УВОД

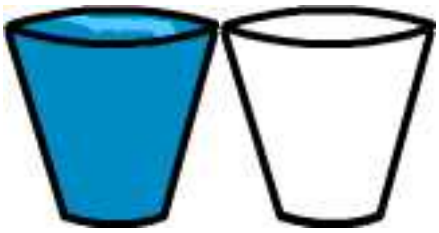
As complexity rises, precise statements lose meaning and meaningful statements lose precision.
—Lotfi Zadeh

Почетоците на фази логиката можат да се бараат во онај момент кога човештвото дошло до сознание дека не постои Архимедова целосна лага ниту целосна вистина тука задоволителен степен на лага или вистина.

Концептот на фази логиката како идеа била презентирана од Lotfi Zadeh, професор на „Беркли" универзитетот во Калифорнијаа не како контролна методологија, но како начин да се обработуваат податоците дозволувајќи делумна припадност во множество (*фази множество*) во однос на „тврда" (*crisp*) припадност или неприпадност во множество (*класично множество*). Zadeh, гледајќи дека многу множествата во светот што не опкружува се дефинирани без јасни граници (високи луѓе, планинини, океани, броеви поголеми од , решил да ја прошири дво-вредносната логика, дефинирана со бинарниот {0,1} (вистина или невистина) на целиот интервал [0,1] на тој начин претставувајќи градиентна транзиција од лага до вистина. За претставување на степенот на вклопување на елементот во одредено фази множество, Zadeh предложил степен на припадност (*grade of membership*) μ.

Можеби еден од најдобрите примери за презнтирање на фази логичко размислување е примерот со „чашата со вода".
Дефинирано е множество од исполнети чаши.

Класичното размислување би разликувало дали е чашата полна (чашата припаѓа на дефинираното множество $\mu=1$) или чашата е празна (чашата не припаѓа на дефинираното множество $\mu=0$)



Во случај на исполнетост на чашата од пример 77% ваквото размислување ја губи својата сила односно се поставува прашањето дали чашата е полна или празна.

Фази размислувањето би разликувало 77% припадност на чашата во дефинираното множество , односно степен на припадност $\mu=0.77$

Истотака Lotfi Zadeh го предложил концептот на лингвистички или „фази" променливи со што се направило лингвистичко изразување на реалните броеви или вредности (температура, напон, плата, влажност, брзина, исполнетост...) во специфичен опсег т.е фази множество.

Со класичното размислување во споменатиот пример би можеле да дефинираме „полна" или „празна" чаша, додека лингвистички изразувања за опсезите на исполнетоста (често сретнувани во природниот јазик) како „скоро празна" (пример од 0-12% исполнетост), „многу" полна(пример од 80-100% исполнетост), „ептен многу" полна (пример од 90-100% исполнетост),, „средно" полна(пример од 45-55% исполнетост),... можат да опстојуваат само во фази размислувањето.

Фази-логичко управување

Се сметаше дека фази логиката е подобар метод за сортирање и ракување со податоците, но се докажа и како одличен избор за многу системи на контрола бидејќи ја имитира логиката на човечка контрола. Околу 1975, Ebrahim Mamdani и S. Assilian на „Queen Mary()" колеџот на Универзитетот во Лондон (Англија) публиковале насловен како "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," каде фисибилити на фази логичкиот контолер беше докажан со апликација на фази логичка контрола врз парната машина. Денес фази логиката се користи за изградба од мали, рачни производи до големи компјутеризирани процесни контролни системи.

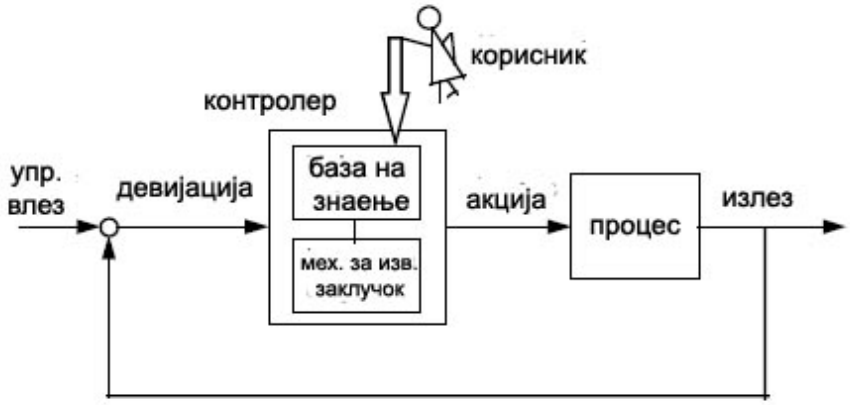
Фази логиката нуди неколку уникатни особини кои ја прават добар избор за многу контролни проблеми:

- На системот на контрола не му потребна прецизност, влезови без шум и можи да биди програмиран сигурност во случај на откажување ако одзивот од сензорот престани или е уништен
- Бидејќи фази контролерот процесира корисничко-дефинирани правила водејќи го одредениот контролен систем, лесно можи да биди модификован и подесен за да го подобри или драстично да ги смени перформансите на системот. Новите сензори можат лесно да се инкорпорираат во ситемот едноставно со генерирање на соодветни правила
- Контролата не е лимитирана на неколку влезови на повратната врска и еден или два контролни излези, ниту е потребно да се мерат или пресметува фрекфенцијата на промена на параметрите за тие да бидат имплементирани. Секакови податоци од сензор кој обезбедува индикација на акциите и реакциите на системот е од повеќе. Ова овозможува сензорите да се евтини и непрецизни на тој начин задржувајќи ја цената и комплексноста на целиот систем ниска.
- Заради операцијата базирана на правила, секој разумен број на влезови кои можат да се процесираат (1-8 или повеќе) и бројни излези (1-4 или повеќе) да се генерираат, и покрај тоа дефинирањето на базата на знаење станува многу комплексна ако многу влезови и излези се избрани за имплементација бидејќи и меѓусебните релации мора да бидат дефинирани истотака.
- Можи да контролира нелинерани системи кои би било тешко или невозможно да се моделира математички, што овозможува отворање на вратите за нивна автоматизација.

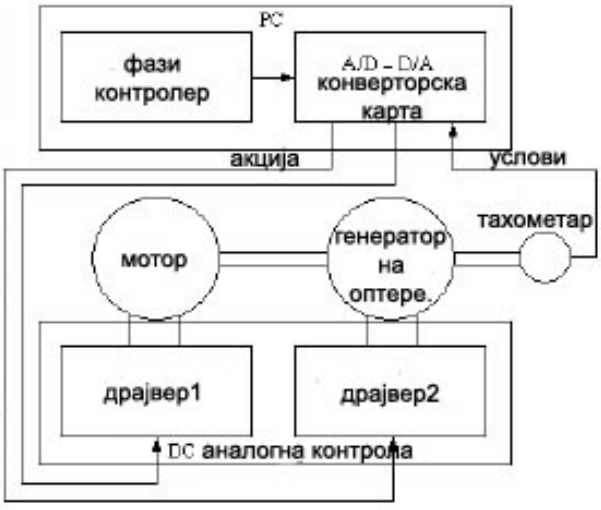
Фази контролерите се користат во најразлични контролни шеми (начини на контрола):

- **Директна контрола** (*direct control*)

  Ова контрола е најупотребувана и се каркатеризира со тоа што фази контролерот е на дирекниот пат на конторолниот систем со повратна врска сл. Излезот на процесот се споредува со референцата, и ако постои девијација, конторлерот превзема акција согласно со контролната стратегија.



Показниот експериментален систем користи директна контрола со фази контролер од PI – тип затоа што ги користи знаењата од класичен PI сл().



Фази контролерот за контрола на DC моторот е имплементиран со користење на PC. Моторот е оптертен со генератор на оптеретување за да обезбеди реални услови и исто така е контролиран од страна на софтверот на PC-то. Аналогно-дигитално и дигитално-аналогната конверзија при комуникацијата меѓу контролата и процесот е остварена со конверторска A/D – D/A карта. Моменталните услови во процесот т.е моменталната брзина на моторот се зема од тахометарот, додека контролната акција во вид на

контролен напон се испраќа до драјверите.

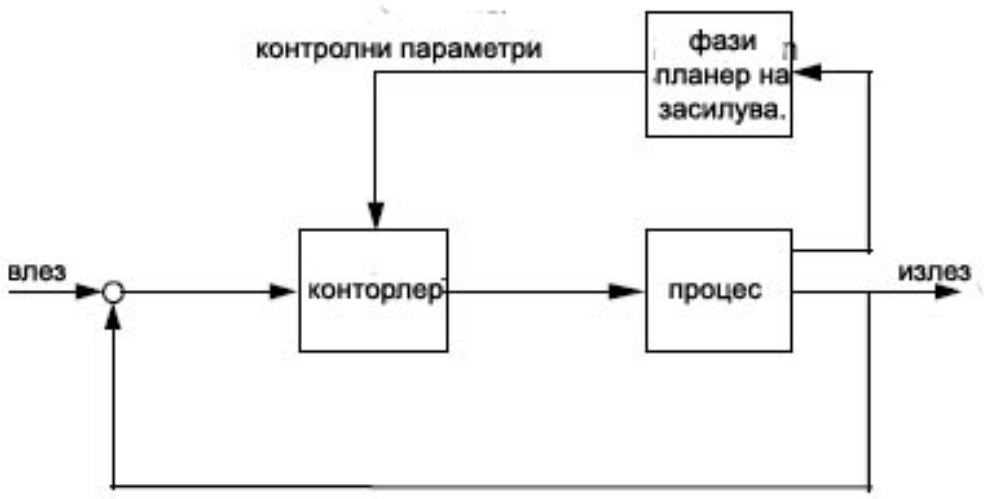
- **Компензациона контрола** (*feedforeward control*)

Ваквиот начин на контрола бара добар математички модел, но ако математичкиот модел е тежок или скап за изведба, фази модел може да биде корисен. На сликата (сл.)

е прикажан контролер или фази компензатор, процесот и повратната врска се изоставени. Начинот, и покрај дистурбацискиот влез, може да се гледа како колаборација на линерени и нелинерни контролни акции. Контролерот може да биде линерен PID контролер, додека компензаторот е фази нелинеарен контролер.

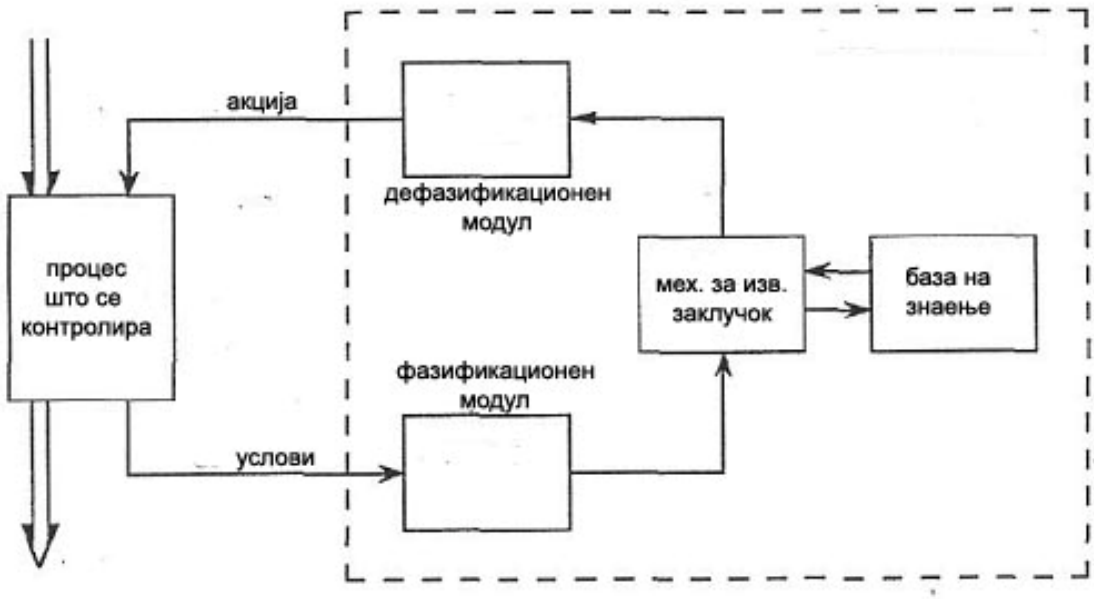
- **Контрола преку адаптација на параметрите на засилување**

Фази правилата истотака можат да се користат за подесување на параметрите сл().



Ако нелинерниот процес ја промени работната точка, можно е да се променат параметрите на контролерот во согласност на секоја работна точка. Ова се вика планирање на засилувањето (*gain scheduling*) бидејќи оригинално било користено за промена на засилувањето на процесот. Планирањето на засилувањето на контролерот содржи линерен контролер чии параметри се променливи во функција на работната точка на претходно програмиран начин. Потребно е прецизно знаење на процесот, но често е добар начин за компензација на нелинеарностите и варијациите на параметрите.
Мерењата на сензорите се користат како планирани променливи (*scheduling variables*) кои ја раководат промената на параметрите на контролерот, често како пребарување по табела.

Блок шема на фази контолер е дадена на слика() која можат да се уочат



следните модули:

- фазификационен модул
- модул за изведување на заклучок (мех. за изведување заклучок и база на знаење)
- дефазификационен модул

Претпроцесирање

Пред да се изврши процесот на фазификација се врши прилагодување на влезната управувачка величина, добиена од некоја опрема за мерење, на доменот (универзумот на говор) на фази контролерот. Овај чекор ги опфаќа следните процесни акции:

- Квантизација

Се заокружува влезната вредност на најблискиот квант во доменот (пр. 4.56 на 5 во случај кога доменот е во облик $u = \{-5, -4, ...0, ..4, 5\}$)

- Нормализација или скалирање

Опсегот на вредности (домен) на управувачка влезната величина се скалира линеарно или нелинеарно(види сл.) на стандардниот опсег(домен [-5,5]) на фази контролерот.

Често користени домени се:

- реални броеви во опсег 1 до -1
- процентуален опсег -100 до -100
- опсег [0,4095] што одговара на 12 битен A/D конвертор
- типот на променливата во програмскиот јазик исто така може да влијае во избор на опсегот

- Филтрирање

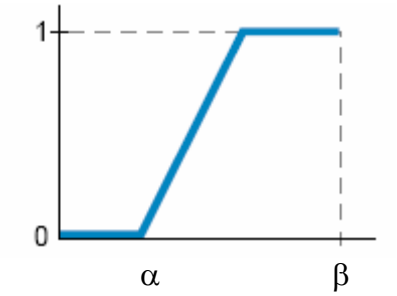
Се врши филтрирање на влезот со цел отстранување на шумови.

- Комбинација на повеќе мерења

Резултатите од повеќе мерења се комбинираат за да се обезбедат клучни индикатори

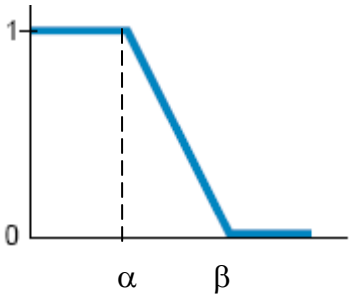
Функции на припадност

Како прв чекор на дизајнот на фази-логички контролиран систем е идентификувањето на состојбите на влезно-излезните променливи и нивно преставување со фази множества наречени функции на припадност (*membership functions*). Најчесто обликот на функциите на припадност е триаголен но, можат да имаат најразличен облик (види сл.)



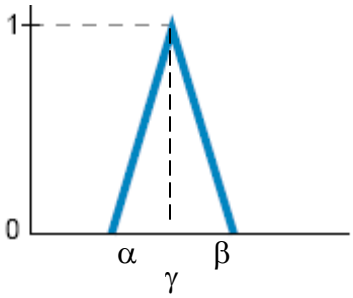
$$\Gamma(u;\alpha,\beta)=\begin{cases}0, u<\alpha\\ (u-\alpha)/(\beta-\alpha), a\le u\le\beta\\ 1, u>\beta\end{cases}$$

Γ - фази множество



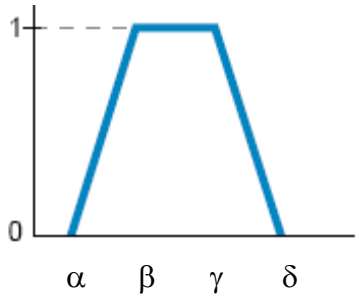
$$L(u;\alpha,\beta)=\begin{cases}1, u<a\\ (u-\alpha)/(\beta-\alpha), a\le u\le\beta\\ 0, u>\beta\end{cases}$$

L - фази множество



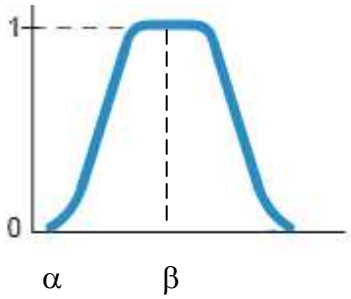
$$\mathrm{A}(u;\alpha,\beta,\gamma)=\begin{cases}0, u\le\alpha\\ (u-\alpha)/(\beta-\alpha), a<u\le\beta\\ (u-\gamma)/(\beta-\gamma), \beta<u\le\gamma\\ 0, u>\gamma\end{cases}$$

Λ - фази множество



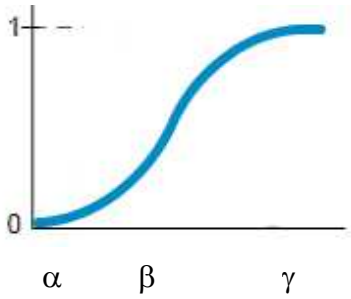
$$\Pi(u;\alpha,\beta)=\begin{Bmatrix}0,u<\alpha\\(u-\alpha)/(\beta-\alpha),a\le u<\beta\\1,\beta\le u<\gamma\\(u-\delta)/\gamma-\delta),\gamma\le u\le\delta\\0,u>\delta\end{Bmatrix}$$

Π - фази множество



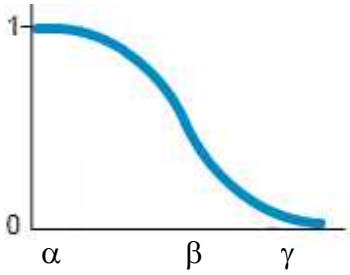
$$\pi(u;\alpha,\beta)=\begin{Bmatrix}0,u\le\alpha\\2((u-\alpha)/(\beta-\alpha))^2,a\le u\le(a+\beta)/2\\1-2((u-3)/(3-\alpha))^2,(\alpha+\beta)/2<u\le(3\beta-\alpha)/2\\2((u-2\beta+\alpha)/(3\beta-2\alpha-2\beta+\alpha))^2,3\beta-\alpha<u\le2\beta-\alpha\\0,u>2\beta-\alpha\end{Bmatrix}$$

π - фази множество



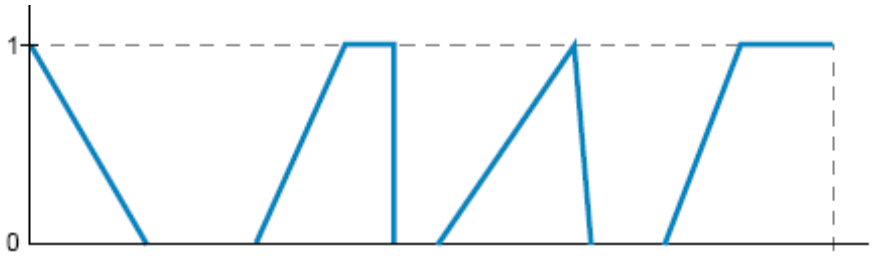
$$S(u;\alpha,\beta,\gamma)=\begin{Bmatrix}0,u<\alpha\\\left(\frac{u-\alpha}{\gamma-\alpha}\right)^2,a<u\le\beta\\1-2\left(\frac{u-\gamma}{\gamma-\alpha}\right)^2,\beta<u\le\gamma\\1,u>\gamma\end{Bmatrix}$$

S- фази множество

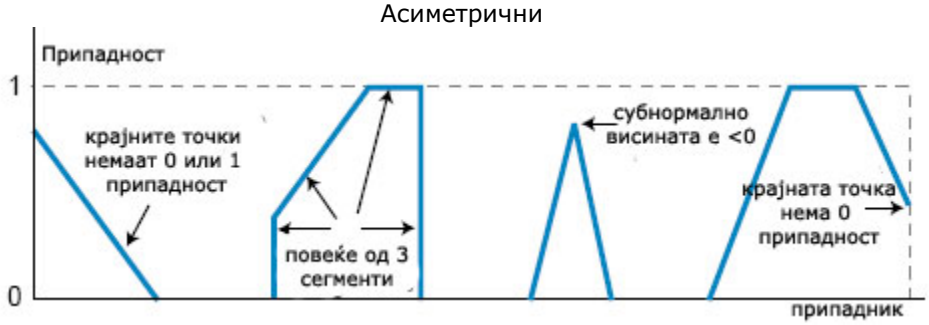


$$Z(u;\alpha,\beta,\gamma)=\begin{Bmatrix}\\\\\end{Bmatrix}$$

Z- фази множество



Асиметрични



Некоректни

и тоа во зависност колку добро ја отсликува припадноста на реалната вредност на влезно-излезните променливи во преставената состојба.

За означување на функциите на припадност се користат лингивистички променливи. Секоја лингвистичка променлива *LX* дефинирана е со четворката

$$(X, \xi X, XD, Mx)$$

каде,

X - *симболичко име (брзина, напон, температура, грешка ...)*

ζX - *лингвистички квантификатор (многу големо, мало,повеќе или помалку,многу многу големо, нула, средно, многу мало...)*

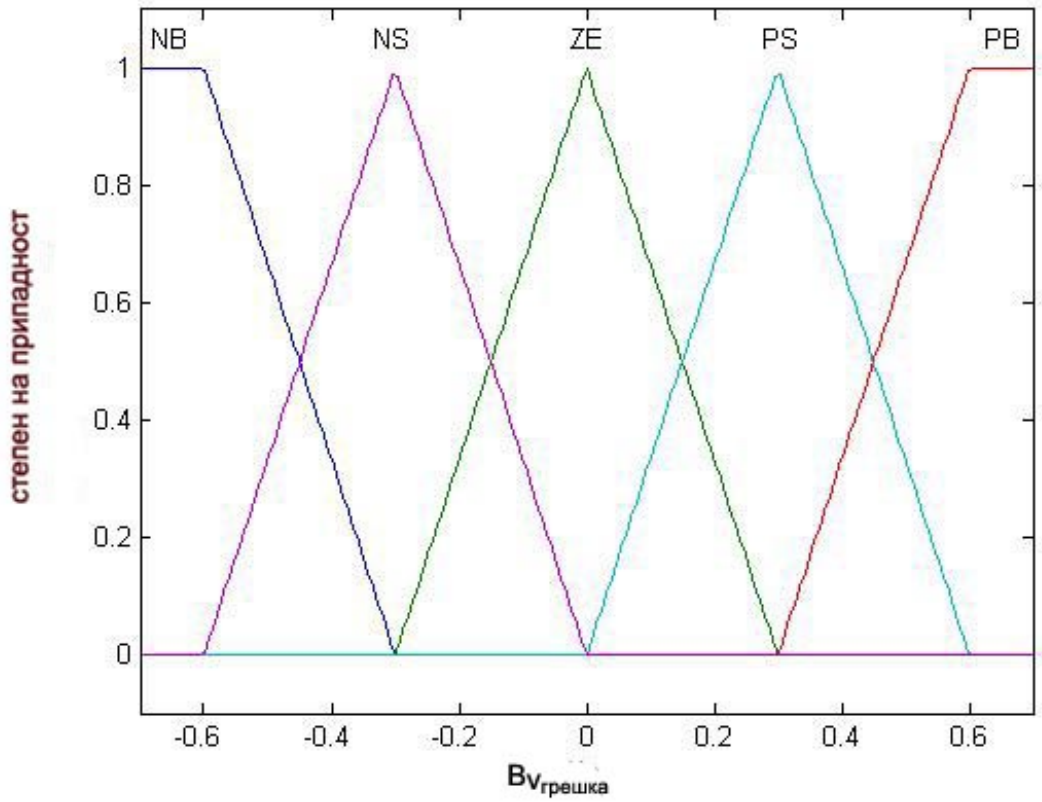
XD - *домен на дискретизирани влезни вредности (пр. [-100,100],[0,4095], [-128,127]...)*

Mx - *семантичка функција* $Mx : LX \to \tilde{LX}$ *(каде* $\tilde{LX}$ *е функција на припадност)*

Најчесто во фази-контролерите се користи стандардизирано множество на лингвистички променливи во однос на нормализиран симетричен домен на позитивни и негативни вредности(позитивно големо/PB-negative big, позитивно мал /PS-negative small, нула/ZE-zero equal, негативно големо/NB-negative big....)

Во егзампларниот систем идентификувани се следните состојби на влезната променлива $v_{грешка}$ – *грешка во брзината*,

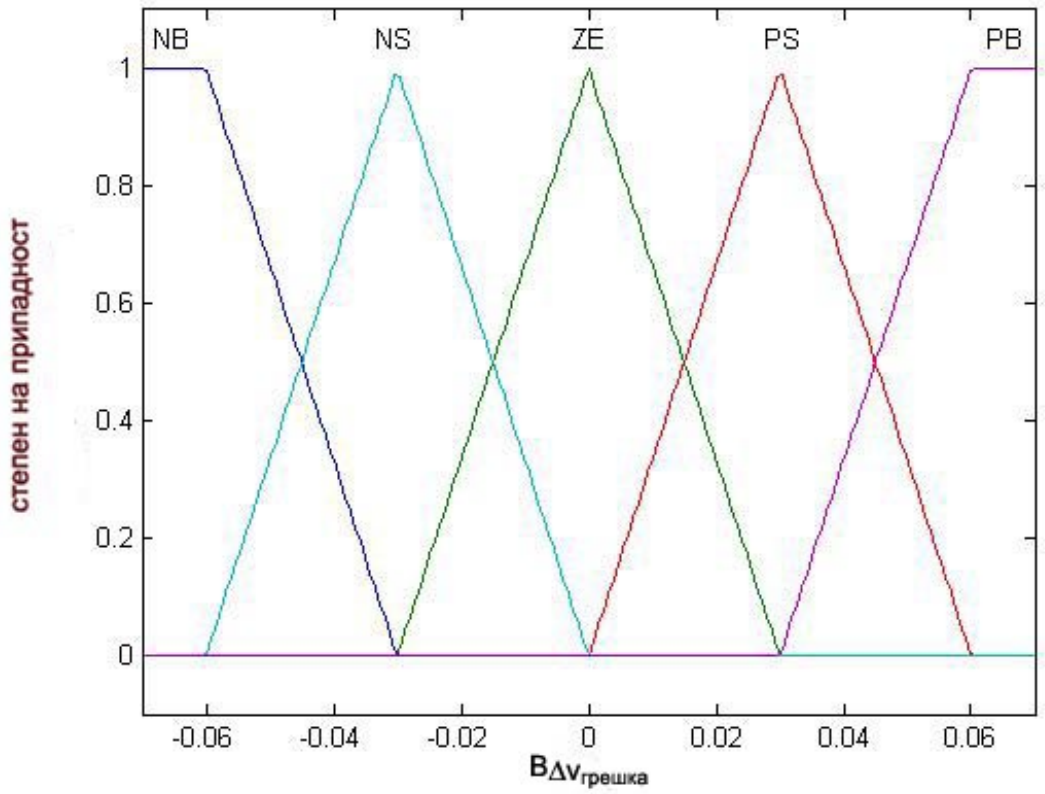
- Негативна голема *грешка* (NB-negative big)
- Негативна мала *грешка* (NS-negative small)
- Занемарлива или нула *грешка* (ZE-zero)
- Позитивна мала *грешка* (PS-positive small)
- Позитивна голема *грешка* (PB-positive big)



и се претставени со триаголни функции на припадност како на сл().

Во егзампларниот систем идентификувани се следните состојби на влезната променлива $\Delta v_{грешка}$ – *промена на грешката*,

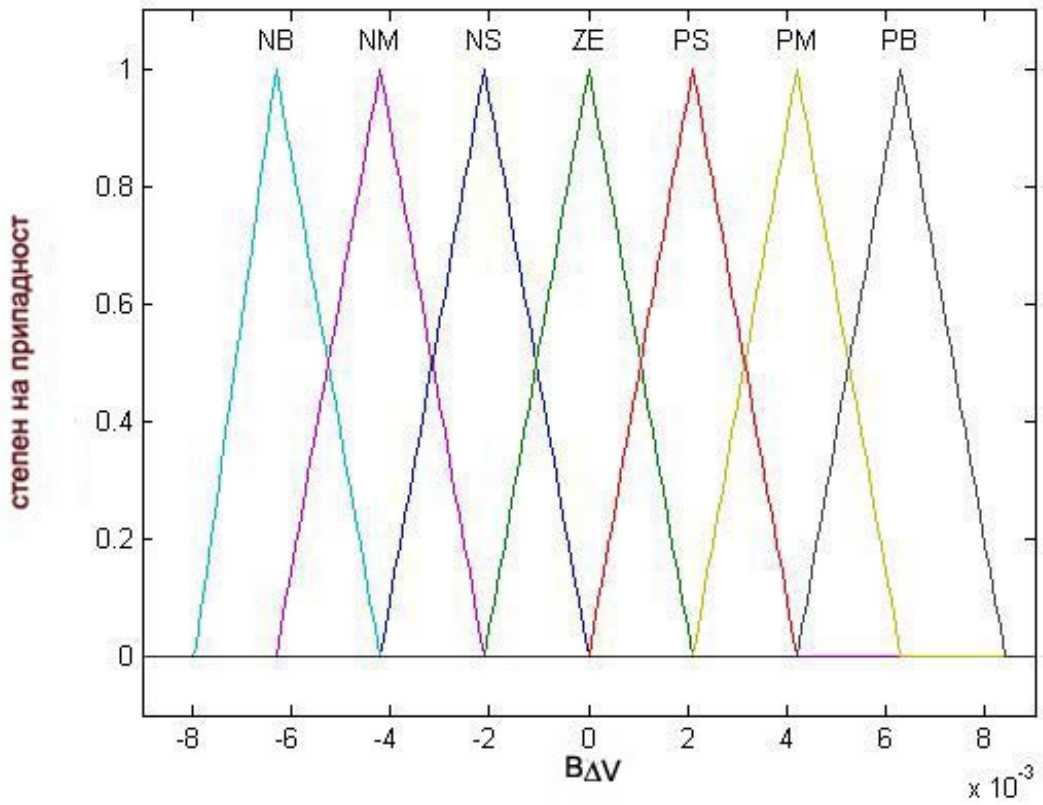
- Негативна голема *промена на грешката* (NB-negative big)
- Негативна мала *промена на грешката* (NS-negative big)
- Занемарлива или нула *промена на грешката* (ZE-zero)
- Позитивна мала *промена на грешката* (PS-positve small)
- Позитивна голема *промена на грешката* (PB-positive big)



и се претставени со триаголни функции на припадност како на сл().

Во егзампларниот систем идентификувани се следните состојби на излезна променлива $\Delta V_{мотор}$ – *промена напонот*,

- Негативна голема *промена напонот* (NB-negative big)
- Негативна средна *промена напонот* (NM-negative middle)
- Негативна мала *промена напонот* (NS-negative small)
- Занемарлива или нула *промена напонот* (ZE-zero equal)
- Позитивна мала *промена напонот* (PS-positive small)
- Позитивна средна *промена напонот* (PM-positive middle)
- Позитивна голема *промена напонот* (PB-positive big)



и се претставени со триаголни функции на припадност како на сл().
При дизајнирањето на функциите на припадност користени се релациите:

$$B_{\Delta v_{брзина}} = \alpha \bullet B_v, B_{\Delta V} = K_I \bullet B_v$$

$$\alpha = K_I / K_P$$

каде,

B_V - параметар за подесување по избор на дизјанерот
K_I – интеграциоен коефициент на PI контолерот
K_P– дифернцијален коефициент на PI контолерот

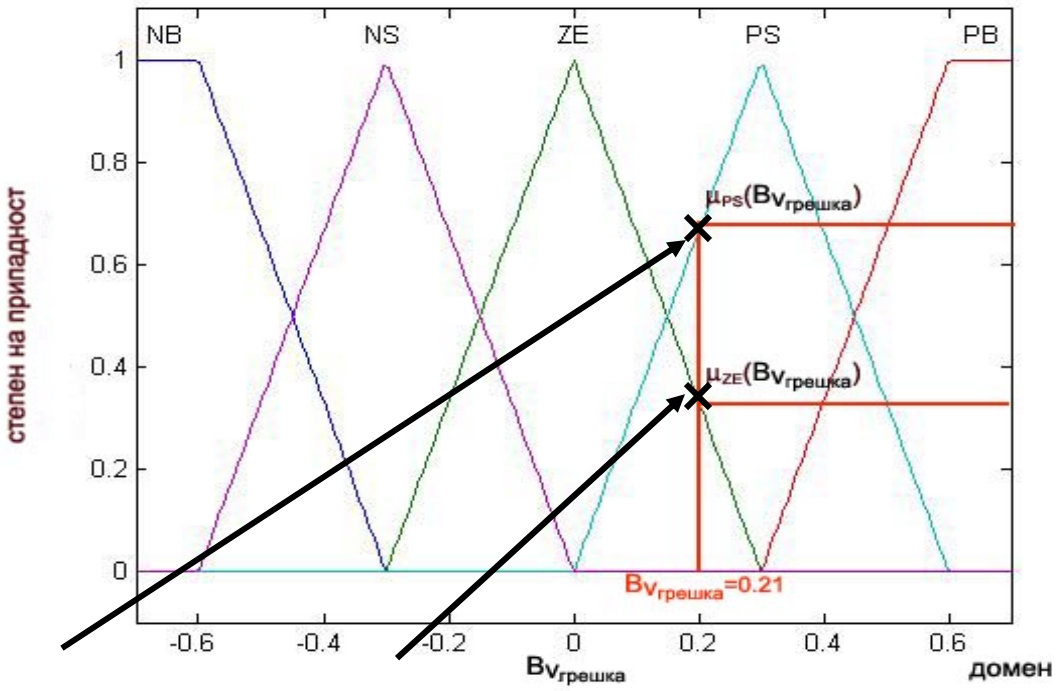
Фазификација

За време на фазификациониот процес фази контролерот прима нормализирана влезна управувачка вредност, уште позната како фази променлива, и ја анализира согласно на корисничките функции на припадност (*membership function*). Потоа и доделува степен на припадност (*degree of membership*) од 0 до 1 во зависност колку таа вредност се вклопува(припаѓа) во секоја од функциите на припадност:

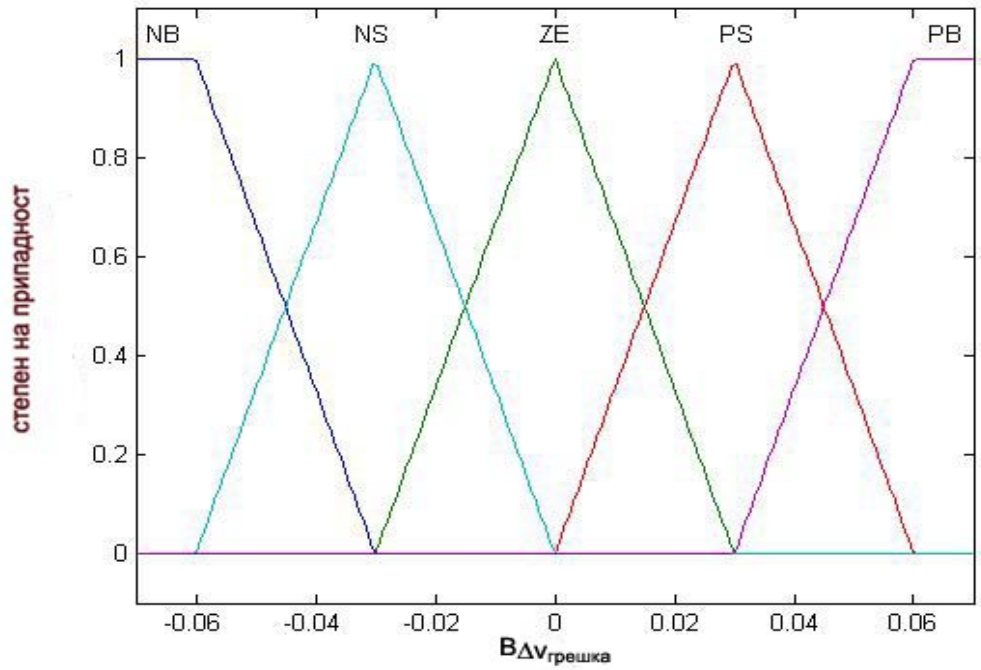
$$\mu_F(u) \in [0,1]$$

каде
μ_F – *степен на припадност на нормализирана влезна управувачка променлива "u"*
F – *функција на припадост (пример NB,NS,ZE..)*
u – *нормализирана влезна управувачка променлива*

За експериментални вредност на нормализираната управувачка фази променлива $v_{грешка}$, во процесот на фазификација, се доби припадност во 2 фази множества (лингвистичка означени како ZE и PS) со одреден степен на припадност како што се гледа на сликата.



За експериментални вредност на нормализираната управувачка фази променлива $B_{\Delta}v_{грешка}$, во процесот на фазификација, се доби припадност во 2 фази множества (лингвистичка означени како ZE и PS) со одреден степен на припадност како што се гледа на сликата.


NB
NS
ZE
PS
PB
1
0.8
0.6
0.4
0.2
0
степен на припадност
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
BΔvгрешка

База на знаење

Правила на заклучување врз кој се изведува заклучок ја сочинуваат базата на знаења на фази логичкиот контролер. Тие се математичките реченици составени од фази-пропозиции кои се поврзани со сврзниците **и**(and), **или**(or), **ако-тогаш** if-then (или **импликација**(-implies) најчесто од Мамдани тип), **ако и само ако**(if and only if) и често се во следниот облик:

АКО <фази-пропозиција1> **сврзник** <фази-пропозиција2>

сврзник ... **сврзник** <фази-пропозиција2> ТОГАШ <фази-пропозиција>

АКО/IF <услов/antescedent> ТОГАШ/THEN <последица/consequent>

АКО $v_{брзина}$ е *многу голема* **И** $A_{оптеретување}$ е *нула* **ТОГАШ** V_{motor} е *многу голем*

Дизајнирани врз база на:

- Искуство на експерт или инжинерско знаење на контролата

Правила добиени со испитување на експертот или операторот со внимателно организиран прашалник или со набљудување и запишување на контролнит акции на операторот.

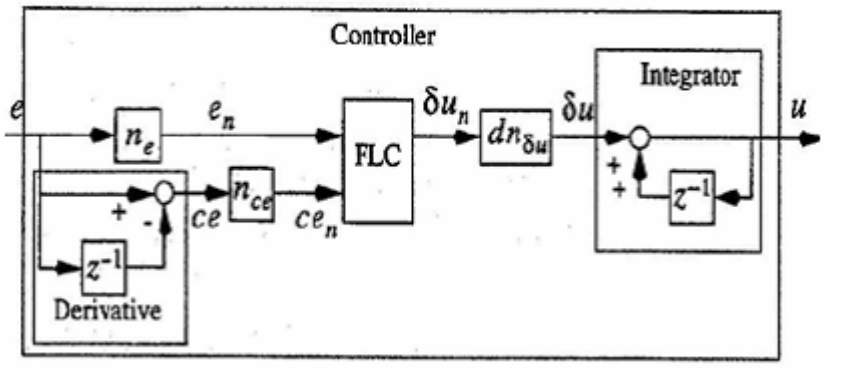
- Базиран на фази моделот на процесот

Базата на знаење може да се гледа како инверзен модел на процесот па правилата можата да се добијат со инверзија на фази моделот на процесот. Други методи се фази идентификација (*fuzzy identification*) или методи базирани на искуства од класичните контролери.

- Базирано на учење

Правилата ги пронаоѓа самиот конторлер на база на вградената вештачка интелигенција и неворнски мрежи

Фази логичкиот контролер од PID тип (сл.),



базиран на размислувањата на конвенционален **PID** контролер со дискретен облик:

$$\Delta u = K_P \cdot \Delta e + K_I \cdot e + K_D \cdot \partial e$$

има лингвистичка репрезентација на Ако-Тогаш(If-Then) правилото од облик:

АКО *e* <лингивистичка променлива> **И** *Δe* е <лингивистичка променлива>

И ∂e е <лингивистичка променлива> **ТОГАШ** *u* е <лингивистичка променлива>
каде,
e – грешка
Δe – промена на грешката ($\Delta e = e(k) - e(k-1)$)
∂e *– сума на грешките* ($\partial_e = \sum_{i=0}^{k-1} e(i)$)

процесни променливи на условот додека,

u – управувачки излез ($u = y_{sp} - y(k)$)
Δu – промена на управувачкиот излез ($\Delta u = u(k) - u(k-1)$)

се променливи на управувачкиот излез на последицата во моментите на одбирање *0,1,2...к* .

<лингивистичка променлива> - зема вредност од множеството на лингвистички вредности

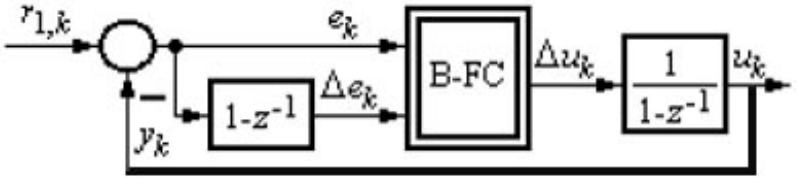
Фази логичкиот контролер на PD тип базиран на размислувањата на конвенционален **PD** контролер со дискретен облик:

$$u = K_P \cdot e + K_D \cdot \Delta e$$

има лингвистичка репрезентација на Ако-Тогаш(If-Then) правилото од облик:

IF *e* <лингивистичка променлива> AND *Δe* е <лингивистичка променлива> THEN *u* е <лингивистичка променлива>

Фази логичкиот контролер на PI тип (сл.),



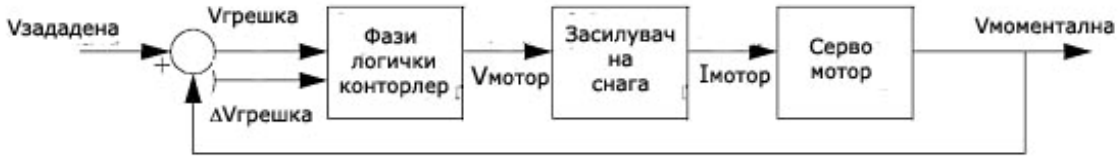
базиран на размислувањата на конвенционален **PI** контролер со дискретен облик:

$$\Delta u = K_P \cdot \Delta e + K_I \cdot e$$

има лингвистичка репрезентација на Ако-Тогаш(If-Then) правилото од облик:

АКО *е* <лингивистичка променлива> **И** *Δе* е <лингивистичка променлива>
ТОГАШ *u* е <лингивистичка променлива>

Блок шемата на егзампларниот фази логички контролер од PI тип е прикажана на сликата:



а со оглед на к-ката на PI класичен контролер со параметри од конкретниот случајот во следниот облик,

$$V_{motor} = K_P \cdot v_{грешка} + K_I \int v_{грешка} dt$$

од каде произлегува дискретниот облик,

$$\Delta V_{мотор} = K_P \cdot \Delta v_{грешка} + K_I \cdot v_{грешка} =$$

$$= K_P \cdot (\Delta v_{грешка} + \alpha \bullet v_{грешка})$$

каде:

$v_{грешка}$ - грешка (разлика меѓу зададената и моменталната брзина)

$\Delta v_{грешка}$ - промена на грешката

V_{motor} - напон на моторот

K_I *– интеграционен коефициент на PI контолерот во случај на Tustin-ова дискретизација:*

$$K_I = k_C \, T_s / T_c$$

K_P *– диференцијален коефициент на PI контолерот во случај на Tustin-ова дискретизација:*

$$K_P = k_C \, [1 - T_s / (2T_c)]$$

α *– односен коефициент:*

$$\alpha = K_I / K_P = 2T_s /(2T_c - T_s)$$

T_s – *период на дискретизација*

k_p – *процесно засилување (во системот* $k_p = 4900$ *)*

T_Σ – *сума на паразитните константи (во системот* $T_\Sigma = 0,035s$ *)*

T_c – *интеграциона временска константа*

$$T_c = \beta T_\Sigma$$

k_c – *интеграционо засилување*

$$k_c = 1/(\beta\sqrt{\beta}T_\Sigma^2 k_P)$$

β – *подесувачки параметар во опсег 1 < β < 20* согласно со „Проширениот симетрички метод" за постигнување оптимум *(Extended Symmetrical Optimum method*) во однос на *брз одзив и кратко време на смирување. (во системот избрано е* β=6*)*

Позитивна грешка

$$v_{грешка(k)} = v_{зададена} - v_{(k)} < 0 (_{пример}. v_{грешка(k)} = NB)$$

значи дека излезот има поголема вредност од зададениот и обратно за негативна грешка,
и позитивна промена на грешката

$$\Delta v_{грешка} = v_{грешка(k)} - v_{грешка(k-1)} =$$

$$= v_{зададена} - v_{(k)} - v_{зададена} + v_{(k-1)} =$$

$$= [v_{(k-1)} - v_{(k)}] > 0 (_{пример}. \Delta v_{грешка(k)} = PS)$$

значи тековниот излез има повисока вредност од вредноста во претходниот семплирачки момент и обратно за негативна промена.

Земајќи во предвид горенаведеното како и дискретниот облик на PI контролер имаме пример

$$\Delta V_{motor} = NB + PS = NM$$

и правилата на фази логичкиот контролер се со следниот облик:

АКО $v_{брзина}$ е *негативно голема* (NB) **И** $\Delta v_{грешка}$ е *позитивно голема* (PB) **ТОГАШ**

ΔV_{motor} е *нула*(ZE)

АКО $v_{брзина}$ е *негативно голема* (NB) **И** $\Delta v_{грешка}$ е *позитивно мала* (PS) **ТОГАШ** ΔV_{motor} е *негативно мал* (NS)

АКО $v_{брзина}$ е *негативно голема* (NB) **И** $\Delta v_{грешка}$ е *нула* (ZE) **ТОГАШ** ΔV_{motor} е *негативно среден* (NM)

….

За преставување на фази логичките правила, а со оглед на реалниот начинот на имплементација, прегледност, измени и одржување на правилата, се користи табеларен приказ (*rule matrix*):

Δe_k \ e_k	NB	NS	ZE	PS	PB
PB	ZE	PS	PM	PB	PB
PS	NS	ZE	PS	PM	PB
ZE	NM	NS	ZE	PS	PM
NS	NB	NM	NS	ZE	PS
NB	NB	NB	NM	NS	ZE

Матрицата во егзампларниот систем го има следниот облик:

$\Delta v_{грешка}$ \ $v_{грешка}$	(-) голема NB	(-) мала NS	(нула) ZO	(+) мала PS	(+) голема PB
(+) голема PB	V_{motor} нула	V_{motor} (+) голем	V_{motor} (+)среден	V_{motor} (+) голем	V_{motor} (+) голем
(+) мала PS	V_{motor} (-) мал	V_{motor} нула	V_{motor} (+) мал	V_{motor} среден	V_{motor} (+) голем
(нула) ZO	V_{motor} (-) среден	V_{motor} (-) мал	V_{motor} нула	V_{motor} (+) мал	V_{motor} (+)среден
(-) мала NS	V_{motor} (-) голем	V_{motor} (-) среден	V_{motor} (-) мал	V_{motor} нула	V_{motor} (+) мал
(-) голема NB	V_{motor} (-) голем	V_{motor} (-) голем	V_{motor} (-)среден	V_{motor} (-)мал	V_{motor} нула

Во случај на n - влезни променливи (E1,E2,…) би требало n – димензинални табели но наместо тоа се користат релациони бинарни табели како на примерот сл. со три влезни променливи.

E_1	E_2	E_3	U
-100	-100	-100	-100
-100	-100	-67	-89
-100	-100	0	-67
-100	-100	67	-178
-100	-100	100	-33
-100	-67	-100	-89
-100	-67	-67	-189
-100	-67	0	-56
$\dots$	$\dots$	$\dots$	$\dots$
100	100	100	100

Фази логичкиот контролер на база на правила од Такаги-Сугено тип има лингвистичка репрезентација опишана со терми на Ако-Тогаш(If-Then) правилото:

АКО *f*(e_1 е <лингивистичка променлива>**…** e_n <лингивистичка променлива>)
ТОГАШ *u* е *g*(e_1,… e_n)

каде:

f – логичка (И,ИЛИ..) функција
g - функција од (0,1,..) ред

АКО $v_{брзина}$ е *негативно голема*(NB) **И** $\Delta v_{грешка}$ е *позитивно голема* (PB) **ТОГАШ**

V_{motor} е *g*($v_{брзина}$, $\Delta v_{грешка}$)

каде:

g($v_{брзина}$, $\Delta v_{грешка}$) е функција од 0-ред т.е *g*($v_{брзина}$, $\Delta v_{грешка}$)
c - константа

АКО $v_{брзина}$ е *негативно голема*(NB) **И** $\Delta v_{грешка}$ е *позитивно голема* (PB) **ТОГАШ**

V_{motor} е *g*($v_{брзина}$, $\Delta v_{грешка}$)

каде:

g($v_{брзина}$, $\Delta v_{грешка}$) е функција од 1-ред т.е *g*($v_{брзина}$, $\Delta v_{грешка}$)= $a*e+b(\Delta e)+c$
a,b,c - константи

Механизам за изведување на заклучок

Улогата на механизмот за изведување заклучок (*inferance engine*) процесирање на АКО-ТОГАШ правилата т.е пронаоѓање на оние правила чишто услов е исполнет за зададениот влез (АКО делот на правилото) и евалуира излез на база на излезите (ТОГАШ деловите на правилата) на тие правила.

Најкористен начин за изведување на заклучок е темелено врз композиција *compositional rule of inference (CROI).*

Процесот на изведување на заклучок се изведува во 3 фази процеси:

- Примена на фази операции
- Импликација
- Агрегација

Во правата фаза за оние АКО-ТОГАШ правила,

АКО x_1 е $LX_1 \otimes x_2$ е $LX_2 \otimes ... \otimes x_2$ е LX_n

за кој е исполнет условот(*antecedent*) се изведуваат фази операциите над истиот и се добива степенот на припадност како:

$$\mu_{ant}{}^{(\kappa)} = (\mu_{LX_1}(x_1) \oplus \mu_{LX_2}(x_2) \otimes ... \mu_{LXn}(x_n))$$

каде:

к – *то правило*
L – *лингвистичка променлива (позитивно голема, мала, средна, негативно голема,...)*
⊗*- фази операција (**И**(AND),**ИЛИ**(OR) односно **min, max**)*

$$\mu_{LX_1}(x_1) \otimes \mu_{LX_2}(x_2) = \min(\mu_{LX_1}(x_1), \mu_{LX_2}(x_2)), (\otimes = and)$$

$$\mu_{LX_1}(x_1) \otimes \mu_{LX_2}(x_2) = \max(\mu_{LX_1}(x_1), \mu_{LX_2}(x_2)), (\otimes = or)$$

или

$$\mu_{LX_1}(x_1) \otimes \mu_{LX_2}(x_2) = \mu_{LX_1}(x_1) \bullet \mu_{LX_2}(x_2), (\otimes = and)$$

$$\mu_{LX_1}(x_1) \otimes \mu_{LX_2}(x_2) = \mu_{LX_1}(x_1) + \mu_{LX_2}(x_2) - \mu_{LX_1}(x_1) \bullet \mu_{LX_2}(x_2), (\otimes = or)$$

x1,x2...xn – управувачки влезови

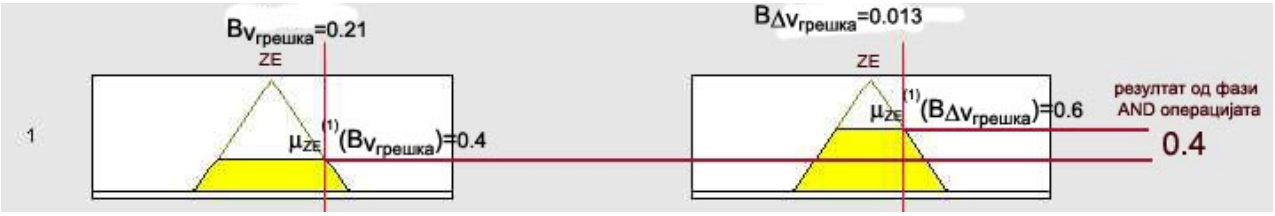
Делот на условот(*antecedent*) на првото исполнето АКО-ТОГАШ правило е следното:

АКО $v_{грешка}$ е *нула* (ZE) **И** $\Delta v_{грешка}$ е *позитивно голема* (ZE)

Со изведување на **И**(AND) фази операцијата,

$$\mu_{ant}{}^{(1)} = (\mu_{ZE}(v_{грешка}) and\, \mu_{ZE}(\Delta v_{грешка})) = \min(\mu_{ZE}(v_{грешка}), \mu_{ZE}(\Delta v_{грешка}))$$

се добива степенот на припадност на условот μ_{ant} што се гледа и од сликата.



Импликација

Влезот на импликацискиот процес е единствен број даден од условот, а излез е фази множество. Но, пред да се примени импликацискот метод треба да се земат во обзир тежините, во опсет од 0 до 1, доделени на равилата. Генерално вредноста на тежините е 1, па не е потребна примена над резултатот од условот.

Најчесто АКО-ТОГАШ правилата ја користат Мамдани импликацијата:

$$\mu_{ant}{}^{(k)} \Rightarrow \mu_{cns}{}^{(k)} = \mu_{ant}{}^{(k)} \wedge \mu_{cns}{}^{(k)}$$

па следи

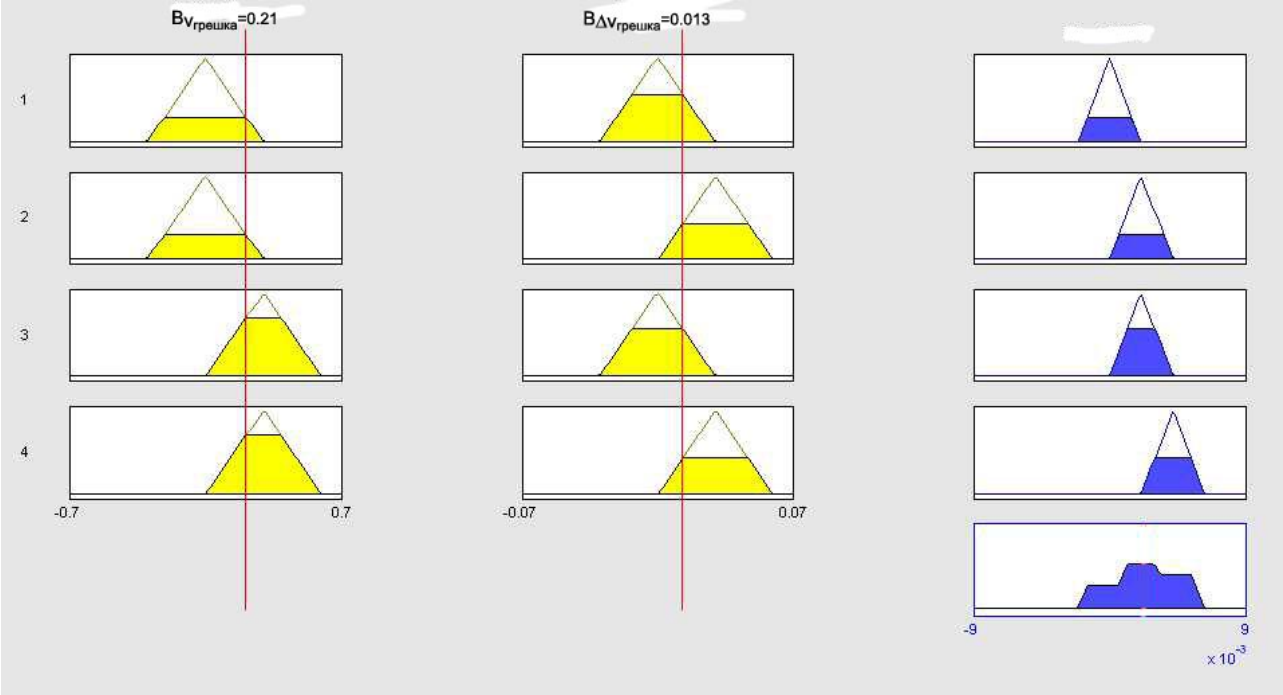
$$\mu_{CLU}{}^{(k)} = \min(\mu_{ant}{}^{(k)}, \mu_{cns}{}^{(k)})$$

Резултатот од импликацијата во нашиот систем е прикажан на сл. :

Агрегација

Некогаш агрегацијата се нарекува исполнување(*fulfillment*) или сила на испукувањето(*firing strength*). Тоа е процес во кој листата на потсечени фази множества добиени при импликацијата се комбинира во единствено фази множество.

Резултатот од агрегацијата во нашиот систем е прикажан на сл. :



Дефазификација

Потсечени фази множества од процесот на агрегација мора да се конвертираат во единствена вредност која се праќа до појачувачот како контролен сигнал. Овој процес се вика дефазификација и постојат десетина методи на изведба:

- центар на подрачје (CoA)
- центар на суми (CoS)
- среден максимум (mean of maxima MoM)
- прв максимум (FoM)
- највисок максимум, максимум најлево или максимум најдесно (HM)(LM)(RM)
- центар на најголемо подрачје (CLA)

Центар на подрачје (CoA)

Методата центар на подрачје или центроида е најпозната и најкористена метода. Во дискретен случај остра вредност на управувачкиот сигнал се добива со релацијата:

$$u^* = \frac{\sum_{i=1}^{k} u_i \bullet \max \mu_{CLU}{}^{(k)}(u_i)}{\sum_{i=1}^{k} \mu_{CLU}{}^{(k)}(u_i)}$$

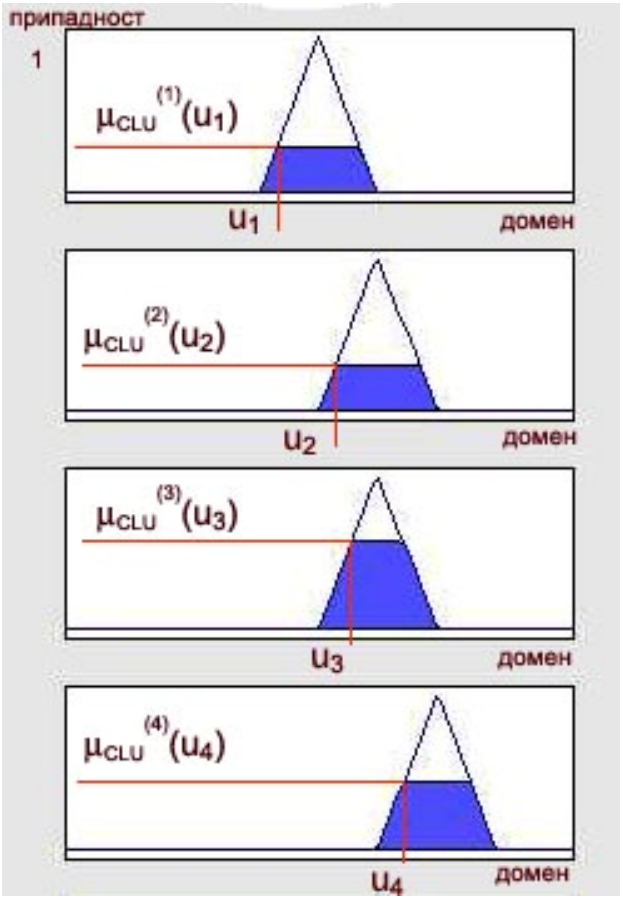
каде:

k - *исполнети правила на базата на знаење*

$\mu_{CLU}{}^{(k)}(u_i)$ - *степен на припадност на потсеченото фази множество добиено со импликација на к-тото правило*

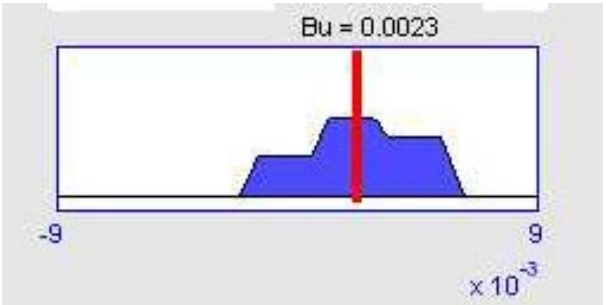
u_i - вредноста од доменот при $\mu_{CLU}{}^{(k)}(u_i)$

Центроидалната метода на дефазификација е применета на потсечените фази множества на 4-те исполнети правила



за пронаоѓање на единствена вредност на управувачкиот сигнал :

$$Bu^* = \frac{u_1 \bullet \mu_{CLU}^{(1)}(u_1) + u_2 \bullet \mu_{CLU}^{(2)}(u_2) + u_3 \bullet \mu_{CLU}^{(3)}(u_3) + u_4 \bullet \mu_{CLU}^{(4)}(u_4)}{\mu_{CLU}^{(1)}(u_1) + \mu_{CLU}^{(2)}(u_2) + \mu_{CLU}^{(3)}(u_3) + \mu_{CLU}^{(4)}(u_4)} = 0.0023$$



Постпроцесирање

По завршувањето на дефазификациониот процес добиената величина се денормализира и се филтрира и како таква се испраќа на управуваниот процес. Може слободно да се каже дека постпроцесирањето е обратен процес од предпроцесирањето.