna Rule Viewer).

Surface Viewer zobrazuje graf závislosti výstupu na vstupu. V případě jednoho vstupu a jednoho výstupu se zobrazí dvourozměrný graf. Když je více vstupů a výstupů, zobrazí se

trojrozměrná charakteristika. Tento graf lze pomocí myši libovolně otáčet. Pod ním je nabídka, sloužící pro výběr, které dva vstupy a jaký výstup se mají zobrazit.



Obr. 7.6 Surface Viewer

7.2 LFLC 2000

LFLC 2000 je softwarový balík, vyvíjen na Ústavu pro výzkum a aplikace fuzzy modelování (ÚVAFM) při Ostravské Univerzitě v Ostravě. Tento program slouží k návrhu a testování soustav fuzzy pravidel IF-THEN.[11]

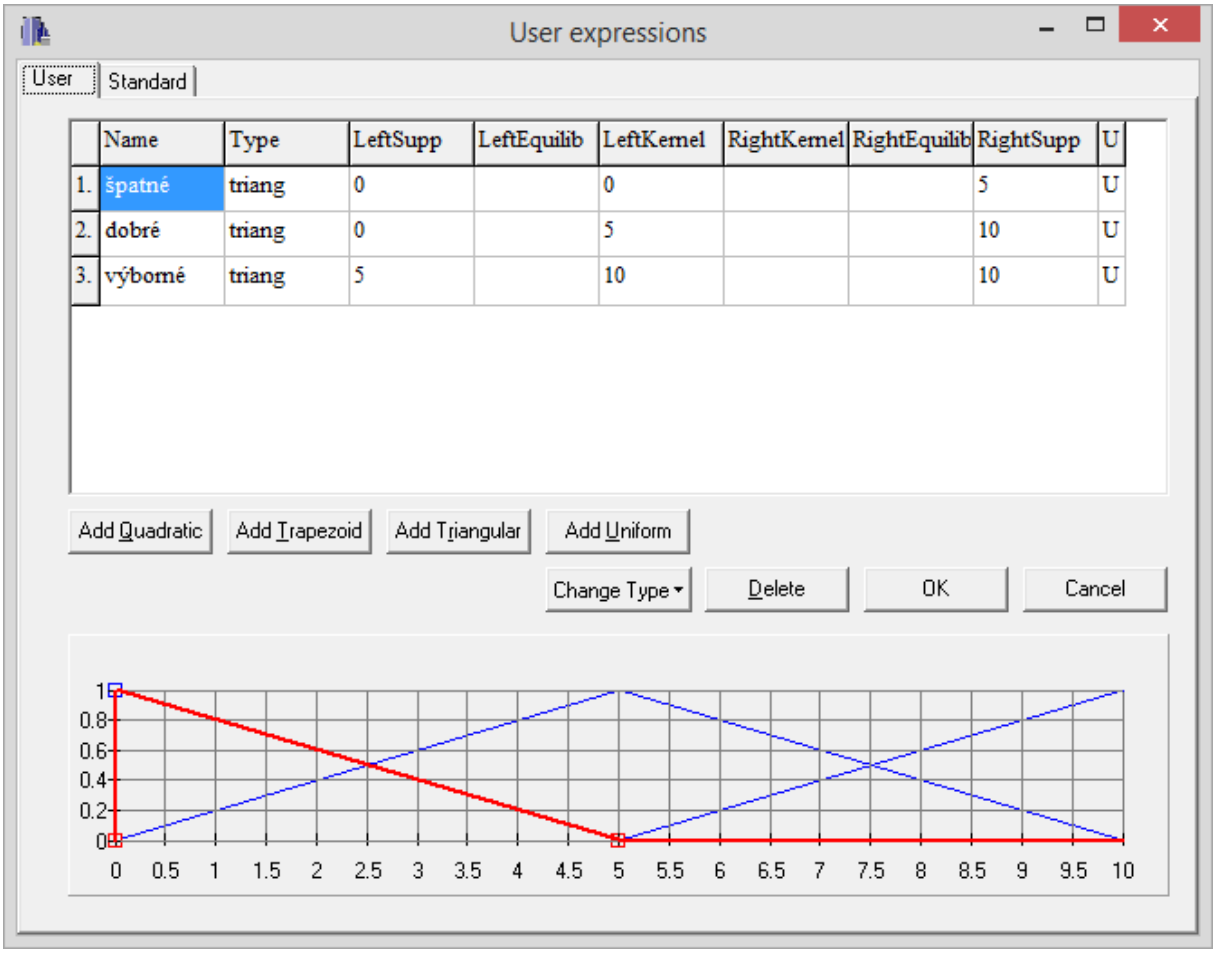
Po spuštění programu se pomocí tlačítka *New* otevře okno s pěti záložkami, kde je možné navrhnout systém.

7.2.1 General

Jako první je záložka General. Na této záložce je možné nastavit obecné vlastnosti systému jako název, typ inference, typ defuzzifikace a popis systému.

7.2.2 Input variables, Output variables

Na této záložce se vytvářejí vstupní proměnné. Stisknutím tlačítka *Add variable* se otevře okno, kde se vyplňuje jméno proměnné, nastavuje její rozsah a diskretizace. Diskretizace určuje počet bodů, v níž se počítá funkce příslušnosti. V tomto okně je dále tlačítko *Edit exoressions...*. Tímto tlačítkem se vyvolá editor funkcí příslušností.



Obr. 7.7 Okno pro editaci funkcí příslušnosti

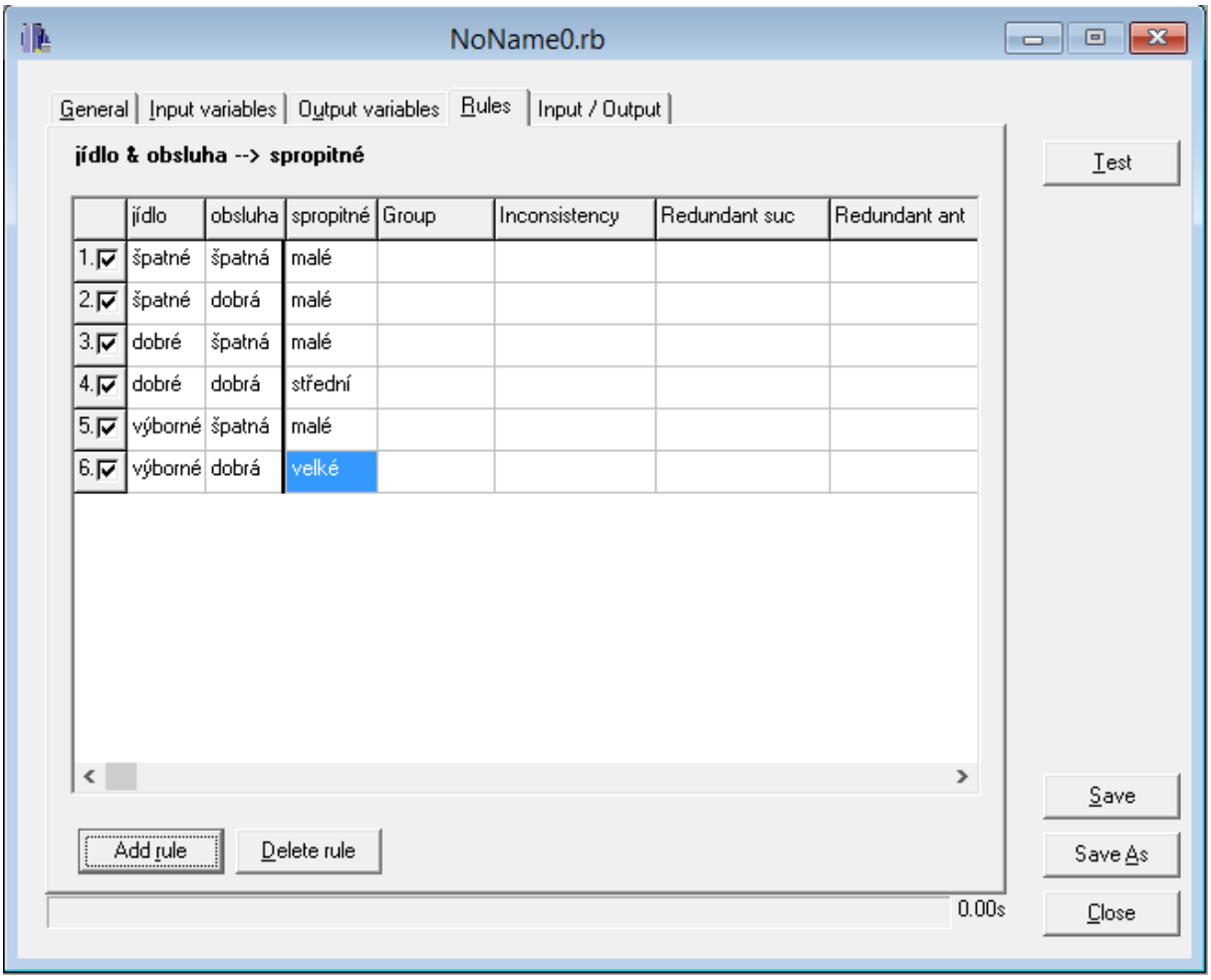
Funkce příslušnosti se přidává jedním z tlačítek *Add Quadratic*, *Add Trapezoid*, *Add Trialgular*, *Add Uniform*. Funkce příslušnosti se objeví v tabulce a také se graficky znázorní pod tabulkou, její parametry se upravují v tabulce.

7.2.3 Rules

Záložka Rules slouží k vytváření IF-THEN pravidel. Pravidla jsou v tabulce, kam se přidávají tlačítkem *Add rule* a mažou tlačítkem *Delete rule*. Každé pravidlo se skládá z několika sloupců. V prvním sloupci je číslo pravidla a zaškrtávací políčko k aktivaci a deaktivaci pravidla v případě testování. Další sloupce jsou vstupní a výstupní proměnné.

Dále jsou zde čtyři informační slupce.

- **Group** – oznamuje shodná pravidla.
- **Inconsistency** – oznamuje, která pravidla mají stejný antecedent, ale jiný konsekvent.
- **Redundand suc, Redundant ant** – oznamuje, která pravidla jsou navíc, v případě kdy jsou dvě pravidla se stejným konsekventem, ale antecedent jednoho má větší rozsah než druhého. Nebo je stejný antecedent a různé konsekventy.



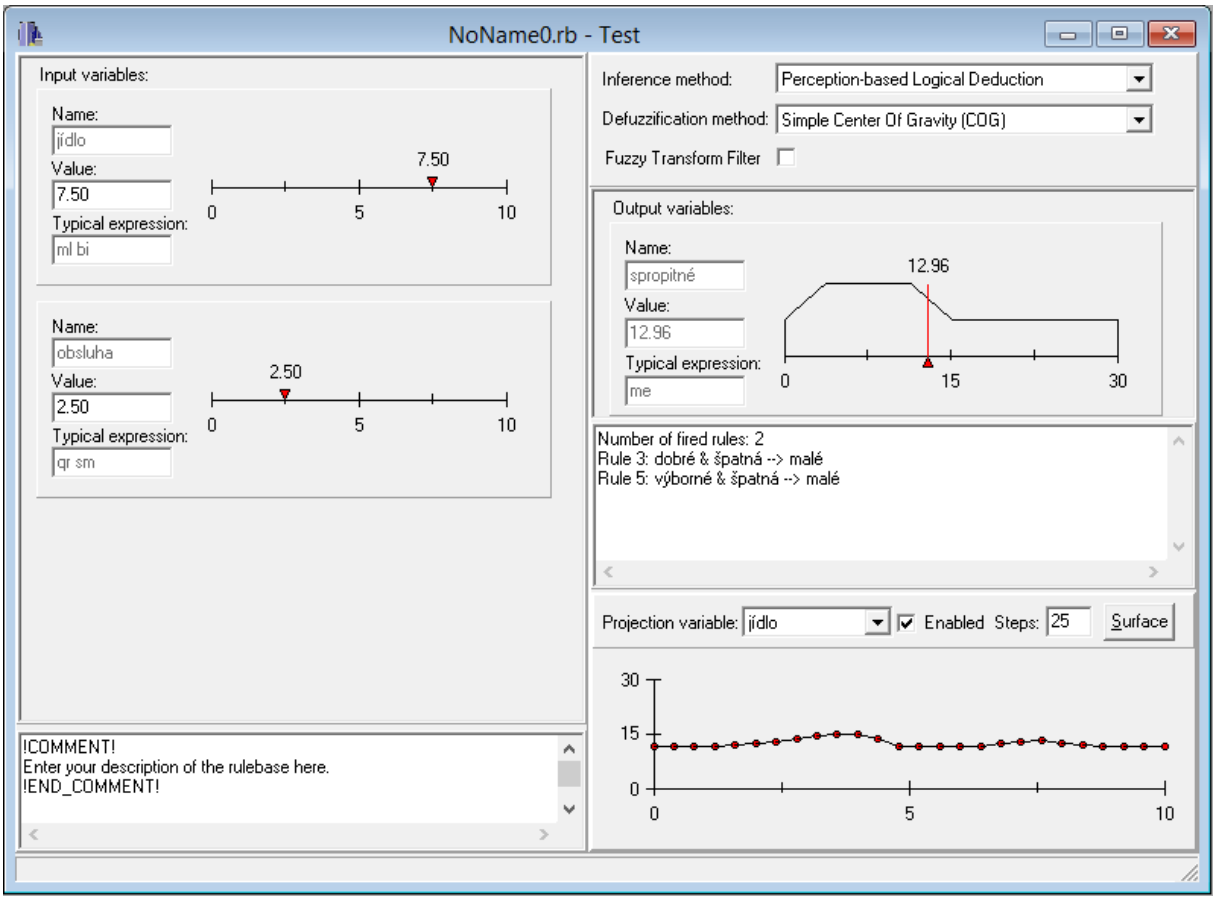
Obr. 7.8 Záložka s editací pravidel

7.2.4 Input / Output

Na poslední záložce je možné pracovat s daty uloženými v souboru.[11]

7.2.5 Test

Po zadání všech parametrů popisu systému, je možnost testu. Tento nástroj se spustí tlačítkem *Test*. V tomto okně jsou vlevo vstupní proměnné. Jejich hodnota se nastavuje posuvníkem nebo se zadá číslo do políčka *Value*. V pravé části nahoře jdou měnit metody inference a defuzzifikace. Vpravo uprostřed se zobrazuje výsledná fuzzy množina a výsledek. Vpravo dole je dvourozměrné zobrazení funkce vytvořené jazykovým popisem. V případě více vstupů, je možné tlačítkem *Surface* zobrazit trojrozměrnou charakteristiku.

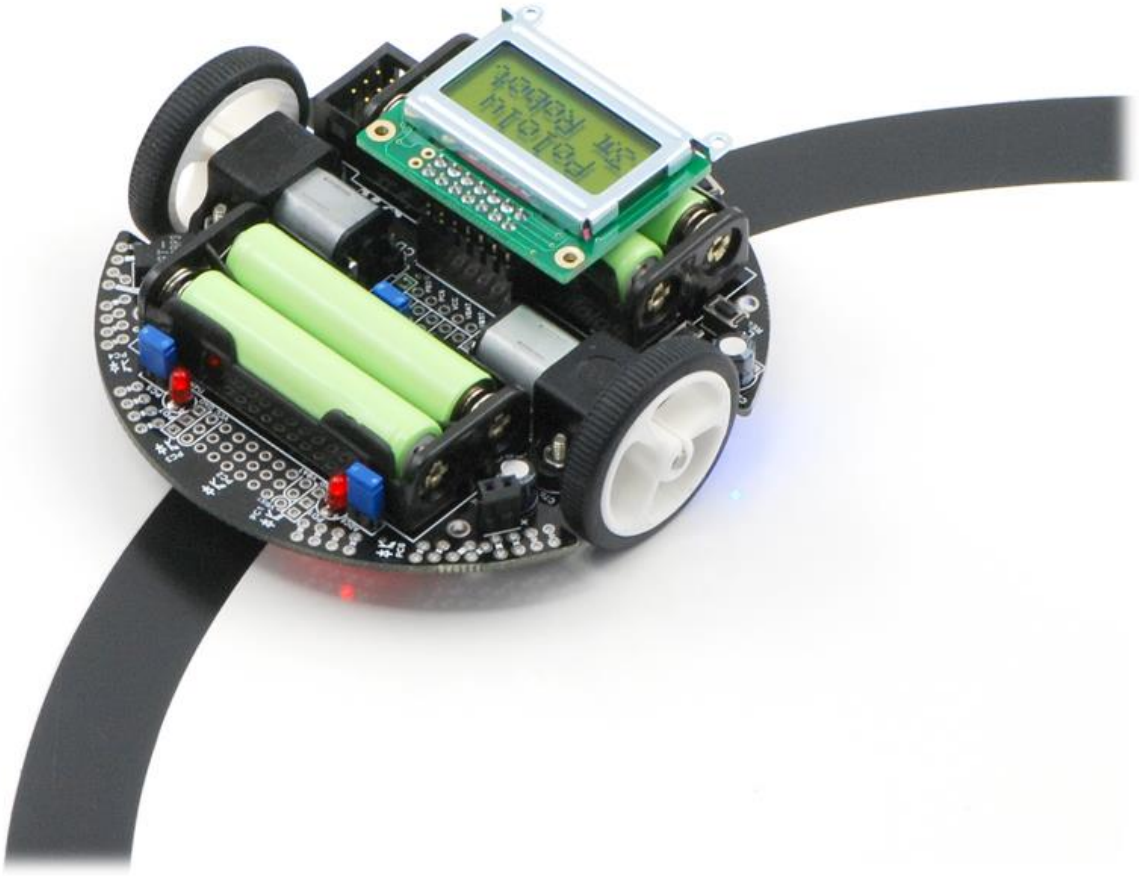


Obr. 7.9 Okno pro testování

8 Robot

K realizaci byl vybrán robot Pololu 3pi. Robot je navržen tak, aby vynikal v úkolech „line-follower“ a „maze-solver“. Jde o malého dvoukolového robota s průměrem 9,5 cm s maximální rychlostí 100 cm/s. O napájení se starají čtyři AAA baterie. Motory pracují s napětím 9,25 V nezávisle na stavu baterií.[12]

Robot je vybaven dvěma motory, LCD 8x2 znaků, pěti odrazovými snímači, bzučákem a třemi uživatelskými tlačítky. Dále je možno připojovat jiné příslušenství, jako například bluetooth, které jsem také využil.[12]



Obr. 8.1 Robot Pololu 3pi (převzato z [9])

Základem robota je mikrokontrolér Atmel ATmega328 s taktem 20 MHz, 32 KB flash paměti, 2 KB RAM a 1 KB EEPROM. Programování probíhá pomocí externího programátoru. Jako vývojové prostředí lze využít Atmel Studio. A protože mikrokontrolér je stejný jako v Arduinu, může se použít i Arduino IDE.[12]

9 Fuzzy knihovny

9.1 Embedded Fuzzy Logic Library

Embedded Fuzzy Logic Library je všestranná, lehká a výkonná knihovna umožňující pracovat s fuzzy logikou na embedded systémech.

Knihovna je napsána v C++/C a používá pouze standardní jazyk C knihovny *stdlib.h*, takže knihovna je navržena nejen pro Arduino, ale i pro jiné embedded systémy.[13]

Knihovna nemá žádná omezení, která se týkají vstupů, výstupů nebo pravidel. Jejich počet je limitován pouze výpočetním výkonem a pamětí mikrokontroléru.[13]

Výpočet se uskutečňuje pomocí metod MAX, MIN, typ MAMDANI a metoda defuzzifikace je CENTER OF AREA.[10]

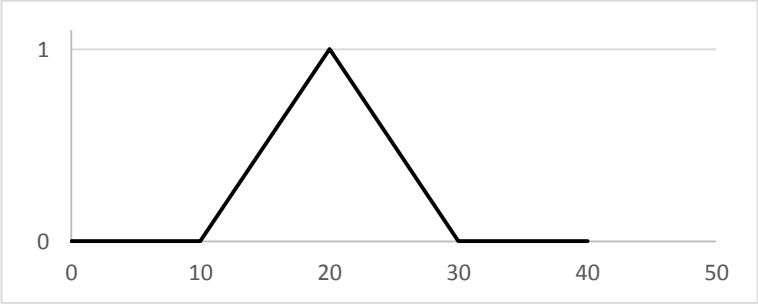
9.1.1 Dokumentace

- **Fuzzy objekt** – obsahuje všechny části fuzzy systému.[14]
- **FuzzyInput objekt** – obsahuje všechny funkce příslušnosti pro vstup.[14]
- **FuzzyOutput objekt** – je obdoba FuzzyInput, ale pro výstup.[14]
- **FuzzySet objekt** – definuje funkci příslušnosti. Podporované jsou Λ - funkce, Π - funkce a singletony. Jsou dány pomocí čtyř bodů, které se zadávají pomocí konstruktoru.[14]

Příklady:

Λ – funkce

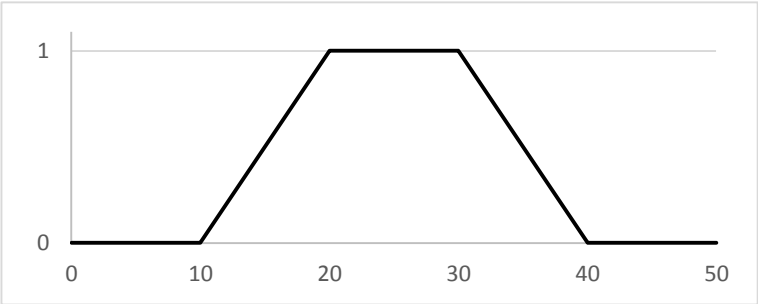
```
FuzzySet* fs = FuzzySet(10, 20, 20, 30);
```



Obr. 9.1 Λ – funkce [14]

Π – funkce

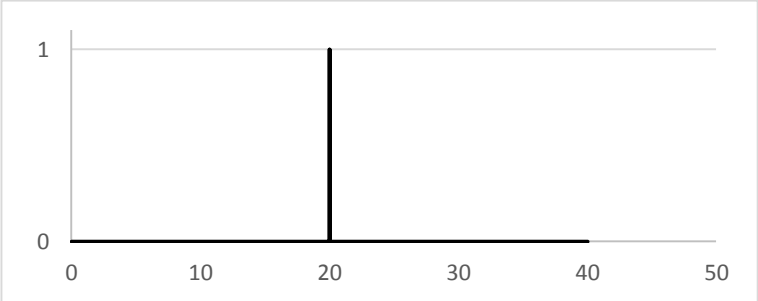
```
FuzzySet* fs = FuzzySet(10, 20, 30, 40);
```



Obr. 9.2 Π – funkce [14]

Singleton

```
FuzzySet* fs = FuzzySet(20, 20, 20, 20);
```



Obr. 9.3 Singleton [14]

- **FuzzyRule objekt** – obsahuje fuzzy pravidlo.
- **FuzzyRuleAntecedent objekt** – obsahuje antecedent pravidla.
- **FuzzyRuleConsequente objekt** – obsahuje konsekvent fuzzy pravidla.

Přidání pravidla do fuzzy objektu se provádí metodou *addFuzzyRule(FuzzyRule* fuzzyRule)*.

Uvedené objekty slouží pro nadefinování fuzzy systému. Pro jeho použití jsou důležité tři následující metody.

- **Metoda pro zadávání hodnot vstupů.**

```
bool setInput(int id, float value);
```

Parametr *id* určuje vstup a *value* je hodnota vstupu.

- **Metoda pro spuštění fuzifikace.**

```
bool fuzzify();
```

- **Metoda pro dokončení fuzifikačního procesu.**

```
float defuzzify(int id);
```

Parametr *id* určuje výstup.

Někdy je vhodné vědět, jestli byla použita určitá funkce příslušnosti. K tomu slouží funkce *float getPertinence()*. Nebo jestli bylo aktivováno určité pravidlo. To se zjistí funkcí *bool isFiredRule(int ruleId)*.

Příklady pro uvedené objekty a funkce viz příloha B.

9.2 Fuzzy Logic Library for Microsoft .Net

Tato knihovna je snadno použitelnou komponentou, která implementuje fuzzy inferenční systém (podporován je systém Mamdani i Sugeno). Knihovna je napsána v jazyce C#.[15]

10 Program pro robota

Pro robota Pololu 3pi jsem vytvořil fuzzy regulátor v úloze „line-follower“, to znamená, že musí jet po vyznačené čáře.

Jak jsem uvedl v kapitole o robotovi. Kontrolér je stejný, jako v Arduinu, proto jsem použil pro fuzzy logiku knihovnu Embedded Fuzzy Logic Library. K návrhu fuzzy regulátoru posloužil popsaný FLT Toolbox.

10.1 Návrh regulátoru

Regulátor jsem zvolil typu PD. Pravidla tohoto regulátoru mají tvar[2]:

$$IF\ (e(k)\ is\ A)\ and\ (\Delta e(k)\ is\ B)\ THEN\ (u(k)\ is\ C). \qquad (10.1)$$

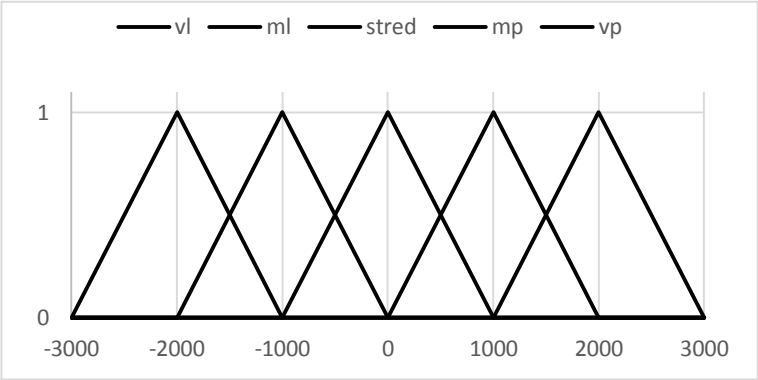
e(k) – odchylka, Δe(k) – změna odchylky, u(k) – akční zásah a A, B, C – jazykové výrazy.

10.1.1 Vstupní a výstupní proměnné

Odchylka

Pro vstupní proměnnou odchylka jsem zvolil pět Λ - funkcí příslušnosti.

- vl – velká-levá
- ml – malá-levá
- stred – žádná odchylka
- mp – malá-pravá
- vp – velká-pravá



Obr. 10.1 Funkce příslušnosti pro odchylku

Univerzum této proměnné je interval ⟨-2000, 2000⟩. Tyto hodnoty jsem získal úpravou intervalu ⟨0, 4000⟩, který je možný získat ze senzorů.

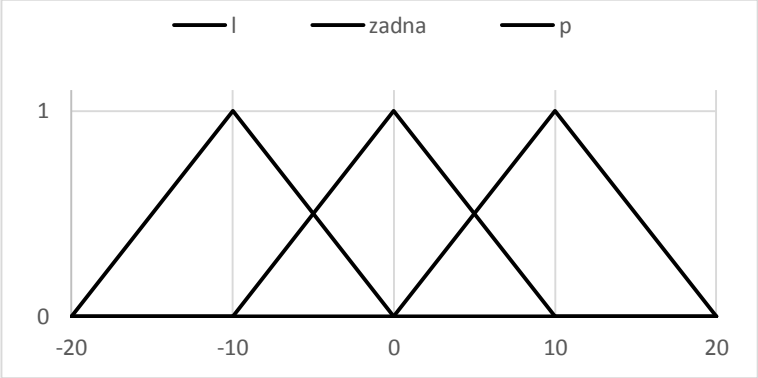
Pro získání hodnoty slouží funkce *readLine()*.

```
unsigned int pos = qtr3pi.readLine(sensors, 1);
int e = pos - 2000;
```

Změna odchylky

Pro tuto vstupní proměnnou jsem zvolil tři Λ-funkce příslušnosti.

- l – záporná změna
- zadna – žádná změna
- p – kladná změna



Obr. 10.2 Funkce příslušnosti pro změnu odchylky

Zde jsem zvolil interval ⟨-20, 20⟩. Tento interval považuji za dostatečný.

Změna odchylky se získá rozdílem aktuální a předchozí odchylky.

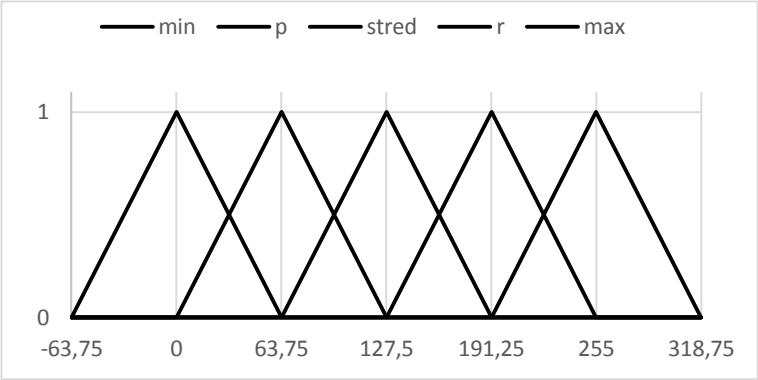
```
int de = pos - lastPos;
```

Levý a pravý motor

Výstupní proměnné jsou dvě. Jedna pro levý motor a druhá pro pravý. Tyto proměnné jsou definovány stejně. Mají pět Λ – funkcí příslušnosti.

- min – minimální rychlost
- p – pomalu
- stred – střední rychlost
- r – rychle

- max – maximální rychlost



Obr. 10.3 Funkce příslušnosti pro výstupy (levý a pravý motor)

Vzhledem k možnosti nastavení rychlosti motoru v intervalu hodnot $\langle 0, 255 \rangle$, jsem použil tento interval i pro výstupní proměnné.

10.1.2 Báze pravidel

IF (odchylka is vl) THEN (leve is min)(prave is max)
IF (odchylka is ml) AND (změna is l) THEN (leve is stred)(prave is r)
IF (odchylka is ml) AND (změna is zadna) THEN (leve is p)(prave is r)
IF (odchylka is ml) AND (změna is p) THEN (leve is min)(prave is r)
IF (odchylka is stred) THEN (leve is stred)(prave is stred)
IF (odchylka is mp) AND (změna is l) THEN (leve is r)(prave is min)
IF (odchylka is mp) AND (změna is zadna) THEN (leve is r)(prave is p)
IF (odchylka is mp) AND (změna is p) THEN (leve is r)(prave is stred)
IF (odchylka is vp) THEN (leve is max)(prave is min)

10.2 Program

Program se skládá ze dvou základních funkcí: setup() a loop(). Funkce setup() je vyvolána při startu programu, zatímco funkce loop() je spouštěna stále dokola.

Na začátku programu je nutné nejdříve přidat potřebné knihovny.

```
…
#include <PololuQTRSensors.h>
#include <FuzzyRule.h>
…
```

Potom jsou definovány konstanty pro tlačítka, LED, senzory a další objekty.

```
…
const int pinButtonA = 9;
…
```

A definice funkcí příslušnosti.

```
…
FuzzySet* odchylka_vl = new FuzzySet(-3000, -2000, -2000, -1000);
FuzzySet* odchylka_ml = new FuzzySet(-2000, -1000, -1000, 0);
…
```

Funkce fuzzy_set() je funkce, ve které jsem vytvořil vstupy, výstupy a pravidla. Tato funkce je vyvolána v setup().

```
…
FuzzyInput* odchylka = new FuzzyInput(1);
odchylka->addFuzzySet(odchylka_vl);
…
fuzzyreg->addFuzzyInput(odchylka);
…
…
FuzzyRuleAntecedent* ifOdchylka_vl = new FuzzyRuleAntecedent();
ifOdchylka_vl->joinSingle(odchylka_vl);

FuzzyRuleConsequent* thenLeve_min_0Prave_max_0 = new
FuzzyRuleConsequent();
thenLeve_min_0Prave_max_0->addOutput(leve_min);
thenLeve_min_0Prave_max_0->addOutput(prave_max);
FuzzyRule* rule1 = new FuzzyRule(1, ifOdchylka_vl,
thenLeve_min_0Prave_max_0);
fuzzyreg->addFuzzyRule(rule1);
…
```

Funkce loop() je hlavní částí programu, protože běží stále dokola. Proto v ní probíhá čtení hodnot ze senzorů, fuzzifikace, defuzzifikace.

```
…
unsigned int pos = qtr3pi.readLine(sensors, 1);
e = pos - 2000;
de = pos - lastPos;
```

```
// vstupy fuzzy
fuzzyreg->setInput(1,e);
fuzzyreg->setInput(2,de);
// fuzzifikace
fuzzyreg->fuzzify();
// vystupy fuzzy, defuzzifikace
float leve = fuzzyreg->defuzzify(1);
float prave= fuzzyreg->defuzzify(2);
…
```

Funkce pro nastavení rychlostí motorů.

```
…
motors.setSpeeds((int) leve, (int) prave);
…
```

Dále jsem vytvořil podmínku pro změnu rychlosti. V případě odchylky menší než 100 se rychlost postupně zvyšuje o 1 během každého proběhnutí funkce *loop()*. Když je odchylka větší, rychlost se snižuje o 5.

```
…
if(abs(e)<100)
{
  plus = plus + 1;
}
else
{
  plus = plus - 5;
  if(plus < 0) plus = 0;
}
…
```

11 Software pro převod systému FLT do C kódu

Pro zjednodušení aplikace fuzzy regulátoru pomocí eFLL jsem vytvořil aplikaci, generující C kód ze souboru, který ukládá Fuzzy Logic Toolbox.

Soubor fis se skládá z několika částí.

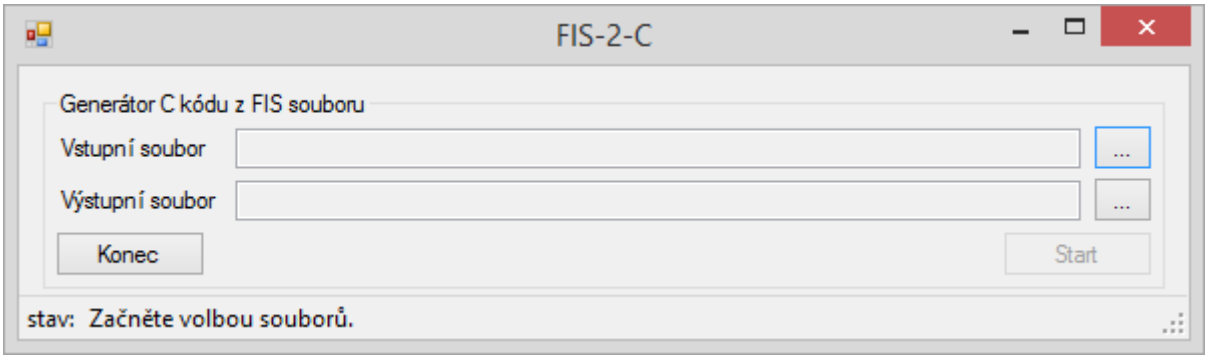
- **System** – obsahuje základní informace o systému jako název, typ, počet vstupů, výstupů a použité metody.
- **Input, Output** – další části jsou jednotlivé vstupy a výstupy. Každý vstup nebo výstup má svůj blok, ten nese informaci o názvu, rozsahu, počtu funkcí příslušnosti a parametry samotných funkcí příslušnosti.
- **Rules** – obsahuje seznam pravidel. Tyto pravidla jsou zapsána ve zjednodušeném tvaru.

Pravidlo: 1 1, 1 5 (1) : 1

Čísla před čárkou rozdělené mezerou jsou jednotlivé vstupy. Samotná hodnota pak určuje ID funkce příslušnosti. Za čárkou jsou stejně uvedeny výstupy. Číslo v závorce je váha pravidla. Poslední číslo udává spojení antecedentu (1 pro AND, 2 pro OR).

Ukázkový fis soubor viz příloha C.

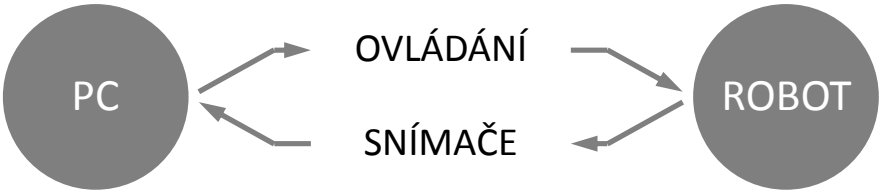
Použití aplikace je velmi jednoduché. Po spuštění se vybere vstupní soubor (*.fis), ten se získá exportem fuzzy systému ve Fuzzy Logic Toolboxu. Vybere se umístění a název výstupního souboru. Generování kódu se spustí tlačítkem Start. Výsledný kód se otevře v textovém editoru. Pro použití kódu stačí překopírovat do vývojového prostředí.



Obr. 11.1 Program pro převod kódu

12 Ovládání robota z počítače

Nejdříve jsem hledal vhodné nástroje, které by umožňovaly řízení robota, kde by roli regulátoru zastával počítač. Robot by pouze odesílal hodnotu ze snímačů a přijímal data pro nastavení rychlosti obou motorů.



Obr. 12.1 Schéma ovládání pomocí PC

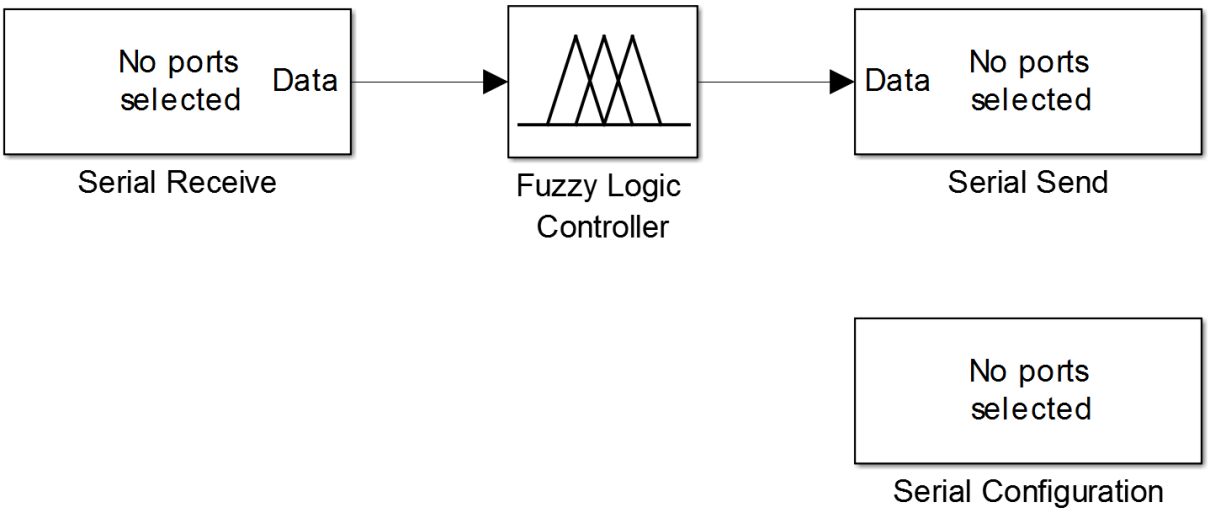
12.1 Simulink

Jako první možnost mě napadlo využít Simulink. Simulink je součást MATLABu pro modelování a simulaci dynamických systémů.

Pro samotnou regulaci jsem využil blok Fuzzy Logic Controller. V tomto bloku se nastavuje FIS, který se navrhne pomocí Fuzzy Logic Toolboxu.

- Bloky Serial Receive, Serial Send

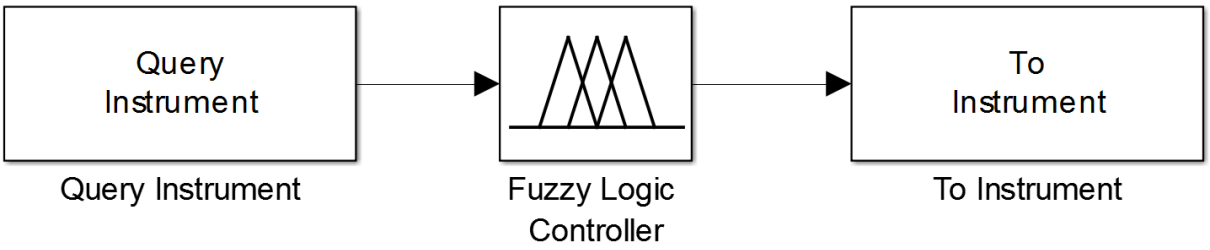
Pro přenos dat po seriové lince je možné využít bloky *Serial Receive*, *Serial Send* a s nimi související *Serial Configuration*.



Obr. 12.2 Schéma s bloky Serial Receive a serial Send

- Bloky Query Instrument, To Instrument

Další možností přenosu dat po sériové lince je pomocí bloků *Query Instrument* a *To Instrument*.



Obr. 12.3 Schéma s bloky Query Instrument a To Instrument

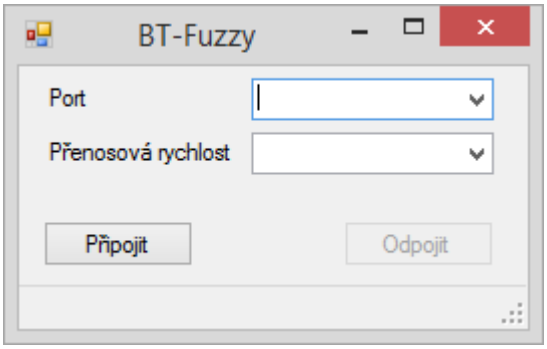
Přestože jsem našel dva způsoby jak přenášet data po sériové lince, nepodařilo se mi bloky nakonfigurovat, tak aby fungoval alespoň ukázkový příklad.

12.2 Vlastní software pro ovládání robota z PC

Jelikož jsem byl při použití Simulinku neúspěšný, rozhodl jsem se vytvořit vlastní program.

Program jsem napsal v C# a využil jsem již zmíněnou knihovnu fuzzynet (Fuzzy Logic Library for Microsoft .Net). Tento program jsem vytvořil jako ukázku možnosti ovládání z PC.

Fuzzy regulátor je pevně zadán ve zdrojovém kódu. Definován byl podle návrhu ve Fuzzy Logic Toolboxu. Ovládání programu je tedy velmi jednoduché. Nejdříve se vybere příslušný COM port a přenosová rychlost. Potom stačí stisknout tlačítko *Připojit*.



Obr. 12.4 Program pro ovládání robota

K tomuto účelu jsem musel vytvořit i program pro robota, který s tímto programem komunikuje.

Tento program pro robota má stejný základ jako program, který obsahoval fuzzy regulátor. Rozdíl je v tom, že neobsahuje fuzzy logiku, ale je v něm dopsána komunikace přes bluetooth.

Ve funkci *setup()* přibylo.

```
…
pinMode(A7, OUTPUT);
digitalWrite(A7, 1);
delay(50);
digitalWrite(A7, 0);
Serial.begin(19200);
while (!Serial)
{
  buzzer.playFrequency(1200, 20, 15);
  delay(30);
}
…
```

Dále jsem dopsal funkci *serialEvent()*. Tato funkce se vyvolá v případě, že přichází data. Přenos jsem si vytvořil ve tvaru *100x120k*. Čtení probíhá po znacích, až přijde znak *x*, který označuje ukončení první hodnoty (hodnota pro levý motor) a znak *k*, ten označuje ukončení druhé hodnoty (hodnota pro druhý motor) a celého přenosu.

```
void serialEvent()
{
  leftMotor = "";
  rightMotor = "";
  char c;
  boolean left = true;

  while (Serial.available())
  {
    c = (char)Serial.read();
    if (left) {
      if (c == 'x') {
        left = false;
        continue;
      }
      leftMotor = leftMotor + c;
    }
    else {
      if (c == 'k') stringComplete = true;
      rightMotor = rightMotor + c;
    }
  }
}
```

Ve funkci *loop()* se zjistí hodnota ze senzorů a odešle se do počítače. Poté je zpoždění, aby proběhl přenos dat. Když se dokončí přenos dat z počítače, nastaví se rychlost motorů.

```
…
Serial.print(pos);
Serial.print('k');
delay(13);
if(stringComplete)
{
  stringComplete = false;
  lm = leftMotor.toInt();
  rm = rightMotor.toInt();
}
…
```

13 Závěr

V první části této bakalářské práce jsem se zaměřil na fuzzy množiny a rozdíl oproti klasickým množinám. Popsal jsem důležité pojmy týkající se fuzzy množin. Uvedl jsem operace s fuzzy množinami a příklady k nim. Dále jsem popsal proces fuzzifikace a defuzzifikace, které jsou důležitou součástí fuzzy systému. Také jsem popsal systémy Mamdani a Takagi-Sugeno-Kang. Tato část je důležitá k pochopení fuzzy logiky a regulace.

V další části jsem se podle zadání zaměřil na popis softwarových nástrojů pro návrh fuzzy systému. Popsal jsem Fuzzy Logic Toolbox (FLT), který je součástí a MATLABu a LFLC 2000 softwarový balík, vyvíjen na Ústavu pro výzkum a aplikace fuzzy modelování při Ostravské Univerzitě v Ostravě.

Na základě získaných znalostí o fuzzy logice jsem vytvořil program pro malého dvoukolového robota, který sleduje čáru. V tomto programu byl vytvořen fuzzy regulátor pomocí knihovny Embedded Fuzzy Logic Library, kterou jsem si vyhledal. Tuto knihovnu jsem popsal a uvedl příklady požití. Fuzzy regulátor jsem navrhl pomocí FLT.

K usnadnění přepisování fuzzy systému z FLT do C kódu, jsem vytvořil program, který dokáže generovat C kód z výstupního souboru FLT.

Nakonec jsem uvedl možnosti ovládání robota z počítače. První možností bylo ovládání skrze Simulink, které se mi ale nepovedlo prakticky realizovat. Zaměřil jsem se tedy na vytvoření vlastního programu, který obsahuje fuzzy regulátor a přes bluetooth ovládá robota. V rámci tohoto řešení jsem také vytvořil program pro robota, umožňující ovládání z počítače.

Seznam literatury a informačních zdrojů

[1] ZADEH, Lotfi A. Fuzzy sets. *Information and Control*. 1965, vol. 8, issue 3. Dostupné z: http://www.cs.berkeley.edu/~zadeh/papers/Fuzzy%20Sets-Information%20and%20Control-1965.pdf

[2] TŮMA, František. *Automatické řízení 2: diskrétní systémy, logické systémy, nelineární sytémy, fuzzy systémy*. 2., upr. vyd. V Plzni: Západočeská univerzita, 2007, 183 s. ISBN 978-80-7043-569-4.

[3] NOVÁK, Vilém. *Základy fuzzy modelování: diskrétní systémy, logické systémy, nelineární sytémy, fuzzy systémy*. Vyd. 1. Praha: BEN, 2000, 176 s. ISBN 80-730-0009-1.

[4] ONDROUŠEK, Vít a Jaroslav PULCHART. *Úvod do fuzzy logiky a fuzzy regulátory* [online]. Brno: Vysoké učení technické v Brně, 2007 [cit. 2014-04-17]. Dostupné z: http://autnt.fme.vutbr.cz/lab/a4-716/vyuka/rir//pdf/fuzzy.pdf

[5] ZADEH, Lotfi A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*. 1973, vol. SMC-3, no. 1. Dostupné z: http://www.cs.berkeley.edu/~zadeh/papers/1973-Outline%20of%20a%20New%20Approach%20to%20the%20Analysis%20of%20Complex%20Systems%20and%20Decision%20Processes.pdf

[6] BONISSONE, Piero P. *Fuzzy Sets & Expert Systems in Computer Eng. (5): Linguistic Variables*. 1998. Dostupné z: http://homepages.rpi.edu/~bonisp/fuzzy-course/99/L5/formalgrammar.pdf

[7] MODRLÁK, Osvald. *Fuzzy řízení a regulace* [online]. Liberec: Technická univerzita v Liberci, 2002 [cit. 2014-04-18]. Dostupné z: https://www.kirp.chtf.stuba.sk/~bakosova/wwwRTP/tar2fuz.pdf

[8] What Is Mamdani-Type Fuzzy Inference?. *MathWorks - MATLAB and Simulink for Technical Computing - B* [online]. © 1994-2014 [cit. 2014-05-15]. Dostupné z:http://www.mathworks.com/help/fuzzy/what-is-mamdani-type-fuzzy-inference.html

[9] SEDONÍK, Jiří. *Implmentace báze pravidel a fuzzy regulátoru v rozhodovacích problémech demografických studií* [online]. Olomouc, 2012 [cit. 2014-05-21]. Dostupné z: http://theses.cz/id/sqtk22/Sedonik_DP.pdf. Magisterská práce. Univerzita Palackého v Olomouci.

[10] Build Mamdani Systems (GUI). *MathWorks - MATLAB and Simulink for Technical Computing - B* [online]. © 1994-2014 [cit. 2014-04-29]. Dostupné z:http://www.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html

[11] ÚSTAV PRO VÝZKUM A APLIKACE FUZZY MODELOVÁNÍ OSTRAVSKÁ UNIVERZITA V OSTRAVĚ. *LFLC 2000 Linguistic Fuzzy Logic Controller*. Ostrava, 2003.

[12] Pololu 3pi Robot. *Pololu Robotics and Electronics* [online]. © 2001–2014 [cit. 2014-05-02]. Dostupné z: http://www.pololu.com/product/975/

[13] EFLL - C++ Fuzzy Library: Introduction. *ZEROKOL* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://zerokol.com/product/51e93616e84c5571b7000018/2/en

[14] EFLL - C++ Fuzzy Library: Documentation. *ZEROKOL* [online]. 2014 [cit. 2014-05-02]. Dostupné z: http://zerokol.com/product/51e93616e84c5571b7000018/4/en

[15] Fuzzy Logic Library for Microsoft .Net. *SourceForge* [online]. 2008 [cit. 2014-05-03]. Dostupné z: http://fuzzynet.sourceforge.net/

Přílohy

Příloha A – DVD se zdrojovými kódy

Příloha B – příklady použití eFLL

Příklad – jednoduchý antecedent.

```
"IF distance = small THEN velocity = slow"

FuzzyRuleAntecedent* ifDistanceSmall = new FuzzyRuleAntecedent();
ifDistanceSmall->joinSingle(small);
```

Metoda joinSingle() slouží ke tvorbě jednoduchých pravidel IF – THEN.

Příklad – složitější antecedent.
Pro tvorbu složitějších pravidel existují metody joinWithAnd() a joinWithOr().

```
"IF temperature = hot AND pressure = hight THEN rick = big"

FuzzyRuleAntecedent* ifTemperatureHotAndPressureHight = new
FuzzyRuleAntecedent();
ifTemperatureHotAndPressureHight->joinWithAND(hot, hight);
"IF temperature = hot OR pressure = hight THEN rick = big"

FuzzyRuleAntecedent* ifTemperatureHotAndPressureHight = new
FuzzyRuleAntecedent();
ifTemperatureHotAndPressureHight->joinWithOR(hot, hight);
```

Příklad – spojování.
Metody joinWithAnd() a joinWithOr() mohou spojovat nejen funkce příslušnosti, ale také antecedenty.

```
bool joinWithAND(FuzzySet* fuzzySet, FuzzyRuleAntecedent*
fuzzyRuleAntecedent);
bool joinWithAND(FuzzyRuleAntecedent* fuzzyRuleAntecedent, FuzzySet*
fuzzySet);
bool joinWithOR(FuzzySet* fuzzySet, FuzzyRuleAntecedent*
fuzzyRuleAntecedent);
bool joinWithOR(FuzzyRuleAntecedent* fuzzyRuleAntecedent, FuzzySet*
fuzzySet);
bool joinWithAND(FuzzyRuleAntecedent* fuzzyRuleAntecedent1,
FuzzyRuleAntecedent* fuzzyRuleAntecedent2);
bool joinWithOR(FuzzyRuleAntecedent* fuzzyRuleAntecedent1,
FuzzyRuleAntecedent* fuzzyRuleAntecedent2);
```

```
"IF (velocity = hight AND distance = small) OR fuel = low THEN velocity =
small AND consumption = short"

FuzzyRuleAntecedent* speedHightAndDistanceSmall = new
FuzzyRuleAntecedent();
speedHightAndDistanceSmall->joinWithAND(hight, small);

FuzzyRuleAntecedent* fuelLow = new FuzzyRuleAntecedent();
fuelLow->joinSingle(low);

FuzzyRuleAntecedent* ifSpeedHightAndDistanceSmallOrFuelLow = new
FuzzyRuleAntecedent();
ifSpeedHightAndDistanceSmallOrFuelLow-
>joinWithOR(speedHightAndDistanceSmall, fuelLow);
```

nebo zjednodušeně

```
FuzzyRuleAntecedent* speedHightAndDistanceSmall = new
FuzzyRuleAntecedent();
speedHightAndDistanceSmall->joinWithAND(hight, small);

FuzzyRuleAntecedent* ifSpeedHightAndDistanceSmallOrFuelLow = new
FuzzyRuleAntecedent();
ifSpeedHightAndDistanceSmallOrFuelLow-
>joinWithOR(speedHightAndDistanceSmall, low);
```

Příklad – jednoduchý konsekvent.

```
"IF disctance = small THEN velocity = slow"

FuzzyRuleConsequent* thenSpeedSlow = new FuzzyRuleConsequent();
thenSpeedSlow->addOutput(slow);
```

Fuzzy pravidlo

```
FuzzyRule* fuzzyRule = new FuzzyRule(2, ifDistanceSmall, thenSpeedSlow);
```

Příklad – konsekvent pro dva výstupy.

```
FuzzyRuleConsequent* thenSpeedSmallAndFeedTine = new FuzzyRuleConsequent();
thenSpeedSmallAndFeedSmall->addOutput(small);
thenSpeedSmallAndFeedSmall->addOutput(tine);
```

Fuzzy pravidlo

```
FuzzyRule* fuzzyRule = new FuzzyRule(2,
ifSpeedHightAndDistanceSmallOrFuelLow, thenSpeedSmallAndFeedTine);
```

Příklad – zjištění použití funkce příslušnosti.

```
FuzzySet* hot = new FuzzySet(30, 50, 50, 70);
...
... // po fuzzifikaci ->fuzzyfy();
...
float pertinenceOfHot = hot->getPertinence();
```

Příklad – zjištění aktivace pravidla.

```
FuzzyRule* fuzzyRule = new FuzzyRule(2, ifDistanceSmall, thenSpeedSlow);
...
... // po fuzzifikaci ->fuzzyfy();
...
bool wasTheRulleFired = fuzzy->isFiredRule(2);
```

Příloha C – FIS soubor – návrh fuzzy regulátoru

```
[System]
Name='lineFollower'
Type='mamdani'
Version=2.0
NumInputs=2
NumOutputs=2
NumRules=9
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='odchylka'
Range=[-2000 2000]
NumMFs=5
MF1='vl':'trimf',[-3000 -2000 -1000]
MF2='ml':'trimf',[-2000 -1000 0]
MF3='stred':'trimf',[-1000 0 1000]
```

```
MF4='mp':'trimf',[0 1000 2000]
MF5='vp':'trimf',[1000 2000 3000]

[Input2]
Name='zmena'
Range=[-10 10]
NumMFs=3
MF1='l':'trimf',[-20 -10 0]
MF2='zadna':'trimf',[-10 0 10]
MF3='p':'trimf',[0 10 20]

[Output1]
Name='leve'
Range=[0 255]
NumMFs=5
MF1='min':'trimf',[-63.75 0 63.75]
MF2='p':'trimf',[0 63.75 127.5]
MF3='stred':'trimf',[63.75 127.5 191.3]
MF4='r':'trimf',[127.5 191.3 255]
MF5='max':'trimf',[191.3 255 318.8]

[Output2]
Name='prave'
Range=[0 255]
NumMFs=5
MF1='min':'trimf',[-63.75 0 63.75]
MF2='p':'trimf',[0 63.75 127.5]
MF3='stred':'trimf',[63.75 127.5 191.3]
MF4='r':'trimf',[127.5 191.3 255]
MF5='max':'trimf',[191.3 255 318.8]

[Rules]
1 2, 1 5 (1) : 1
2 1, 3 4 (1) : 1
2 2, 2 4 (1) : 1
2 3, 1 4 (1) : 1
3 2, 3 3 (1) : 1
4 1, 4 1 (1) : 1
4 2, 4 2 (1) : 1
4 3, 4 3 (1) : 1
5 2, 5 1 (1) : 1
```