BAB II

LANDASAN TEORI

2.1 Pemrograman Borland Delphi versi 6.0

2.1.1 Sejarah bahasa pemrograman Delphi

Beberapa tahun lalu seorang *programmer* harus benar - benar memeras otak untuk bisa memebuat program berbasis *Windows*. Kesulitan ini mengilhami **Charlie Calvert** dan **Zack Urlocker**, dua *programmer* ***Borland International***, untuk membuat pemrograman *Windows* menjadi lebih mudah.

Berawal dari bahasa pemrograman ***Pascal*** muncullah ide untuk mengembangkan bahasa pemrograman ini yang nantinya lebih dikenal dengan sebutan **"DELPHI"**, mampu memberikan tampilan visual dari pemrograman Pascal. Pada saat bahasa pemrograman *Windows* dengan Pascal dirasa cukup sulit maka **Borland International Incoporation** merilis **Delphi 1** untuk memenuhi kebutuhan tersebut pada tahun 1995.

Di tahun 1996 **Borland International Incoporation** lewat bahasa pemrograman **Delphi 2**_nya telah mampu dijalankan pada *Windows* 95 dan Windows NT. **Borland International Incoporation** berganti nama menjadi **Inprise Corporation** dan merintis perkembangan bahasa pemrograman ini.

2.1.2 IDE Delphi

Secara umum tampilan bidang kerja dalam lingkungan Delphi atau yang lebih lazim dikenal dengan IDE mempunyai 6 bagian utama yaitu :

1. ***Menu Bar***

   Menyediakan kelompok perintah – perintah yang berhubungan dengan IDE Delphi.

2. ***ToolBar***

   Berisi kumpulan tombol sebagai pengganti beberapa item menu yang sering digunakan, bertujuan untuk meringkas atau mempercepat pekerjaan.

3. ***Component Pallete***

   Berisi kumpulan icon yang melambangkan komponen – komponen pada VCL .

4. ***Object Inspector***

   Adalah sarana yang digunakan untuk mengubah karakteristik sebuah komponen.

5. ***Window Form***

   Tempat untuk merancang jendela pada aplikasi *windows*.

6. ***Code Editor***

   Tempat untuk menuliskan program dalam *Object Pascal*.

2.1.3 Program dan proyek

A. Susunan program

Dalam pemrograman Delphi sepenuhnya menerapkan metode pemrograman *Object Oriented Programming* dimana segala sesuatu yang dimanipulasi dalam Prgram adalah objek.

Delphi memilah program menjadi dua bagian utama yaitu bagian *primer* dan *sekunder*. Bagian *primer* berupa *file* program yang mengkoordinasi keseluruhan program, *file* ini disebut *file* proyek. Secara umum bentuk urutan program *primer* Delphi sebgai berikut :

```
Program nama program
Uses { kumpulan unit – unit }
Const { kumpulan konstanta }
Type { kumpulan type }
Var { kumpulan variable }
Begin
{ Statement – statement }
End
```

Sedangkan bagian *sekunder* disebut sebagai unit, secara umum sebuah unit terdiri atas

```
Unit { nama unit }
Interface
Uses { daftar unit }
Implementation
{ implementasi procedure dan function }
End.
```

B. Susunan proyek

Sebuah proyek adalah suatu kumpulan *file* yang bersama – sama membentuk sebuah aplikasi.

- *File Source* kode proyek (*.dpr)

  File *.dpr adalah *file* program utama yang digunakan Delphi untuk mengkompilasi semua unit – unit. Contoh kode proyek sebagai berikut :

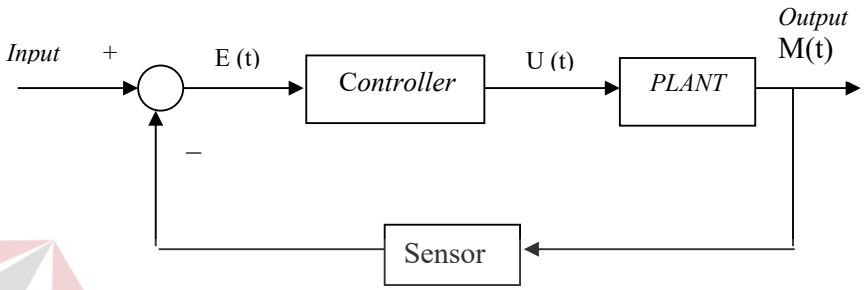
```
program Project1;
uses
Forms,
Unit1 in 'Unit1.pas' {Form1};
{$R *.res}
begin
Application.Initialize;
Application.CreateForm(Tform1, Form1);
Application.Run;
end.
```

- *File source* kode *.pas

  Ekstensi *.pas menunjukkan bahwa *file* tersebut adalah *file* unit, dimana unit adalah bagian proyek yang berfungsi sebagai bahan utama pembangunan aplikasi yang berisi *source* kode Pascal yang merupakan elemen aplikasi.

2.2. *Controller* PID

Controller jenis ini merupakan kombinasi dari tiga jenis *controller* yaitu *Proportional, Integral dan Derivative.*



Gambar 2.1. Blok Diagram *Controller* PID

Controller PID merupakan kombinasi dari tiga jenis *controller.* Jika ketiga jenis *controller* tersebut berdiri sendiri, maka hasil yang dicapai kurang bagus . Sebab masing – masing memilki kelebihan dan kelemahan sendiri – sendiri. Dikombinasikannya ketiga jenis *controller* tersebut menjadi satu sistem kontrol tunggal yang diharapkan memberikan kontribusi dari kelebihannya masing – masing.

Controller P adalah suatu penguat linier yang dapat diatur penguatannya. *Controller* PI merupakan perubahan dari *output* kontrol *integral*, berubah dengan fungsi waktu yang sebanding dengan sinyal kesalahan. Dimana sinyal kesalahan adalah sama dengan *Error* E(t) yang merupakan selisih antara *output* dan *setpoint*. *Controller* D sering disebut disebut kontrol laju, karena besar *output controller* sebanding dengan laju perubahan sinyal kesalahan. Tetapan waktu turunan adalah

selang waktu bertambah majunya respon *controller* P yang disebabkan oleh aksi laju.

Gabungan *controller proportional, controller integral* dan *controller derivative* memiliki keunggulan dalam memperbaiki kesalahan sinyal dibandingkan dengan jika ketiga *controller* tersebut berdiri sendiri. Adapun persamaan PID adalah sebagai berikut :

$$M(t) = K_p\, e(t) + K_p\, T_d \frac{\mathrm{de(t)}}{\mathrm{dt}} + \frac{K_p}{T_i} \int_0^t e(t)\, dt \qquad (2.1)$$

atau fungsi alihnya :

$$\frac{M(s)}{E(s)} = K_p \left(1 + T_d s + \frac{1}{T_i s} \right) \qquad (2.2)$$

dimana :

K_p = *proportional gain*

T_d = waktu turunan

T_i = waktu integral

K_p akan memberikan efek mengurangi waktu naik, tetapi tidak menghapus kesalahan keadaan tunak. K_i akan memberi efek menghapus kesalahan keadaan tunak, tetapi berakibat memburuknya respon transient. K_d akan memberikan efek meningkatnya stabilitas sistem, mengurangi overshoot, dan menaikkan respon transfer. Efek dari setiap *controller* (K_p, K_i, K_d) dalam sistem *loop* tertutup diperlihatkan pada tabel 2.1 berikut .

Tabel 2.1. Efek masing – masing *controller*

Respon Loop Tertutup	**Waktu Naik**	***Overshoot***	**Waktu Turun**	**Kesalahan Keadaan Tunak**
Kp	Menurun	Meningkat	Perubahan kecil	Menurun
Ki	Menurun	Meningkat	Meningkat	Hilang
Kd	Perubahan kecil	Menurun	Menurun	Perubahan kecil

Dari tabel 2.1, terlihat bahwa hubungan korelasi tersebut mungkin tidak sepenuhnya akurat, karena K_p, K_i dan K_d saling bebas. Pada kenyataannya, mengubah salah satu variabel dapat mengubah dua yang lainnya. Karena alasan tersebut, tabel hanya digunakan sebagai referensi saat kita menentukan nilai untuk K_p, K_i dan K_d.

2.2.1 *Controller* P

Untuk *controller* P , hubungan antara *output* M(t) dan sinyal *error* E(t) adalah :

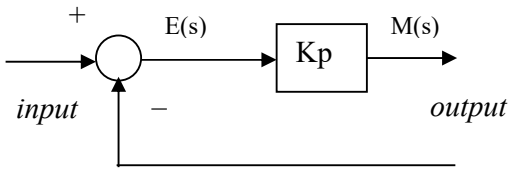
$$M(t) = K_p E(t) \qquad (2.3)$$

dalam besaran trasformasi Laplace

$$\frac{M(s)}{E(s)} = K_p \qquad (2.4)$$

Dimana :

Kp = *proportional gain*



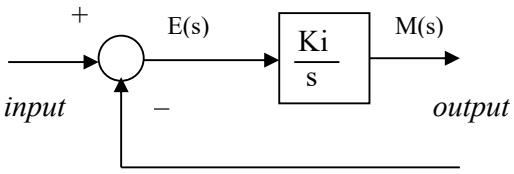
Gambar 2.2. Diagram blok *controller* P

Berdasarkan tabel 2.1 kita ketahui bahwa Kp mengurangi waktu naik (*rise time*) dan meningkatkan *overshoot* dan mengurangi kesalahan keadaan tunak .

Apapun wujud mekanisme yang sebenarnya dan apapun jenis daya penggeraknya, *controller* P pada dasarnya merupakan penguat dengan penguatan yang dapat diatur.

2.2.2 *Controller* I

Pada aksi kontrol ini akan terjadi pengubahan harga *output* M(t) menjadi laju yang sebanding dengan sinyal *error* E(t).



Gambar 2.3. Diagram blok *controller* I

$$\frac{dm(t)}{dt} = K_i\, E(t) \qquad (2.5)$$

atau

$$M(t) = K_i \int_0^t E(t)\, dt \qquad (2.6)$$

Dimana :

Ki = *integral gain*

Jika harga E(t) diduakalikan, maka harga M(t) berubah dengan laju perubahan menjadi dua kali semula. Jika kesalahan penggerak atau *error* sama dengan nol, maka harga M(t) tetap stationer.

2.2.3 *Controller* D

Controller D adalah jenis sinyal yang menghasilkan sinyal *output* berbanding langsung dengan sinyal *error*. *Controller* ini dikenal sebagai rate kontrol atau *anticipatory* kontrol. Kontrol jenis ini tidak dapat digunakan tersendiri karena ketika perubahan sinyal *error* sama dengan nol, maka *controller* menghasilkan sinyal kontrol sebesar nol. Jadi *controller derivative* efektif selama periode *transient*. Hubungan sinyal kontrol dan sinyal *error* sebagai berikut :

$$M(t) = K_p\, T_d \frac{dE(t)}{dt} \qquad (2.7)$$

2.3 Himpunan *Fuzzy*

Suatu himpunan *fuzzy (fuzzy set)* A dalam semesta pembicaraan *(universe of discourse)* U dinyatakan dengan fungsi keanggotaan *(membership function)* μ_A, yang harganya berada dalam interval [0,1]. Secara matematika hal ini dinyatakan dengan :

$\mu_A : U \rightarrow [0,1]$ (2.8)

Himpunan *fuzzy* A dalam semesta pembicaraan U biasa dinyatakan sebagai sekumpulan pasangan elemen u(u anggota U) dan besar derajat keanggotaan *(grade of membership)* elemen tersebut, $\mu_{A,}$ sebagai berikut:

$A \{ (u, \mu_A(u) / u \in U) \}$ (2.9)

*T*anda '/' digunakan untuk menghubungkan sebuah elemen dengan derajat keanggotaannya. Jika *U* adalah diskrit, maka A biasanya dinyatakan dengan :

$A \ \mu_A (u_1) / u_{1+ \ldots +} \ \mu_A (u_u) / u_n$ (2.10)

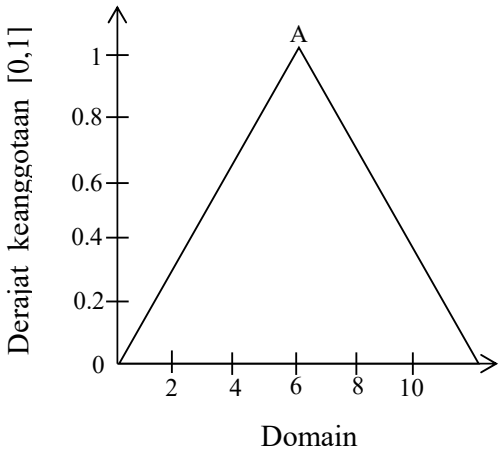
atau

$$A = \sum_{n=1}^{n} \mu_A (\mu_1) / \mu_1 \quad (2.11)$$

Dan jika U adalah diskrit , maka A biasanya dinyatakan dengan : A $\int \mu_A$ *(u) u.* Sebagai contoh, untuk semesta pembicaraan 'bilangan cacah yang kurang dari 10' dan himpunan *fuzzy* A yang didefinisikan sebagai bilangan yang dekat dengan 5', dinyatakan :

A = 0/0 + 0.2/1 + 0.4/2 + 0.6/3 + 0.8/4 + 1.0/5 + 0.8/6 + 0.6/7 + 0.4/8 + 0.2/9

Proses untuk mendapatkan besarnya derajat keanggotaan *input*yang berupa suatu variabel numerik non-*fuzzy* (elemen himpunan) dalam suatu himpunan *fuzzy* disebut *fuzzy*fikasi *(fuzzycation)*



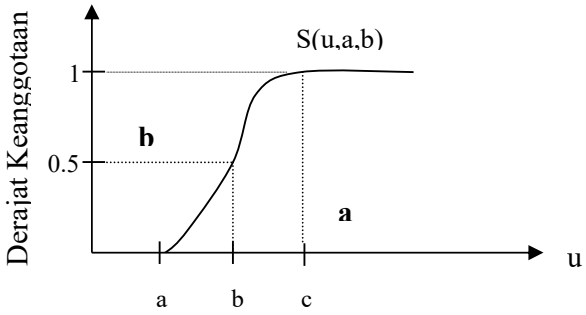
Gambar 2.4. Definisi himpunan *fuzzy* A
secara diagram

Penentuan keanggotaan suatu himpunan *fuzzy* tidak dibatasi oleh aturan-aturan tertentu. Pada umumnya terdapat tiga macam fungsi keanggotaan himpunan *fuzzy* yaitu

1. S-*function*; menurut Kuswadi (2000:29) dinyatakan dengan persamaan berikut ini

$$S(u,a,b)=\begin{cases} 0 & ; \ u \le a \\ 2\left(\dfrac{u-a}{b-a}\right)^2 & ; \ a \le u \le \dfrac{(a+b)}{2} \\ 1-2\left(\dfrac{b-u}{b-a}\right)^2 & ; \ \dfrac{(a+b)}{2} \le u \le b \\ 1 & ; \ u \ge b \end{cases} \qquad (2.12)$$

Definisi *S-funtion*

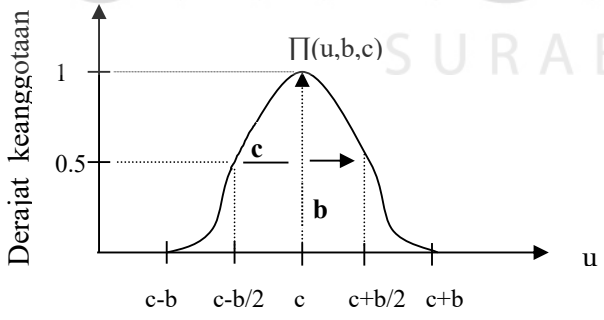


Gambar 2.5. Diagram S – *function*

2. Π -*Function* menurut Kuswadi (2000:29) adalah

$$\pi(u,b,c)=\begin{cases} S\left(u;c-b,c-\frac{b}{2},c\right) \;;\; u\le c \\ 1-S\left(u;c,c+\frac{b}{2},c+b\right) \;;\; u\ge c \end{cases} \qquad (2.13)$$

Definisi Π - *Function*

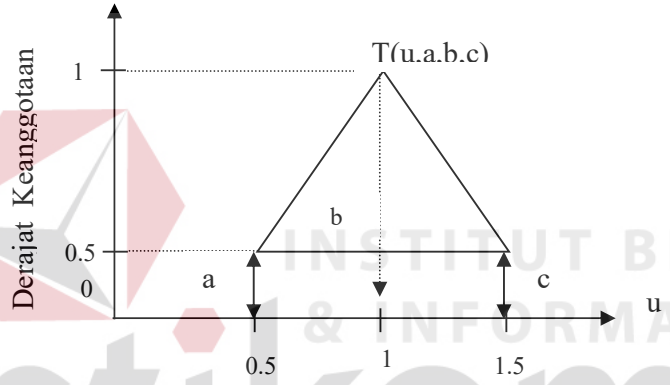


Gambar 2.6. Diagram Π -Function

3. T- *Function* menurut Kusumadewi (2002:49-50) adalah

$$T(u;a,b,c)=\begin{cases}0 & ; \quad u \le a \\ \dfrac{u-a}{b-a} & ; \quad a \le u \le b \\ \dfrac{c-u}{c-b} & ; \quad b \le u \le c \\ 0 & ; \quad u \ge c \end{cases} \qquad (2.14)$$

Definisi T - *Function*



Gambar 2.7. Diagram T- *Function*

2.3.1 Operasi Himpunan *Fuzzy*

Menurut Kuswandi (2000:30) operasi himpunan *fuzzy* dapat disimpulkan sebagai berikut :

1. *Equality*

$$\mu_A = \mu_B(u), u \varepsilon U \qquad (2.15)$$

2. *Union*

$$\mu_{(A \cup B)}(u) = \max\{\mu_A(u), \mu_B(u)\}, u \varepsilon U \qquad (2.16)$$

3. *Intersection*

$\mu_{(A \cup B)}(u) = \min \{\mu_A(u), \mu_B(u)\}, u \varepsilon U$ (2.17)

4. *Complement*

$\mu_A(u) = 1 - \mu_A(u), u \varepsilon U$ (2.18)

Jika himpunan A dan Ā adalah komplemen, maka berlaku aturan sebagai berikut:

$$\mu_{\left(A \cap \overline{A}\right)}(u) = \min \left\{\mu_A(u), \mu_{\overline{A}}(u)\right\} \leq 0{,}5 \quad (2.19)$$

$$\mu_{\left(A \cap \overline{A}\right)}(u) = \max \left\{\mu_A(u), \mu_{\overline{A}}(u)\right\} \geq 0{,}5 \quad (2.20)$$

5. *Normalization*

$\mu_{NORMAL(A)}(u) = \mu_A(u)/\max(\mu_A(u)), u \varepsilon U$ (2.21)

6. *Consentration*

$\mu_{CONV(A)}(u) = (\mu_A(u))^2, u \varepsilon U$ (2.22)

7. *Dilation*

$\mu_{DIL(A)}(u) = (\mu_A(u))^{0.5}, u \varepsilon U$ (2.23)

8. *Algebric product.*

$\mu_{(A \bullet B)}(u) = \mu_A(u) \bullet \mu_B(u), u \varepsilon U$ (2.24)

9. *Bounded sum*

$$\mu_{(A \oplus B)}(u) = \min \{1, \mu_A(u) + \mu_B(u)\}, u \varepsilon \cup \quad (2.25)$$

10. *Bounded product*

$$\mu_{(A \ominus B)}(u) = \max \{0, \mu_A(u) + \mu_B(u) - 1\}, u \varepsilon \cup \quad (2.26)$$

11. *Intensification*

$$\mu_{INT(A)}(u) = \begin{cases} 2(\mu_A(u)) & 0 \leq \mu_A(u) < 0.5 \\ 1 - 2(1 - \mu_A(u))^2 & 0.5 \leq \mu_A(u) \leq 1 \end{cases} \quad (2.27)$$

12. *Drastic Product*

$$\mu_A(u) \otimes \mu_B(u) = \begin{cases} \mu_A(u) & \mu_B(u) = 1 \\ \mu_B(u) & \mu_A(u) = 1 \\ 0 & \mu_A(u), \mu_B(u) < 1 \end{cases} \quad (2.28)$$

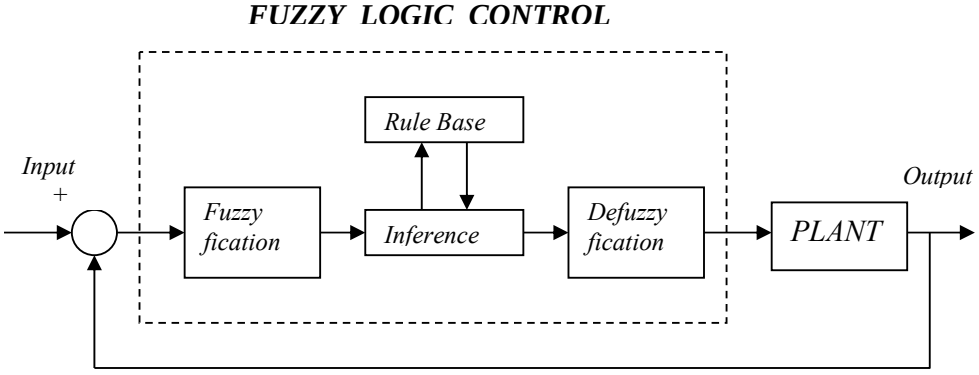
2.3.2 Variabel lingusitik

Suatu himpunan *fuzzy* biasa didefinisikan sebagai (u, T(u), U, R, S) dengan *u* adalah nama variabel linguistik, *T(u)* adalah himpunan *term (Linguistic Value /* Linguistik Label) pada *u* dan masing-masing *term* didefinisikan dengan fungsi keanggotaan yang normal (mempunyai harga maksimum sama dengan 1) dan *convex* pada U; R adalah sintetis untuk menghasilkan nama nilai-nilai pada *u*; dan *S* adalah aturan semantik untuk menghubungkan tiap nilai dengan artinya. Suatu contoh variabel linguistik misalnya kecepatan, maka himpunan *term* *T*(kecepatan) yang mungkin misalnya :

T(kecepatan) = {lambat,sedang,cepat}

Lambat, sedang, cepat adalah harga linguistik (*linguistic value)* dari variabel linguistic kecepatan, dan masing-masing didefinisikan dalam suatu himpunan *fuzzy*, sekumpulan *term* disebut *linguistic qualifiers*.

2.4 Struktur Dasar *Controller* Logika *Fuzzy*



Gambar 2.8. Blok Diagram Sistem Logika *Fuzzy*

2.4.1 Fuzzyfikasi

Proses fuzzyfikasi merupakan proses untuk mengubah variabel *non fuzzy* (*variabel numeric*) menjadi variabel *fuzzy* (*variabel linguistic*). Nilai masukan-*input* yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali logika *fuzzy* harus diubah terlebih dahulu kedalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun maka dari nilai-nilai *input* tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy* pula. Proses ini disebut **fuzzyfikasi.**

Dengan kata lain fuzzyfikasi merupakan pemetaan titik numerik (*crisp points)* $\underline{x} = (x^1, ..., x^n)^T \varepsilon\, U$ ke himpunan *fuzzy* A di U . U adalah semesta pembicaraan. Paling tidak ada dua kemungkinan pemetaan, yaitu:

- Fuzzyfikasi *singleton* : A adalah *fuzzy singleton* dengan mendukung $\underline{x}$, artinya, $\mu_A(\underline{x}') = 1$ untuk $\underline{x}' = \underline{x}$ dan

$\mu_A(\underline{x}') = 0$ untuk selain $\underline{x}' \varepsilon U$ dengan $\underline{x}' \neq \underline{x}$

- Fuzzyfikasi *nonsingleton* : $\mu_A(\underline{x}) = 1$ dan $\mu_A(\underline{x}')$ menurunkan dari 1 sebagaimana $\underline{x}'$ bergerak menjauh dari *x*. sebagai contoh :

$$\mu_A(X') = \exp\left[-\frac{(\underline{x}' - \underline{x})^T (\underline{x}' - \underline{x})}{\sigma^2}\right] \quad (2.29)$$

Dimana σ^2 adalah parameter yang menentukan bentuk dari $\mu_A(\underline{x}')$. Sejauh ini yang banyak digunakan adalah fuzzyfikasi *singleton,* tetapi pemakaian non *singleton* juga telah dirintis terutama untuk masukan-*input*yang banyak dimasuki oleh derau (*noise).*

2.4.2 *Inference* dan Penyusunan *Rule Base*

Pada umumnya, aturan-aturan *fuzzy* dinyatakan dalam bentuk 'IF-THEN' yang merupakan inti dari relasi *fuzzy*, dinyatakan dengan *R,* juga disebut implikasi *fuzzy*. Relasi *fuzzy* dalam pengetahuan dasar dapat didefinisikan sebagai himpunan pada implikasi *fuzzy*.

Aturan dasar *fuzzy* adalah dalam bentuk umum:

$R^{(1)}$: IF x_1 is F_1^1 AND … AND … x_n is F_n^1, THEN y is G^{11}

Dimana F_i^1 dan G^1 adalah himpunan *fuzzy* masing-masing di $U_1 \subset$ R dan V $\subset$ R, dan) $\underline{x} = (x^1, ..., x^n)^T \in U_1 X ... X U_n$ dan y $\in$ V adalah variabel linguistik.

Untuk mendapatkan aturan IF-THEN di atas ada dua cara utama, yaitu:

(1) Menanyakan ke operator manusia yang dengan cara manual telah mampu mengendalikan sistem tersebut dikenal dengan istilah *"Human Expert"* .

(2) Dengan menggunakan algoritma pelatihan berdasarkan data-data *input* dan *output*.

Cara pertama tersebut merupakan cara langsung seluruh aturan-aturan tersebut, karena seringkali terjadi bahwa operator tersebut mengendalikan sistem atas dasar perasaan semata dan refleks yang sulit dijelaskan. Karena keterbatasan-keterbatasan tersebut maka banyak rekayasawan menawarkan ide untuk menggunakan data *output* dan *input* sebagai dasar penyusunan aturan secara otomatik.

Dalam penalaran logika *fuzzy*, ada dua tipe utama untuk pengambilan keputusan *fuzzy*, yaitu *generalized modus ponens* (GMP) dan *generalized modus tollens* (GMT). *Generalized modus ponens* disebut juga dengan *direct reasoning*, sedangkan *generalized modus tollens* disebut juga *indirect reasoning.*

Jika himpunan *fuzzy* dinotasikan dengan A, A', B, B' dan variabel linguistic dinotasikan dengan x dan y, maka GMP dan GMT dapat dinyatakan sebagai berikut :

Generalized Modus Ponens (GMP)

Pernyataan 1 (aturan) : *if x is A then y is B*

Pernyataan 2 (fakta) : *x is A'*

Penyelesaian : y is B'

Dalam hal ini penyelesaian B' dapat dinotasikan dengan :

$B' = A' \circ R$

Dengan R adalah relasi *fuzzy* dari implikasi *fuzzy 'if A then B'*, tanda $\circ$ adalah operator komposisi, dan A' adalah himpunan *fuzzy* yang mana mempunyai bentuk : A, sangat A, lebih atau kurang A, tidak A dan lain sebagainya.

Generalizes Modus Tollens (GMT)

Pernyataan 1 (aturan)	:	*if x is A then y is B*
Pernyataan 2 (fakta)	:	*y is B'*
Penyelesaian	:	y is A'

Dalam hal ini penyelesaian B' dapat dinotasikan dengan :

$A' = R \circ B'$

Bermacam-macam pendekatan dapat digunakan menentukan hubungan korespodensi pada aturan-aturan *fuzzy*. Tabel menunjukkan penggunaan fungsi implikasi untuk mendefinisikan aturan-aturan *fuzzy* dalam bentuk 'IF x is A THEN y is B', dimana A ε U, x ε U, B ε V, y ε V.

Table 2.2. Fungsi-fungsi Implikasi *Fuzzy*

Tipe Operasi	Fungsi-fungsi implikasi untuk 'IF x is A THEN y is B'
Mini Rule [Mamdani]	$R_c = A \times B = \int_{u \times v} \mu_A(u) \wedge \mu_B(v)/(u,v)$
Product Rule [Larsen]	$R_p = A \times B = \int_{u \times v} \mu_A(u) \bullet \mu_B(v)/(u,v)$
Max-min Rule [Zadeh]	$R_m = (A \times B) \cup (notA \times V) = \int_{u \times v} (\mu_A(u) \wedge \mu_B(v)) \vee (1 - \mu_A(u))/(u,v)$
Arithmetic rule [Zadeh]	$R_a = (notA \times V) \oplus (U \times B) = \int_{u \times v} 1 \wedge (1 - \mu_A(u) + \mu_B(u))/(u,v)$
Boolean	$R_b = (notA \times V) \oplus (U \times B) = \int_{u \times v} 1 \wedge (1 - \mu_A(u) \vee \mu_B(u))/(u,v)$

2.4.3 Defuzzyfikasi

Dalam sistem kontrol secara umum, terdapat suatu hubungan sebab-akibat yang spesifik antara *input* dan *output* sistem tersebut. Karakteristik hubungan inilah yang membedakan satu sistem dengan sistem yang lain.

Pengendali yang menggunakan logika *fuzzy* juga membutuhkan spesifikasi hubungan antara *input*dan *output*, yang secara umum dinyatakan dengan :

IF (A_1) THEN (B_1)

…

IF (A_n) THEN (Bn)

A_1, …, An adalah *antecedent,* yaitu *input*yang telah di*fuzzy*fikasi, sedangkan B_1, …, Bn adalah *consequent*, yaitu aksi pengendalian (*output*). Hubungan antara

antecedent dan *consequent* disebut aturan (*rule*), dan antara satu *rule* dengan yang lain tidak terdapat hubungan sebab-akibat.

Proses untuk mendapatkan aksi *output* dari suatu kondisi *input* dengan mengikuti *rule-rule* yang telah ditetapkan disebut *inference* atau *reasoning* (pengambilan keputusan).

Keputusan yang dihasilkan dari proses penalaran ini masih dalam bentuk *fuzzy,* yaitu berupa derajat keanggotaan *output*. Hasil ini harus diubah kembali menjadi variabel numerik *non-fuzzy* melalui proses defuzzifikasi. Dua metode defuzzifikasi yang umum digunakan menurut Kuswadi (2000:37) adalah :

A. *Maximum of Mean* (MOM)

Metode ini didefinisikan sebagai :

$$v_0 = \sum_{j=1}^{j} \frac{v_j}{J} \qquad (2.30)$$

$$v_j = \max_{v \varepsilon V} \mu_V(v) \qquad (2.31)$$

v_0 = nilai *output*

J = jumlah harga maksimum

V_j = nilai *output* maksimum ke-j

$\mu_v(v)$= derajat keanggotaan elmen-elemen pada *fuzzy* set v

V = semesta pembicaraan

B. *Centre of Area* (COA)

Metode ini didefinisikan sebagai :

$$v_0 = \frac{\sum_{k=1}^{m} v_k \mu_v(v_k)}{\sum_{k=1}^{m} \mu_v(v_k)} \qquad (2.32)$$

v_o = nilai *output*

m = tingkat kuantisasi

v_k = elemen ke-k

$\mu_v(v_k)$ = derajat keanggotaan elmen-elemen pada *fuzzy* set v

V = semesta pembicaraan

Kesulitan yang ada adalah dalam penentuan MF dalam rule base yang akan kita buat. Nilai yang kita masukkan haruslah sesuai, untuk itu kita perlu mengadakan suatu eksperimen sampai kita mendapat nilai MF yang paling sesuai untuk sistem kita. Semakin sesuai MF yang kita masukkan dengan sistem yang kita buat maka *output* atau *output* yang dihasilkan sistem akan semakin baik.

2.5 *Adaptive Neuro Fuzzy Inference System*

ANFIS adalah arsitktur yang secara fungsional sama dengan *fuzzy rule base* Sugeno. Arsitektur ANFIS juga sama dengan jaringan syaraf dengan fungsi radial dengan sedikit batasan tertentu. Bisa dikatakan bahwa ANFIS adalah suatu metode yang mana dalam melakukan penyetelan aturan digunakan algoritma pembelajaran terhadap sekumpulan data. Pada ANFIS juga memungkinkan aturan – aturan untuk beradaptasi.

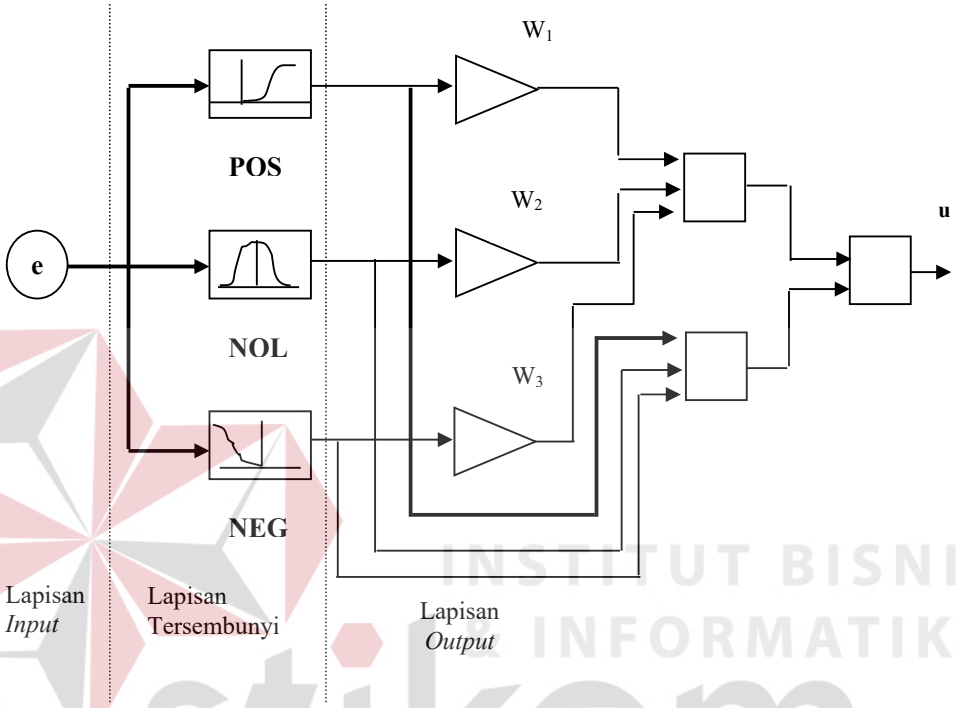
Pada jaringan syaraf kita pasti telah mengenal adanya fungsi aktivasi. Sebenarnya fungsi aktivasi itu sendiri dapat dikatakan seperti fungsi keanggotaan pada logika *fuzzy*. Misalkan kita memiliki aturan – aturan sebagai berikut :

If ***e*** *is* ***POS*** *then* ***u*** *is 100*

If ***e*** *is* ***NOL*** *then* ***u*** *is 0*

If ***e*** *is* ***NEG*** *then* ***u*** *is -100*

Dengan *input* **e** adalahh *error* dan *output* **u** adalah sinyal kontrol. Metode Inferensi dapat digambarkan digambarkan sebagai diagram blok yang menyerupai jaringan syaraf.



Gambar 2.9. Blok Diagram Untuk ANFIS dengan 3 Aturan

Jaringan memiliki satu lapisan *input*, satu lapisan tersembunyi, dan satu lapisan *output*. *Input* yang terhubung dengan lapisan tersembunyi ini berkaitan dengan bagian **IF** dari aturan. Tiap – tiap neuron pada lapisan tersembunyi berisi fungsi aktivasi, disini tidak terjadi penjumlahan, sebab setiap neuron hanya menerima satu *input*-an. Bobot untuk setiap neuron pada lapisan berupa nilai **-100**, **0** dan **100.** Neuron pada lapisan *output* merupakan hasil rata – rata terbobot sebagai metode *centroid* dalam defuzzy.

Jaringan ***Backpropagation*** dapat diaplikasikan untuk kasus ini jika setiap lapisan bersifat differentiable. Ada 2 pembelajaran yang akan dilakukan. Pertama, pengaturan bobot pada lapisan *output* hingga tercapai *error* minimum. Kedua, mengatur bentuk fungsi keanggotaan dengan cara mengatur parameter – parameter yang bersesuaian. Jaringan ini dapat digambarkan sebagai jaringan *feedforward* dengan satu jaringan lapisan *input*, satu lapisan tersembunyi dan satu lapisan *output*. Pada jaringan tesebut ada pemetaan *nonlinier* dari lapisan *input* kelapisan tersembunyi, kemudian dilanjutkan dengan pemetaan linier dari lapisan tersembunyi ke lapisan *output*. Lebih tepatnya, jaringan ini mirip dengan jaringan dengan fungsi basis radial *(radial basis function)*.

Beberapa jaringan yang menggunakan fungsi basis radial digunakan untuk pencocokan kurva pada ruangan multidimensi. Teknik yang digunakan oleh fungsi basis radial adalah memilih **F** sedemikian sehingga :

$$\begin{aligned} \mathrm{F(u)} &= w^{T} f\left(\| u - u_k \|\right) \\ &= \left[w_1\ w_2 \cdots\ w_k\right] \begin{bmatrix} \mathrm{f}\left(\|\mathrm{u} - \mathrm{u}_1\|\right) \\ \mathrm{f}\left(\|\mathrm{u} - \mathrm{u}_2\|\right) \\ \vdots \\ \mathrm{f}\left(\|\mathrm{u} - \mathrm{u}_k\|\right) \end{bmatrix} \end{aligned} \qquad (2.33)$$

dengan $u \in \Re^m$ adalah vektor *input* , $w_k \in \Re^m$ (k = 1,2,…,k) adalah vektor yang berisi data – data yang akan dilatih, $w \in \Re^k$ adalah vektor bobot, $f\left(\| u - u_k \|\right)$ adalah himpunan fungsi basis radial (nonlinier), dan $\|-\|$ adalah bentuk normal (biasanya normal Eclidian). Titik – titik data u_k akan menjadi pusat dari fungsi – fungsi basis radial. Fungsi $f(u, u_{\mathrm{k}})$ akan maksimum jika *input* u terletak pada pusat u_k .

Tiap – tiap u_k harus memenuhi persamaan :

$$W^T f(u_k) = d_k \qquad (2.34)$$

Dengan d_k adalah respon yang diharapkan yang berhubungan dengan u_k, dengan demikian nilai bobot W yang belum diketahui dapat dicari melalui persamaan :

$$\begin{bmatrix} w_1 & w_2 & \dots & w_k \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & & f_{1k} \\ f_{21} & f_{22} & & f_{2k} \\ & & & \\ f_{k1} & f_{k2} & & f_{kk} \end{bmatrix} = \begin{bmatrix} d_1 & d_2 & \cdots & d_k \end{bmatrix} \qquad (2.35)$$

yang dapat dinotasikan sebagai:

$$W^T \phi = d^T \qquad (2.36)$$

dengan vektor d menunjukkan vektor respon yang diharapkan

$$f_{jk} = f\left(\| u_j - u_k \| \right); \text{ dengan j,k = 1,2,…,K} \qquad (2.37)$$

Matrik $\phi = \{ f_{jk} \}$ disebut sebagai matriks interpolasi, Pada fungsi basis radial (khususnya fungsi Gauss), matriks interpolasi ini bersifat definit positif. Dengan demikian kita bisa menyelesaikan dengan

$$w^T = d^T \phi^{-1} \qquad (2.38)$$

2.5.1 Arsitektur ANFIS

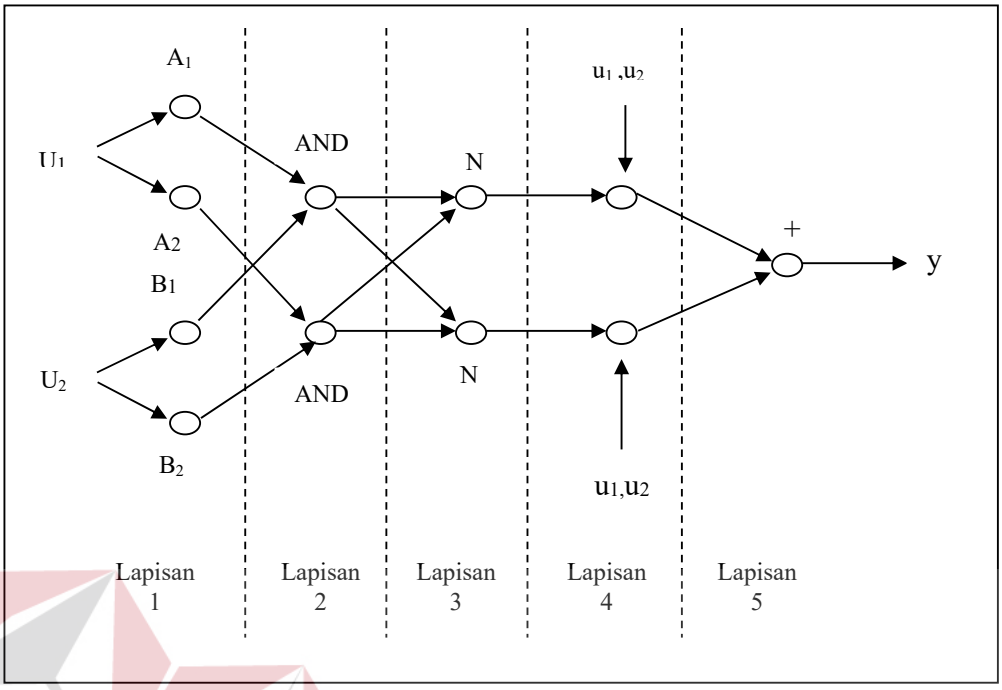
Misalkan ada 2 *input* **u_1**, **u_2** dan satu *output* **y.** Ada 2 aturan yang harus diperhatikan yaitu :

If u_1 is A_1 and u_2 is B_1 Then $y_1 = c_{11}u_1 + c_{12}u_2 + c_{10}$

If u_1 is A_2 and u_2 is B_2 Then $y_2 = c_{21}u_1 + c_{22}u_2 + c_{20}$

Jika α predikat untuk aturan kedua aturan adalah α_1 dan α_2, maka dapat dihitung rata – rata terbobot :

$$y = \frac{\alpha_1 y_1 + \alpha_2 y_2}{\alpha_1 + \alpha_2} = \overline{\alpha}_1 y_1 + \overline{\alpha}_2 y_2 \qquad (2.39)$$



Gambar 2.10. Struktur Jaringan ANFIS

Jaringan ANFIS pada gambarr 2.10 terdiri dari lapisan – lapisan sebagai berikut :

1. Tiap –tiap *node* i pada lapisan pertama adadtif terhadap parameter suatu fungsi aktivasi. *Output* dari tiap *node* berupa derajat keanggotaan yang diberikan oleh fungsi keanggotaan *input*, yaitu $\mu_{A1}[u_1]$, $\mu_{B1}[u_2]$, $\mu_{A2}[u_1]$ atau $\mu_{B2}[u_2]$. Sebagai contoh, misalkan fungsi keanggotaan diberikan sebagaimana dicontohkan pada buku Kusumadewi (2002:209).

$$\mu[x] = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \qquad (2.40)$$

dimana {a,b,c} adalah parameter – parameter *input*. Jika nilai parameter – parameter ini berubah, maka bentuk kurva yang terjadi pun ikut berubah.

Parameter – parameter pada lapisan itu biasanya kenal dengan nama *primise parameter.*

2. Tiap – tiap *node* pada lapisan kedua berupa *node* tetap yang *output*-nya adalah hasil *input*. Biasanya digunakan operator AND. Tiap – tiap *node* mempresentasikan α predikat dari aturan ke-i.

3. Tiap – tiap *node* pada lapisan ketiga berupa *node* tetap yang merupakan hasil perhitungan perbandingan dari α predikat dari aturan ke-i terhadap jumlah dari keseluruhan α predikat.

$$\overline{\alpha}_i = \frac{\alpha_i}{\alpha_1 + \alpha_2}, \text{ dengan i =1,2.} \qquad (2.41)$$

Hasil ini dikenal dengan nama *normalized firing strength*.

4. Tiap – tiap *node* pada lapisan keempat merupakan merupakan *node* adaptif terhadap suatu *output*.

$$\overline{\alpha}_i y_i = \overline{\alpha}_i (c_{i1} u_1 + c_{i2} u_2 + c_{i0}) \text{ ,dengan i = 1,2.} \qquad (2.42)$$

dengan α_i adalah *normalized firing strength* pada lapisan ketiga dan {c_{i1}, c_{i2}, c_{i0} }adalah parameter – parameter pada *node* tersebut. Parameter – parameter pada lapisan tersebut tersebut disebut dengan nama *consequent parameters.*

5. Tiap – tiap *node* pada lapisan kelima adalah *node* tetap yang merupakan penjumlah dari semua masukan.

2.5.2 Algoritma pembelajaran ANFIS

Pada saat *premise* parameter ditemukan *output* yang terjadi akan merupakan kombinasi liniear dari *consequent parameters*, yaitu :

$$y = \frac{\alpha_1}{\alpha_1+\alpha_2} y_1 + \frac{\alpha_2}{\alpha_1+\alpha_2} y_2 \quad (2.43)$$

$$= \overline{\alpha}_1(c_{11}u_1 + c_{12}u_2 + c_{10}) + \overline{\alpha}_2(c_{21}u_1 + c_{22}u_2 + c_{20}) \quad (2.44)$$

$$= (\overline{\alpha}_1 u_1)c_{11} + (\overline{\alpha}_1 u_2)c_{12} + \overline{\alpha}_1 c_{10} + (\overline{\alpha}_2 u_2)c_{21} + (\overline{\alpha}_2 u_2)c_{22} + \overline{\alpha}_2 c_{20} \quad (2.45)$$

adalah linear terhadap parameter c_{ij} (i = 1,2 dan j = 0,1,2).

Algoritma hybrid akan mengatur parameter – parameter c_{ij} secara maju *(forward)*dan akan mengatur parameter – parameter {a_i, b_i, c_i} secara mundur *(backward)*.

Pada langkah maju *(forward)*, *input* jaringan akan merambat maju sampai pada lapisan keempat, dimana parameter – parameter c_{ij} akan diidentifikasi dengan menggunakan metode *least-square*. Sedangkan pada langkah mundur *(backward)*, *error* sinyal akan merambat mundur dan parameter – parameter {a_i, b_i, c_i} akan diperbaiki dengan menggunakan metode *gradient-decent*.

2.6 Algoritma Backpropagasi

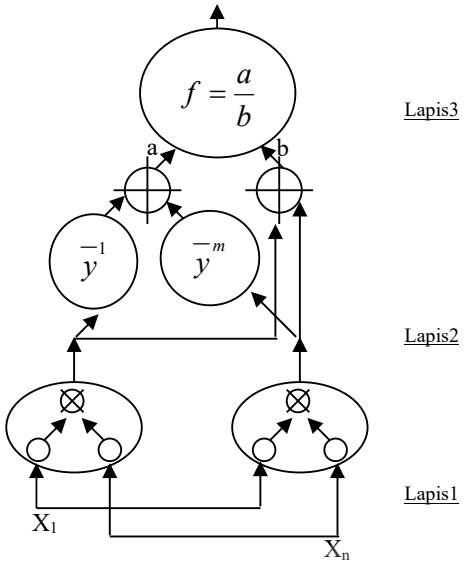
Obyektif dari algoritma ini : mendapatkan bentuk persamaan dan nilai koefisien dalam formula dengan meminimalkan jumlah *Sum Square Error* melalui model yang dikembangkan (*training set*).

1. Dimulai dengan lapisan *input*, hitung *output* dari setiap elemen pemroses melalui lapisan *output*.
2. Hitung kesalahan pada lapisan *output* yang merupakan selisih antara data aktual dan target.
3. Transformasikan kesalahan tersebut pada kesalahan yang sesuai di sisi *input* elemen pemroses.
4. Propagasi balik kesalahan – kesalahan tersebut ke lapisan tersembunyi. Transformasikan kesalahan ini pada *output* setiap elemen pemroses ke kesalahan yang terdapat pada masukan. Ulangi proses ini sampai lapisan *input* tercapai.
5. Ubah seluruh bobot dengan menggunakan kesalahan pada sisi *input* elemen pemroses dan *output* elemen pemroses yang terhubung.

Tujuan dari perubahan bobot untuk setiap lapisan, bukan merupakan hal yang sangat penting. Perhitungan kesalahan merupakan pengukuran bagaimana jaringan dapat belajar dengan baik. Kesalahan pada *output* dari jaringan merupakan selisih antara *output* aktual (*current output*) dan *output* target (*desired output*).

Berikut ini akan dipaparkan penurunan dari teknik '*back - propagation*' tersebut. Misalkan dimiliki pasangan *input output* (x^p, d^p), $x^p \in U \subset R^n$, $d^p \varepsilon V \subset R$. Tugas kita menentukan sistem logika *fuzzy* dalam bentuk persaman sedemikian rupa sehingga:

$$e^p = \frac{1}{2}\left[f\left(x^p\right) - d^p\right]^2 \qquad (2.46)$$



Gambar 2.11. Jaringan dari Sistem logika *fuzzy*

Diminimalkan bila diasumsikan bahwa M (jumlah aturan) dapat ditentukan,maka masalahnya adalah bagaimana mendapatkan harga-harga dari parameter sistem logika *fuzzy* ,yaitu $\bar{y}^l$, x_1^l *dan* σ_1^l sedemikian rupa sehingga e^p dapat diminimalkan. Berikut ini digunakan *e, f* dan *d* untuk menyatakan e^p , $f\left(x^p\right)$ dan d^p . Untuk melatih y^{-l} digunakan :

$$\bar{y}^l\left(k+1\right) = \bar{y}^l\left(k\right) - \alpha \frac{\partial e}{\partial \bar{y}^l}\bigg|_k \qquad (2.47)$$

dimana l = 1,2,…,M, k = 0,1,2,… dan α adalah suatu konstanta pembelajaran (learning constan). Dari gambar 2.11 maka dapat dilihat f (demikian pula e) tergantung kepada $\bar{y}^l$ hanya melalui a ,dimana $f = a/b$ dan

$$a = \sum_{l=1}^{M}\left(\bar{y}^l z^l\right) \qquad (2.48)$$

$$b = \sum_{l=1}^{M} (z^l) \qquad (2.49)$$

$$\text{dan } z^l = \prod \exp\left[-\left[\frac{x_1 - \bar{x}^l}{\sigma_1^l} \right]^2 \right] \qquad (2.50)$$

sehingga dengan menggunakan dalil rantai (chain rule) maka didapat :

$$\frac{\partial e}{\partial \bar{y}^l} = (f-d)\frac{\partial f}{\partial a}\frac{\partial a}{\partial \bar{y}^l} = (f-d)\frac{1}{b}z^l \qquad (2.51)$$

Subsitusikan persamaan (2.50) ke persamaan (2.51) maka akan didapatkan algoritma pembelajaran untuk $\bar{y}^l$:

$$\bar{y}^l(k+1) = \bar{y}^l(k) - \alpha \frac{f-d}{b} z^l \qquad (2.52)$$

dimana $l = 1,2,\ldots,M$, dan $k = 0,1,2,\ldots$ untuk melatih $\bar{x}_1^l$ digunakan

$$\bar{x}_1^l(k+1) = \bar{x}_1^l(k) - \alpha \frac{\partial e}{\partial \bar{x}_1^l}\bigg|_k \qquad (2.53)$$

dimana $i = 1,2,\ldots,N$, $l = 1,2,\ldots, M$ dan $k = 0,1,2,\ldots$

Dapat dilihat dari gambar bahwa f (demikian pula e) tergantung pada $\bar{x}_1^l$ hanya melalui z^l ; sehingga dengan dalil rantai kita dapatkan :

$$\frac{\partial e}{\partial \bar{x}_1^l} = (f-d)\frac{\partial f}{\partial z^l}\frac{\partial z^l}{\partial \bar{x}_1^l} = \frac{f-d}{b}\left(\bar{y}^l(k) - f\right)z^l \frac{2\left(x_1 - \bar{x}_1^l(k)\right)}{\sigma_1^l(k)^2} \qquad (2.54)$$

Dengan mensubtitusikan persamaan (2.54) ke dalam persamaan (2.55) maka akan didapatkan :

$$\bar{x}_1^l(k+1) = \bar{x}_1^l(k) - \alpha \frac{f-d}{b}\left(\bar{y}_1^l(k) - f\right)z^l \frac{2\left(x_1 - \bar{x}_1^l(k)\right)}{\sigma_1^l(k)^2} \qquad (2.55)$$

Dimana i = 1,2,….,N , l = 1,2,….,M dan k = 0,1,2,… Dengan menggunakan metode pembelajaran yang sama maka dapat diperoleh algoritma pembelajaran untuk σ_1^l :

$$\sigma_1^l(k+1) = \sigma_1^l(k) - \alpha \frac{f-d}{b}\left(\bar{y}^l(k) - f\right) z^l \frac{2\left(x_1 - \bar{x}_1^l(k)\right)}{\sigma_1^l(k)^2} \qquad (2.56)$$

dimana i = 1,2,…,N ; l = 1,2,…….,M dan k = 0,1,2,… Algoritma pembelajran persamaan (2.52), (2.54), dan (2.56). menggunakan prosedur *'error back propagation'*. Pembelajaran dari sistem logika *fuzzy* diatas meliputi dua langkah.Pertama, dengan menggunakan *input* $\underline{x}^p$ yang tersedia hitung *output* sistem logika *fuzzy* untuk memperoleh z^l (l = 1,2,…..,M), a, b dan f. Kemudian melatih parameter jaringan secara mundur menggunakan persamaan (2.52),(2.53), dan (2.56).

Metode pemilihan harga-harga awal parameter secara on-line merupakan hal yang sangat penting ,mengingat teknik *'back-propagation'* memiliki kelemahan yaitu mudahnya ia terperangkap dalam harga optimum yang lokal. Dengan pemilihan harga awal yang tepat maka kelemahan tersebut sangat mungkin untuk dihindari. Juga, proses pembelajaran akan menjadi lebih cepat.

2.7 Motor Induksi

Pada prinsipnya motor induksi tiga fasa bekerja berdasarkan induksi elektromagnet dan induksi tegangan. Putaran pada motor induksi tiga fasa ditmbulkan oleh karena adanya medan putar (*fluks* yang berputar) yang dihasilkan oleh kumparan stator, jika dihubungkan dengan sumber tegangan tiga fasa.

2.7.1 Tegangan stator

Persamaan tegangan stator motor induksi tiga fasa dapat dituliskan sebagai berikut

$$\overline{V}_s = R_s \overrightarrow{I}_s + \frac{d}{dt}\vec{\lambda}s + j\omega_s \vec{\lambda}_s \quad (2.57)$$

Dimana :

$$\vec{V}_s = v_{ds} + jv_{qs} \quad (2.58)$$

$$\vec{i}_s = \vec{i}_{ds} + j\,\vec{i}_{qs} \quad (2.59)$$

$$\overrightarrow{\lambda}_s = \vec{\lambda}_{ds} + j\vec{\lambda}_{qs} \quad (2.60)$$

2.7.2 Tegangan rotor

Persamaan tegangan rotor motor induksi tiga fasa dapat dituliskan sebagai berikut

$$\vec{V}_r = R_r \overrightarrow{I}_r + \frac{d}{dt}\vec{\lambda}_r + j(\omega_s - \omega_r)\vec{\lambda}_r \quad (2.61)$$

Dimana :

$$\vec{V}_r = v_{dr} + jv_{qr} \quad (2.62)$$

$$\vec{i}_r = \vec{i}_{dr} + j\,\vec{i}_{qr} \quad (2.63)$$

$$\overrightarrow{\lambda}_r = \vec{\lambda}_{dr} + j\vec{\lambda}_{qr} \quad (2.64)$$

2.7.3 Torsi elektormagnetik

Torsi Elektromagnetik *(T_e)* merupakan fungsi dari arus stator dan arus rotor, yang dituliskan sebagai berikut :

$$T_e = pM\,(i_{dr}\,i_{qs} - i_{qr}\,i_{ds}) \quad (2.65)$$

dimana : p = Jumlah pasangan kutub

M = Induktansi gandeng (H)

i_{dr} = Arus rotor pada sumbu d (A)

i_{qs} = Arus stator pada sumbu q (A)

i_{qr} = Arus rotor pada sumbu q (A)

i_{ds} = Arus stator pada sumbu d (A)

2.7.4 Kecepatan angular

Kecepatan putaran rotor, dinyatakan sebagai fungsi dari torsi beban, dan torsi elektromagnetik, yang dinyatakan sebagai berikut :

$$\frac{J}{p}\frac{d}{dt}\omega_r + K_g\omega_r = T_e - T_l \qquad (2.66)$$

dimana : K_g = Konstanta gesekan (kg.m^2/dt)

J = Momen inersia(kg.m^2)

ω_r = Kecepatan Angular

2.7.5 *Flux linkage*

Flux linkage didefinisikan sebagai besarnya medan putar *(fluks)* pada kumparan (baik stator maupun rotor) dengan jumlah N lilitan. Masing-masing harga *flux linkage* baik pada rangkaian stator dan rotor adalah :

$$\vec{\lambda}_s = L_s\vec{I}_s + M\vec{I}_r \qquad (2.67)$$

$$\vec{\lambda}_r = L_r\vec{I}_r + M\vec{I}_r \qquad (2.68)$$

dimana : λ_s = *Fluks Linkage* pada kumparan stator dengan N lilitan

λ_r = *Fluks Linkage* pada kumparan rotor dengan N lilitan

L_s = Induktansi diri kumparan stator (H)

L_r = Induktansi diri kumparan rotor (H)

i_r = Arus rotor (A)

i_s = Arus stator (A)

persamaan arus stator dan arus rotor dapat dinyatakan dalam persamaan berikut

$$\frac{d}{dt}i_{ds} = \frac{1}{D}(L_r A_1 + MA_2) \quad (2.69)$$

$$\frac{d}{dt}i_{qs} = \frac{1}{D}(L_r B_1 - MB_2) \quad (2.70)$$

$$\frac{d}{dt}i_{ds} = \frac{1}{D}(-MA_1 + L_s A_2) \quad (2.71)$$

$$\frac{d}{dt}i_{qs} = \frac{1}{D}(-MB_1 + L_s B_2) \quad (2.72)$$

Sedangkan persamaan sudut (posisi) dan kecepatan dari putaran rotor, dinyatakan dengan :

$$\frac{d}{dt}\theta_r = \omega_r \quad (2.73)$$

$$\frac{d}{dt}\omega_r = \frac{1}{J}(T_e - T_l) - K_g\omega_r \quad (2.74)$$