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*.

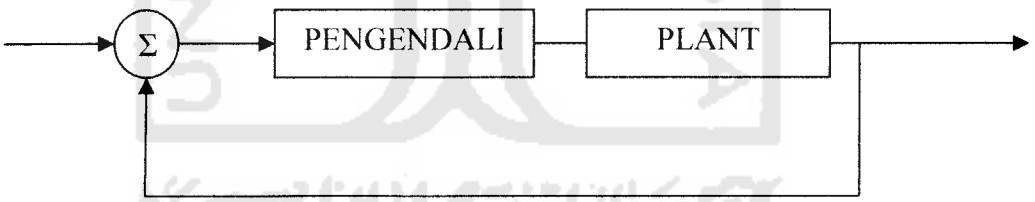
Pengujian sistem menggunakan blok simulink Matlab. Pada blok motor DC menggunakan blok *DC machine*. Parameter dari motor DC diperoleh dari motor DC sebenarnya yang terdapat pada Laboratorium Instalasi dan Mesin Listrik. Masukan ANFIS dan JST adalah kecepatan sedangkan keluarannya adalah tegangan. Setelah 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

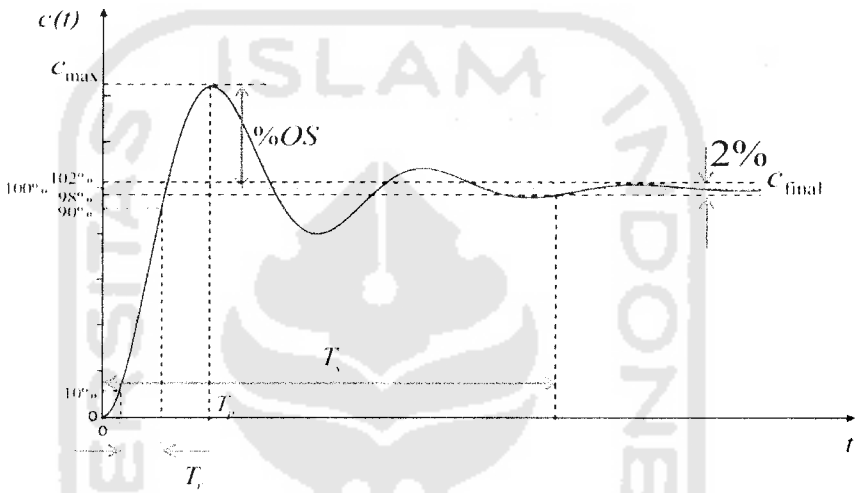


Gambar 2.1 Blok diagram Sistem Kendali

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 :

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

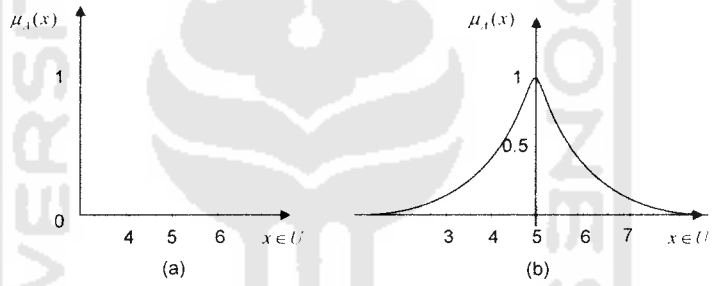
yang ditunjukkan oleh gambar 2.3 (a).

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

yang mana fungsi karakteristiknya adalah :

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

yang ditunjukkan oleh gambar 2.3 (b).

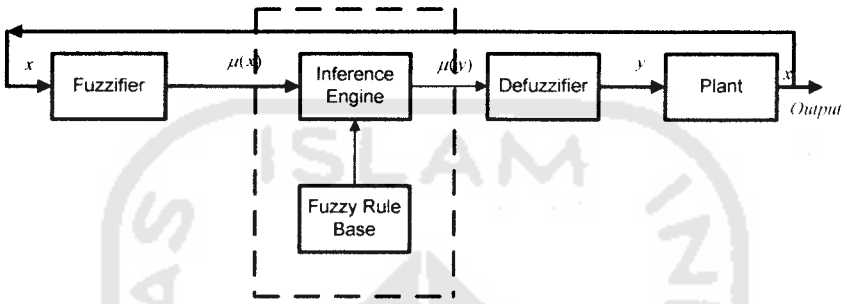


Gambar 2.3 Fungsi karakteristik himpunan klasik A dan himpunan *fuzzy* $\tilde{A}$

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

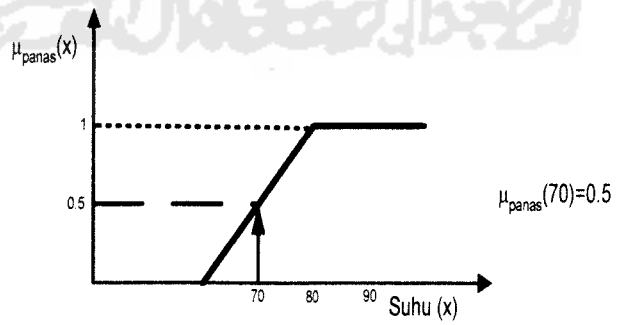
Tujuan dari sistem pengendali *fuzzy* adalah untuk menghitung nilai variabel kendali atau aksi dari variabel keadaan hasil observasi atau pengukuran proses yang dikendalikan sedemikian rupa sehingga diperoleh hasil yang diinginkan. Sehingga, pemilihan secara tepat dari variabel keadaan dan juga variabel 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 :

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

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.

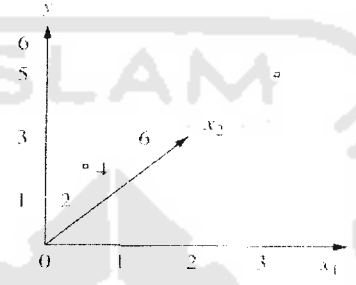
Masukan dari proses defuzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *keluaran* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka harus dapat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

2.5 Sistem Automatisasi *Fuzzy*

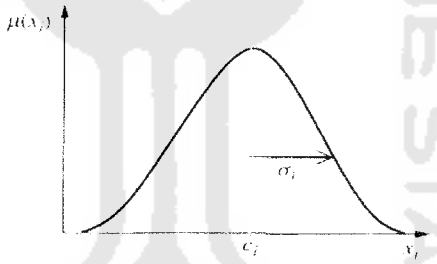
Untuk mengetahui sebuah sistem, analisis sangat bergantung pada informasi dan banyaknya pengetahuan yang diperoleh dari penelitian atau hasil percobaan untuk mengembangkan sebuah model dan untuk memprediksi keluaran. Tetapi untuk sistem yang baru, sedikitnya pengetahuan dan penelitian seorang analisis 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 $\mu(x)$,

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

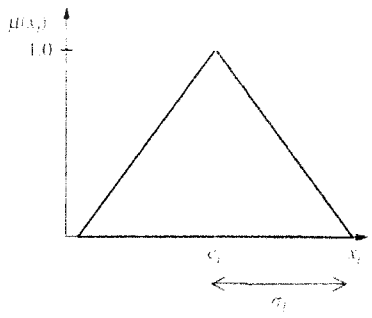
x_i adalah variabel masukan, c_i adalah pusat fungsi keanggotaan (fungsi keanggotaan mencapai nilai maksimum), σ_i adalah relasi konstan penyebaran fungsi keanggotaan. Gambar 2.7 ilustrasi tipe fungsi keanggotaan gaussian dan parameternya.



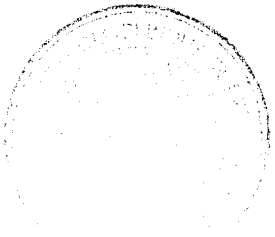
Gambar 2.6 Contoh dari 2 masukan, 1 keluaran untuk 3 titik

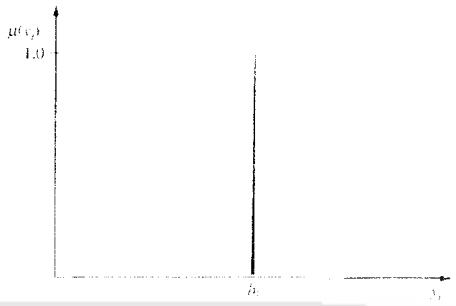


Gambar 2.7 Fungsi keanggotaan Gaussian



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 :

$$f(x|\theta) = \frac{\sum_{i=1}^{R} b_i \prod_{j=1}^{n} \exp\left[-\frac{1}{2}\left(\frac{x_i - c_j^i}{\sigma_j^i}\right)^2\right]}{\sum_{i=1}^{R} \prod_{j=1}^{n} \exp\left[\frac{-1}{2}\left(\frac{x_j - c_j^i}{\sigma_j^i}\right)^2\right]} \qquad (2.6)$$

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_1 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.

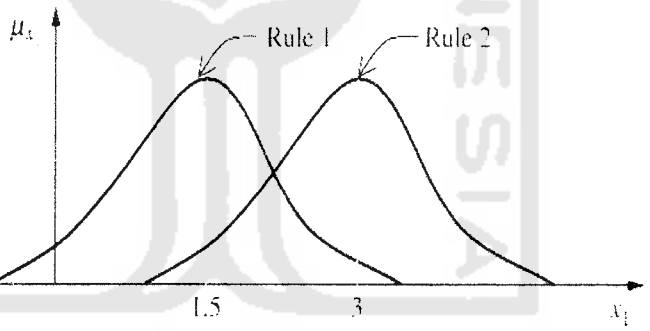
Langkah selanjutnya adalah memilih *respective spread,* σ_j^i, untuk fungsi keanggotaan yang di seleksi. Sebagai contoh $\sigma_j^i = 2$, untuk i =1, 2 dan j = 1, 2, Fungsi keanggotaan masukan untuk *Rule 1* dan *Rule 2* adalah fungsi keanggotaan Gaussian dan ditampilkan pada Gambar 2.10 dan 2.11. Fungsi 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 :

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

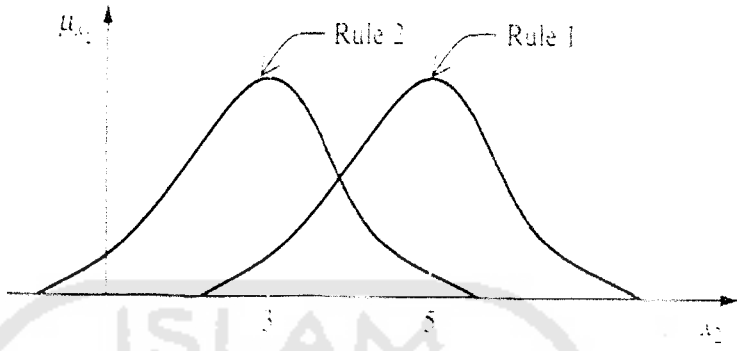
Dimana n = 2, x adalah masukan data dan c_j^i dan μ_j^i adalah parameter basis aturan.

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

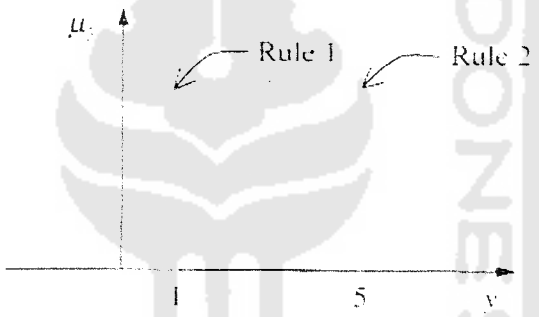
$$f(x \mid \theta) = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \qquad (2.8)$$



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 ($x^m|\theta$), dengan nilai keluaran aktual (y^m), maka fungsi kuadrat *error* e^m, dinamakan " *error surface* ".

Persamaan untuk *error surface* sebagaimana pada persamaan 2.9.

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

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 :

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

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) :

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

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

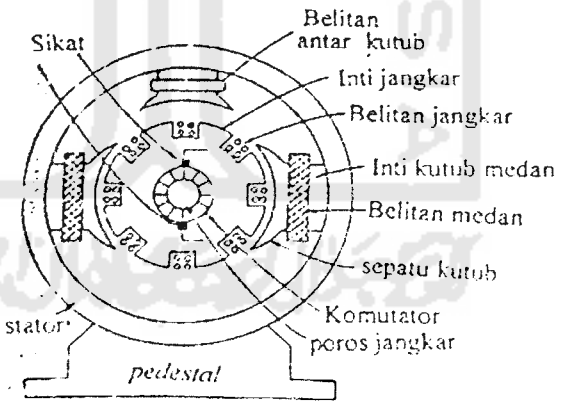
$$\sigma_j^i(k) = \sigma_j^i(k-1) - \lambda_3 * \varepsilon_k(k-1) * \left(\frac{b_i(k-1) - f(x^k I\theta(k-1))}{\sum_{i=1}^{R} \mu_i(x^k, k-1)} \right) * \mu_i(x^k, k-1) * \left(\frac{(x_j^k - c_j^i(k-1))^2}{(\sigma_j^i((k-1))^3} \right) \qquad (2.13)$$

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 beukuran 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)

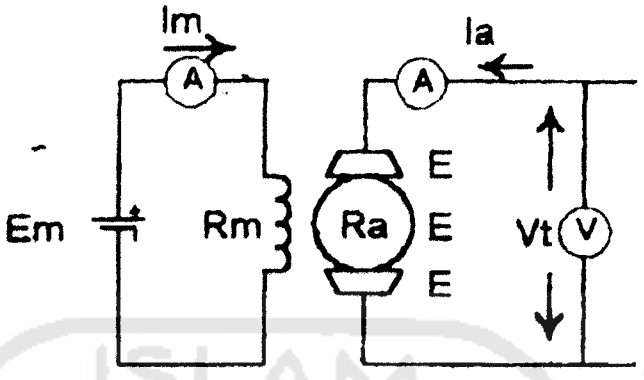
2.7.2 Prinsip Motor DC

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

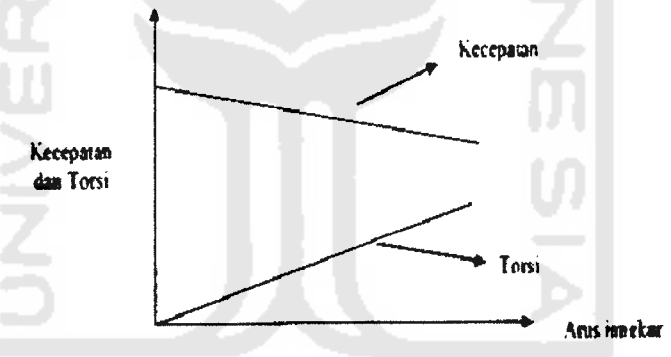
Ra = Resistansi Jangkar

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.