BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia industri, komputer dapat melakukan sebuah proses pemodelan terhadap sistem nyata / sebenarnya. Pemodelan adalah suatu bentuk tiruan, rekayasa dari sistem nyata yang dibuat dalam sebuah komputer dengan sebuah program. Hal ini merupakan tahapan sebelum pelaksanaan pengendalian pada sistem nyata. Keuntungan dengan memodelkan sebuah sistem nyata diantaranya :

1. Dapat mengetahui jalannya sebuah sistem nyata sebelum sistem itu dibuat
2. Dapat bereksperimen terhadap model yang akan dibuat
3. Dapat memperkecil *error* / kesalahan
4. Dapat menganalisa sebuah sistem nyata dari pemodelan sistem

Pemodelan dapat dibuat dalam komputer dengan *software* Matlab. Matlab adalah bahasa pemrograman dengan performansi tinggi untuk komputasi teknis. Beberapa kegunaan dari matlab ini adalah untuk matematika dan komputasi, pengembangan algoritma, pemodelan, simulasi, analisa data, eksplorasi, visualisasi, dan dapat membangun aplikasi.

Salah satu aplikasi yang terdapat dalam *software* Matlab adalah logika *fuzzy*. Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang input ke

dalam suatu ruang output. Antara input dan output terdapat suatu relasi yang dapat menghubungkan keduanya untuk saling bekerja menghasilkan suatu keputusan.

Untuk mengetahui sebuah sistem, sangat bergantung pada informasi dan banyaknya pengetahuan yang diperolch dari penelitian atau hasil percobaan untuk mengembangkan sebuah model dan untuk memprediksi keluaran. Tetapi untuk sistem yang baru, sedikitnya pengetahuan dan penelitian seorang analisis merupakan keterbatasan untuk mengembangkan sebuah model dengan menggunakan sistem *fuzzy* konvensional. Pada situasi seperti ini, sistem automatisasi *fuzzy* sangat praktis dan dapat digunakan untuk mengembangkan model untuk sistem dengan keterbatasan informasi yang tersedia.

Salah satu aplikasi pemodelan sistem adalah pada motor DC. Motor DC (motor arus searah) banyak digunakan dalam kehidupan sehari-hari. Baik dalam dunia industri maupun rumah tangga. Berdasarkan karakteristiknya, motor arus searah mempunyai daerah pengaturan putaran yang luas, sehingga sampai sekarang masih banyak digunakan pada pabrik-pabrik yang mesin produksinya memerlukan pengaturan putaran yang luas. Motor-motor yang digunakan di dunia industri akan lebih menghasilkan produk yang bagus dan memiliki tingkat ketelitian yang tinggi apabila kesalahan dari faktor manusia dapat diperkecil.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan diatas, maka dapat diambil suatu rumusan masalah sebagai berikut : "Bagaimana merancang FLC dengan metode gradien untuk mengendalikan motor DC".

1.3 Batasan Masalah

Dengan adanya batasan masalah, penulis dapat lebih menyederhanakan dan mengarahkan penelitian dan pembuatan sistem agar tidak menyimpang dari apa yang diteliti dan dikembangkan. Batasan-batasannya adalah sebagai berikut :

1. Pada Tugas Akhir ini, perancangan dan simulasi sistem dibuat pada perangkat lunak Matlab R2007a.
2. Membuat model sistem motor DC pada *simulink* berdasarkan karakteristik motor DC sebenarnya yang terdapat pada Laboratorium Instalasi dan Mesin Listrik dengan perhitungan matematis.
3. Pelatihan menggunakan sistem automatisasi *fuzzy* dengan Metode Gradien.

1.4 Tujuan Penelitian

Adapun Tujuan dari penelitian dan perancangan sistem ini adalah :

1. Merancang dan mensimulasikan sebuah sistem pengendalian kecepatan motor DC dengan menggunakan Sistem Automatisasi *Fuzzy* dengan Metode Gradien.

2. Mempelajari, mendesain, dan menganalisa sistem *fuzzy* menggunakan *simulink* pada Matlab sebagai media pelatihan.
3. Dapat memahami dengan jelas dan benar tentang konsep logika *fuzzy* yang digunakan sebagai media kontrol dalam suatu aplikasi kendali.
4. Memperoleh nilai *error* yang kecil dari sistem.

1.5 Sistematika Penulisan

Sistematika penulisan laporan dari Tugas Akhir akan dibagi dalam lima bab, dengan isi masing - masing bab diuraikan sebagai berikut :

BAB I PENDAHULUAN

Berisi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, dan sistematika penulisan.

BAB II STUDI PUSTAKA

Menguraikan tentang teori-teori yang menjadi acuan dalam pembuatan tugas akhir, diantaranya teori logika *fuzzy*, teori sistem automatisasi *fuzzy* metode gradien, dan teori motor DC.

BAB II

STUDI PUSTAKA

2.1 Tinjauan Pustaka

Tugas akhir ini memiliki persamaan dari penelitian atau tugas akhir sebelumnya. Perbedaan yang utama terdapat pada metode untuk mengendalikan motor DC. Tugas akhir sebelumnya dengan judul "Simulasi Kendali Kecepatan Motor DC berbasis Algoritma ANFIS", yang dikerjakan oleh Muhammad Rifky Indriarto dan tugas akhir yang berjudul "Simulasi Jaringan Syaraf Tiruan Berbasis Metode *Back Propagation* sebagai Pengendali Kecepatan Motor DC", yang dikerjakan oleh Romy Wiryadinata.

Pelatihan menggunakan 150676 data. Proses pelatihan membutuhkan waktu yang sangat lama dalam hitungan jam. Namun dengan menggunakan metode tersebut, proses dapat dipersingkat dan menghasilkan *error* / galat yang kecil. Pada proses pelatihan, data yang digunakan diseleksi agar pelatihan tidak terlalu berat dan memakan waktu yang lama, dan juga dapat memperkecil *error*.

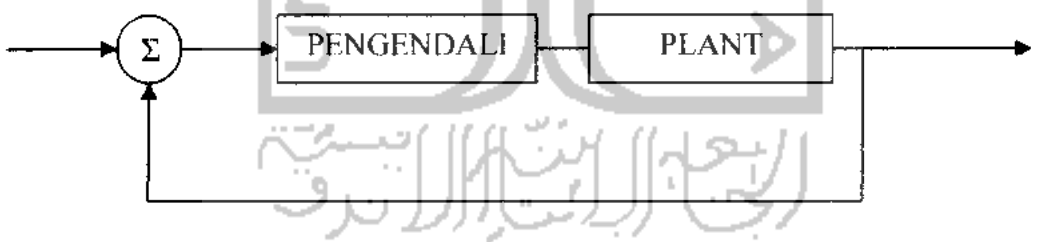
Pengujian sistem menggunakan blok simulink Matlab. Pada blok motor DC menggunakan blok *DC machine*. Parameter dari motor DC diperoleh dari motor DC sebenarnya yang terdapat pada Laboratorium Instalasi dan Mesin Listrik. Masukan ANFIS dan JST adalah kecepatan sedangkan keluarannya adalah tegangan. Setelah dilakukan pengujian sistem, menghasilkan sistem yang cukup baik. Secara keseluruhan keluaran motor dapat mengikuti *setpoint* dan dapat menghasilkan selisih rata – rata kecepatan yang kecil. Pada JST sebesar 4.14 rad / s. Sedangkan pada ANFIS sebesar 1.4 rad / s.

2.1.1 Analisis Tinjuan Pustaka

Perbedaan dari tugas akhir ini dengan tugas akhir sebelumnya terdapat pada metode yang digunakan dalam pengendalian motor DC dan blok *simulink* yang digunakan. Pada tugas akhir sebelumnya, motor DC tetap menggunakan blok motor DC pada simulink. Sedangkan pada tugas akhir ini, blok motor DC diganti dengan blok *transfer function* yang diperoleh secara perhitungan matematis dan analisa.

Penelitian terhadap sistem kendali motor DC yang dilakukan sebelumnya, akan dijadikan sebagai acuan dalam menentukan blok simulink yang digunakan untuk proses pelatihan dan proses pengujian sistem pada tugas akhir ini.

2.2 Sistem Kendali

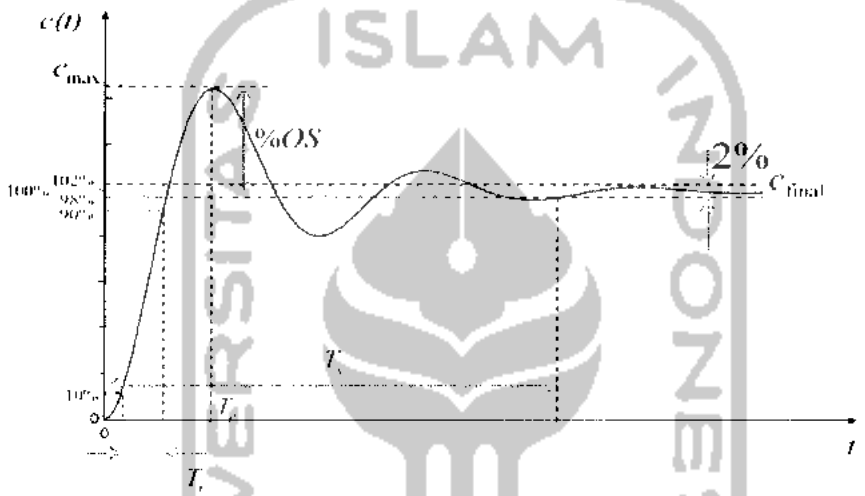


Gambar 2.1 Blok diagram Sistem Kendali

Pengendali dapat didefinisikan sebagai suatu komponen dalam sistem yang berfungsi mengontrol sinyal *error* (selisih dari *setting point* dengan keluaran *feedback*) menjadi sinyal kontrol sehingga respon keluaran pada keadaan mantap mendekati *setting point/ error steady state* minimal. Sedangkan *Plant* adalah

seperangkat peralatan, mungkin hanya terdiri dari beberapa mesin yang bekerja bersama, yang digunakan untuk melakukan suatu operasi tertentu.

Pada sistem kendali jika dijalankan akan menghasilkan gelombang keluaran atau respon sistem seperti pada gambar 2.2.



Gambar 2.2 Respon Sistem Kendali

Dari gambar 2.2 dapat diketahui bahwa waktu (t) adalah dalam satuan detik Matlab, bukan satuan detik dalam keadaan sebenarnya. Sedangkan untuk memudahkan mengetahui maksud gambar 2.2, berikut ini adalah penjelasannya:

- Waktu tunda (*delay time*), td : adalah waktu yang diperlukan oleh respon untuk mencapai setengah nilai akhir untuk waktu yang pertama.
- Waktu naik (*rise time* = tr) adalah waktu yang diperlukan oleh tanggapan untuk naik dari 0 % menjadi 100 % dari nilai akhir.

- Waktu turun (*settling time* = ts) adalah waktu yang diperlukan untuk menanggapi kurva agar dapat mencapai dan tetap berada dalam persentase nilai akhir tertentu dan biasanya digunakan batasan 5 %.
- Maksimum o*vershoot* (Mp) adalah nilai puncak kurva tanggapan diukur dari satuan waktu, digunakan untuk mengukur kestabilan relatif dari sistem.
- Waktu puncak (*peak time* = tp) adalah waktu yang diperlukan tanggapan untuk mencapai puncak pertama kali.

2.3 Logika *Fuzzy*

Logika *fuzzy* pertama kali diperkenalkan oleh ilmuan Amerika, Lotfi A. Zadeh, pada tahun 1965 ketika mempublikasikan papernya berjudul "*Fuzzy Sets*". Zadeh menunjukkan bahwa logika *fuzzy* adalah dasar bagi logika lainnya. Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan ruang masukan ke dalam ruang keluaran, dengan prinsip logika *fuzzy* mencoba menjawab keterbatasan yang dimiliki oleh struktur logika biner bolean yang hanya memiliki dua kondisi pernyataaan yaitu benar (*true*) atau salah (*false*).

Himpunan *fuzzy* mempunyai batas yang dapat berpindah, artinya elemen dari himpunan *fuzzy* tidak hanya merepresentasikan "hitam" dan "putih", namun juga warna *gray* diantara kedua warna tersebut, atau dengan kata lain bahwa logika *fuzzy* mencoba untuk menjembatani kondisi yang tidak hanya bisa diselesaikan dengan

pernyataan ya atau tidak dan logika *fuzzy* juga mendeskripsikan kondisi-kondisi pertengahan, kondisi diantara situasi ya dan tidak ke dalam formulasi matematis.

Himpunan *fuzzy* berbeda dengan himpunan tegas (*crisp*), yang mana himpunan *crisp* nilai keanggotaan dalam suatu himpunannya hanya memiliki 2 kemungkinan, yaitu 0 atau 1. Apabila x memiliki nilai keanggotaan *fuzzy* $\mu_A[x]=0$ berarti x tidak menjadi anggota himpunan A, demikian pula apabila x memiliki nilai keanggotaan *fuzzy* $\mu_A[x]=1$ berarti x menjadi anggota penuh pada himpunan A.

Himpunan *fuzzy* berbeda dengan himpunan *crips* karena satu nilai pernyaatan dalam himpunan *fuzzy* bisa berada dalam dua keadaan yang berbeda. Himpunan *fuzzy*, di sisi lain memperkenalkan *vagueness* (ketidakjelasan / samar) dengan mengeliminasi batas yang tajam yang memisahkan anggota dan bukan anggota pada himpunan klasik. Jadi, transisi antara anggota penuh dan bukan anggota bersifat gradual dan bukan tajam. Sehingga himpunan *fuzzy* dapat dilihat sebagai ekstensi dan generalisasi konsep dasar himpunan klasik, meskipun beberapa teori bersifat unik dan berlaku pada himpunan *fuzzy* saja.

Perbedaan himpunan klasik dengan himpunan *fuzzy* dapat diuraikan dengan jelas pada permasalahan berikut. Misalkan U adalah garis riil R dan himpunan klasik A mewakili " bilangan riil lebih besar atau sama dengan 5 " maka diperoleh :

$$A = \{(x, \mu_A(x)) \mid x \in U\} \qquad (2.1)$$

yang mana fungsi karakteristiknya adalah :

$$\mu_A(x) = \begin{cases} 1 & , x \geq 5 \\ 0 & , x < 5 \end{cases} \qquad (2.2)$$

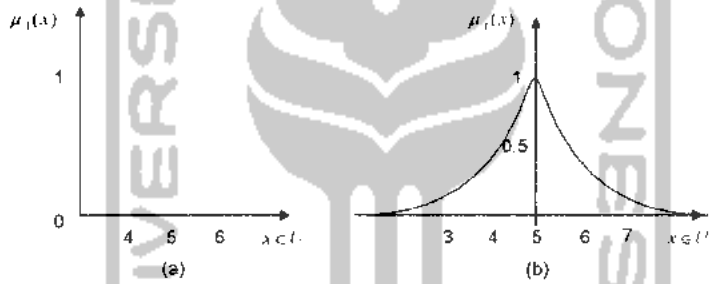
yang ditunjukkan oleh gambar 2.3 (a).

Misalkan himpunan *fuzzy* $\widetilde{A}$ mewakili " bilangan riil yang mendekati 5" maka diperoleh $\widetilde{A} = \{(x, \mu_{\widetilde{A}}(x)) \mid x \in U\}$

yang mana fungsi karakteristiknya adalah :

$$\mu_{\widetilde{A}}(x) = \frac{1}{1+10(x-5)^2} \qquad (2.3)$$

yang ditunjukkan oleh gambar 2.3 (b).

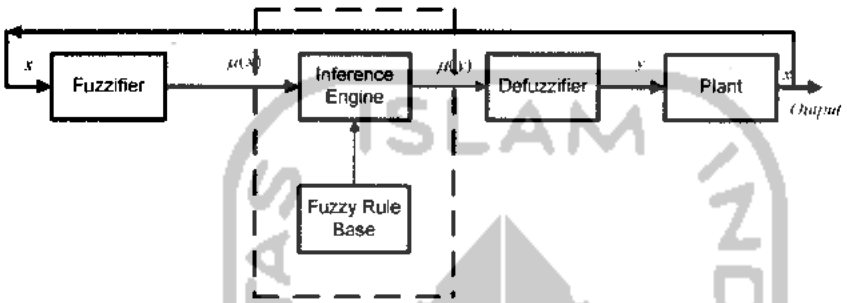


Gambar 2.3 Fungsi karakteristik himpunan klasik A dan himpunan *fuzzy* $\widetilde{A}$

2.4 Sistem Kendali Logika *Fuzzy*

Selama beberapa dekade yang lalu, *fuzzy logic control* (FLC), yang pertama kali diperkenalkan oleh Mamdani dan Assilian [1975], telah berkembang menjadi bidang penelitian yang aktif dan menjanjikan sebagai aplikasi teori himpunan *fuzzy*, logika *fuzzy*, dan *fuzzy reasoning*. Aplikasi tersebut tersebar mulai dari kendali proses industri sampai dengan diagnosa medis dan *securities trading*. Berbeda dengan sistem kendali konvensional, FLC lebih tepat digunakan pada sistem yang sulit

didefinisikan (*ill-defined*), yang dapat dikendalikan dengan operator manusia dengan tanpa mengetahui sifat dinamis dalam sistem tersebut. Struktur dan operasi sistem kendali logika *fuzzy* diperlihatkan pada gambar 2.4.



Gambar 2.4 Struktur sistem kendali logika *fuzzy*

Pada dasarnya FLC terdiri atas empat bagian utama yaitu *fuzzifier*, *fuzzy rule base*, *inference engine* dan *defuzzifier*.

2.4.1 Ruang Masukan dan Keluaran

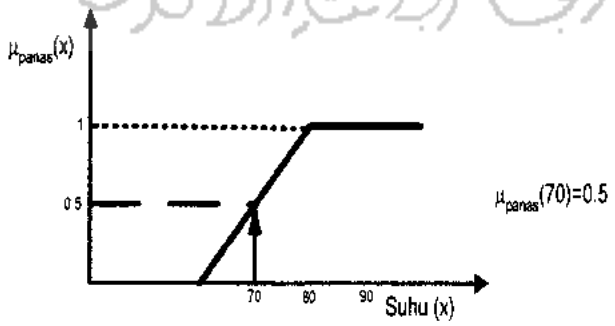
Tujuan dari sistem pengendali *fuzzy* adalah untuk menghitung nilai variabel kendali atau aksi dari variabel keadaan hasil observasi atau pengukuran proses yang dikendalikan sedemikian rupa sehingga diperoleh hasil yang diinginkan. Sehingga, pemilihan secara tepat dari variabel keadaan dan juga variabel kendali sangat penting dalam menentukan karakteristik operasi FLC dan mempunyai pengaruh yang esensial terhadap unjuk kerja FLC. Pengalaman pakar dan pengetahuan teknisi memegang peranan penting dalam proses seleksi variabel keadaan dan variabel kendali. Pada umumnya, Masukan ke FLC merupakan keadaan (*state*), *error* keadaan

(*error state*), turunan *error* keadaan (*state error derivative*), integral *error* keadaan (*state error integral*), dan lain sebagainya.

2.4.2 *Fuzzifier*

Fuzzifier melakukan proses fuzzifikasi yaitu mengubah nilai variabel numerik ke nilai variabel linguistik. Dengan kata lain, fuzzifikasi merupakan pemetaan dari ruang masukan ke himpunan *fuzzy* yang didefinisikan pada semesta pembicaraan variabel masukan.

Proses fuzzifikasi dapat diilustrasikan dengan contoh berikut : Misalkan Plant berupa sistem pemanas air sehingga variabel Masukan ke FLC yaitu x merupakan suhu terukur. Suhu x yang berupa variabel numerik mempunyai interval nilai 15^0 hingga 100^0. Variabel suhu dinyatakan sebagai variabel linguistik dengan lima nilai linguistik yaitu dingin, agak dingin, sedang, agak panas, dan panas. Misalkan sensor suhu menghasilkan pembacaan data 70^0, maka proses fuzzifikasi nilai numerik suhu 70^0 pada nilai linguistik panas dapat diilustrasikan pada gambar 2.5.



Gambar 2.5 Proses fuzzifikasi

2.4.3 *Fuzzy Rule Base*

Fuzzy rule base berisi kumpulan pengetahuan empiris yang dimiliki seorang operator pakar tentang proses operasi suatu sistem. Aturan kendali *fuzzy* dinyatakan dengan kumpulan aturan IF-THEN yang mana prekondisi dan konsekuennya berupa variabel linguistik. Kumpulan aturan kendali *fuzzy* tersebut merupakan relasi masukan-keluaran dari sebuah sistem. Bentuk umum dari aturan kendali *fuzzy* pada sistem banyak masukan satu keluaran (MISO) adalah :

$$R^l : \text{IF x is } A_i \text{ ..... AND y is } B_i \text{ THEN z is } C_i \qquad (2.4)$$

yang mana *x* dan *y* merupakan variabel linguistik masukan yang merepresentasikan variabel keadaan sistem (*state variabel*) sedangkan *z* merupakan variabel linguistik keluaran yang merepresentasikan variabel kendali (*control variabel*). A_i, ... B_i, dan C_i berturut-turut merupakan nilai linguistik variabel *x*, ..., *y*, dan *z* pada semesta U, ... , V, dan W. Persamaan di atas merupakan sistem *fuzzy* tipe Mamdani yang digunakan dalam perancangan sistem dalam tugas akhir ini.

Aturan-aturan IF-THEN yang ada akan membentuk memory asosiatif *fuzzy* (*fuzzy assosiative memory*, FAM). Perlu diingat bahwa tidak semua kombinasi aturan IF-THEN akan masuk dalam FAM, tetapi hanya aturan-aturan yang mungkin dieksekusi saja yang masuk dalam FAM. Aturan yang menggambarkan keadaan yang tidak mungkin terjadi tidak dimasukkan. Demikian pula aturan yang dapat diabaikan karena telah ada aturan lain yang serupa tidak dimasukkan ke dalam FAM, misalnya :

aturan 1 : jika x = A1 dan y = B1 maka z = C1

aturan 2 : jika x = A1 maka z = C1 .

pada keadaan ini, aturan 1 dapat diabaikan dan karena itu tidak perlu dimasukkan ke FAM.

FAM ini berupa matriks yang menyatakan hubungan masukan - keluaran sesuai dengan aturan IF-THEN yang ada. Yang dimaksud dengan masukan di sini adalah masukan yang telah melalui proses fuzifikasi, sehingga sudah dalam bentuk nilai linguistik dengan derajat keanggotaan tertentu. Keluarannya pun dalam bentuk nilai linguistik.

2.4.4 *Inference Engine*

Inference engine merupakan inti dari FLC dalam memodelkan cara berpikir manusia dalam konsep logika *fuzzy* dan *approximate reasoning* yang memegang peranan penting dalam proses inferensi ini. Terdapat empat tipe operator komposisi yang bisa digunakan pada aturan komposisi inferensi yaitu :

- *Max-min*
- *Max-product*
- *Max bounded product*
- *Max drastic product*

Pada FLC, operator komposisi *max-min* dan *max-product* paling banyak digunakan dan paling umum karena perhitungannya sederhana dan efisien.

2.4.5 *Defuzzifier*

Defuzzifikasi merupakan pemetaan dari ruang aksi kendali *fuzzy* yang didefinisikan pada semesta pembicaraan keluaran ke ruang aksi kendali *nonfuzzy* (numerik). Proses ini penting karena pada aplikasi praktis aksi kendali numerik diperlukan untuk melakukan pengendalian. Sehingga *defuzzifier* diperlukan jika sistem *fuzzy* yang digunakan adalah tipe Mamdani.

Strategi defuzzifikasi membantu menemukan aksi kendali *nonfuzzy* yang paling baik dalam mewakili distribusi peluang aksi kendali *fuzzy* hasil inferensi. Hanya saja, tidak terdapat satupun prosedur yang sistematik dalam memilih strategi defuzzifikasi tersebut. Salah satu metode defuzzifikasi yang umum digunakan adalah *mean of maximum* (MOM) dan diaplikasikan pada perancangan kendali *fuzzy* untuk sistem pegendalian motor DC. Metode MOM menghasilkan kondisi transien yang lebih baik pada respon sistem. Metode MOM menentukan aksi kendali yang mewakili nilai rataan (*mean*) dari aksi kendali lokal yang fungsi keanggotaannya mencapai maksimum.

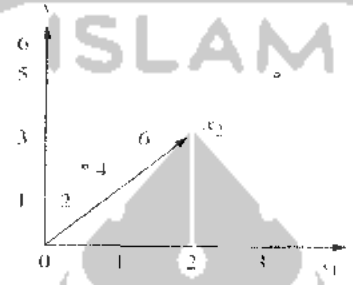
Masukan dari proses defuzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *keluaran* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka harus dapat diambil suatu nilai *crisp* tertentu sebagai *keluaran*.

2.5 Sistem Automatisasi *Fuzzy*

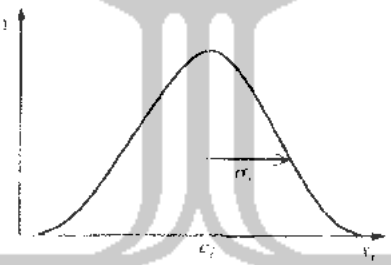
Untuk mengetahui sebuah sistem, analisis sangat bergantung pada informasi dan banyaknya pengetahuan yang diperoleh dari penelitian atau hasil percobaan untuk mengembangkan sebuah model dan untuk memprediksi keluaran. Tetapi untuk sistem yang baru, sedikitnya pengetahuan dan penelitian seorang analisis merupakan keterbatasan untuk mengembangkan sebuah model dengan menggunakan sistem konvensional. Pada situasi seperti ini, pemodelan *fuzzy* sangat praktis dan dapat digunakan untuk mengembangkan model untuk sistem dengan keterbatasan informasi yang tersedia. *Batch Least Squares, Recursive Least Squares, Gradient Methode, Learning from Example, Modified Learning From Example,* dan *Clustering Methode* adalah beberapa algoritma yang mengembangkan sebuah model *fuzzy*. Enam metode tersebut merupakan metode automatisasi *fuzzy*. Gambaran dari enam metode automatisasi dijelaskan dengan *software* Matlab. Pada tulisan ini hanya 2 masukan dan 1 keluaran yang diilustrasikan tetapi algoritma tersebut dapat digunakan untuk sistem dengan banyak masukan satu keluaran dan bahkan untuk sistem dengan banyak masukan banyak keluaran. Sebagai contoh dari 2 masukan 1 keluaran sistem diilustrasikan pada gambar 2.6, ada tiga titik, masukannya adalah x_1, x_2, keluarannya adalah y. Kebanyakan dari algoritma menggunakan fungsi keanggotaan Gaussian untuk masukan $\mu(x)$,

$$\mu(x) = \exp\left[-\frac{1}{2}\left(\frac{x_i - c_i}{\sigma_i}\right)^2\right] \qquad (2.5)$$

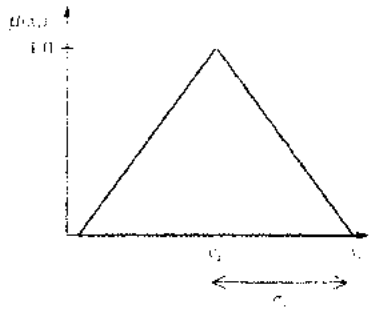
x_i adalah variabel masukan, c_i adalah pusat fungsi keanggotaan (fungsi keanggotaan mencapai nilai maksimum), σ_i adalah relasi konstan penyebaran fungsi keanggotaan. Gambar 2.7 ilustrasi tipe fungsi keanggotaan gaussian dan parameternya.



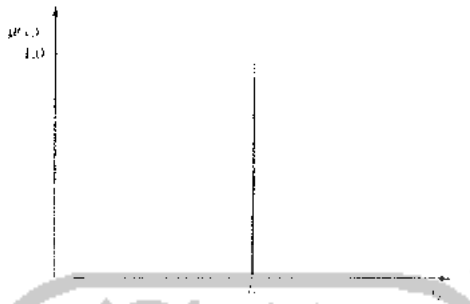
Gambar 2.6 Contoh dari 2 masukan, 1 keluaran untuk 3 titik



Gambar 2.7 Fungsi keanggotaan Gaussian



Gambar 2.8 Fungsi keanggotaan Segitiga



Gambar 2.9 Fungsi keanggotaan Delta

Pada kebanyakan algoritma, fungsi keanggotaan segitiga banyak digunakan sebagai fungsi keanggotaan keluaran sistem seperti pada gambar 2.8. Sedangkan pada gambar 2.9, merupakan gambar fungsi keanggotaan delta. Dimana b_i adalah nilai puncak fungsi keanggotaan dan nilai yang lain adalah nol.

Enam metode automatisasi mengembangkan basis aturan atau menggunakan basis aturan sebelumnya untuk memprediksi keluaran berdasarkan masukannya. Sebuah aturan terdiri dari klausa premis dan konsekuen. Contoh aturan yaitu :

IF *premise*$_1$ and *premise*$_2$ THEN *consequence*

Aturan itu dikembangkan oleh algoritma untuk memprediksi dan menentukan keluaran sistem dari nilai masukan yang diberikan. Pada algoritma *Batch Least Squares* (BLS), *Recursive Least Squares* (RLS) dan Metode Gradien, basis aturan ini harus dispesifikasikan oleh pengguna algoritma dari prosedur automatisasi. Akan tetapi Metode Gradien mampu meng *up date* parameter dari basis aturan (parameter dari fungsi keanggotan). Pada Metode *Clustering* (CM) dan *Modified Learning From Example* (MLFE), basis aturan dibentuk dari masukan dan keluaran sistem,

yang digunakan untuk memodelkan sistem. Algoritma *Learning From Example* (LFE) menggunakan semua fungsi keanggotaan, namun hanya membangun aturan - aturan berdasarkan basis aturan. Oleh karena itu, beberapa algoritma dapat digunakan secara bersama untuk memperbaiki dan memperhalus sistem.

Sebagai contoh, dari enam algoritma tersebut menggunakan fungsi keanggotaan Gaussian untuk masukan dan fungsi keanggotaan segitiga untuk keluaran. Berikut ini merupakan persamaan untuk memprediksi keluaran berdasarkan masukan data x_j :

$$f(x10) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left[-\frac{1}{2}\left(\frac{x_i - c^i_j}{\sigma^i_j}\right)^2\right]}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left[\frac{-1}{2}\left(\frac{x_j - c^i_j}{\sigma^i_j}\right)^2\right]} \qquad (2.6)$$

Keterangan:

R = jumlah aturan

n = jumlah masukan

c = masukan *membership function center*

j = nomor masukan

σ = *respective spread*

2.6 Metode Gradien

Metode gradien merupakan salah satu dari enam metode automatisasi *fuzzy*. Metode Gradien mampu memprediksi keluaran berdasarkan data pelatihan yang diperbaiki dan mampu memodifikasi parameter masukan. Tidak hanya itu, metode ini menyediakan cara untuk mengatur parameter dari model *fuzzy* (parameter basis aturan).

Berikut ini merupakan ilustrasi penggunaan Metode Gradien berdasarkan pada Tabel 2.1.

Tabel 2.1 Data Pelatihan Z = {([x_1, x_2], y)}

x_1	x_2	y
0	2	1
2	4	5
3	6	6

Data Z dapat digunakan sebagai data pelatihan yang digunakan untuk model *fuzzy*. Data Z terdiri dari 2 variabel, yaitu variabel masukan dan variabel keluaran. Variabel masukan = x_i dan x_2 (n = 2) dan variabel keluaran = y . Masing – masing variabel mempunyai 3 data, m = 3. Setelah itu mendesain jumlah aturan (2 aturan, R = 2) dan parameter aturan. Konsekuen dari masing – masing aturan dinotasikan sebagai pusat fungsi keanggotaan (b_1 dan b_2). Dua premis dari masing – masing

aturan dinotasikan sebagai pusat fungsi keanggotaan masukan (c_i) dan *respective spread* (σ_i). :

IF *premise*$_1$ and *premise*$_2$ THEN *consequence*

Premis dan konsekuen dari basis aturan diperoleh berdasarkan masukan. Pusat fungsi keanggotaan masukan c_j^i, dimana i adalah nomor aturan dan j adalah nomor masukan :

$$c_1^1 = 1.5$$
$$c_1^2 = 3$$
$$c_2^1 = 3$$
$$c_2^2 = 5$$

Rule 1 : if x_1 "about 1.5" and x_2 is about 3" then b_1 is 1.

Rule 2 : if x_1 "about 3" and x_2 is about 5" then b_1 is 5.

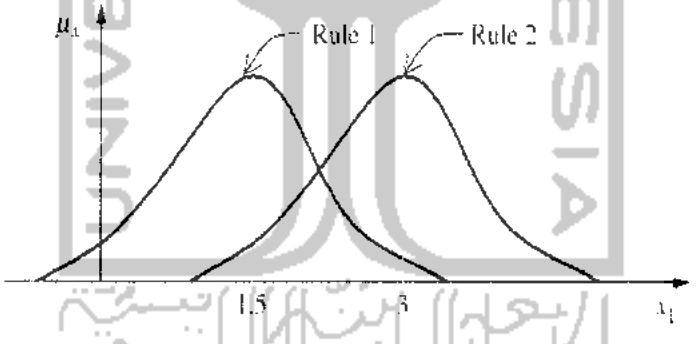
Langkah selanjutnya adalah memilih *respective spread,* σ_j^i, untuk fungsi keanggotaan yang di seleksi. Sebagai contoh $\sigma_j^i = 2$. untuk i =1, 2 dan j = 1, 2. Fungsi keanggotaan masukan untuk *Rule 1* dan *Rule 2* adalah fungsi keanggotaan Gaussian dan ditampilkan pada Gambar 2.10 dan 2.11. Fungsi keanggotaan keluaran adalah fungsi keanggotaan delta seperti pada Gambar 2.12. Nilai keanggotaan dapat dihitung dari masing – masing masukan data pada basis aturan. Hasil dari nilai keanggotaan pada masukan data merupakan aturan yang diteliti. Berikut adalah persamaannya :

$$\mu_i(x) = \prod_{j=1}^{n} \exp\left[-\frac{1}{2}\left(\frac{x - c_j^i}{\sigma_j^i} \right)^2 \right] \qquad (2.7)$$

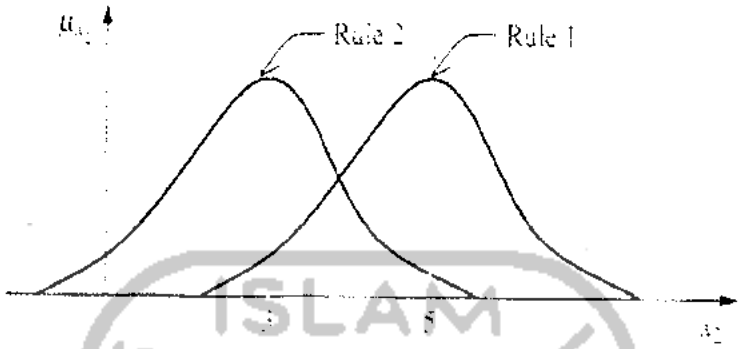
Dimana n = 2, x adalah masukan data dan c_j^i dan μ_j^i adalah parameter basis aturan.

Setelah direduksi, persamaan 2.6 dapat ditulis ulang sebagaimana persamaan 2.8.

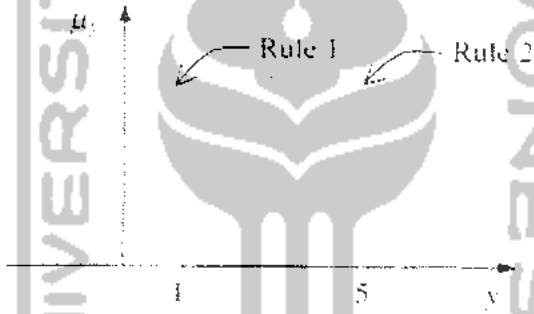
$$f(x \mid \theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \qquad (2.8)$$



Gambar 2.10. Fungsi keanggotaan masukan untuk x_1



Gambar 2.11. Fungsi keanggotaan Masukan untuk x_2



Gambar 2.12. Fungsi keanggotaan keluaran untuk y

Penggunaan data pelatihan dari Tabel 2.1 dapat digunakan sebagai ilustrasi untuk mengembangkan model *fuzzy* dengan Metode Gradien. Setelah aturan dibuat, Metode Gradien mampu mengatur parameter yang menghubungkan dengan basis aturan. Oleh karena itu data yang digunakan pada data pelatihan sangat penting untuk mencapai nilai yang diinginkan. Pada metode ini memiliki tujuan untuk memperkecil nilai *error* (e^m) antara nilai keluaran yang diprediksi f ($x^m|0$), dengan nilai keluaran aktual (y^m), maka fungsi kuadrat *error* e^m, dinamakan " *error surface* ".

Persamaan untuk *error surface* sebagaimana pada persamaan 2.9.

$$e^m = \frac{1}{2} [f(x^m|\theta) - y^m]^2 \qquad (2.9)$$

m = merupakan jumlah data masukan dan keluaran pada data pelatihan. Untuk mendapatkan nilai minimum pada " *error surface* ", ditentukan ketika model mencapai nilai yang diprediksi. Kemudian data pelatihan di *up date* berdasarkan parameter basis aturan untuk mengurangi perbedaan antara keluaran yang diprediksi dan keluaran aktual. Persamaannya adalah :

$$\varepsilon_m = f(x^m|\theta) - y^m \qquad (2.10)$$

Data pelatihan dapat dilatih dengan *step* ((k), k = 0,1,2,..)) dengan memodifikasi parameter aturan, untuk menurunkan ε_m dan memperoleh sistem *fuzzy* yang diperbaiki. Metode Gradien juga menggunakan step size λ untuk tiga parameter yang diatur (bi, c_j^i, σ_j^i), yang digunakan untuk memperoleh *up date* parameter aturan dan mengurangi nilai *error*. Masing – masing untuk nilai λ pada pusat keanggotaan keluaran (bi), pusat keanggotaan Masukan (c_j^i), dan *spreads* keanggotaan masukan (σ_j^i) bernilai 1, karena digunakan untuk mempermudah perhitungan. Untuk memperoleh nilai parameter yang baru dari basis aturan, digunakan persamaan (2.11), (2.12), (2.13).

Persamaan untuk *up date* pusat keanggotaan keluaran (*bi*) :

$$b_i(k) = b_i(k-1) - \lambda(\varepsilon_k(k-1))\frac{\mu_i(x^k, k-1)}{\sum_{i=1}^{R}\mu_i(x^k, k-1)} \qquad (2.11)$$

Persamaan untuk *up date* pusat keanggotaan masukan (c_j^i) :

$$c_j^i(k) = c_j^i(k-1) - \lambda_2\varepsilon_k(k-1)\left(\frac{b_i(k-1) - f(x^k|\theta(k-1))}{\sum_{i=1}^{R}\mu_i(x^k, k-1)}\right) * \mu_i(x^k, k-1)\left(\frac{x_j^k - c_j^i(k-1)}{(\sigma_j^i((k-1))^2}\right) \qquad (2.12)$$

Persamaan untuk *up date spreads* keanggotaan masukan (σ_j^i) :

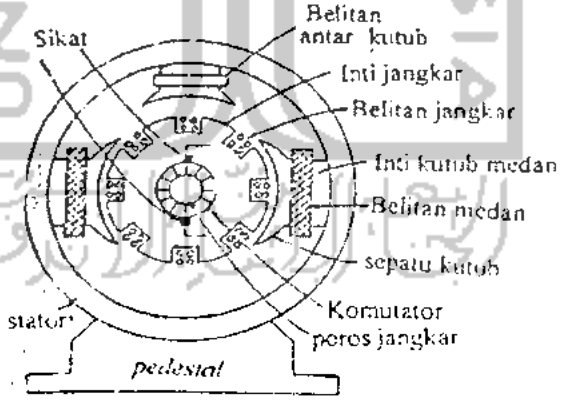
$$\sigma_j^i(k) = \sigma_j^i(k-1) - \lambda_3 * \varepsilon_k(k-1) * \left(\frac{b_i(k-1) - f(x^k|\theta(k-1))}{\sum_{i=1}^{R}\mu_i(x^k, k-1)}\right) * \mu_i(x^k, k-1) * \left(\frac{(x_j^k - c_j^i(k-1))^2}{(\sigma_j^i((k-1))^3}\right) \qquad (2.13)$$

2.7 Motor DC

Motor DC adalah suatu sistem mesin yang berfungsi mengubah tenaga listrik arus searah (listrik DC) menjadi gerak atau tenaga mekanis. Motor arus searah atau motor DC hampir dapat dijumpai di setiap peralatan baik rumah tangga, kendaraan bahkan dalam dunia industri sekalipun, dari yang berukuran mikro sampai dengan motor-motor yang memiliki kekuatan ribuan daya kuda.

Antara motor arus searah dengan generator arus searah tidak ada perbedaan konstruksi. Pada prinsipnya motor arus searah dapat digunakan sebagai generator arus searah, begitu juga sebaliknya.

2.7.1 Konstruksi Motor DC



Gambar 2.13. Kontruksi Mesin DC

Pada motor DC terlihat bahwa kumparan jangkar pada rotor (bagian yang berputar) dan kumparan medan pada stator (bagian yang tidak berputar / diam).

Rotor terdiri dari :

- Poros jangkar : bagian yang membawa inti jangkar, kumparan jangkar, dan komutator untuk ikut berputar.
- Inti jangkar, terbuat dari plat baja yang masing – masing dilapisi isolator.
- Kumparan jangkar, masing – masing kumparan terisolasi dari yang lain, terletak pada alur ("*slot*") dan terhubung secara elektrik dengan komutator.

  Ada dua macam bentuk :

  a. *Ring winding*
  b. *Drum winding*: - gelung ("*lap*")
     - gelombang ("*wave*")
- Komutator, terbuat dari segmen – segmen tembaga yang masing – masing diisolasi dengan bahan mika.
- Sikat, terbuat dari karbon dan grafit.

Stator terdiri dari :

- Gandar ("*yoke*") merupakan kerangka baja.
- Kutub, terdiri dari :
  - Inti kutub, merupakan plat baja yang dilapisi isolator
  - Sepatu kutub
  - Kumparan kutub (kutub utama dan kutub bantu)

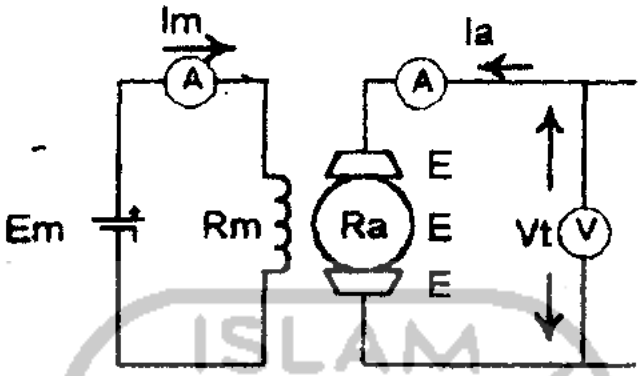
2.7.2 Prinsip Motor DC

Motor DC adalah segulung kawat yang dialiri arus listrik dan di tempatkan di dalam suatu medan magnet. Akibatnya, gulungan kawat ini akan mengalami suatu gaya yang sebanding dengan arus dan kekuatan medan magnetnya. Arah gaya membentuk sudut siku-siku terhadap arus dan arah medan magnet. Arah gaya ini akan terbalik jika arus atau arah medan magnetnya dibalik. Jika arah medan magnet dan arus keduanya dinaikan, maka arah gayanya tidak akan berubah. Sifat ini memungkinkan motor-motor tertentu dapat berputar dengan arus searah (DC) maupun bolak-balik (AC).

Pada prinsipnya mesin listrik dapat berlaku sebagai motor maupun sebagai generator. Perbedaannya hanya terdapat pada konversi dayanya, pada motor masukan tenaga listrik diubah menjadi tenaga mekanik, sedangkan pada generator masukan yang berupa tenaga mekanik diubah menjadi daya keluaran listrik. Maka dengan membalik generator arus searah, dimana tegangan Vt menjadi sumber dan tegangan Ea merupakan GGL (Gaya Gerak Listrik) lawan, mesin arus searah ini akan berlaku sebagai motor.

2.7.3 Karakteristik Motor DC

Gambar 2.14 merupakan rangkaian ekivalen motor DC *shunt* dengan eksitasi terpisah beserta persamaan yang berlaku :



Gambar 2.14 Rangkaian ekivalen motor DC

$$Ea = Vt - Ia\,Ra\,, \qquad (2.14)$$

$$Ea = C\,n\,\phi\,, \qquad (2.15)$$

$$n = \frac{Vt - Ia\,Ra}{C\,\phi}\,, \qquad (2.16)$$

Keterangan :

Ea = Tegangan Jangkar (*Armature*)

Vt = Tegangan yang dibangkitkan oleh rangkaian jangkar

Ia = Arus Jangkar

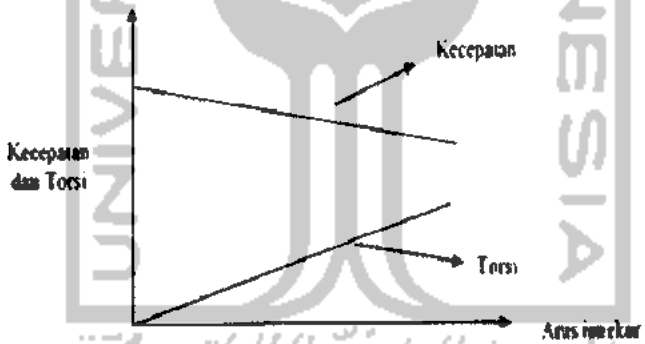
Ra = Resistansi Jangkar

C = Konstanta

n = Kecepatan

ϕ = Fluks Medan

Dari persamaan 2.16 diketahui bahwa pada motor *shunt*, bertambahnya kopel (arus jangkar bertambah) mengakibatkan kecepatan (n) menurun. Pada motor seri, bertambahnya kopel (arus) akan menyebabkan bertambahnya harga fluks (ϕ), karena fluks pada motor seri merupakan merupakan fungsi arus jangkar. Untuk harga arus jangkar sama dengan nol, harga fluks juga nol sehingga dari persamaan 2.15, diperoleh harga n menuju tak terhingga. Sedangkan untuk harga Ia yang cukup besar, harga n akan mendekati nol. Dengan demikian karakteristik kecepatan-kopel untuk motor *shunt* dan seri dapat digambarkan pada gambar 2.15 :



Gambar 2.15 Karakteristik kecepatan-kopel motor *shunt* dan seri

Pada motor dengan medan magnet permanen, medan magnetnya dihasilkan oleh satu atau beberapa magnet permanen. Magnet-magnet ini digenggam oleh besi atau baja, atau terkadang oleh rangka motor itu sendiri. Magnet ini merupakan bagian motor yang diam di tempatnya (stator). Kawat yang mengalirkan arus listrik digulung pada bagian motor yang berputar (rotor). Rotor yang terdapat pada motor sederhana, dibuat menjadi tiga buah kutub kumparan yang dibuat dari logam berlapis.

Fluks magnet menempuh suatu lintasan tertutup melalui rangka motor. Mengalirnya arus di dalam suatu kumparan kawat akan menimbulkan gaya. Gaya ini akan menggerakan rotor sampai arah gayanya sejajar dengan arah medan magnet. Pada keadaan seperti ini, rotor akan tetap diam dan tidak akan berputar lagi. Akan tetapi, jika arusnya dihubungkan ke salah satu kumparan lainnya, maka motor akan bergerak kembali sampai berada pada posisi sejajar yang baru.

Arus dialirkan ke kumparan rotor melalui dua buah sikat yang sekaligus merupakan kontak dengan cincin penghantar pada rotor (komutator). Komutator dibagi menjadi tiga bagian yang disusun berdekatan. Kedua sikat akan memindahkan arus dari satu kumparan ke kumparan lainnya sesuai dengan putaran motor. Salah satu kekurangan dari motor sederhana ini adalah adanya perubahan torsi pada setiap putaran motor. Pada kecepatan tinggi, perubahan torsi tidak masalah dan masih ada beban dan kelembaman atau *inertia* yang memperhalus gerakan motor. Pada kecepatan rendah, perubahan torsi membuat putaran motor cenderung meloncat dari satu posisi ke posisi lainnya.

2.7.4 Pengaturan Kecepatan Motor DC

Pengaturan kecepatan memegang peranan penting dalam motor arus searah, karena motor arus searah mempunyai karakteristik kopel-kecepatan yang menguntungkan dibandingkan dengan motor lainnya. Dari persamaan 2.14 sampai

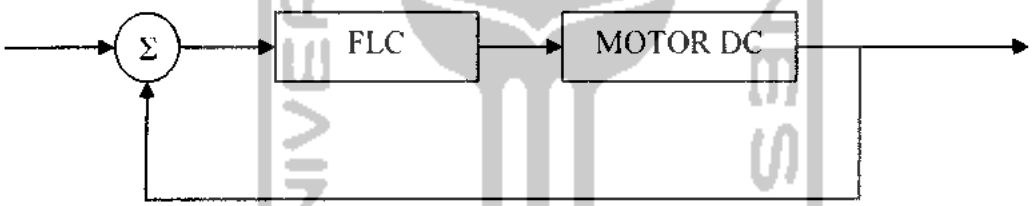
2.16, dapat dilihat bahwa kecepatan (n) dapat diatur dengan mengubah besaran ϕ,Vt dan Ra.

1. Pengaturan kecepatan dengan mengatur medan *shunt* (ϕ), dengan menyisipkan tahanan variabel yang dipasang seri terhadap kumparan medan (motor *shunt*), maka dapat diatur arus medan dan fluksnya. Rugi panas yang ditimbulkan sangat kecil pengaruhnya. Karena besarnya fluks yang dicapai oleh kumparan medan terbatas, kecepatan yang diaturpun akan terbatas.
2. Pengaturan kecepatan dengan mengatur tegangan (Vt), dikenal dengan metode *Ward Leonard*. Pengaturan jenis ini menghasilkan suatu pengaturan kecepatan yang sangat halus dan banyak dipakai untuk lift, mesin bubut dan lain-lain. Satu-satunya kerugian dalam sistem ini adalah biaya untuk penambahan generator dan penggerak awal.
3. Pengaturan kecepatan dengan mengatur tahanan (Ra), dengan menyisipkan tahanan variabel terhadap tahanan jangkar. Cara ini jarang dipakai, karena penambahan tahanan seri terhadap tahanan jangkar menimbulkan rugi panas yang cukup besar.

BAB III

METODOLOGI

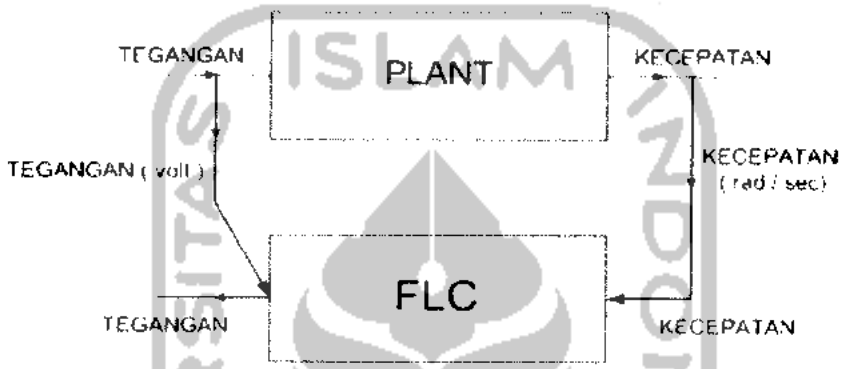
Pada perancangan sistem, ada dua buah masukan dalam FLC (*Fuzzy Logic Controller*) yaitu berupa kecepatan yang diinginkan (*error*) dan perubahan kecepatan (*derivative error*). Sedangkan keluarannya adalah perubahan tegangan. Sebagai keluaran motor dan juga sebagai hasil akhir dari sistem adalah kecepatan. 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)

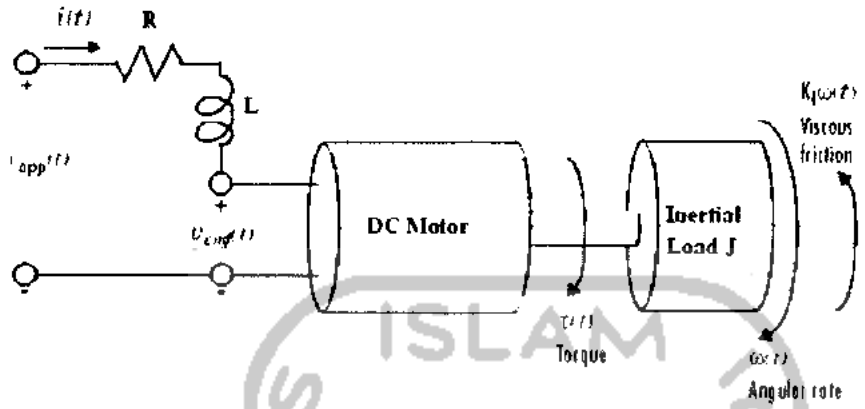
K_f / b = Koefisien gesek (N-m / rad /sec)

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, schingga

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

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 :

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

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

Maka persamaan tempat kedudukan sistem motor DC adalah sebagai berikut :

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

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

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

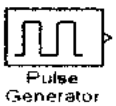
SIMULINK adalah perangkat lunak yang digunakan untuk memodelkan, melakukan simulasi, dan analisa terhadap sistem dinamis. *Simulink* dapat digunakan

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*

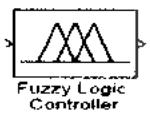


Merupakan blok yang digunakan untuk membangkitkan pulsa pada pengambilan data pelatihan.

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 :

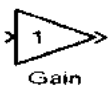
a. Sinyal vektor menjadi skalar atau vektor yang lebih kecil.

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*



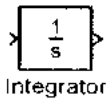
Blok *gain* merupakan blok penguat. Masukan pada blok dapat berupa besaran skalar, vektor atau matrik. Dapat diperolch secara *trial and error* maupun perhitungan matematis.

9. *Saturation*

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

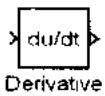
10. *Integrator*

10. Integrator



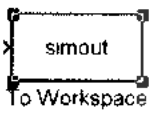
Blok ini merupakan integral dari keluaran. Merubah nilai keluaran menjadi perubahan / *integral* dari keluaran.

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) :

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

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

$$= 2.036 \text{ Nm}$$

$$T = K_i\, i_a$$

$$2.036 = K_i\, 3.2$$

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

$$Ki = 0.636$$

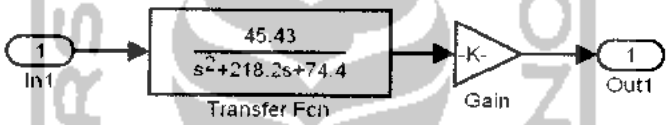
2. Perhitungan koefisien emf balik

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

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

$$K_b = 0.863$$

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

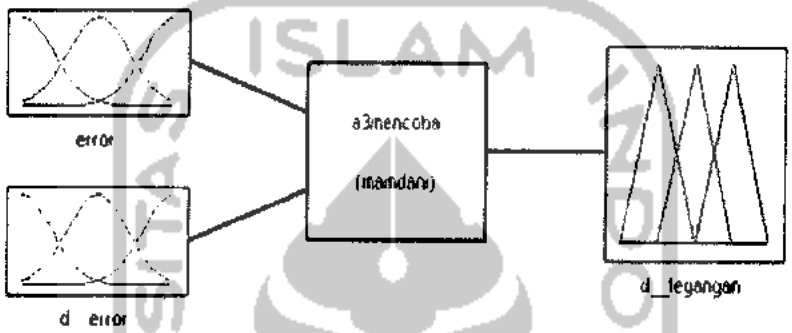
Pada gambar 3.4 setelah fungsi alih diperoleh kemudian ditambahkan dengan nilai *gain* / penguatan, agar diperoleh nilai keluaran yang diinginkan (1750 rpm) dengan masukan (150 volt). Nilai penguatannya adalah 60 / 3.14, yang diperoleh dari perubahan rpm (*radian per menit*) menjadi (*radian per second*).

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.



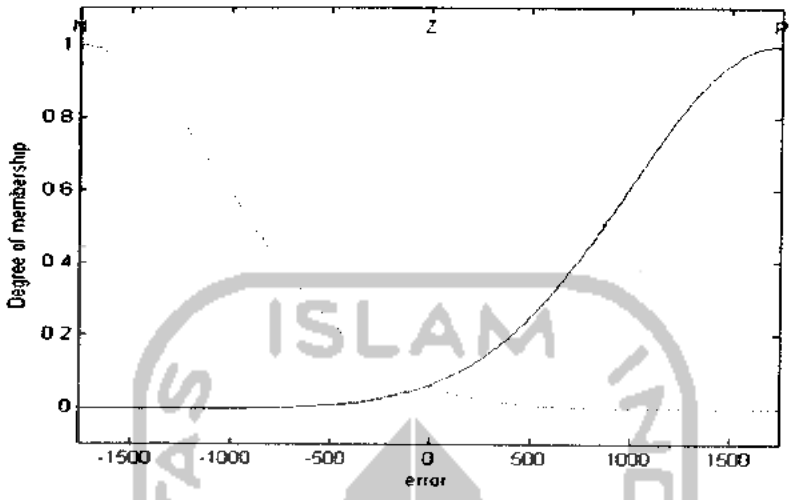
Gambar 3.5 Variabel masukan dan keluaran.

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

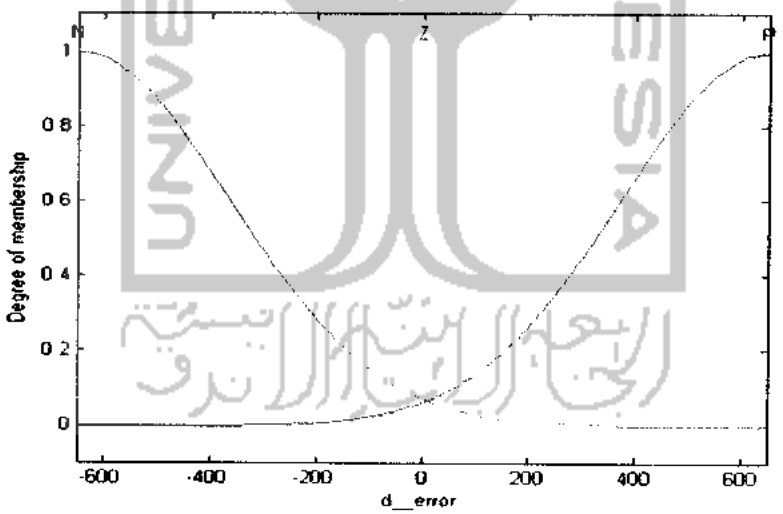
Jangkauan masukan dan keluaran ditentukan berdasarkan nilai maksimal dan minimal (*error* dan *d_error*) yang diperoleh saat simulasi. Keluaran dari kendali *fuzzy* merupakan besarnya perubahan tegangan. Untuk jangkauannya menyesuaikan dengan batas maksimal dari tegangan yang masuk ke motor DC.

3. Penentuan fungsi keanggotaan masukan dan keluaran FLC.

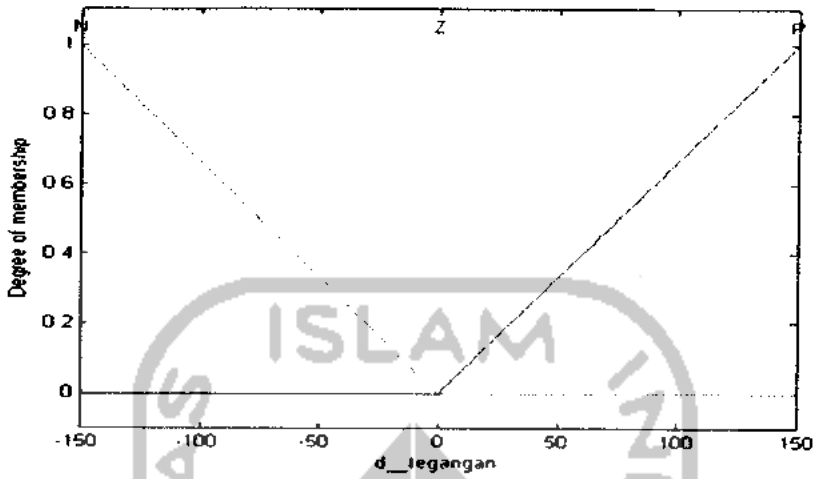
Parameter dari setiap fungsi keanggotaan masukan (*error* dan *d_error*) maupun keluaran, sesuai dengan ketentuan pada sistem automatisasi *fuzzy*. Fungsi keanggotaan masukan (*error* dan *d_error*) berupa fungsi keanggotaan *Gaussian* sedangkan fungsi keanggotaan keluaran berupa fungsi keanggotaan segitiga.



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 :

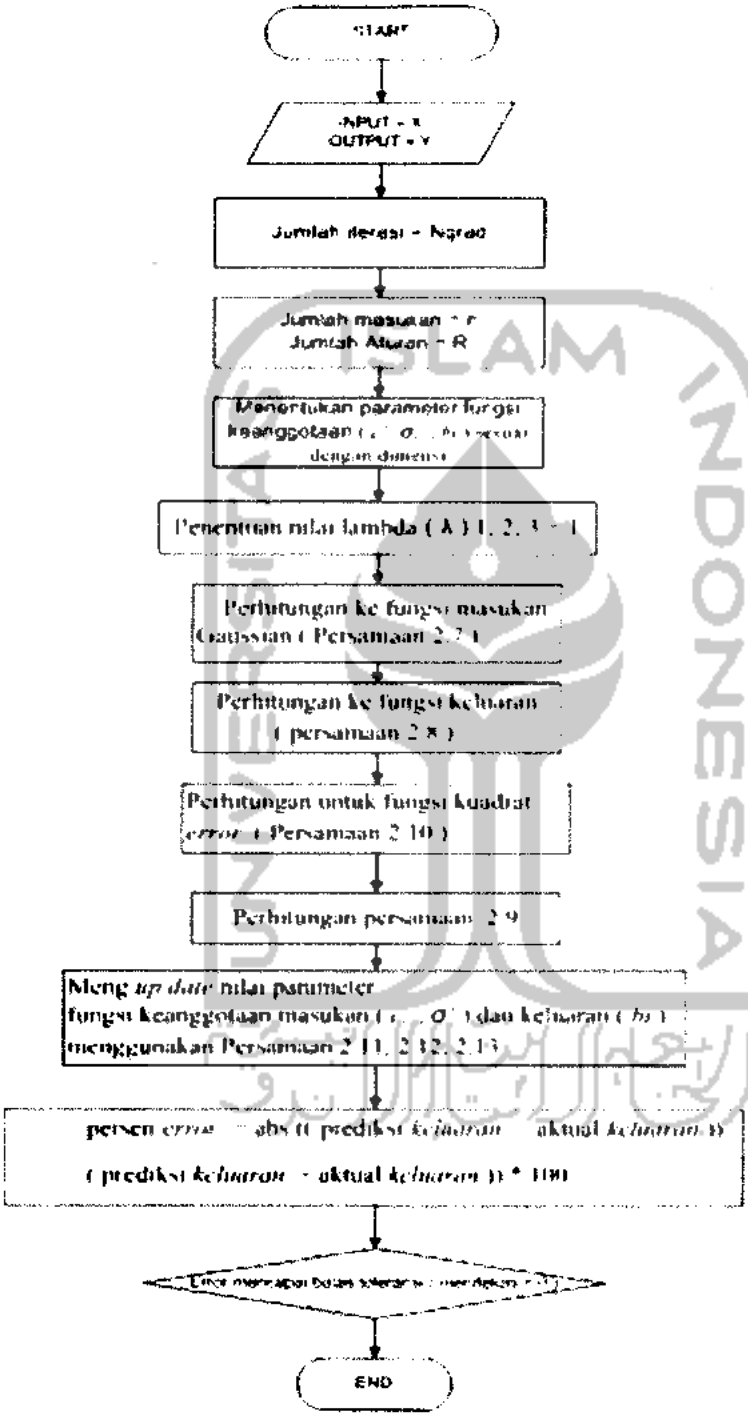
- Penentuan data masukan (X) dan data keluaran (Y) yang akan dilatih.
- Penentuan jumlah iterasi yang akan dilakukan (Ngrad).
- Penentuan jumlah masukan (n) dan jumlah aturan (R).
- Penentuan parameter fungsi keanggotaan masukan (c_j^i, σ_j^i) dan parameter fungsi keanggotaan keluaran (bi) sesuai dengan 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_i').

- 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

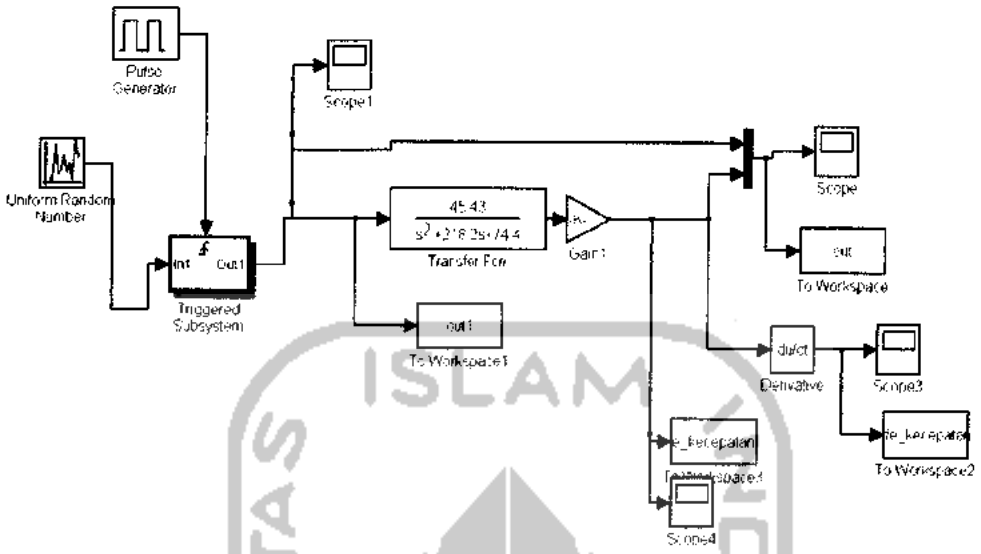
BAB IV

ANALISA DAN PEMBAHASAN SISTEM

Proses pelatihan dengan sistem automatisasi *fuzzy* metode gradien, terlebih dahulu dilakukan penentuan kondisi dari *plant* sesuai dengan keadaan yang ada pada motor sebenarnya. Masukan berupa tegangan (0 – 150 Volt) untuk mendapatkan keluaran berupa selisih kecepatan (*error*) dan perubahan kecepatan (*d_error*). Hasil simulasi disimpan kedalam *workspace* Matlab untuk dijadikan sebagai data masukan dan data keluaran pada pelatihan pengendali kecepatan motor DC.

4.1 Pelatihan *Fuzzy* dengan Metode Gradien

Pada pelatihan *fuzzy* metode gradien, data yang akan dilatih diambil dari hasil simulasi. Pada simulasi, masukan berupa nilai tegangan (0 – 150) pada blok *Uniform Random Number*, untuk memperoleh nilai kecepatan dan perubahan kecepatan. Hasil keluaran dari simulasi dalam bentuk gelombang kotak. Untuk membangkitkan gelombang kotak dapat diperoleh dengan menggunakan blok *Pulse Generator*. Simulasi menggunakan waktu 1000 detik. Hasil simulasi kemudian disimpan di *workspace* dengan menggunakan blok *from workspace*, yang disimpan dalam format *array*. Gambar 4.1 merupakan rangkaian blok *simulink* untuk pengambilan data pelatihan, seperti yang terlihat pada gambar dibawah ini :



Gambar 4.1. Simulink Pengambilan Data Pelatihan Open Loop

Data pelatihan masukan diambil dari nilai *error* kecepatan dan *derivative error* kecepatan, sedangkan data keluaran diambil dari nilai *out*1. Diperoleh data pelatihan sebanyak 64534. Data tersebut digunakan untuk melakukan pelatihan. dengan beragam nilai iterasi. Pada saat pelatihan dilakukan dengan semua data, membutuhkan waktu yang sangat lama dalam hitung jam. Namun tidak diperoleh nilai *error* yang diinginkan.

Pada pelatihan menggunakan 500 data berurutan diambil secara bebas. Proses pelatihan dilakukan dengan data pelatihan dan iterasi yang beragam, untuk memperoleh nilai *error* pelatihan yang diingkan. Jika *iterasi* diperbesar maka waktu pelatihan bertambah lama, *error* pelatihan bertambah kecil. Pelatihan menggunakan 200 *iterasi*, dapat dilakukan hanya dalam hitungan menit untuk menghasilkan nilai *error* yang diinginkan (mendekati 0). *Error* pelatihan menggunakan 3 aturan

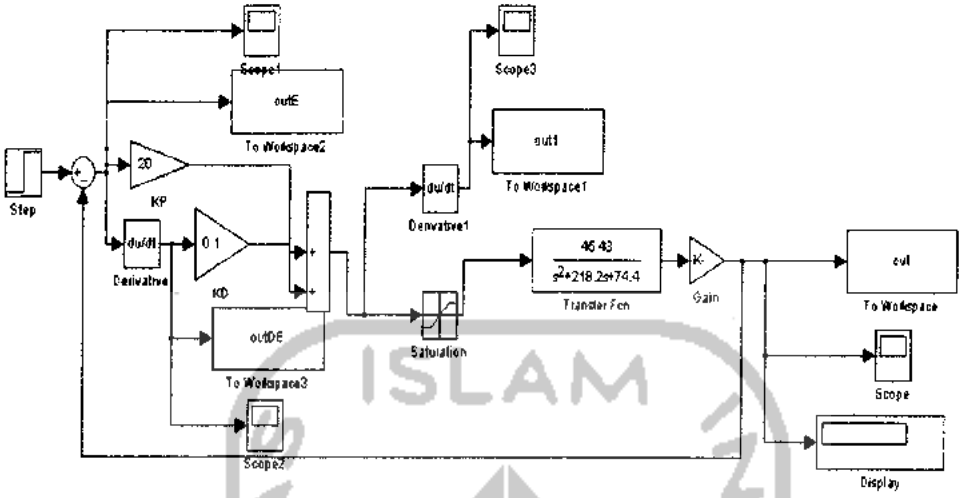
sebesaar 0 sampai 0.0278. *Error* pelatihan menggunakan 4 aturan sebesar 0 sampai 3.019. *Error* pelatihan menggunakan 5 aturan sebesaar 0 sampai 2.995. Aturan diperoleh secara *trial and error.*

Pada proses pelatihan data dengan nilai 0 tidak digunakan, karena akan menghasilkan nilai *error* 100% setelah dilakukan nilai *iterasi* yang berbeda – beda. Hal ini disebabkan pada keadaan sebenarnya, jika motor DC diberi masukan 0 volt maka tidak ada nilai tegangan yang dibangkitkan motor. Motor tidak akan mendapatkan keluaran berupa nilai kecepatan atau motor dalam keadaan diam.

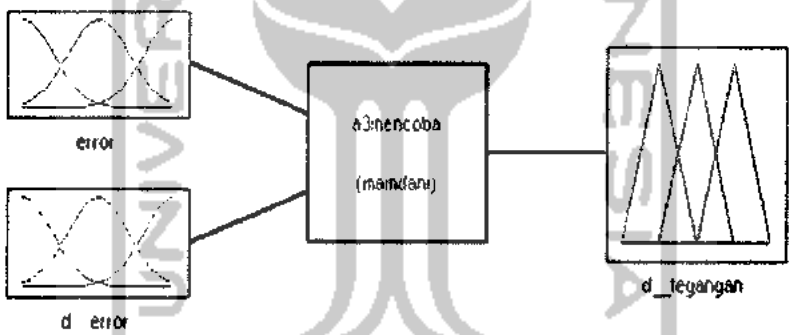
4.2 Simulasi Motor DC dengan Kendali *Fuzzy*

4.2.1 Simulasi Motor DC Sebelum Pelatihan

Tahap pertama adalah penentuan jangkauan untuk masukan dan keluaran pada sistem kendali logika *fuzzy* berdasarkan gambar 4.2. Jangkauan masukan dan keluaran ditentukan berdasarkan nilai maksimal dan minimal *error* dan *d_error* yang diperoleh saat simulasi. Nilai jangkauan *error* [-1750 , +1750], nilai jangkauan *d_error* [-650, +650]. Keluaran dari kendali *fuzzy* merupakan perubahan tegangan. Nilai jangkauan perubahan tegangan [-150, +150].

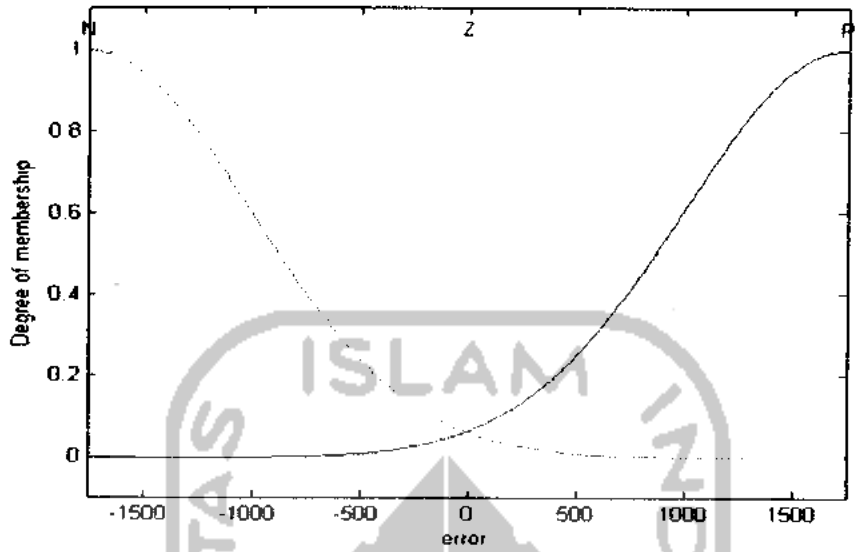


Gambar 4.2 Pengambilan jangkauan variabel masukan dan keluaran.

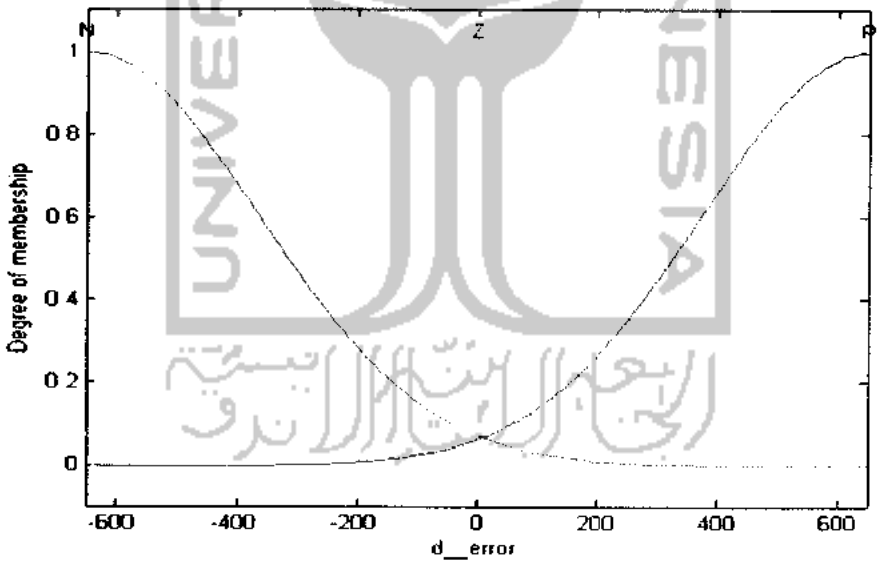


Gambar 4.3 Dua variabel masukan dan satu variabel keluaran

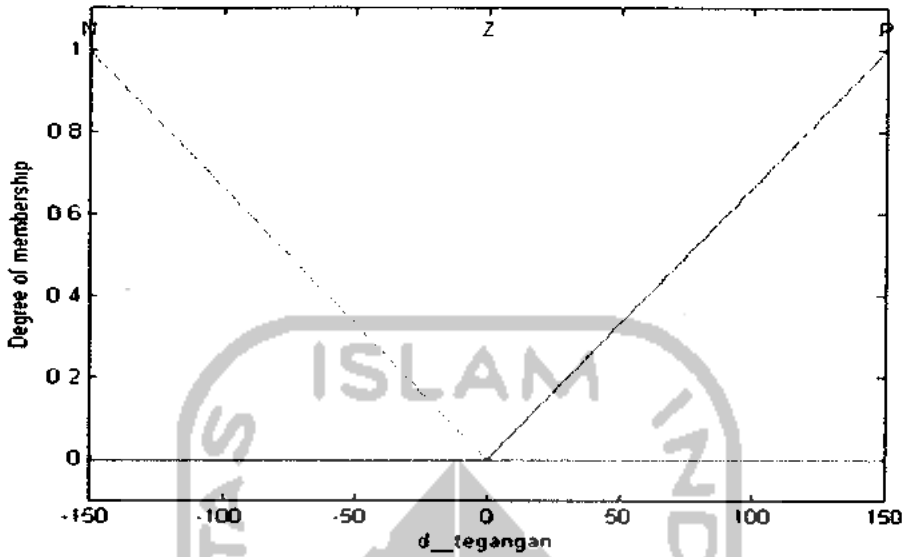
Tahap kedua yaitu penentuan fungsi keanggotaan untuk variabel masukan dan variabel keluaran. Variabel masukan menggunakan 3 fungsi keanggotaan Gaussian dan variabel keluaran menggunakan 3 fungsi keanggotaan segitiga.



Gambar 4.4 Fungsi keanggotaan untuk *error*



Gambar 4.5 Fungsi keanggotaan untuk *d error*



Gambar 4.6 Fungsi keanggotaan untuk perubahan tegangan

Tahap ketiga adalah penentuan aturan *fuzzy*. Penentuan aturan *fuzzy* merupakan tahap akhir dari proses desain sistem *fuzzy*. Berdasarkan jumlah masukan (2) dan jumlah fungsi keanggotaan dari masing – masing masukan (3), diperoleh aturan maksimal yang dapat digunakan yaitu $3^2 = 9$ aturan. Jumlah 9 aturan tidak digunakan semua pada desain aturan *fuzzy*. Hanya 3, 4 dan 5 aturan yang digunakan dalam aturan *fuzzy*. Penggunaan variasi aturan diperoleh secara *trial and error* berdasarkan pada tabel 3.1.

Penggunaan 3 aturan seperti dibawah ini, setelah dilakukan beberapa kali percobaan.

1. if (er is N) and (d_er is N) then (tegangan is P)
2. if (er is P) and (d_er is Z) then (tegangan is N)
3. if (er is P) and (d_er is P) then (tegangan is N)

Penggunaan 4 aturan diperoleh dari beberapa percobaan seperti yang terlihat pada tabel 4.1.

Tabel 4.1 Tabel Percobaan 4 Aturan

Percobaan	Aturan yan digunakan	Keterangan
1	1,2,8,9	Parameter b no 2 tidak berubah.
2	1,3,8,9	Berhasil
3	1,4,8,9	Parameter b no 2 tidak berubah.
4	1,5,8,9	Parameter b no 2 tidak berubah.
5	1,6,8,9	Parameter b no 2 tidak berubah.

Berdasarkan pada tabel 4.1, maka 4 aturan seperti dibawah ini :

1. *if (er is N) and (d_er is N) then (d_tegangan is N)*
2. *if (er is N) and (d_er is P) then (d_tegangan is Z)*
3. *if (er is P) and (d_er is Z) then (d_tegangan is P)*
4. *if (er is P) and (d_er is P) then (d_tegangan is P)*

Penggunaan 5 aturan diperoleh dari beberapa percobaan seperti yang terlihat pada tabel 4.2.

Tabel 4.2 Tabel Percobaan 5 Aturan

Percobaan	Aturan yan digunakan	Keterangan
1	1,2,3,8,9	Berhasil.
2	1,3,4,8,9	Parameter b ke 2 tidak berubah.
3	1,3,5,8,9	Parameter b ke 2 dan 3 tidak berubah.
4	1,3,6,8,9	Parameter b ke 3 tidak berubah.

Berdasarkan pada tabel 4.2, maka 5 aturan seperti terlihat dibawah ini :

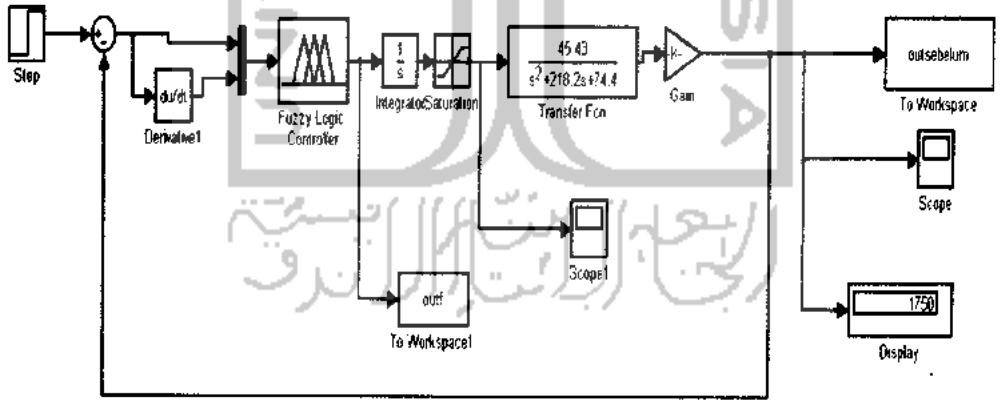
1. *if (er is N) and (d_er is N) then (d_tegangan is N)*
2. *if (er is N) and (d_er is Z) then (d_tegangan is N)*
3. *if (er is N) and (d_er is P) then (d_tegangan is Z)*
4. *if (er is P) and (d_er is Z) then (d_tegangan is P)*
5. *if (er is P) and (d_er is P) then (d_tegangan is P)*

Tabel 4.1 dan 4.2 merupakan contoh untuk memperoleh 4 dan 5 aturan *fuzzy*. Karena tidak semua variasi aturan dapat digunakan dalam perancangan *fuzzy* yang

baru untuk mendapatkan *error* sistem kecil pada saat pengujian. Beberapa hal dapat disebabkan karena parameter b dari nomor aturan tertentu tidak berada dalam nilai jangkaun setelah dilakukan pelatihan dengan *iterasi* yang beragam dan memperlebar jangkauan parameter b.

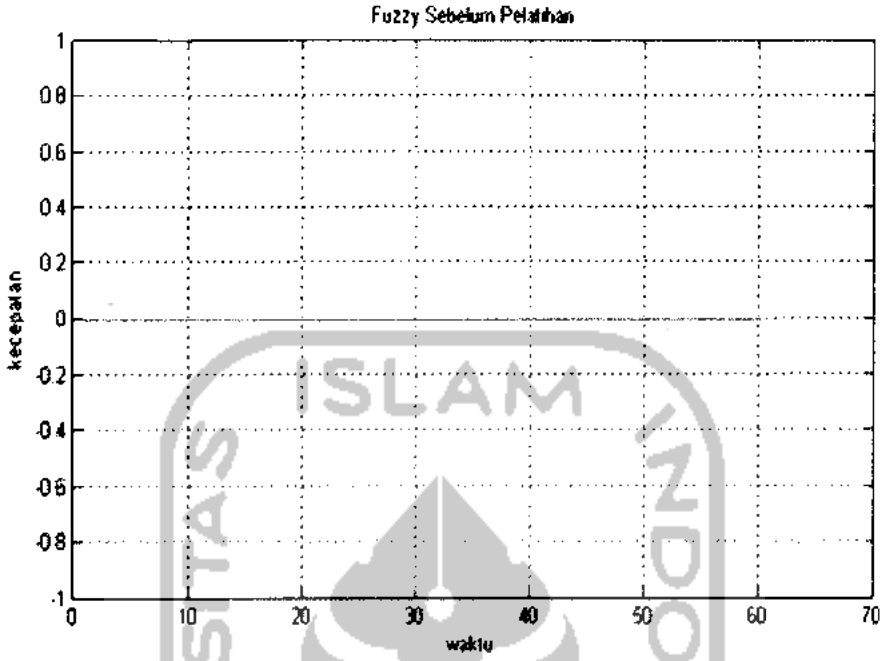
Penggunaan aturan *fuzzy* yang baru dapat diperoleh dengan metode *trial and error* untuk mendapatkan variasi aturan yang beragam. Metode tersebut membutuhkan waktu yang lama untuk memperoleh aturan yang diinginkan.

Proses tahapan perancangan *fuzzy* selesai dilakukan, *fuzzy* disimpan dengan format .fis, langkah selanjutnya *export* data kendali *fuzzy* ke *workspace.* Untuk dapat menjalankan simulasi sistem dengan menggunakan *simulink* seperti pada gambar 4.7, blok FLC diberi nama sesuai dengan nama *fuzzy* yang disimpan.



Gambar 4.7 Rangkaian simulasi motor DC sebelum pelatihan

Simulasi kendali motor DC menggunakan waktu simulasi selama 100 detik. Setelah di jalankan didapatkan grafik respon sistem seperti pada gambar 4.8.



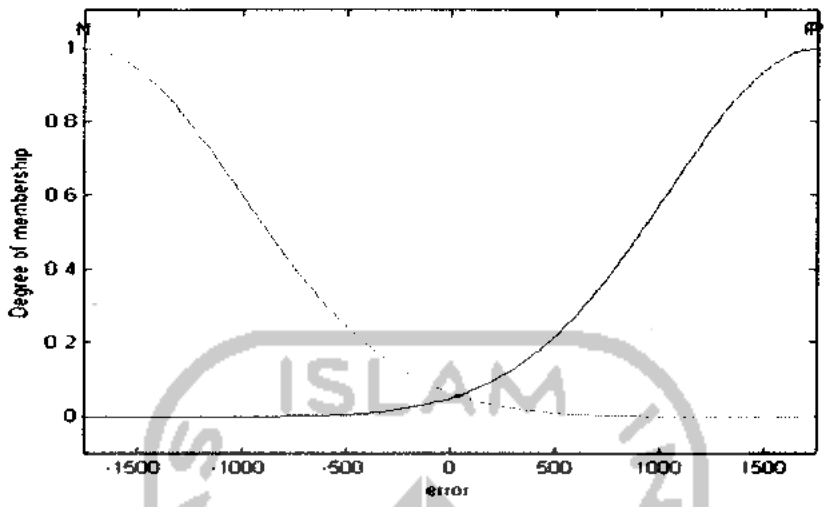
Gambar 4.8 Keluaran sistem *fuzzy* sebelum pelatihan

4.2.2 Simulasi Motor DC Setelah Pelatihan

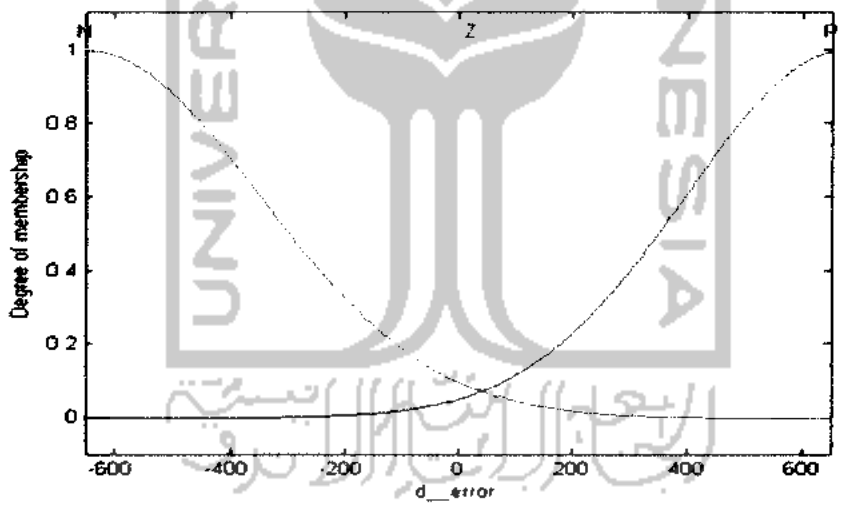
Proses perancangan *fuzzy* yang baru dilakukan setelah tahap pelatihan *fuzzy* seperti pada subbab 4.1. Pada saat diperoleh *error* yang diinginkan dan perubahan parameter fungsi keanggotaan (bi, c_j^l, σ_j^l). Langkah selanjutnya melakukan perancangan *fuzzy* yang baru. Tahap melakukan perancangan *fuzzy* setelah pelatihan. hampir sama seperti perancangan *fuzzy* sebelum pelatihan. Perbedaannya adalah pada parameter fungsi keanggotaan masukan (c_j^l, σ_j^l) dan keluaran (bi). Perubahan hasil parameter fungsi keanggotaan (bi, c_j^l, σ_j^l) dimasukan ke perancangan *fuzzy*.

A. Penggunaan 3 Aturan

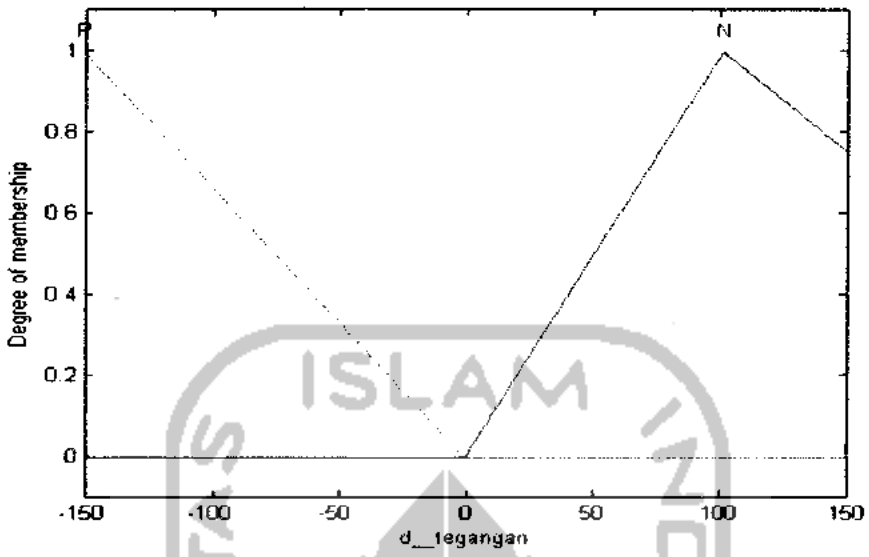
Gambar 4.9, 4.10, 4.11 merupakan hasil perubahan parameter fungsi keanggotaan *fuzzy* dengan 3 aturan.



Gambar 4.9 Fungsi keanggotaan masukan *error fuzzy* setelah Pelatihan



Gambar 4.10 Fungsi keanggotaan masukan *derivative error fuzzy* setelah Pelatihan



Gambar 4.11 Fungsi keanggotaan keluaran *fuzzy* setelah Pelatihan

Gambar 4.9, 4.10, 4.11 diperoleh dari tabel 4.3, 4.4, 4.5 yang merupakan tabel perbandingan antara parameter fungsi keanggotaan masukan dan keluaran sebelum dan setelah pelatihan.

Tabel 4.3 Parameter fungsi keanggotaan masukan *error*

Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-1750	-1749.7	750	750.8557
1741	1725.6	750	778.4098
1741	1757.8	750	717.3058

Tabel 4.4 Parameter fungsi keanggotaan masukan *d_error*

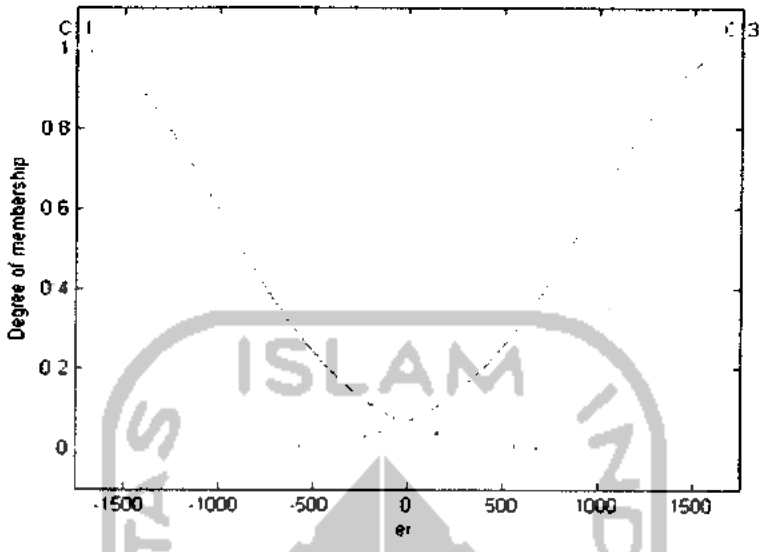
Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-650	-649.2	300	302.3384
0	19.9	300	317.4746
650	674.6	300	275.7636

Tabel 4.5 Parameter fungsi keanggotaan keluaran (b)

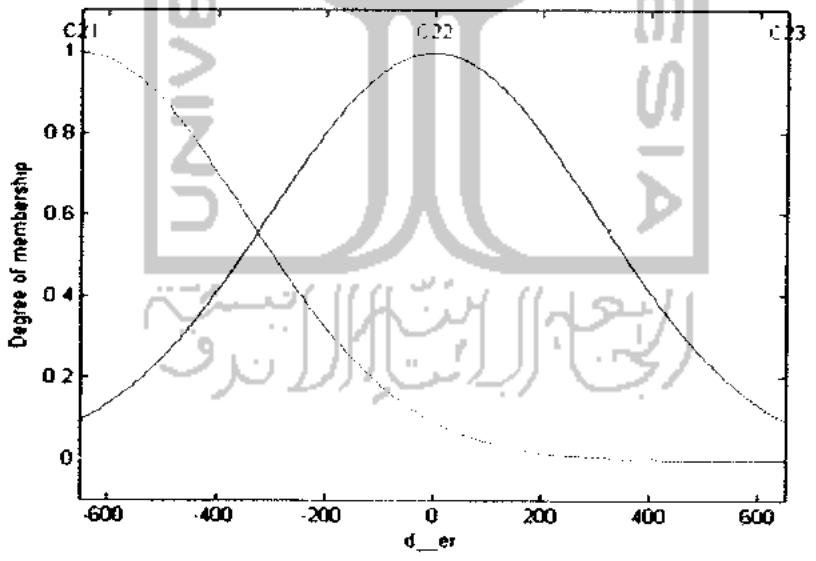
Sebelum Pelatihan	**Setelah Pelatihan**
150	150.5427
-150	101.8506
-150	101.3777

B. Penggunaan 4 Aturan

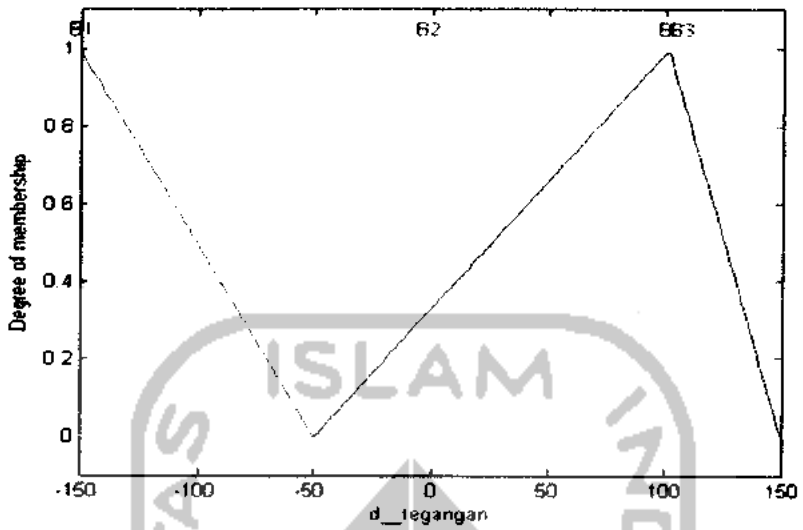
Gambar 4.12, 4.13, 4.14 merupakan hasil perubahan parameter fungsi keanggotaan *fuzzy* dengan 4 aturan.



Gambar 4.12 Fungsi keanggotaan masukan *error fuzzy* setelah Pelatihan



Gambar 4.13 Fungsi keanggotaan masukan *derivative error fuzzy* setelah Pelatihan



Gambar 4.14 Fungsi keanggotaan keluaran *fuzzy* setelah Pelatihan

Gambar 4.12, 4.13, 4.14 diperoleh dari tabel 4.6, 4.7, 4.8 yang merupakan tabel perbandingan antara parameter fungsi keanggotaan masukan dan keluaran sebelum dan setelah pelatihan.

Tabel 4.6 Parameter fungsi keanggotaan masukan *error*

Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-1750	-1749.9	750	750.2174
-1750	-1749.3	750	752.0831
1741	1740.7	750	750.6396
1741	1741.6	750	748.9025

Tabel 4.7 Parameter fungsi keanggotaan masukan *d_error*

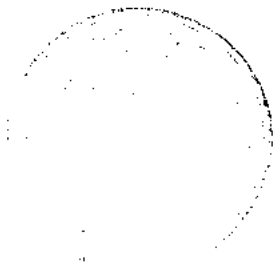
Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-650	-649.8	300	300.4882
650	649.2	300	301.2250
0	0.5	300	300.4372
650	651.2	300	298.6372

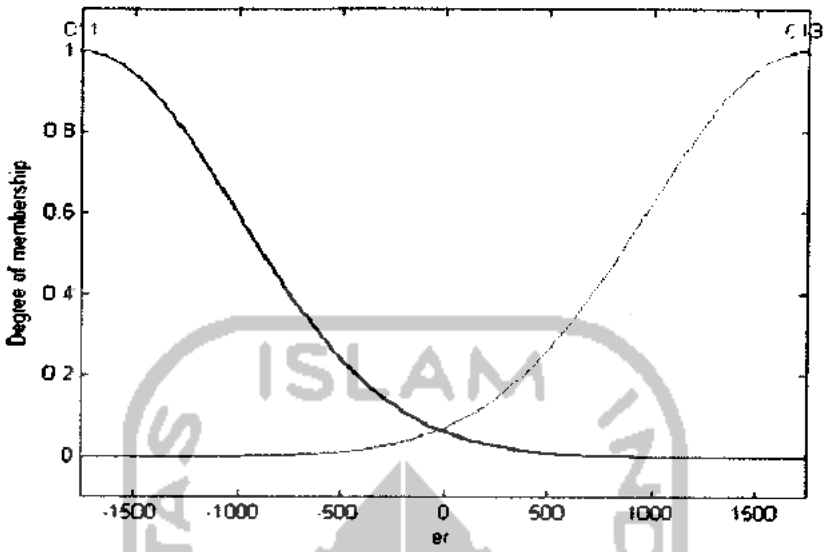
Tabel 4.8 Parameter fungsi keanggotaan keluaran (b)

Sebelum Pelatihan	**Setelah Pelatihan**
-150	-150.0631
-150	-0.8612
0	101.3855
150	106.2847

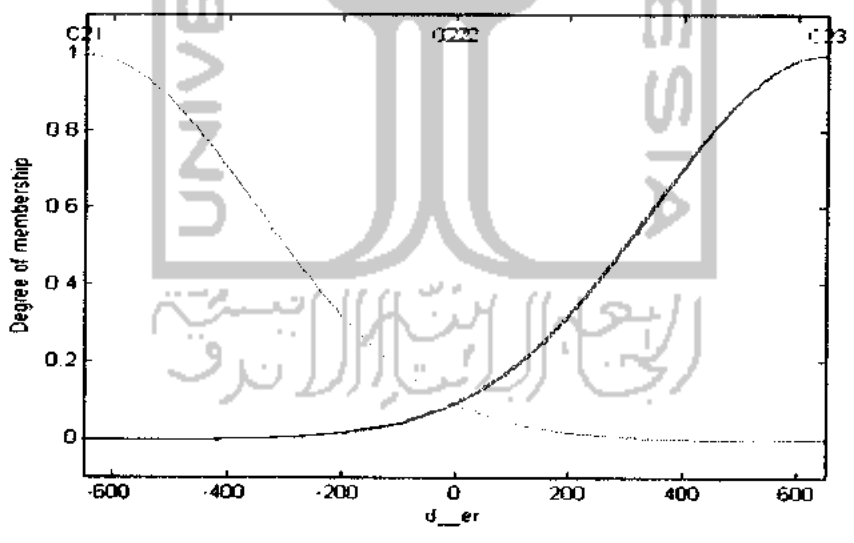
C. Penggunaan 5 Aturan

Gambar 4.15, 4.16, 4.17 merupakan hasil perubahan parameter fungsi keanggotaan *fuzzy* dengan 5 aturan.

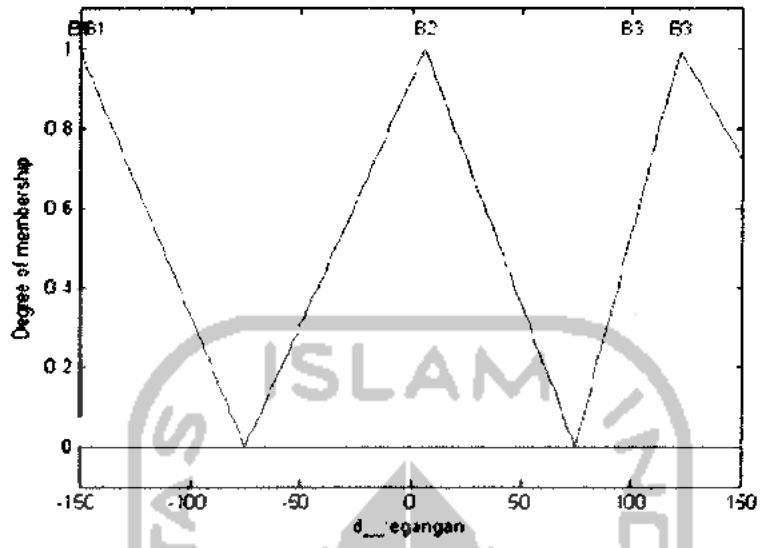




Gambar 4.15 Fungsi keanggotaan masukan *error fuzzy* setelah Pelatihan



Gambar 4.16 Fungsi keanggotaan masukan *derivative error fuzzy* setelah Pelatihan



Gambar 4.17 Fungsi keanggotaan keluaran *fuzzy* setelah Pelatihan

Gambar 4.15, 4.16, 4.17 diperoleh dari tabel 4.9, 4.10, 4.11 yang merupakan tabel perbandingan antara parameter fungsi keanggotaan masukan dan keluaran sebelum dan setelah pelatihan.

Tabel 4.9 Parameter fungsi keanggotaan masukan *error*

Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-1750	-1750	750	750.1349
-1750	-1753.9	750	741.5995
-1750	-1751.9	750	744.9383
1741	1737.9	750	755.5481
1741	1738.8	750	755.6004

Tabel 4.10 Parameter fungsi keanggotaan masukan *d_error*

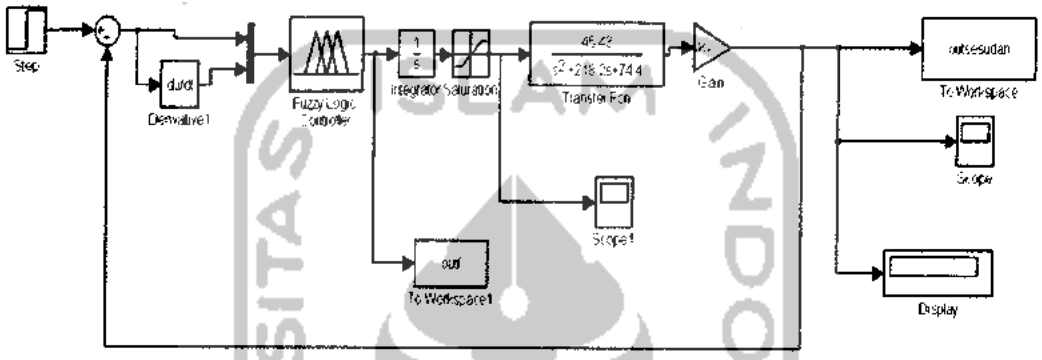
Perubahan Parameter C		Perubahan Parameter σ	
Sebelum Pelatihan	**Setelah Pelatihan**	**Sebelum Pelatihan**	**Setelah Pelatihan**
-650	-650	300	299.8790
0	-5.9	300	293.1283
650	652	300	297.9885
0	4.1	300	303.3245
650	648.9	300	299.8649

Tabel 4.11 Parameter fungsi keanggotaan keluaran (b)

Sebelum Pelatihan	**Setelah Pelatihan**
-150	-150.0013
-150	-144.1906
0	6.5860
0	99.8263
150	121.8568

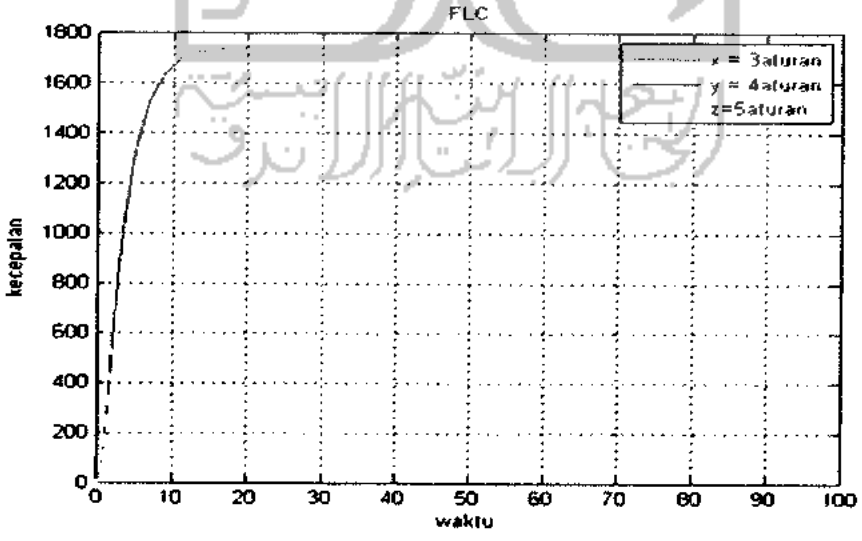
4.3 Pengujian Sistem

Perancangan *fuzzy* selesai dilakukan, langkah selanjutnya *export* data kendali *fuzzy* ke *workspace* untuk dapat menjalankan simulasi sistem dengan menggunakan *simulink* seperti pada gambar 4.18.



Gambar 4.18 Rangkaian simulasi motor DC setelah pelatihan

Simulasi kendali motor DC menggunakan waktu simulasi selama 100 detik. Pengujian dilakukan dengan masukan 1750, didapatkan respon sistem dan keluaran *step respon* setelah pelatihan seperti terlihat pada gambar 4.19.



Gambar 4.19 Hasil keluaran setelah pelatihan

Tabel 4.12 Respon Sistem Sebelum dan Setelah Pelatihan dengan masukan 1750 rad / s.

Karakteristik	**Sebelum**	**Setelah**		
		3 Aturan	4 Aturan	5 Aturan
Rise time	-	6.7192	6.5987	6.5633
Settling time	-	12.4683	12.3139	12.2748
Settling minimal	-	1.5755e+003	1.5755e+003	1.5757e+003
Settling maximal	-	1.7502e+003	1.7502e+003	1.7502e+003
Overshoot (%)	-	0.0104	0.0104	0.0104
Undershoot (%)	-	0	0	0
Peak	-	1.7502e+003	1.7502e+003	1.7502e+003
Peak time	-	99.8007	99.9484	99.4044

Berdasarkan tabel 4.12, respon sistem setelah pelatihan (menggunakan 3, 4, 5 aturan) lebih baik dibandingkan sebelum pelatihan. Hal ini disebabkan oleh penggunaan data pelatihan pada proses pelatihan untuk memperbaiki parameter masukan dan keluaran.

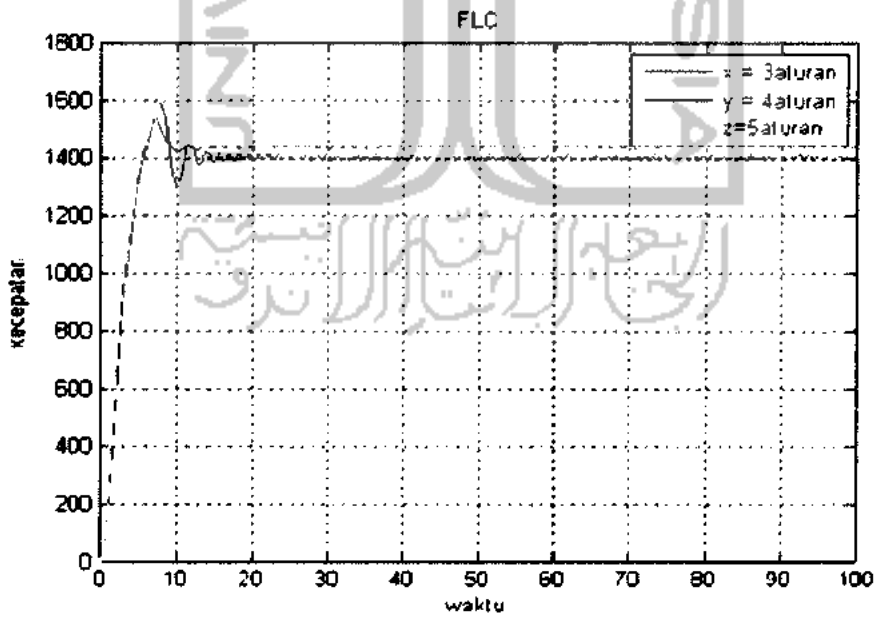
Hasil pengujian menggunakan 3, 4 dan 5 aturan dilakukan dengan memberikan nilai masukan (*setpoint*) yang berbeda.

1. Pengambilan data 1400 rad / s.

   Hasil respon sistem seperti terlihat pada table 4.13

Tabel 4.13 Respon Sistem Setelah Pelatihan dengan masukan 1400 rad / s.

Karakteristik	Setelah		
	3 Aturan	4 Aturan	5 Aturan
Rise time	3.8324	3.7038	3.6642
Settling time	8.0210	12.4936	21.1036
Settling minimal	1.2621e+003	1.2613e+003	1.2620e+003
Settling maximal	1.5380e+003	1.5917e+003	1.6206e+003
Overshoot (%)	9.8558	13.6929	15.7553
Undershoot (%)	0	0	0
Peak	1.5380e+003	1.5917e+003	1.6206e+003
Peak time	7.3275	7.9811	8.5293



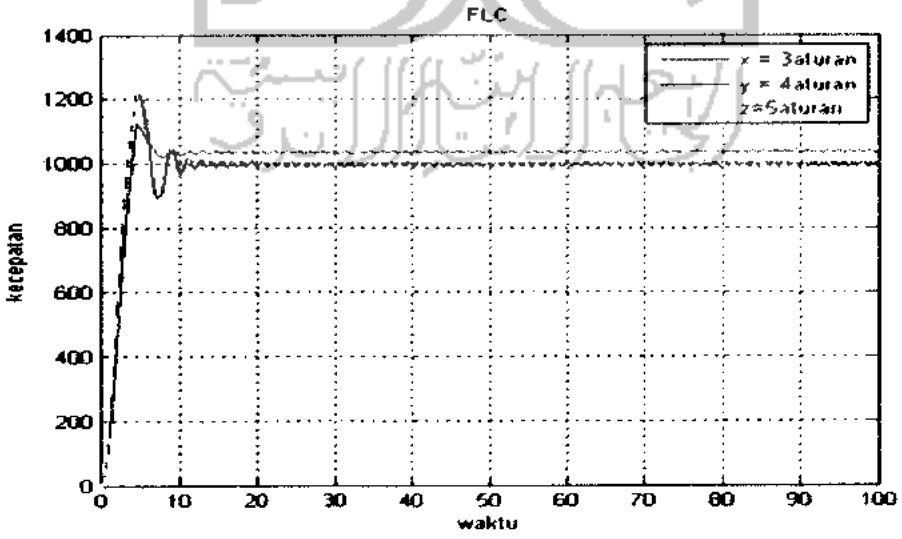
Gambar 4.20 *Step Respon* Pengambilan data 1400 rad / s

2. Pengambilan data 1000 rad / s.

Hasil respon sistem seperti terlihat pada table 4.14.

Tabel 4.14 Respon Sistem Setelah Pelatihan dengan masukan 1000 rad / s.

Karakteristik	**Setelah**		
	3 Aturan	4 Aturan	5 Aturan
Rise time	2.7494	2.3078	2.2198
Settling time	8.4045	11.3915	18.6656
Settling minimal	903.6929	896.8396	902.5205
Settling maximal	1.1277e+003	1.2265e+003	1.2816e+003
Overshoot (%)	12.7705	22.6508	28.1552
Undershoot (%)	0	0	0
Peak	1.1277e+003	1.2265e+003	1.2816e+003
Peak time	4.9202	4.7657	4.9580



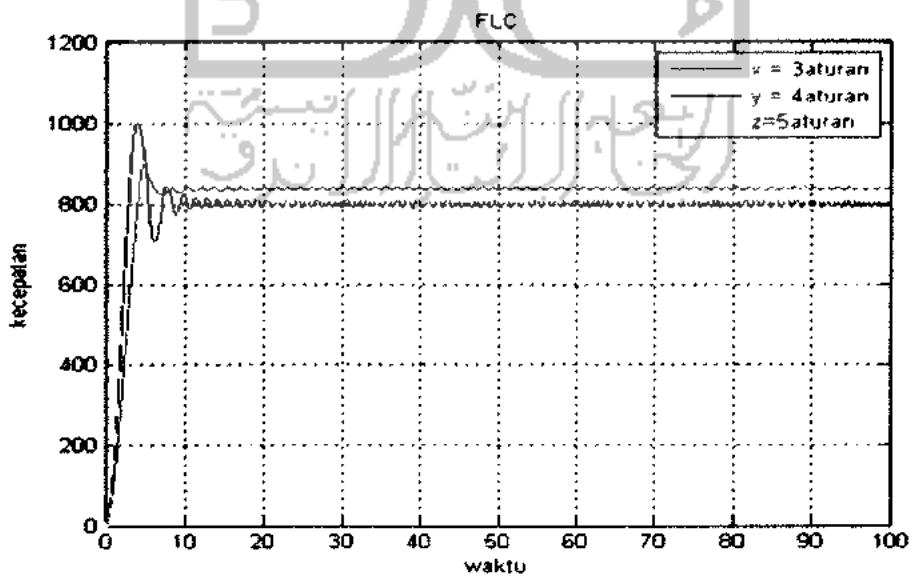
Gambar 4.21 *Step Respon* Pengambilan data 1000 rad / s.

3. Pengambilan data 800 rad / s.

Hasil respon sistem seperti terlihat pada table 4.15.

Tabel 4.15 Respon Sistem Setelah Pelatihan dengan masukan 800 rad / s.

Karakteristik	**Setelah**		
	3 Aturan	4 Aturan	5 Aturan
Rise time	2.7983	1.9207	1.7688
Settling time	8.3013	10.3492	18.1977
Settling minimal	720.2586	705.6440	725.9562
Settling maximal	901.5193	1.0012e+003	1.0614e+003
Overshoot (%)	12.6899	25.1533	32.6688
Undershoot (%)	0	0	0
Peak	901.5193	1.0012e+003	1.0614e+003
Peak time	4.8999	3.9329	3.9390



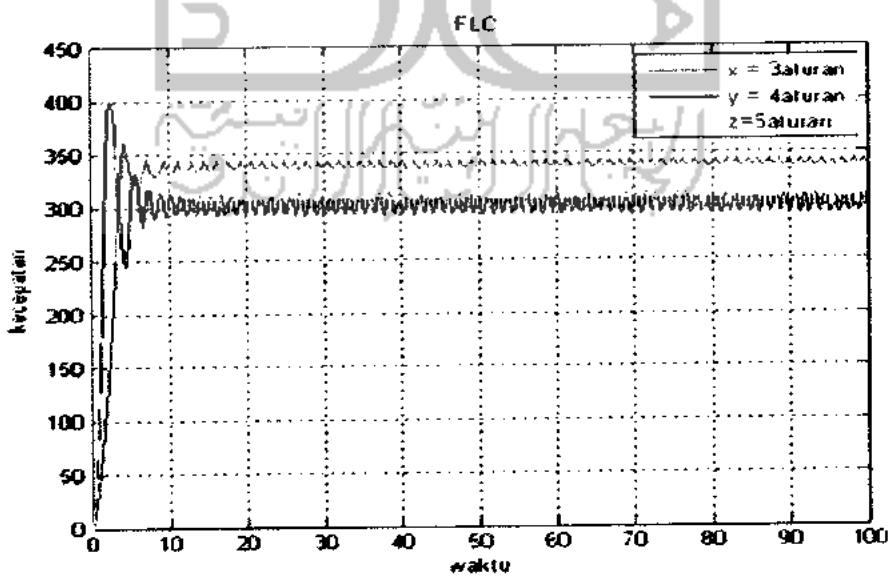
Gambar 4.22 *Step Respon* Pengambilan data 800 rad / s.

4. Pengambilan data 300 rad / s.

Hasil respon sistem seperti terlihat pada table 4.16.

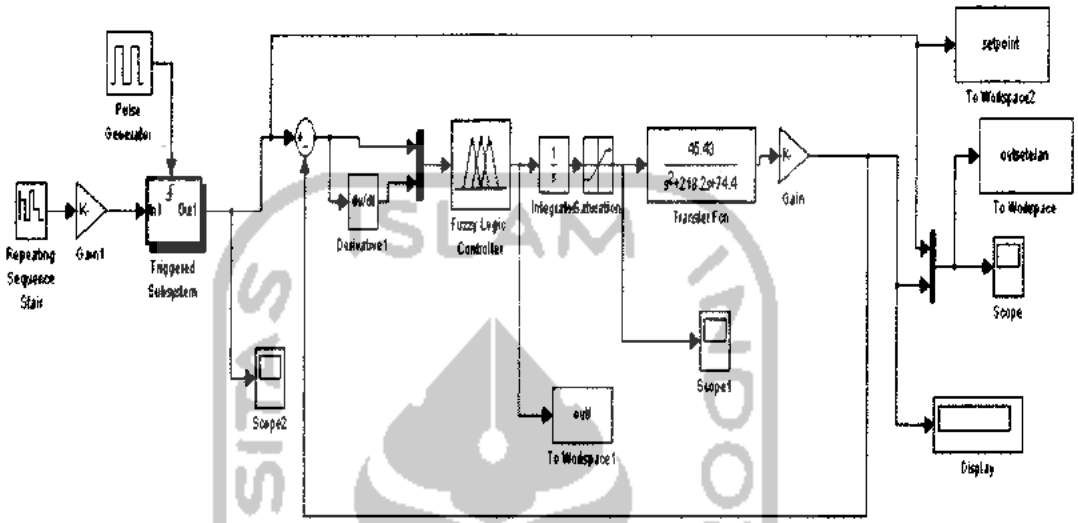
Tabel 4.16 Respon Sistem Setelah Pelatihan dengan masukan 300 rad / s.

Karakteristik	**Setelah**		
	3 Aturan	4 Aturan	5 Aturan
Rise time	2.4619	1.1634	0.9710
Settling time	NaN	NaN	NaN
Settling minimal	270.0671	245.7802	272.3831
Settling maximal	361.2320	400.8773	440.7591
Overshoot (%)	20.4107	33.6258	46.9197
Undershoot (%)	0	0	0
Peak	361.2320	400.8773	440.7591
Peak time	4.3774	2.4820	2.2566

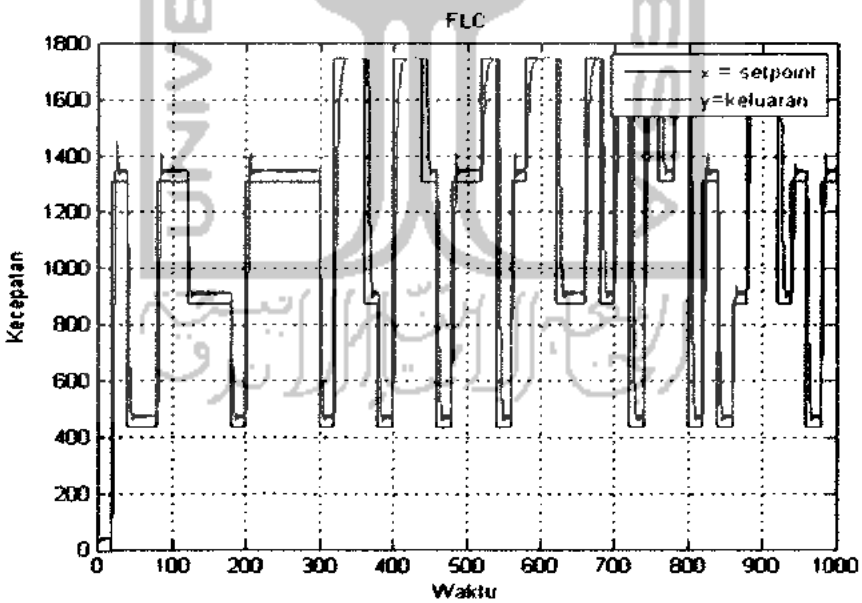


Gambar 4.23 *Step Respon* Pengambilan data 300 rad / s.

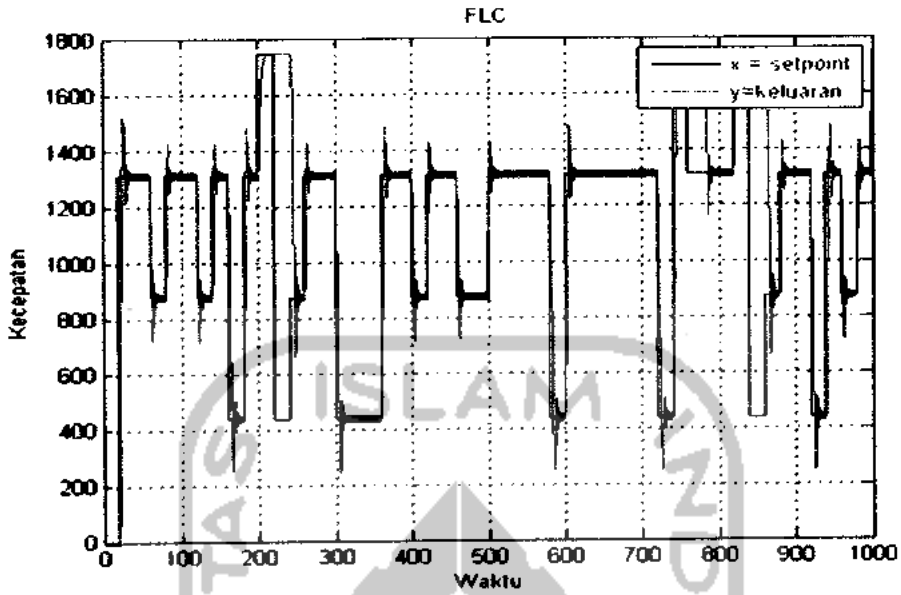
Setelah nilai karakteristik dari respon sistem diketahui, kemudian dilakukan pengujian menggunakan rangkaian simulasi menggunakan gambar 4.24.



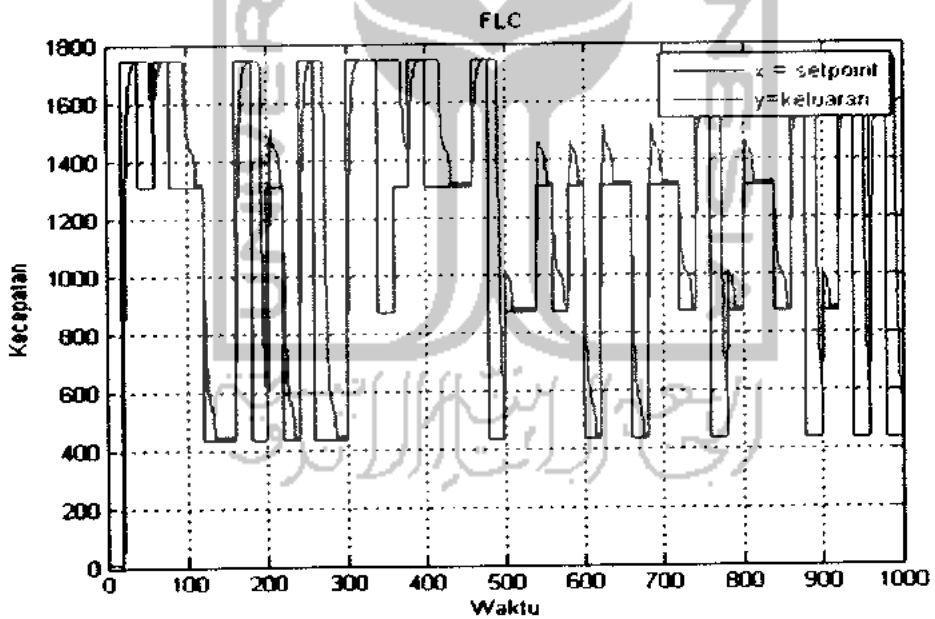
Gambar 4.24 Rangkaian simulasi pengujian terakhir.



Gambar 4.25 Hasil keluaran pengujian dengan 3 aturan



Gambar 4.26 Hasil keluaran pengujian dengan 4 aturan.



Gambar 4.27 Hasil keluaran pengujian dengan 5 aturan.

Pada hasil *step respon* dengan penggunaan 5 aturan menghasilkan respon sistem yang lebih cepat dari penggunaan 3 aturan dan 4 aturan. Berdasarkan hasil respon sistem, kendali *fuzzy* dapat meredam *overshoot* yang dihasilkan sistem pada nilai

akhir 1750, sedangkan pada nilai lain terdapat *overshoot*. Pada Gambar 4.25 diatas, penggunaan 3 aturan, grafik keluaran (berwarna biru) sudah hampir mengikuti grafik *set point* (berwarna merah). Setiap mencapai puncak nilai 1750 tidak ada osilasi, sedangkan pada saat keadaan turun untuk mencapai puncak pada nilai yang lain terdapat osilasi dan pada waktu turun ada beberapa grafik keluaran yang tidak mengikuti dengan grafik *set point*. Begitu juga pada penggunaan 4 aturan dan 5 aturan. Hal ini karena pengaruh dari jangkauan dan perubahan parameter fungsi keanggotaan (masukan dan keluaran) dan aturan *fuzzy* yang digunakan. Namun secara keseluruhan grafik tersebut membuktikan bahwa pengujian sistem sudah cukup baik namun bukan yang terbaik.

Berikut ini adalah tabel perbandingan hasil pengujian sebelum dan setelah pelatihan *fuzzy*, dimana nilai masukan kecepatan dipilih secara bebas.

Tabel 4.17 Perbandingan Pengujian Sebelum dan Setelah Pelatihan

NO	Masukan (rad / s)	Sebelum (rad / s)	Setelah (rad / s)		
			3 Aturan	4 Aturan	5 Aturan
1	1750	-	1750	1750	1750
2	1400	-	1439	1406	1411
3	1000	-	1041	995.4	1011
4	800	-	839	794.4	808.4
5	300	-	338.7	297.3	314.8

Tabel 4.18 Pengujian kecepatan *set point* dengan kecepatan motor setelah pelatihan (3 aturan)

NO	Masukan (rad / s)	Keluaran (rad / s)	Selisih (rad / s)
1	1750	1750	0
2	1400	1439	39
3	1000	1041	41
4	800	839	39.2
5	300	338.7	38.7

Dari tabel 4.18, rata-rata selisih kecepatan setelah pelatihan sebesar 31.58 rad / s.

Tabel 4.19 Pengujian kecepatan *set point* dengan kecepatan motor setelah pelatihan (4 aturan)

NO	Masukan (rad / s)	Keluaran (rad / s)	Selisih (rad / s)
1	1750	1750	0
2	1400	1406	6
3	1000	995.4	4.6
4	800	794.4	5.6
5	300	297.3	2.7

Dari tabel 4.19, rata-rata selisih kecepatan setelah pelatihan sebesar 3.15 rad / s.

Tabel 4.20 Pengujian kecepatan *set point* dengan kecepatan motor setelah pelatihan (5 aturan)

NO	Masukan (rad / s)	Keluaran (rad / s)	Selisih (rad / s)
1	1750	1750	0
2	1400	1411	11
3	1000	1011	11
4	800	808.4	8.4
5	300	314.8	14.8

Dari tabel 4.20, rata-rata selisih kecepatan setelah pelatihan sebesar 7.53 rad / s.

Berdasarkan tabel 4.18, 4.19, dan 4.20 diperoleh nilai rata – rata selisih kecepatan. Nilai tersebut dapat diperoleh dari persamaan berikut :

$$selisih\ rata - rata = \frac{\Sigma\ \text{selisih}}{\text{banyaknya data}} \qquad (4.1)$$

Penggunaan 4 aturan, selisih rata – rata kecepatannya lebih kecil dari penggunaan 3 dan 5 aturan. Hasil penelitian ini membuktikan bahwa simulasi dengan sistem automatisasi *fuzzy* metode gradien sebagai kendali kecepatan motor DC sudah cukup baik namun bukan yang terbaik. Penggunaan variasi aturan berpengaruh terhadap perubahan parameter masukan dan keluaran, hasil respon sistem, dan selisih rata – rata kecepatan

Dalam pelatihan tidak menggunakan seluruh data. Data yang digunakan sebanyak 500 data berurutan. Hal ini juga disebabkan keterbatasan komputer yang digunakan saat pelatihan dan simulasi, sehingga dengan data masukan yang banyak, perlu dicoba menggunakan komputer dengan tingkat proses komputasi yang lebih tinggi.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil simulasi, analisa dan pembahasan yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Pada proses pelatihan dengan 500 data berurutan, jika *iterasi* diperbesar maka waktu pelatihan bertambah lama, *error* pelatihannya semakin kecil.
2. Pada pengendalian kecepatan motor DC dengan sistem automatisasi *fuzzy* metode gradien menghasilkan sistem kendali yang cukup baik. Penggunaan 3 aturan rata – rata selisih kecepatan sebesar 31.58 rad / s. Penggunaan 4 aturan rata – rata selisih kecepatan sebesar 3.15 rad / s. Penggunaan 5 aturan rata – rata selisih kecepatan sebesar 7.53 rad / s.
3. Pada hasil *step respon* dengan penggunaan 5 aturan menghasilkan respon sistem yang lebih cepat dari penggunaan 3 aturan dan 4 aturan.
4. Penggunaan variasi aturan berpengaruh terhadap perubahan parameter masukan dan keluaran, hasil respon sistem, dan selisih rata – rata kecepatan.
5. Respon sistem setelah pelatihan dengan 3 aturan menghasilkan *rise time* 6.7192 s , *peak time* 99.8007 s, *maksimum overshoot* 0.0104 s , *settling time* 12.4683 s. Penggunaan 4 aturan menghasilkan *rise time* 6.5987s , *peak time* 99.9484 s, *maksimum overshoot* 0.0104 s , *settling time* 12.3139

s. Penggunaan 5 aturan menghasilkan *rise time* 6.5633 s , *peak time* 99.4044 s, *maksimum overshoot* 0.0104 s , *settling time* 12.2748 s.

6. Penggunaan 4 aturan, selisih rata – rata kecepatannya lebih kecil dari penggunaan 3 dan 5 aturan.
7. Pada proses pengujian terdapat osilasi ketika pengujian menggunakan nilai selain 1750. Hal ini disebabkan oleh penggunaan 9 aturan direduksi menjadi 3, 4 dan 5 aturan yang diperoleh secara *trial and error*.

5.2 Saran

Masukan dan saran sangat bermanfaat untuk pengembangan lebih lanjut dari sistem ini, berikut beberapa masukan dan saran yang dapat dipertimbangkan:

1. Perlunya aplikasi ini untuk dibuat dalam perangkat keras (*hardware*), agar diketahui sistem automatisasi *fuzzy* metode gradien dapat bekerja dengan baik pada sistem yang sebenarnya.
2. Penggunaan sistem automatisasi *fuzzy* metode gradien untuk sistem yang lain.
3. Untuk mendukung proses pelatihan agar dapat berlangsung lebih cepat dari penelitian ini, diperlukan komputer / laptop minimal dengan spesifikasi yang lebih baik dari 2 *peripheral* berikut :
   - *Processor* : Intel *Core Duo*
   - RAM : 1 GB