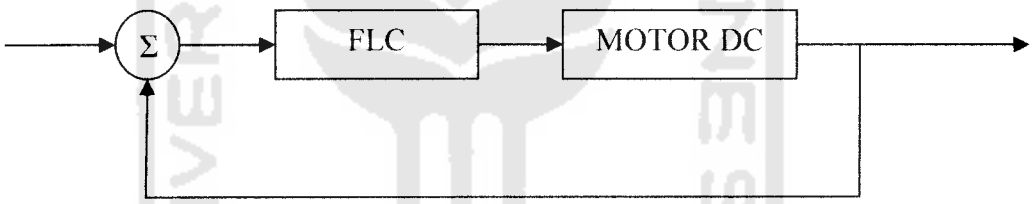
BAB III

METODOLOGI

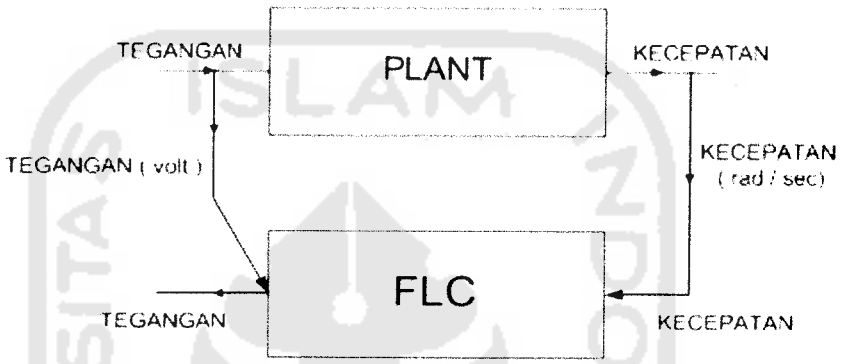
Pada perancangan sistem, ada dua buah masukan dalam FLC (*Fuzzy Logic Controller*) yaitu berupa kecepatan yang diinginkan (*error*) dan perubahan kecepatan (*derivative error*). Sedangkan keluarannya adalah perubahan tegangan. Sebagai keluaran motor dan juga sebagai hasil akhir dari sistem adalah kecepatan. Perancangan sistem kendali ini dapat dilihat pada diagram blok berikut :



Gambar 3.1 Blok diagram sistem pengendali kecepatan motor DC

Pelatihan dari sistem pengendali dirancang dengan menggunakan metode *inverse*, dimana masukan dari *plant* / model motor adalah sebagai keluaran dari *fuzzy*, sehingga keluarannya akan digunakan kembali sebagai masukan. Karena pada pelatihan menggunakan metode *inverse*, maka masukan dan keluaran dari sistem kendali yang sebenarnya akan dibalik pada saat pelatihan. Pada saat pelatihan masukan dari *fuzzy* adalah keluaran dari motor, yaitu kecepatan motor, sedangkan keluaran dari *fuzzy* merupakan masukan motor, yaitu tegangan. Seperti yang terlihat

pada Gambar 3.2. Kemudian kemampuan dari *fuzzy* akan dipergunakan untuk mengidentifikasikan motor. Selanjutnya hasil proses identifikasi dipergunakan pada proses pengendalian kecepatan motor.



Gambar 3.2 Model *inverse* pelatihan

Pada perancangan sistem ini seluruhnya dimodelkan dalam perangkat lunak (*software*), baik untuk pemodelan motor DC, operasi *fuzzy*, dan proses pengendalinya. Perangkat lunak yang digunakan dalam perancangan sistem adalah Matlab R 2007a. Matlab merupakan perangkat lunak yang memiliki bahasa untuk komputasi teknik dan dapat digunakan untuk perhitungan, visualisasi, pemrograman, pengembangan algoritma, pemodelan, simulasi, analisa data, dan membangun aplikasi.

3.1 Pemodelan Matematis Motor DC

Pemodelan sistem motor DC berikut merupakan pemodelan ke dalam bentuk matematis (fungsi alih). Untuk mencari nilai fungsi alih ini sebelumnya ditentukan dahulu parameter yang berhubungan dengan sistem motor DC.

R_a = Hambatan jangkar (Ohm)

L_a = Induktansi jangkar (Henry)

i_a = Arus jangkar (Ampere)

i_f = Arus medan (Ampere)

e_a / V_{app} = Tegangan jangkar terpasang (Volt)

e_b / V_{emf} = Tegangan medan / emf balik (Volt)

T = Torsi motor (N-m)

J = Momen Inersia (kg – m^2)

K_i = Koefisien torsi (N-m / Ampere)

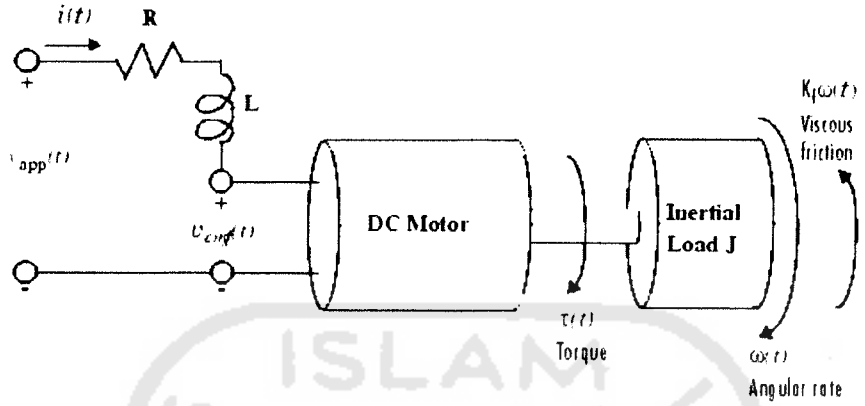
K_f / b = Koefisien gesek (N-m / rad /sec)

K_b = Koefisien emf balik (Volt / rad /sec)

P = Daya (Watt)

n = Kecepatan motor (rpm)

ω / $\frac{d\theta}{dt}$ = Kecepatan sudut (radian)



Gambar 3.3 Diagram skematik Motor DC

Berdasarkan gambar 3.3, dapat diperoleh beberapa persamaan matematis kendali motor DC.

Perhitungan untuk mencari Torsi motor :

$$P = \frac{2\pi n}{60} T \qquad (3.1)$$

$$P = nT / 9.55 \qquad (3.2)$$

$$T = 9.55 \frac{P}{n} \qquad (3.3)$$

Untuk medan arus konstan, fluks juga konstan dan torsi mempunyai arah arus sesuai arus kumparan magnet, sehingga :

$$T = K_i \, i_a \qquad (3.4)$$

Untuk fluks konstan, tegangan induksi e_b / V_{emf} berbanding lurus dengan kecepatan sudut $\frac{d\theta}{dt}$,

$$e_b = K_b \frac{d\theta}{dt} = K_b \, \omega_m \qquad (3.5)$$

Kecepatan jangkar magnet motor DC dikontrol oleh tegangan kumparan e_a / $V_{app.}$ Persamaan diferesial rangkaian kumparan magnet adalah :

$$La \frac{d\theta}{dt} + R_a i_a + e_b = e_a \qquad (3.6)$$

Arus jangkar magnet menghasilkan torsi yang bekerja terhadap inersia dan gesekan, sehingga

$$J \frac{d^2\theta}{dt^2} + b \frac{d\theta}{dt} = T = Ki_a \qquad (3.7)$$

Hubungan antara K_i dan K_b seperti terlihat pada persamaan berikut dalam satuan hp (*house power*). Persamaan daya mekanik yang dihasilkan pada jangkar :

$$P = \frac{e_b i_a}{746} hp \qquad (3.8)$$

Persamaan daya mekanik dengan mencakup torsi dan kecepatan sudut, P adalah

$$P = \frac{T \omega}{550} hp \qquad (3.9)$$

dengan T dalam ft-lb dan ω_m dalam rad / sec. Dengan menggunakan persamaan (3.5) dan (3. 6), didapatkan persamaan berikut :

$$\frac{K_b \omega T}{746 K_i} = \frac{T\omega}{550}$$

$$K_b = \frac{746}{550} K_i = 1.356 K_i \qquad (3.10)$$

Maka persamaan tempat kedudukan sistem motor DC adalah sebagai berikut :

$$\frac{d}{dt}\begin{bmatrix} i \\ \omega \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{K_b}{L} \\ \frac{K_i}{J} & -\frac{K_f}{J} \end{bmatrix}\begin{bmatrix} i \\ \omega \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} v_{app}(t)$$

$$y(t) = [\,0\ 1\,] \cdot \begin{bmatrix} i \\ \omega \end{bmatrix} + [\,0\,] \cdot v_{app}(t) \qquad (3.11)$$

3.2 Simulasi Sistem dengan Matlab

Matlab adalah bahasa pemrograman dengan performansi tinggi untuk komputasi teknis. Matlab merupakan integrasi komputasi, visualisasi, dan pemrograman dalam lingkungan yang mudah digunakan dimana masalah dan solusi diekspresikan dalam notasi matematis yang umum. Dengan kemampuan tersebut, Matlab menjadi perangkat bantu yang sangat bermanfaat dalam berbagai bidang ilmu teknik. Berbagai macam komputasi, mulai dari perhitungan sederhana hingga pemodelan yang rumit, sangat terbantu dengan adanya perangkat lunak ini. Beberapa kegunaan dari matlab ini adalah untuk perhitungan, visualisasi, pemrograman, pengembangan algoritma, pemodelan, simulasi, analisa data, eksplorasi, membangun aplikasi dan pembuatan GUI (*Graphical User Interface*).

3.2.1 Perancangan Simulasi Sistem dengan Matlab Simulink

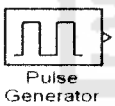
SIMULINK adalah perangkat lunak yang digunakan untuk memodelkan, melakukan simulasi, dan analisa terhadap sistem dinamis. *Simulink* dapat digunakan

untuk sistem linier maupun sistem non linier. *Simulink* menyediakan *block library* yang lengkap meliputi *sinks* (keluaran), *sources* (masukan) komponen linier, non linier dan konektor. *Simulink* dapat digunakan untuk mensimulasikan sistem, artinya mengamati dan menganalisa perilaku dari tiruan sistem. Tiruan sistem diharapkan mempunyai perilaku yang sangat mirip dengan sistem fisik. Jika digunakan dengan benar, simulasi akan membantu proses analisis. Berikut ini adalah beberapa blok *simulink* yang digunakan pada simulasi sistem ini :

1. *Step*

Merupakan blok yang digunakan untuk memberikan nilai masukan yang berupa nilai tertinggi atau nilai maksimal dari suatu sistem.

2. *Pulse generator*

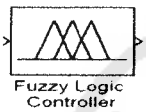


Merupakan blok yang digunakan untuk membangkitkan pulsa pada pengambilan data pelatihan.

3. *Sum*

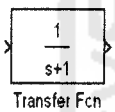
Merupakan blok penjumlah atau pengurang masukan sesuai dengan tanda operasinya. Blok ini menerima masukan berupa skalar, vektor, matrik atau elemen dari vektor tunggal dan blok ini memiliki lebih dari satu masukan.

4. *FLC (Fuzzy Logic Controller)*



Blok yang digunakan untuk memanggil FIS yang telah dibuat, yang sebelumnya telah di simpan dalam *workspace*. Kemudian dapat dipanggil dengan menuliskan nama yang sama seperti yang telah ditulis pada FIS.

5. *Transfer Fcn*



Blok model sistem linier dengan variabel s. Blok dapat diperoleh dari *single-input single-output* (SISO) dan *single-input multiple output* (SIMO). Pada sistem ini, *Transfer function* merupakan pengganti dari motor DC yang sebenarnya yang diperoleh dari hasil perhitungan dan analisa.

6. *Demultiplexer*

Blok yang memiliki satu masukan menjadi beberapa keluaran. Blok ini akan membagi :

a. Sinyal vektor menjadi skalar atau vektor yang lebih kecil.

b. Sinyal bus yang dihasilkan blok *mux* menjadi skalar, vektor, atau sinyal matrik.

7. *Multiplexer*

Blok yang memberikan beberapa masukan menjadi satu keluaran. Masukan dapat berbentuk skalar, vektor, atau matrik. Keluaran blok ini tergantung masukannya, bisa berupa vektor atau sinyal gabungan berisi matrik dan vektor.

8. *Gain*

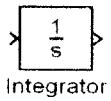
Blok *gain* merupakan blok penguat. Masukan pada blok dapat berupa besaran skalar, vektor atau matrik. Dapat diperoleh secara *trial and error* maupun perhitungan matematis.

9. Saturation

Blok yang memiliki batas bawah dan batas atas pada sebuah sinyal. Agar masukan tidak keluar dari batasan yang telah ditentukan.

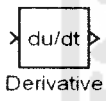
10. Integrator

10. Integrator



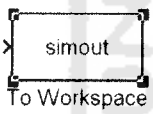
Blok ini merupakan integral dari keluaran. Merubah nilai keluaran menjadi perubahan / *integral* dari keluaran.

11. Derivative



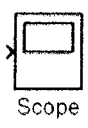
Blok yang merupakan turunan dari nilai masukan. Blok ini digunakan untuk mengetahui nilai perubahan masukan. Pada sistem ini, untuk mengetahuia nilai perubahan kecepatan pada masukan *fuzzy.*

12. *To Workspace*



Blok ini digunakan untuk menyimpan data dari hasil simulasi ke dalam *workspace* Matlab. hasil penyimpanan dapat berupa format struktur dengan waktu, struktur, dan array.

13. Scope



Blok yang dapat menampilkan keluaran dari masukan sistem dengan respon waktu simulasi.

Pada penulisan ini, *plant* motor DC di modelkan dalam bentuk fungsi alih. Motor yang disimulasikan diambil dari data motor sebenarnya di Laboratorium Instalasi dan Mesin Listrik. Berikut ini adalah karakteristik motor DC yang sebenarnya :

Type : GSDT

Exciting shunt

Rpm = 1750

Armature control = ~ Rpm

Field control = ~ Rpm

Bearing : DE 6203ZZ NDE 6204ZZ

Serial number R2A308012

Weight : 19 Kg

Date : 1992

Design : -

Keluaran = 0.5 HP

Rating cont : - *Class ins* F

- *Class ins* f

Armature : - *Voltage* = 150 V

- *Current* = 3.2 A

- *Resistance* = 46.875 Ohm

- *Inductance* = 0.01 H

Field : - *Voltage* = 100 V

- *Current* = 0.48 A

- *Resistance* = 208.33 Ohm

- *Inductance* = 0.03 H

Penggunaan data diatas sangat bermanfaat untuk memperoleh perhitungan matematis fungsi alih. Perhitungan matematis fungsi alih seperti dibawah ini :

1. Perhitungan koefisien torsi (Ki) :

$$T = 9.55\frac{P}{\eta}$$

$$= 9.55\ \frac{(0.5)(746)}{1750}$$

$$= 2.036 \text{ Nm}$$

$$T = K_i\, i_a$$

$$2.036 = K_i\, 3.2$$

$$K_i = \frac{2.036}{3.2}$$

$$Ki = 0.636$$

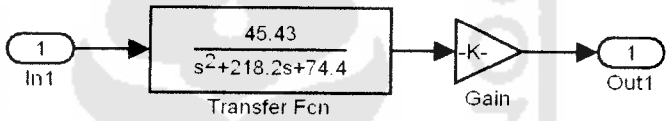
2. Perhitungan koefisien emf balik

$$K_b = \frac{746}{550} K_i$$

$$= \frac{746}{550} 0.636$$

$$K_b = 0.863$$

Nilai L = 0.01 H, sama dengan nilai sebenarnya. Nilai R yang sebenarnya diubah secara *trial and error*. Perubahannya berdasarkan pada teori motor DC yaitu semakin kecil R maka kecepatan semakin besar. Nilai R = 2.18 Ohm. Sedangkan nilai K_f = 0.226 diperoleh secara *trial and error*. Nilai –nilai tersebut dimasukan kedalam *m.file* Matlab. Dimasukkan ke persamaan *state space* seperti pada persamaan 3.11, kemudian disimpan dengan nama tertentu. Langkah selanjutnya dengan menggunakan perintah fungsi alih (tf (nama *state space*)), *m.file* dijalankan diperoleh fungsi alih seperti gambar 3.4.



Gambar 3.4 Fungsi alih dari motor DC

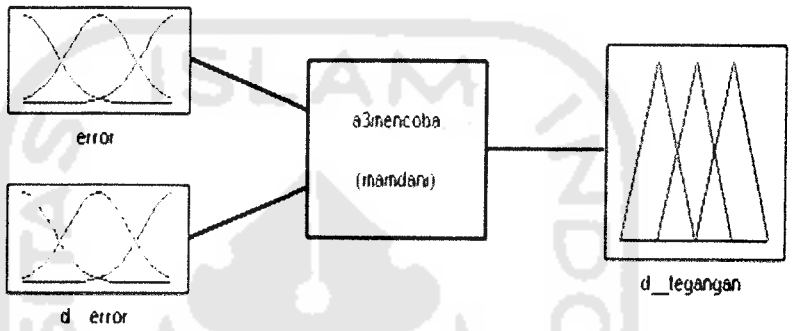
Pada gambar 3.4 setelah fungsi alih diperoleh kemudian ditambahkan dengan nilai *gain* / penguatan, agar diperoleh nilai keluaran yang diinginkan (1750 rpm) dengan masukan (150 volt). Nilai penguatannya adalah 60 / 3.14, yang diperoleh dari perubahan rpm (*radian per menit*) menjadi (*radian per second*).

3.2.2 Perancangan *Fuzzy Logic Controller* (FLC)

FLC merupakan paket program yang berisikan sekumpulan fungsi numeris yang bekerja pada lingkungan matlab untuk membangun sistem berbasis logika *fuzzy*. *Toolbox* ini sangat baik untuk mempetakan masukan ke keluaran yang dapat membangun dan mengedit sistem inferensi *fuzzy* dengan bentuk grafis atau fungsi perintah.

1. Penentuan variabel masukan dan keluaran.

Sistem motor DC ini, memiliki dua buah masukan dan sebuah keluaran pada FLC. Masukan dari *fuzzy* kontrol adalah kecepatan (*error*) dan perubahan kecepatan (*d_error*), sedangkan keluarannya adalah perubahan tegangan.



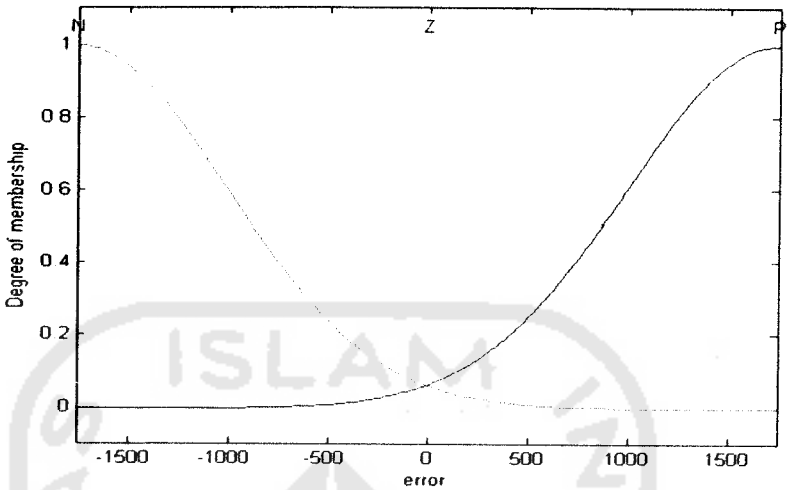
Gambar 3.5 Variabel masukan dan keluaran.

2. Penentuan *range* / jangkauan masukan dan keluaran.

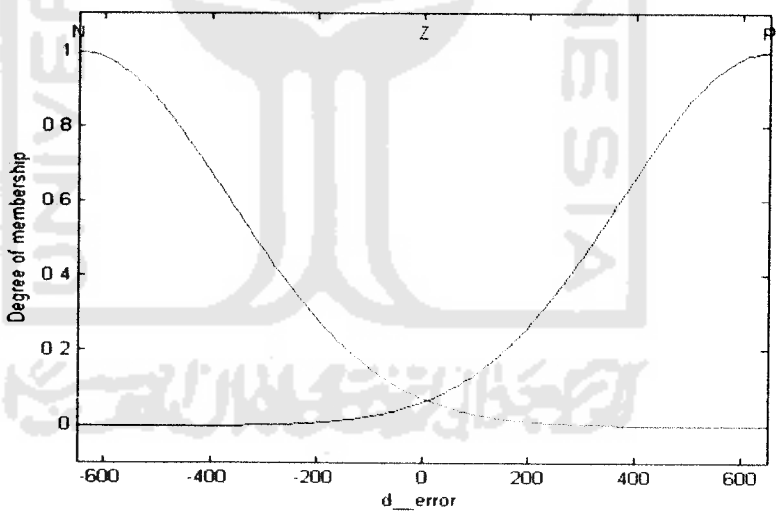
Jangkauan masukan dan keluaran ditentukan berdasarkan nilai maksimal dan minimal (*error* dan *d_error*) yang diperoleh saat simulasi. Keluaran dari kendali *fuzzy* merupakan besarnya perubahan tegangan. Untuk jangkauannya menyesuaikan dengan batas maksimal dari tegangan yang masuk ke motor DC.

3. Penentuan fungsi keanggotaan masukan dan keluaran FLC.

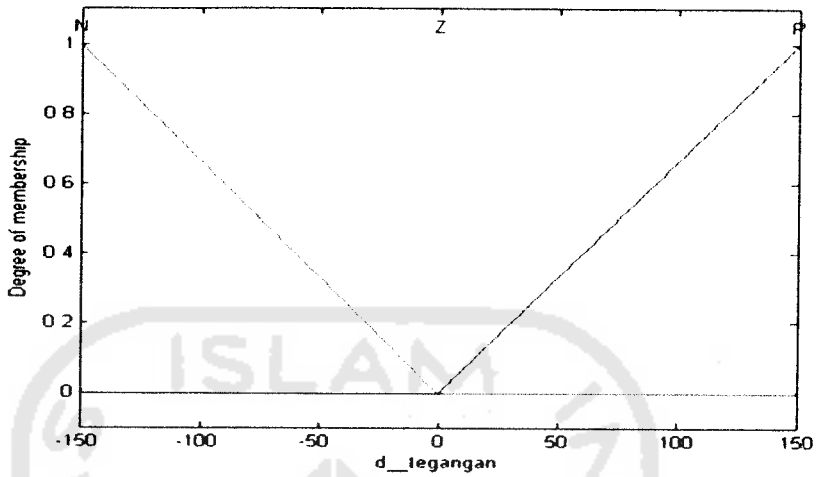
Parameter dari setiap fungsi keanggotaan masukan (*error* dan *d_error*) maupun keluaran, sesuai dengan ketentuan pada sistem automatisasi *fuzzy*. Fungsi keanggotaan masukan (*error* dan *d_error*) berupa fungsi keanggotaan *Gaussian* sedangkan fungsi keanggotaan keluaran berupa fungsi keanggotaan segitiga.



Gambar 3.6 Fungsi keanggotaan *error*



Gambar 3.7 Fungsi keanggotaan *d_error*.



Gambar 3.8 Fungsi keanggotaan keluaran kendali *fuzzy*.

4. Penentuan basis aturan (*rule base*) *fuzzy*.

Proses pembuatan aturan dilakukan dengan menerapkan kemampuan manusia dalam mengendalikan suatu sistem kendali. Aturan-aturan logika *fuzzy* yang akan dipergunakan sangat tergantung pada respon sistem yang dikendalikan. Tidak ada rumusan pasti dalam menentukan aturan-aturan ini. Penentuan aturan *fuzzy* ini merupakan tahap akhir dari proses desain sistem *fuzzy*. Dalam penentuan aturan *fuzzy* sangat sulit bila dilakukan dengan menggunakan kata yang sebenarnya contoh (lambat, sedang, cepat). Untuk mempermudah pembuatan aturan maka (lambat, sedang, cepat) dinotasikan dengan (*negative, zero, positive*). Aturan - aturan *fuzzy* seperti yang terlihat pada tabel 3.1.

Tabel 3.1 Aturan *fuzzy* (*Rule Base*)

Nomor Aturan	Input		Output
	Error	D_error	
1	N	N	N
2	N	Z	N
3	N	P	Z
4	Z	N	N
5	Z	Z	Z
6	Z	P	P
7	P	N	Z
8	P	Z	P
9	P	P	P

P : *Positive*

Z : *Zero*

N : *Negative*

5. *Defuzzyfikasi.*

Pada perancangan *fuzzy* ini menggunakan tipe Mamdani. Tipe Mamdani yang digunakan pada proses ini adalah metode MOM (*Mean of Maximum*). Metode MOM menentukan aksi kendali yang mewakili nilai rata – rata (*mean*) dari aksi

kendali yang fungsi keanggotaannya mencapai maksimum. Metode MOM menghasilkan kondisi transien yang lebih baik pada respon sistem, daripada menggunakan metode *defuzzyfikasi* yang lain (*Centroid, Bisector*, LOM dan SOM).

3.3 Prosedur Pelatihan

. Prosedur pelatihan dari metode gradien dapat dilihat pada langkah-langkah sebagai berikut ini :

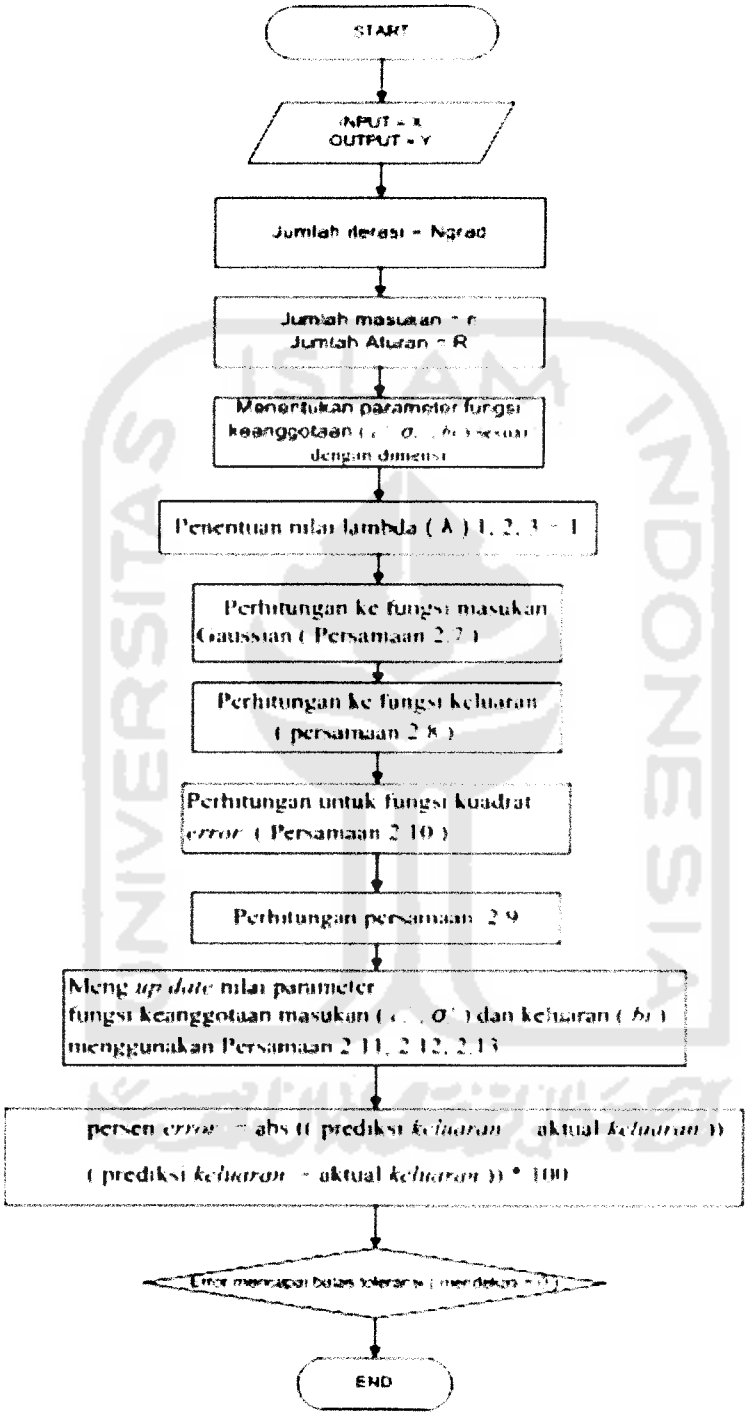
- Penentuan data masukan (X) dan data keluaran (Y) yang akan dilatih.
- Penentuan jumlah iterasi yang akan dilakukan (Ngrad).
- Penentuan jumlah masukan (n) dan jumlah aturan (R).
- Penentuan parameter fungsi keanggotaan masukan (c_j^i, σ_j^i) dan parameter fungsi keanggotaan keluaran (*bi*) sesuai dengan dimensi (vektor dan kolom).
- Penentuan nilai lambda (λ) 1, 2, 3 = 1.
- Perhitungan untuk fungsi masukan Gaussian (Persamaan 2.7)
- Perhitungan untuk fungsi keluaran delta (Persamaan 2.8)
- Perhitungan untuk fungsi kuadrat *error* (Persamaan 2.10)
- Perhitungan persamaan 2.9.
- Di *up date* nilai parameter fungsi keanggotaan masukan (c_j^i , σ_j^i) dan keluaran (*bi*).
    a. Persamaan 2.11 untuk *up date* pusat keanggotaan keluaran (*bi*).
    b. Persamaan 2.12 untuk *up date* pusat keanggotaan masukan (c_j^i).

c. Persamaan 2.13 untuk *up date spreads* keanggotaan masukan (σ_j^l).

- Nilai persen *error* dapat dihitung dengan persamaan berikut :

  persen *error* = abs ((prediksi keluaran – aktual keluaran)) / (prediksi keluaran + aktual keluaran)) * 100.

- Nilai *error* belum mendekati nol maka iterasi diperbesar. Setelah nilai *error* yang ditoleransi (mendekati nol) telah diperoleh maka pelatihan selesai dilakukan. Kemudian data hasil pelatihan di masukan ke dalam *fuzzy* yang baru.



Gambar 3.9 *Flowchart* Pelatihan