UNIVERSITAS INDONESIA

PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN UNTUK KENDALI VEKTOR MOTOR INDUKSI TIGA FASA

SKRIPSI

GEORGE HUTAURUK
0606073934

FAKULTAS TEKNIK
DEPARTEMEN TEKNIK ELEKTRO
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2011

UNIVERSITAS INDONESIA

PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN UNTUK KENDALI VEKTOR MOTOR INDUKSI TIGA FASA

SKRIPSI

Skripsi ini diajukan untuk memenuhi persyaratan menjadi
Sarjana Teknik

GEORGE HUTAURUK

0606073934

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

DESEMBER 2011

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama : George Hutauruk
NPM : 0606073934
Tanda Tangan :

Tanggal : 19 Januari 2012

HALAMAN PENGESAHAN

SKRIPSI INI DIAJUKAN OLEH :

NAMA	: George Hutauruk
NPM	: 0606073934
PROGRAM STUDI	: Teknik Elektro
JUDUL SKRIPSI	: PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN UNTUK KENDALI VEKTOR MOTOR INDUKSI TIGA FASA

TELAH DIPRESENTASIKAN DAN DITERIMA SEBAGAI PERSYARATAN YANG DIPERLUKAN UNTUK MEMPEROLEH GELAR SARJANA TEKNIK PADA PROGRAM STUDI TEKNIK ELEKTRO, FAKULTAS TEKNIK, UNIVERSITAS INDONESIA

Pembimbing : Dr. Abdul Halim, M.Eng .....)

Penguji : Prof. Drs. Benyamin Kusumoputro MEng., Dr.Eng..(.....)

Penguji : Ir. Aries Subiantoro M.SEE.....(.....)

Ditetapkan di : Depok

Tanggal : 19 Januari 2012

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, proses penulisan laporan seminar ini dapat terselesaikan. Penulis juga mengucapkan banyak terima kasih kepada:

(1). Dr. Abdul Halim, M. Eng, sebagai dosen pembimbing yang telah memberikan waktu, tenaga dan pikiran dalam proses penyelesaian skripsi ini.
(2). Dr. Ir. Ridwan Gunawan, MT, yang telah memberikan perhatian dan rujukan literatur selama seminar dan skripsi.
(3). Orang tua dan keluarga saya yang telah memberikan bantuan berupa dukungan material dan moral.
(4). Teman – teman elektro 2006 yang telah membantu dan memberikan dukungan semangat juga diskusi.
(5). Keluarga besar PSKJ atas kebersamaan, doa dan dukungan yang telah diberikan.
(6). Seluruh pihak yang menolong dalam penyelesaian skripsi ini.

Akhir kata, saya berharap agar Tuhan Yang Maha Esa berkenan membalas segala kebaikan dari semua pihak yang telah membantu. Semoga skripsi ini dapat memberikan manfaat bagi pengembangan ilmu.

Depok, 27 Desember 2011

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini :

Nama	: George Hutauruk
NPM	: 0606073934
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (***Non-exclusive Royalty-Free Right***) atas karya ilmiah saya yang berjudul :

PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN UNTUK KENDALI VEKTOR MOTOR INDUKSI TIGA FASA

Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis / pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok,

Pada tanggal : 27 Desember 2011,

Yang menyatakan,

(George Hutauruk)

ABSTRAK

Nama : George Hutauruk
Program Studi : Teknik Elektro
Judul :PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN UNTUK KENDALI VEKTOR MOTOR INDUKSI TIGA FASA

Motor induksi tanpa sensor kecepatan digunakan secara luas pada bidang industri. Skripsi ini memiliki tujuan untuk merancang estimator kecepatan motor induksi menggunakan jaringan syaraf tiruan. Jaringan syaraf tiruan yang digunakan berstruktur lapisan banyak terhubung semua dan dilatih dengan algoritma *backpropagation*. Masukan jaringan adalah tegangan dan arus stator dari motor induksi. Model motor yang digunakan berada pada sumbu stator, sedangkan pengendali vektor menggunakan persamaan model motor dalam kerangka fluks rotor. Pada percobaan dilakukan variasi parameter untuk diperoleh kinerja yang optimal. Kemudian dilakukan percobaan pada beberapa kondisi kerja untuk mengetahui kemampuan dari jaringan yang telah dirancang. Hasil simulasi menunjukan estimator kecepatan berbasis jaringan syaraf tiruan dengan CMEX S-Function Matlab/Simulink 7.8.0 memberikan hasil yang baik.

Kata kunci : Motor induksi, estimator kecepatan, jaringan syaraf tiruan

ABSTRACT

Name : George Hutauruk
Study Program : Electrical Engineering
Title : Speed Estimator Design Based on Neural Network for Vector Control of Three Phase Induction Motor

Sensorless induction motor is used widely in industrial fields. This thesis has the aim to design an induction motor speed estimator based on neural network. Artificial neural network that being used are the multilayer fully connected of structure and trained with the backpropagation algorithm. The network input is the voltage and current stator of an induction motor. Model motor that being used are in stationary reference frame, while the vector control using the motor model equations in rotor flux oriented reference frame. In the experiments, some variations of parameters are carried out to obtain optimal performance. Further, experiments on some working conditions to determine the ability of the speed estimator that has been designed. The results of the simulation shows the speed estimator based on neural network with S-Function CMEX Matlab/Simulink 7.8.0 is in good performance.

Keyword : Induction motor, speed estimator, neural network

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Motor induksi adalah motor yang paling banyak digunakan pada industri, karena kesederhanaannya, konstruksi yang kokoh, dan biaya pembuatan juga perawatannya yang relatif murah. Hal ini terjadi karena motor ini berdiri sendiri, tidak terhubung dengan luar. Hal ini dimungkinkan karena arus rotor dihasilkan dengan induksi putaran medan magnet. Bersamaan dengan makin banyaknya penggunaan motor induksi, untuk mendapatkan kinerja yang bagus maka pengembangan metode pengendalian motor induksi terus dilakukan.

Pada masa sekarang, banyak dilakukan penelitian untuk meniadakan sensor kecepatan pada motor induksi, tetapi tanpa menurunkan respons dinamisnya. Perkiraan kecepatan merupakan hal yang menarik untuk dibahas pada penggerak motor induksi karena kecepatan mekanis rotor berbeda dengan kecepatan medan magnet. Keuntungan dari penggerak motor induksi tanpa sensor adalah menurunnya kerumitan perangkat keras dan lebih rendahnya biaya, mesin penggerak yang lebih kecil, tidak diperlukannya kabel sensor, ketahanan terhadap gangguan yang lebih baik, peningkatan kehandalan dan kebutuhan perawatan yang rendah. Tetapi motor induksi tanpa sensor juga memiliki kekurangan yaitu jika digunakan pada frekuensi rendah, karena estimator tidak dapat memperoleh nilai variabel yang diperlukannya untuk mengetahui kecepatan motor induksi.

Pengendalian motor induksi bukanlah hal yang mudah, karena terdapat beberapa permasalahan yaitu, mesin yang tidak linear, ketidakmampuan mengukur variabel kondisi, beberapa parameter mesin terutama hambatan rotor yang memiliki nilai yang sangat bervariasi karena perubahan suhu dan *skin effect*.

Pada saat ini banyak pemecahan masalah yang dikembangkan untuk mengatasi kerumitan pengendalian tersebut. Salah satunya yang sedang berkembang adalah teknologi jaringan syaraf tiruan, yaitu jaringan pemproses yang dimodelkan berdasarkan jaringan syaraf manusia, merupakan sistem adaptif yang dapat memodifikasi dirinya untuk memecahkan masalah berdasarkan pelatihan nilai input dan output yang diberikan kepadanya.

Penggunaan jaringan syaraf tiruan dirasakan tepat untuk menghadapi permasalahan tersebut karena teknologi ini memiliki kemampuan untuk memodelkan sistem yang tidak linear seperti motor induksi. Kelebihan lainnya adalah kemampuannya untuk memperoleh, dan menyimpan informasi berdasarkan pengalaman. Sehingga pengembangan dan penelitian mengenai hal ini menjadi sangat diperlukan.

1.2. Tujuan Penulisan

Tujuan penulisan skripsi ini adalah merancang estimator kecepatan pada motor induksi tanpa sensor berbasis jaringan syaraf tiruan. Selain itu juga memaparkan pengaruh perubahan struktur yaitu jumlah unit pada lapisan tersembunyi, laju pembelajaran dan perbandingan unjuk kerja dengan yang menggunakan sensor kecepatan pada beberapa kondisi pemakaian.

Hal itu dilakukan dengan terlebih dahulu menjelaskan pemodelan motor induksi dalam kerangka acuan stator, prinsip kerja dan penggunaan pengendalian vektor, juga mengenai algoritma pembelajaran backpropagation jaringan syaraf tiruan.

1.3. Pembatasan Masalah

Pada skripsi ini permasalahan dibatasi hanya pada penggunaan kendali vektor mengarahkan fluks rotor dengan jaringan syaraf tiruan sebagai estimator kecepatan untuk motor induksi tiga fasa. Pembatasan yang dilakukan adalah sebagai berikut :

1. Konstruksi rotor yang digunakan adalah jenis sangkar bajing.
2. Model motor induksi menggunakan kerangka acuan stator.
3. Inverter yang digunakan untuk membangkitkan sinyal PWM dianggap ideal sehingga tidak memiliki waktu tunda.
4. Motor induksi dijalankan pada wilayah kecepatan normal.
5. Tidak terjadi perubahan parameter.
6. Program yang digunakan untuk simulasi adalah Simulink C-MEX MATLAB 7.8.0.

1.4. Metodologi Penulisan

Penelitian ini dilakukan dengan dua metodologi, yaitu studi pustaka untuk mempelajari dasar teori dari permasalahan yang ada juga pengamatan dan simulasi dengan menggunakan SIMULINK MATLAB.

1.5. Sistematika Penulisan

Sistematika penulisan skripsi dibagi menjadi lima bagian, antara lain:

a. Pada bab pertama akan dijelaskan latar belakang, tujuan penulisan, batasan masalah, metodologi penulisan, dan sistematika penulisan.
b. Bab dua akan menjelaskan teori dasar yang digunakan untuk penelitian ini yaitu mengenai motor induksi tiga fasa dengan pemodelan pada kerangka acuan stator dan kendali vektor berdasarkan arah fluks rotor.
c. Bab tiga akan menjelaskan mengenai skema pengendalian, perancangan pengendalian, jaringan syaraf tiruan, berbagai kondisi operasi yang akan diuji dan perancangan estimator yang diterapkan pada skripsi ini.
d. Bab empat yaitu analisa hasil dan pembahasan mengenai data – data hasil percobaan, analisis, dan pembahasan unjuk kerja dari estimator yang dirancang.
e. Bab lima adalah kesimpulan dan saran yang diperoleh dari skripsi ini.

BAB 2
TEORI DASAR

Bab ini akan membahas motor induksi tiga fasa mengenai konstruksi, prinsip kerja dan pemodelan matematiknya, pengendalian vektor, dasar dari jaringan syaraf tiruan dan metode pelatihan backpropagation. Hal lain yang juga dibahas adalah transformasi kerangka acuan dan penggerak yang dibutuhkan untuk pengendalian vektor.

Pemodelan motor induksi yang digunakan berdasarkan kerangka acuan stator sedangkan untuk kendali vektor berdasarkan arah fluks rotor. Pemodelan ini kemudian digunakan dalam simulasi.

2.1. Motor Induksi Tiga Fasa

2.1.1 Umum

Motor induksi adalah motor arus bolak-balik yang bekerja dengan menggunakan prinsip induksi elektromagnet. Pada motor induksi, arus bolak-balik disuplai ke lilitan stator secara langsung yang kemudian menghasilkan arus pada lilitan rotor karena terinduksi. Disebut juga sebagai motor tak serempak karena kecepatan putaran mekanik rotor tidak sama dengan kecepatan putaran medan putar stator.

Rangkaian ekivalen dari motor induksi sama dengan transformator, karena memiliki prinsip kerja yang sama yaitu rangkaian primer menginduksi rangkaian sekunder sehingga menimbulkan tegangan pada rotor. Tetapi pada transformator rangkaian sekunder bukan merupakan bagian yang dapat berputar. Pada perancangan pengendali dibutuhkan juga pemodelan dalam bentuk rangkaian dinamis, supaya dapat menggambarkan proses putar dan lebih sesuai dengan keadaan sebenarnya.

Rangkaian dinamis diperoleh dengan cara mentransformasi model tiga fasa ke dua fasa. Transformasi yang dilakukan berdasarkan prinsip trigonometri sederhana dengan memproyeksikannya ke sumbu dua fasa yaitu sumbu dq0. Rangkaian ini tidak diam, tetapi berputar pada suatu kerangka kecepatan tertentu.

Sehingga diperoleh kondisi sistem atau nilai parameter dalam kondisi sesaatnya. Dengan menggunakan rangkaian ini dapat dilakukan analisa respons dinamis dan respons alih.

2. 1.2 Konstruksi

Motor induksi memiliki dua bagian utama, yaitu stator dan rotor. Stator adalah bagian yang tidak berputar yang dihubungkan dengan sumber tegangan sedangkan rotor adalah bagian yang berputar yang tegangannya diperoleh dari induksi medan putar stator.

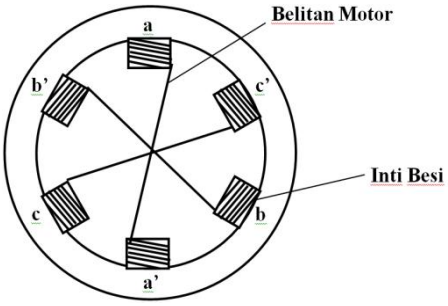
Bagian utama yang lain dari motor induksi adalah rotor. Jenis rotor untuk motor induksi ada dua macam, yaitu rotor belitan (*wound rotor*) dan rotor sangkar bajing (*squirrel cage rotor*). Pada skripsi ini digunakan rotor sangkar bajing. Rotor tipe sangkar bajing memiliki belitan yang didistibusikan pada setiap batang konduktor yang kedua ujungnya terhubung dengan cincin tembaga (shorting rings) sehingga setiap batang konduktor saling terhubung singkat. Motor induksi dengan tipe ini memiliki konstruksi yang lebih sederhana dan kuat dibandingkan dengan tipe belitan.

2. 1.3 Prinsip Kerja

Motor induksi bekerja berdasarkan prinsip induksi elektromagnetik. Pada stator motor induksi tiga fasa terdapat tiga kumparan fasa, yaitu a-a', b-b', dan c-c'. Pada contoh ini digunakan enam buah kumparan, dua buah kumparan untuk setiap fasa. Bekerja secara berpasangan, dihubungkan demikian sehingga jika dialiri arus maka akan menghasilkan kutub magnet yang berlawanan. Dapat dilihat pada Gambar 2.1 setiap pasangan kumparan secara fisik berbeda 120^o. Kumparan stator tersebut dihubungkan dengan sumber tegangan tiga fasa.

Jumlah pasangan kutub dapat lebih dari satu pasang. Jika jumlah pasangan kutub Np ditambah menjadi dua kali lipat dari maka untuk satu kali putaran elektrik θ_e akan terjadi dua kali putaran mekanis θ_m. [1]

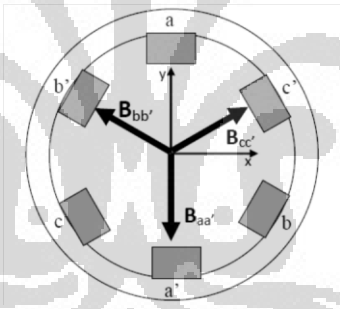
$$\theta_e = N_p \, \theta_m \qquad (2.1)$$



Gambar 2.1 Stator dengan Kumparan

Pada motor induksi tiga fasa, sumbu dari kumparan fasanya terpisah sejauh 120^0 listrik satu sama lain, sehingga $\theta_b = \theta_a - 120$ dan $\theta_c = \theta_a + 120$ maka pada kondisi setimbang arus-arus fasa bernilai sama tetapi dengan beda fasa 120^0. dimana I_m adalah nilai maksimum dari arus dan nilai waktu awal dipilih ketika arus fasa a adalah positif maksimum.

Arus sinusoidal menghasilkan medan magnet yang bersifat sama dengannya ketika dialirkan ke suatu kumparan. Pasangan stator menghasilkan medan magnet sinusoidal yang memiliki perbedaan fasa dan posisi sebesar 120^o.



Gambar 2.2 Medan magnet stator

Gambar 2.2 menunjukan medan magnet yang dihasilkan oleh setiap pasangan kumparan. Medan magnet dari setiap fasa diarahkan ke sumbu – xy. Kemudian setelah menjumlahkan setiap komponen x dan y juga menggunakan identitas trigonometri, persamaan medan total sebagai berikut

$$\vec{B}_{net} = (1.5\, B_M \sin \omega t)\, \hat{\mathbf{x}} - (1.5\, B_M \cos \omega t)\hat{\mathbf{y}} \qquad (2.2)$$

Dari persamaan medan magnet total yang diperoleh dapat diketahui bahwa medan magnet tersebut berputar dengan besar yang tetap dan kecepatan yang sama dengan frekuensi arus stator. Arus yang mengalir dalam kumparan stator menghasilkan medan magnet putar. Medan magnet putar kemudian memotong

batang-batang konduktor kumparan pada rotor, sehingga menghasilkan tegangan induksi.

Tegangan induksi yang dihasilkan menimbulkan arus rotor. Bersama dengan medan magnet putar stator dihasilkan torsi. Jika torsi yang dihasilkan lebih besar dari beban maka rotor dapat berputar. Pada kondisi diberikan gaya putar, motor akan mengalami percepatan terus menerus sampai suatu batas maksimum, yaitu kecepatan sinkronnya. Karena jika rotor berputar pada kecepatan sinkron, maka posisi rotor terhadap medan stator relatif diam. Sehingga tidak akan ada tegangan induksi dan juga arus yang dihasilkan. Jika hal itu terjadi, maka torsi induksi juga bernilai nol dan kecepatan rotor akan turun karena gesekan. Perbedaan relatif kecepatan putar rotor dengan kecepatan medan magnet putar stator disebut slip (s). Sehingga kecepatan stator tidak mungkin sama dengan kecepatan rotor.

2.2 Model Matematika Motor Induksi Tiga Fasa

2.2.1 Rangkaian Pengganti

Motor induksi memiliki rangkaian pengganti yang mirip dengan transformator. Rangkaian pengganti terdiri bagian stator dan rotor yang saling terhubung. Pada bagian stator dari motor induksi terdapat kebocoran induksi yang dimodelkan sebagai reaktansi kebocoran induksi stator X_{ls} dan juga panas yang terbuang, digunakan hambatan stator R_s. Besar tegangan terminal stator berbeda dengan tegangan emf karena adanya tegangan jatuh karena kebocoran impendansi. Sebagai pengganti fluks stator yang menginduksi rangkaian rotor digunakan reaktansi magnetisasi X_m.

Hubungan fasor pada sebuah fasa stator adalah sebagai berikut [1]

$$V_s = E_1 + I_s(R_s + jX_{ls}) \quad (2.3)$$

Seperti pada stator, bagian rotor juga mengalami penurunan tegangan karena adanya kebocoran induksi dan perubahan menjadi panas yang dimodelkan dengan menggunakan reaktansi kebocoran induksi rotor X_{lr} dan hambatan rotor

R_r. Karena rotor dihubung singkat maka fasor antara tegangan, arus, dan slip pada sebuah fasa adalah

$$\frac{E_2}{I_r} = Z_r = R_r + jsX_{lr} \quad (2.4)$$

dimana Z_r adalah impendansi rotor pada setiap fasa terhadap stator, R_r adalah hambatan rotor, dan s X_{lr} sebagai reaktansi kebocoran rotor pada frekuensi slip. Reaktansi dituliskan seperti itu karena memiliki nilai yang berubah sesuai dengan frekuensi rotor dan juga slip.

Sehingga dapat dibentuk rangkaian pengganti yang menghubungkan stator dengan rotor. Kecepatan relatif fluks terhadap kecepatan rotor adalah s dari kecepatannya terhadap stator, hubungan dari nilai emf stator dengan rotor adalah sebagai berikut

$$E_{2s} = s\, E_1 \quad (2.5)$$

Sedangkan arus dari stator memiliki nilai yang sama dengan arus rotor jika diasumsikan tidak ada kebocoran,

$$I_{rs} = I_r \quad (2.6)$$

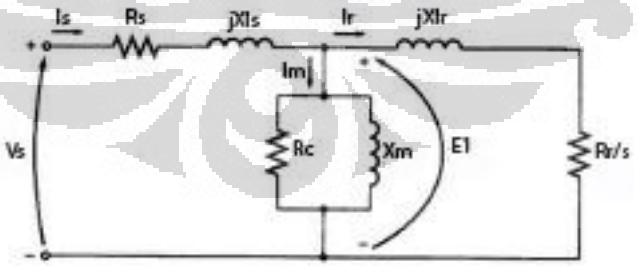
Pembagian antara persamaan 2.5 dengan persamaan 2.6 menghasilkan

$$\frac{E_2}{I_{rs}} = \frac{sE_1}{I_r} \quad (2.7)$$

Jika menggabungkan persamaan 2.4 dengan persamaan 2.7 diperoleh

$$\frac{E_1}{I_r} = \frac{R_r}{s} + jX_{lr} \quad (2.8)$$

Rangkaian pengganti motor induksi ditunjukan pada Gambar 2.3.

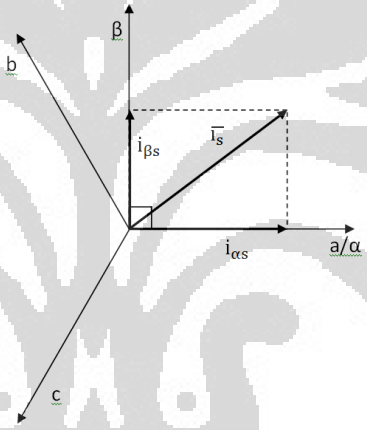


Gambar 2.3 Rangkaian Pengganti untuk Motor Induksi

2.2.2 Transformasi Kerangka Acuan

Motor induksi yang digunakan adalah tiga fasa sedangkan model yang dikembangkan sampai saat ini dua fasa. Sehingga digunakan metode kerangka acuan untuk mentransformasikan besaran dari kerangka *abc* menjadi besaran dalam kerangka berputar *dq0*. Teori ini dapat digunakan untuk mentransformasi besaran-besaran fasa seperti tegangan, arus, dan fluks gandeng. Jika sistem fasa yang ditransformasikan merupakan sistem yang seimbang maka nilai komponen nol pada kuadratur akan sama dengan nol.

Ruang vektor tiga fasa dapat ditampilkan dalam kerangka dua sumbu tegak lurus diam (αβ). Jika sumbu a diasumsikan berhimpit dengan sumbu α maka Gambar 2.4 menunjukan diagram dari kerangka ini.



Gambar 2.4 Diagram arus stator pada sumbu abc dan αβ

Sehingga matriks proyeksi arus stator kerangka tiga fasa menjadi kerangka dua fasa diam adalah sebagai berikut. Transformasi ini dapat digunakan untuk parameter yang lain, tetapi untuk kemudahan pada saat ini digunakan arus. [2]

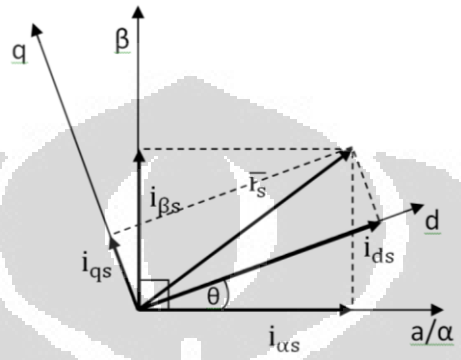
$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \quad (2.9)$$

Dan matriks proyeksi arus stator kerangka dua fasa diam terhadap kerangka tiga fasa adalah sebagai berikut.

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix}\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} \quad (2.10)$$

Konstanta sebesar $\frac{2}{3}$ digunakan untuk *system power non-invariant* sedangkan untuk *system power invariant*, konstanta tersebut bernilai $\sqrt{\frac{2}{3}}$.

Kemudian dilakukan transformasi untuk mengubah kerangka dua fasa diam menjadi kerangka dua fasa bergerak dq. Dengan sumbu-d diasumsikan berhimpit dengan fluks maka Gambar 2.5 menunjukkan diagram kerangka ini.



Gambar 2.5 Diagram arus stator pada sumbu αβ dan dq

Berdasarkan diagram tersebut maka persamaan proyeksi kerangka dua fasa diam terhadap kerangka dua fasa bergerak adalah sebagai berikut.

$$\begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} \quad (2.11)$$

Dan sebaliknya matriks proyeksi kerangka dua fasa bergerak terhadap kerangka dua fasa diam adalah sebagai berikut.

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \quad (2.12)$$

Jika kedua transformasi yang dibahas sebelumnya digunakan maka diperoleh transformasi pengganti dq dari kerangka dua fasa bergerak sebagai berikut :

$$\begin{bmatrix} i_q \\ i_d \\ i_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta_c & \cos\left(\theta_c - \frac{2\pi}{3}\right) & \cos\left(\theta_c + \frac{2\pi}{3}\right) \\ \sin\theta_c & \sin\left(\theta_c - \frac{2\pi}{3}\right) & \sin\left(\theta_c + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \quad (2.13)$$

Posisi kerangka acuan θ_c untuk pengendalian dapat digunakan sudut kerangka acuan pada saat diam, fluks rotor, atau fluks stator.

2.2.3 Pemodelan Dinamis Motor Induksi

Pengendalian motor induksi tanpa menggunakan pemodelan dinamis adalah hal yang sulit. Karena untuk mengetahui sifat dinamis dari sebuah penggerak listrik dibutuhkan pemodelan dalam kondisi bergerak. Seperti koefisien magnetik gandeng antara stator dan rotor yang bervariasi terhadap waktu, dan hal ini tidak dapat diperoleh jika tidak menggunakan model dinamis.

Persamaan dari model umum motor induksi ditunjukan dalam persamaan tegangan dan fluks stator sebagai berikut [3]:

$$\bar{v}_s = R_s\bar{i}_s + p\bar{\lambda}_s + j\omega_c\bar{\lambda}_s \quad (2.14)$$

$$\bar{v}_r = R_r\bar{i}_r + p\bar{\lambda}_r + j(\omega_c - \omega_r)\bar{\lambda}_r \quad (2.15)$$

$$\lambda_s = L_{ls}i_s + L_m(i_s + i_r) = L_s i_s + L_m i_r \quad (2.16)$$

$$\lambda_r = L_{lr}i_r + L_m(i_r + i_s) = L_s i_r + L_m i_s \quad (2.17)$$

$$i_r = \frac{\lambda_r - L_m i_s}{L_r} \quad (2.18)$$

Rangkaian pengganti yang sudah diperoleh merupakan pemodelan pada bagian elektris, Sedangkan untuk memperoleh besar torsi dibutuhkan pemodelan mekanis. Pemodelan untuk bagian mekanis dilakukan dengan menggunakan besar daya pada stator dan rotor.

$$P = v_{as}i_{as} + v_{bs}i_{bs} + v_{cs}i_{cs} + v_{ar}i_{ar} + v_{br}i_{br} + v_{cr}i_{cr} \quad (2.19)$$

Jika ditransformasi pada kerangka dq0 dengan *system power invariant* persamaan daya yang diperoleh adalah sebagai berikut

$$P = v_{qs}i_{qs} + v_{ds}i_{ds} + v_{0s}i_{0s} + v_{qr}i_{qr} + v_{dr}i_{dr} + v_{0r}i_{0r} \quad (2.20)$$

Karena diasumsikan sistem seimbang pada stator maupun rotor, maka komponen 0 dapat dihilangkan. Sehingga persamaan daya pada kerangka dq0s pada sistem seimbang adalah

$$P = v_{qs}i_{qs} + v_{ds}i_{ds} + v_{qr}i_{qr} + v_{dr}i_{dr} \quad (2.21)$$

Persamaan daya induksi pada kerangka acu yang digunakan untuk memperoleh persamaan torsi hanya komponen daya putar saja.

$$P_{mek} = \omega_r T = \{i_{dr}(L_m i_{qs} + L_r i_{qr}) - i_{qr}(L_m i_{ds} + L_r i_{dr})\}(\omega_e - \omega_r) \quad (2.22)$$

Dengan mengetahui bahwa P/2 = (ω_e - ω_r) /ω_r maka dapat diperoleh persamaan torsi elektromagnetis motor induksi dan kecepatan rotor adalah

$$T_e = \frac{P}{2}(\lambda_{qr}i_{dr} - \lambda_{dr}i_{qr})$$

$$T_e = \frac{P}{2}\left(\lambda_{ds} i_{qs} - \lambda_{qs} i_{ds}\right)$$
$$= \frac{P}{2} L_m \left(i_{qs} i_{dr} - i_{ds} i_{qr}\right) \quad (2.23)$$

Kemudian digunakan untuk memperoleh persamaan kecepatan rotor.

$$\frac{d}{dt}\omega_r = \frac{T_e - T_l - B\omega_r}{J} \quad (2.24)$$

dimana T_l, B dan J berturutan adalah torsi beban, koefisien gesekan dan momen inersia.

Pengendalian motor induksi dapat dilakukan dengan beberapa cara, salah satu yang banyak digunakan adalah dengan mengendalikan arus yang diberikan. Pemodelan yang telah dilakukan sebelumnya dapat dilakukan kembali di bagian ini untuk merancang pengendali arus. Pemodelan dilakukan dengan menggunakan kerangka acuan fluks rotor.

Model motor induksi dalam kerangka acuan fluks rotor ($\omega_c = \omega_r$) menjadi seperti berikut [3].

$$\bar{v}'_s = \bar{v}_s e^{-j\theta_e} = v_{ds} + v_{qs} \quad (2.25)$$

$$\bar{i}'_s = \bar{i}_s e^{-j\theta_e} = i_{ds} + i_{qs} \quad (2.26)$$

$$\bar{\lambda}'_s = \bar{\lambda}_s e^{-j\theta_e} = \lambda_{ds} + \lambda_{qs} \quad (2.27)$$

$$\bar{\lambda}'_r = \bar{\lambda}_r e^{-j(\theta_e - \theta_r)} = \lambda_{dr} + \lambda_{qr} \quad (2.28)$$

Tanda petik (‘) digunakan sebagai tanda untuk parameter yang berada pada kerangka acuan fluks rotor. Persamaan di atas kemudian ditransformasikan ke kerangka acuan stator.

$$\bar{v}_s = \bar{v}'_s e^{j\theta_e} \quad (2.29)$$

$$\bar{i}_s = \bar{i}'_s e^{j\theta_e} \quad (2.30)$$

$$\bar{\lambda}_s = \bar{\lambda}'_s e^{j\theta_e} \quad (2.31)$$

$$\bar{\lambda}_r = \bar{\lambda}'_r e^{j(\theta_e - \theta_r)} \quad (2.32)$$

Pada kerangka acuan fluks rotor, sumbu q tidak menghasilkan fluks. Sehingga arus magnetisasi pada sumbu ini bernilai nol ($i_{mrq} = 0$) dan vektor i_{mr} akan sama dengan i_{mr}. Persamaan fluks ditransformasikan pada kerangka acuan fluks rotor.

$$\bar{\lambda}'_s = L_s \bar{i}_s + L_m \bar{i}_r \quad (2.33)$$

$$\bar{\lambda}'_r = L_r \bar{i}_r + L_m \bar{i}_s = L_m \bar{i}_{mr} \quad (2.34)$$

$$\bar{i}_{mr} = i_{mr} = \bar{i}_s + \frac{L_r}{L_m}\bar{i}_r \quad (2.35)$$

Setelah mensubtitusi persamaan fluks di atas kemudian mentransformasikannya ke kerangka acuan rotor diperoleh persamaan yang akan membantu dalam merancang pengendali arus.

$$\bar{v}_s = R_s\bar{i}_s + p\bar{\lambda}_s$$

$$\bar{v}'_s e^{j\theta_e} = R_s\bar{i}'_s e^{j\theta_e} + p\bar{\lambda}'_s e^{j\theta_e} = R_s\bar{i}'_s e^{j\theta_e} + e^{j\theta_e}p\bar{\lambda}'_s + j\omega_e\bar{\lambda}'_s \quad (2.36)$$

$$v_{ds} = R_s i_{ds} + L_s\sigma p i_{ds} - \omega_e L_s\sigma i_{qs} + L_s(1-\sigma)p i_{mr} \quad (2.37)$$

$$v_{qs} = R_s i_{qs} + L_s\sigma p i_{qs} + \omega_e L_s\sigma i_{ds} + L_s(1-\sigma)\omega_e i_{mr} \quad (2.38)$$

Persamaan tegangan di atas akan dilinierasi karena akan digunakan pengendali PI. Persamaan tegangan (2.37 – 2.38) tidak linear karena kedua sumbu dq saling mempengaruhi. Sehingga dilakukan dekopling untuk memisahkan komponen tersebut. Persamaan tegangan stator dipisahkan menjadi tegangan pada sumbunya dan ditambah dengan tegangan koplingnya.

$$v_{ds} = u_{ds} + v_{dc} \quad (2.39)$$

$$v_{qs} = u_{qs} + v_{qc} \quad (2.40)$$

v_{dc} dan v_{qc} merupakan tegangan kopling stator, sehingga persamaan tegangan setelah didekopling sebagai berikut:

$$u_{ds} = v_{ds} - v_{dc} = R_s i_{ds} + L_s\sigma p i_{ds} \quad (2.41)$$

$$u_{qs} = v_{qs} - v_{qc} = R_s i_{qs} + L_s\sigma p i_{qs} \quad (2.42)$$

Motor induksi yang digunakan bertipe sangkar bajing sehingga tegangan rotornya sama dengan nol. Persamaan tegangan rotor setelah ditransformasilkan ke dalam sumbu dq dan dipisahkan menjadi berikut:

$$\bar{v}_r = R_r\bar{i}_r + p\bar{\lambda}_r = 0 \quad (2.43)$$

$$0 = R_r\frac{L_m}{L_r}(i_{mr} - \bar{i}_s) + L_m p i_{mr} + j(\omega_e - \omega_r)L_m i_{mr} \quad (2.44)$$

$$i_{ds} = i_{mr} + \frac{L_r}{R_r}\frac{d}{dt}i_{mr} \quad (2.45)$$

$$i_{qs} = (\omega_e - N_p\omega_r)\frac{L_r}{R_r}i_{mr} \quad (2.46)$$

Kemudian dengan penyederhanaan diperoleh persamaan untuk kecepatan fluks rotor. Dan dari persamaan (2.23) dan (2.34) diperoleh persamaan untuk torsi dari model fluks.

$$\frac{d}{dt}i_{mr} = \frac{R_r}{L_r}(i_{ds} - i_{mr}) \quad (2.47)$$

$$\omega_e = N_p\omega_r + \frac{R_r}{L_r}\frac{i^*_{qs}}{i_{mr}} \quad (2.48)$$

$$\frac{d}{dt}\theta_e = \omega_e \quad (2.49)$$

$$T_e = N_p(1-\sigma)L_s i_{qs} i_{mr} \tag{2.50}$$

Berbeda dengan model pengendali vektor yang menggunakan kerangka acuan rotor, untuk pemodelan motor induksi digunakan kerangka acu tetap ($\omega_c = 0$). Dengan menggunakan persamaan (2.14) – (2.18) didapatkan persamaan arus dan tegangan pada kerangka acu tetap.

$$v_s = R_s i_s + \frac{d\lambda_s}{dt} \tag{2.51}$$

$$v_r = R_r i_r + \frac{d\lambda_r}{dt} - j\omega_r \lambda_r \tag{2.52}$$

Persamaan tegangan rotor sama dengan nol karena konstruksi yang digunakan sangkar bajing.

$$0 = \frac{R_r}{L_r}(\lambda_r - L_m i_s) + \frac{d}{dt}\lambda_r - j\omega_r \lambda_r \tag{2.53}$$

$$\begin{aligned}\frac{d}{dt}\lambda_r &= -\frac{R_r}{L_r}(\lambda_r - L_m i_s) + j\omega_r \lambda_r \\ &= -\frac{R_r}{L_r}\lambda_r + \frac{R_r}{L_r} L_m i_s + j\omega_r \lambda_r \\ &= \frac{R_r}{L_r} L_m i_s + \left(-\frac{R_r}{L_r} + j\omega_r\right)\lambda_r \end{aligned} \tag{2.54}$$

Persamaan (2.52) – (2.54) dimasukan ke persamaan tegangan stator. Untuk memperoleh keadaan arus dan tegangan pada kerangka tetap [4].

$$\begin{aligned} v_s &= R_s i_s + \frac{d}{dt}(L_s i_s + L_m i_r) = R_s i_s + \frac{d}{dt}\left(L_s i_s + \frac{L_m \lambda_r - L_m^2 i_s}{L_r}\right) \\ &= R_s i_s + \frac{d}{dt}\left(L_s i_s - \frac{L_m^2}{L_r} i_s\right) + \frac{L_m}{L_r}\frac{d\lambda_r}{dt} \\ &= R_s i_s + \left(L_s - \frac{L_m^2}{L_r}\right)\frac{di_s}{dt} + \frac{L_m}{L_r}\left[\left(\frac{R_r}{L_r} L_m\right) i_s - \frac{R_r}{L_r}\lambda_r + j\omega_r \lambda_r\right] \\ &= \left(R_s + \frac{L_m^2 R_r}{L_r^2}\right) i_s + \left(L_s - \frac{L_m^2}{L_r}\right)\frac{di_s}{dt} + \frac{L_m}{L_r}\left(-\frac{R_r}{L_r}\lambda_r + j\omega_r \lambda_r\right) \\ &= \left(R_s + \frac{L_m^2 R_r}{L_r^2}\right) i_s + \left(L_s - \frac{L_m^2}{L_r}\right)\frac{di_s}{dt} - \left(\frac{L_m R_r}{L_r^2} - j\omega_r \frac{L_m}{L_r}\right)\lambda_r \end{aligned} \tag{2.55}$$

$$\begin{aligned}\frac{di_s}{dt} &= \frac{L_r}{L_m^2 - L_s L_r}\left[\left(R_s + \frac{L_m^2 R_r}{L_r^2}\right) i_s - \left(\frac{L_m R_r}{L_r^2} - j\omega_r \frac{L_m}{L_r}\right)\lambda_r - v_s\right] \\ &= \frac{L_r}{L_m^2 - L_s L_r}\left[\left(R_s + \frac{L_m^2 R_r}{L_r^2}\right) i_s - \left(\frac{L_m R_r}{L_r^2} - j\omega_r \frac{L_m}{L_r}\right)(L_r i_r + L_m i_s) - v_s\right] \end{aligned} \tag{2.56}$$

Pada bagian rotor

$$\begin{aligned} v_s &= R_s i_s + \frac{d}{dt}\left(L_s \frac{\lambda_r - L_r i_r}{L_m} + L_m i_r\right) = R_s i_s + \frac{d}{dt}\left(\frac{L_s \lambda_r - L_s L_r i_r}{L_m} + L_m i_r\right) \\ &= R_s i_s + \frac{di_r}{dt}\left(\frac{L_m^2 - L_s L_r}{L_m}\right) + \frac{L_s}{L_m}\frac{d\lambda_r}{dt} \\ &= R_s i_s + \frac{di_r}{dt}\left(\frac{L_m^2 - L_s L_r}{L_m}\right) + \frac{L_s}{L_m}\left[\frac{R_r L_m}{L_r} i_s + \left(-\frac{R_r}{L_r} + j\omega_r\right)\lambda_r\right] \end{aligned}$$

$$v_s = \left(R_s + \frac{R_r L_s}{L_r}\right) i_s + \frac{di_r}{dt}\left(\frac{L_m^2 - L_s L_r}{L_m}\right) + \left(j\omega_r \frac{L_s}{L_m} - \frac{L_s R_r}{L_m L_r}\right)\lambda_r \quad (2.57)$$

$$\frac{di_r}{dt} = -\frac{L_m}{L_m^2 - L_s L_r}\left[\left(R_s + \frac{R_r L_s}{L_r}\right) i_s + \left(j\omega_r \frac{L_s}{L_m} - \frac{L_s R_r}{L_m L_r}\right)\lambda_r - v_s\right]$$
$$= -\frac{L_m}{L_m^2 - L_s L_r}\left[\left(R_s + \frac{R_r L_s}{L_r}\right) i_s + \left(j\omega_r \frac{L_s}{L_m} - \frac{L_s R_r}{L_m L_r}\right)(L_r i_r + L_m i_s) - v_s\right] \quad (2.58)$$

Dari persamaan (2.56) dan (2.58) diperoleh persamaan sumbu αβ untuk arus stator dan juga rotor.

$$\frac{d}{dt} i_{\alpha s} = \frac{(R_s L_r i_{\alpha s} - N_p \omega_r L_m^2 i_{\beta s} - R_r L_m i_{\alpha r} - N_p \omega_r L_r L_m i_{\beta r} - L_r v_{\alpha s})}{(L_m^2 - L_r L_s)} \quad (2.59)$$

$$\frac{d}{dt} i_{\beta s} = \frac{(N_p \omega_r L_m^2 i_{\alpha s} + R_s L_r i_{\beta s} + N_p \omega_r L_r L_m i_{\alpha r} - R_r L_m i_{\beta r} - L_r v_{\beta s})}{(L_m^2 - L_r L_s)} \quad (2.60)$$

$$\frac{d}{dt} i_{\alpha r} = -\frac{(R_s L_r i_{\alpha s} - N_p \omega_r L_m L_s i_{\beta s} - R_r L_s i_{\alpha r} - N_p \omega_r L_r L_s i_{\beta r} - L_m v_{\alpha s})}{(L_m^2 - L_r L_s)} \quad (2.61)$$

$$\frac{d}{dt} i_{\beta r} = -\frac{(N_p \omega_r L_m L_s i_{\alpha s} + R_s L_m i_{\beta s} + N_p \omega_r L_r L_s i_{\alpha r} - R_r L_r i_{\beta r} - L_m v_{\beta s})}{(L_m^2 - L_r L_s)} \quad (2.62)$$

Nilai dari torsi, fluks, arus magnetisasi, posisi dan kecepatan dihitung dengan menggunakan persamaan di bawah ini. Besaran ini digunakan sebagai nilai aktual dari motor induksi.

$$T_{e_{act}} = N_p L_m (i_{\beta s} i_{\alpha r} - i_{\alpha s} i_{\beta r}) \quad (2.63)$$

$$\lambda_\alpha = L_r i_{\alpha r} + L_m i_{\alpha s} \quad (2.64)$$

$$\lambda_\beta = L_r i_{\beta r} + L_m i_{\beta s} \quad (2.65)$$

$$i_{mr_{act}} = \frac{\sqrt{\lambda_\alpha{}^2 + \lambda_\beta{}^2}}{L_m} \quad (2.66)$$

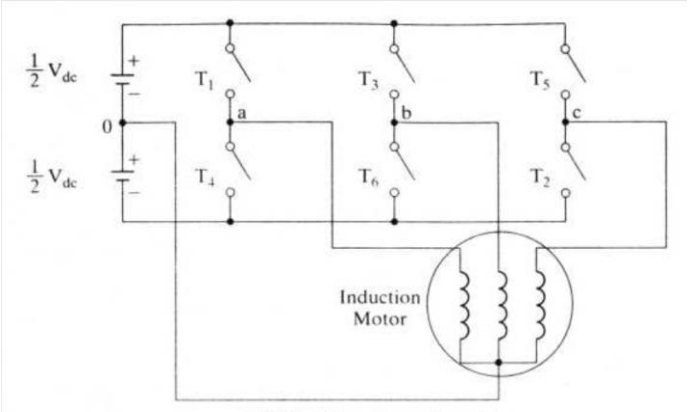
$$\theta_{e_{act}} = \tan^{-1}\left(\frac{\lambda_\alpha}{\lambda_\beta}\right) \quad (2.67)$$

$$\frac{d}{dt}\omega_r = \frac{T_{e_{act}} - T_L - B\omega_r}{J} \quad (2.68)$$

$$\frac{d}{dt}\theta_r = \omega_r \quad (2.69)$$

2.3 Model Penggerak

Struktur umum dari power inverter tiga fasa adalah seperti yang terdapat pada Gambar 2.6, dimana T1, T2, T3, T4, T5 dan T6 merupakan saklar sedangkan Vdc tegangan masukan inverter.



Gambar 2.6 Skema Dasar Inverter Tiga Fasa dengan Motor AC

sumber : Krishnan (2001)

Sebagai saklar dapat digunakan BJT, IGBT, GTO, dan transistor daya lainnya. Saat terjadi kondisi ON – OFF, harus terdapat tiga buah saklar pada kondisi ON dan tiga lainnya pada kondisi OFF dan saklar pada kaki yang sama (T1-T4, T3-T6, T5-T2) bersifat komplementer supaya tidak terjadi hubung singkat.

Prinsip kerja Pulse Width Modulation adalah memotong sinyal masukan sehingga menjadi perintah ON – OFF untuk saklar pada motor dengan tujuan menghasilkan tegangan dc dengan lebar frekuensi ON – OFF yang dapat menjadi pengganti gelombang sinusoidal.

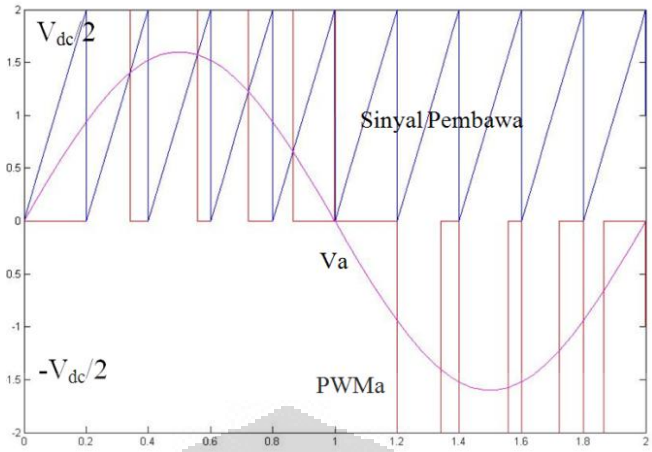
Gambar 2.7 menunjukan gelombang PWM yang dihasilkan dengan menggunakan perpotongan antara gelombang pembawa v_c berbentuk gigi gergaji dengan sinyal yang diperintahkan v_a^*, dalam hal ini sinyal sinusoidal, sehingga dihasilkan sinyal penyalaan untuk saklar inverter.

$$v_{a0} = \frac{1}{2}V_{dc} \; ; 0 < v_a^* \text{ dan } v_c < v_a^*$$

$$v_{a0} = -\frac{1}{2}V_{dc} \; ; 0 > v_a^* \text{ dan } v_c > v_a^*$$

$$v_{a0} = 0; \text{selain kondisi di atas} \qquad (2.70)$$

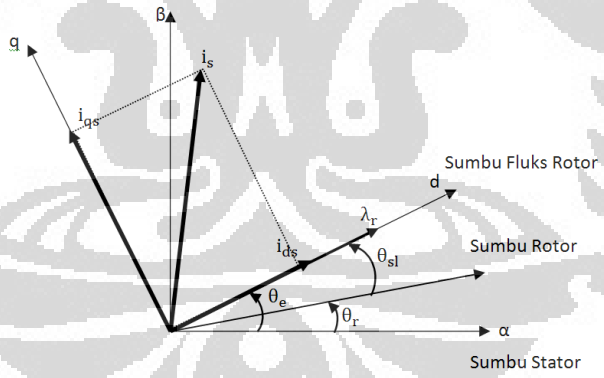
Jika sinyal pembawa lebih kecil dari sinyal yang diperintahkan, maka setengah tegangan DC positif akan dimasukan. Sedangkan jika sinyal pembawa lebih besar, maka setengah tegangan DC negatif akan dimasukan.



Gambar 2. 7 Pembentukan Sinyal PWM

2.4. Kendali Vektor

Kendali vektor atau biasa disebut sebagai pengendali berorientasi medan adalah metode yang digunakan pada penggerak dengan mengubah frekuensi dari arus atau tegangan masukan. Hal itu dilakukan untuk mengendalikan torsi dan juga kecepatan. Pada motor induksi kedua hal tersebut saling mempengaruhi.



Gambar 2.8 Diagram Fasor dari Kendali Vektor

Supaya pengendalian fluks dan torsi dapat tidak saling mempengaruhi, maka arus fasa dari stator harus diubah menjadi searah dengan fluks rotor. Gambar 2.8 merupakan diagram fasor dari komponen yang ada pada motor induksi. Karena berada pada arah yang sama dengan fluks rotor maka dapat diketahui i_{ds} adalah arus penghasil medan. Sedangkan i_{qs} tegak lurus dengannya merupakan arus penghasil torsi. Dapat dilihat i_{ds} dan i_{qs} akan bersifat dc karena

kecepatan relatif terhadap medan rotor sama dengan nol, sehingga tepat untuk digunakan sebagai variabel pengendali.

Posisi medan adalah hal yang penting untuk pengendalian vektor. Hal ini dapat diperoleh dengan menggunakan persamaan berikut

$$\theta_e = \theta_r + \theta_{sl} \quad (2.\ 71)$$

dengan θ_r dan θ_{sl} merupakan posisi rotor dan sudut slip.

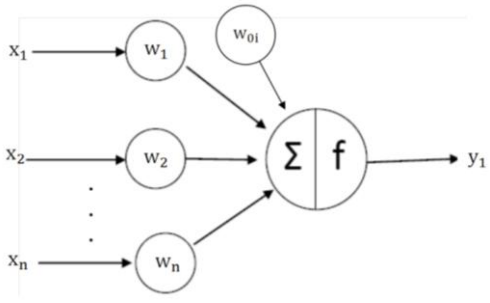
2.5 Jaringan Syaraf Tiruan

Pada bagian ini akan dijelaskan mengenai jaringan syaraf tiruan sebagai dasar untuk perancangan yang akan dilakukan. Hal yang dibahas adalah prinsip kerja, jenis jaringan dan algoritma pelatihan backpropagation.

2.5.1 Prinsip Kerja

Jaringan syaraf tiruan merupakan teknologi perhitungan yang menirukan cara kerja otak. Prinsip dasar dari teknologi ini adalah digunakannya elemen pemproses sederhana yang menirukan neuron pada otak. Kemudian elemen ini disusun sesuai dengan sistem yang dimodelkan. Kemudian jaringan tersebut dilatih dengan sekelompok data, sehingga dapat mengenali bukan hanya pola tersebut jika muncul kembali tetapi juga pola yang mirip dengan pola yang telah dilatih.

Pada umumnya, jaringan syaraf disusun oleh elemen – elemen pemproses sederhana yang terhubung satu dengan yang lain dan membentuk lapisan. Metode pembelajaran ini dapat menggunakan data kelompok sebagai masukan. Pada beberapa jaringan sebuah pengawas (*supervised*) digunakan untuk mengarahkan prosedur, tetapi terdapat juga jaringan yang mampu mengatur jaringannya sendiri sehingga tidak memerlukan pengawas (*unsupervised*).



Gambar 2. 9 Model Neuron

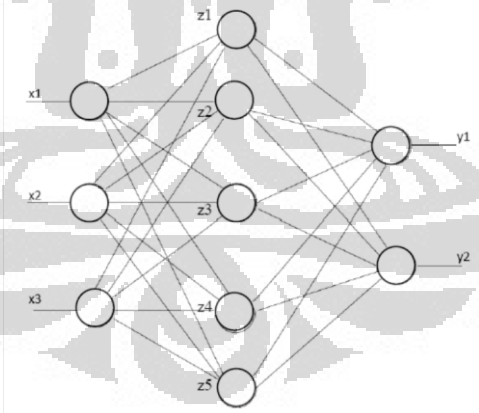
Terdapat beberapa macam elemen node dasar pada jaringan syaraf tergantung dengan jaringan yang dibutuhkan. Tetapi suatu model yang umum ditunjukan oleh Gambar 2.9. Secara berurutan merupakan masukan, bobot dan bias adalah x_1-x_n, w_1-w_n, dan w_{0i}. Perhitungan pada suatu node sesuai dengan persamaan berikut.

$$y_1 = f(\sum_{i=1}^{n} w_i x_i + w_{0i}) = f(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + w_{0i}) \quad (2.72)$$

Dalam persamaan di atas masukan x_i dikalikan dengan bobot masing – masing kemudian dijumlahkan semua juga dengan bias pada node tersebut dan hasil penjumlahannya dihitung ke dalam fungsi aktivasi f.

2.5.2 Jaringan Multi Lapis

Jaringan syaraf tiruan yang umum ditunjukan oleh Gambar 2.10. Jaringan ini memiliki tiga lapisan, yaitu lapisan masukan (x_i), lapisan tersembunyi (z_i), dan lapisan keluaran (y_i). Pada jaringan multi lapis, terdapat node yang terhubung secara langsung dengan semua node di lapisan sebelumnya dan sesudahnya atau terdapat juga yang hanya terhubung sebagian.



Gambar 2. 10 Jaringan Syaraf Multi Lapis Terhubung Penuh

Jaringan ini memiliki, satu lapis masukan, satu atau lebih lapisan tersembunyi, dan satu lapisan keluaran. Lapisan masukan hanya bertugas meneruskan masukan dan tidak melakukan perhitungan, sementara lapisan tersembunyi dan lapisan keluaran melakukan perhitungan. Jumlah neuron pada lapisan masukan sama dengan jumlah ciri pada pola yang akan dikenali, sedang jumlah neuron pada lapisan keluaran sama dengan jumlah kelas pola.

2.5.3 Jaringan Syaraf Tiruan dengan Algoritma Backpropagation

Tujuan dari teknik backpropagation adalah untuk mengatur bobot dan bias pada node sehingga menyebabkan kuadrat kesalahan dari keluaran menjadi kecil. Bobot dan bias pada jaringan syaraf harus diberikan nilai awal. Metode yang dapat digunakan antara lain dengan memasukan nilai secara acak atau dengan menggunakan metode Nguyen – Widrow.

Metode inisialisasi yang akan digunakan adalah metode Nguyen – Widrow. Langkah awal yang dilakukan metode ini adalah mencari faktor skala

$$\beta = 0.7\ (p)^{1/n} \tag{2.73}$$

dengan β adalah faktor skala, n, dan p merupakan jumlah unit masukan dan tersembunyi. Kemudian dilakukan langkah sebagai berikut:

1. Untuk setiap unit pada lapisan tersembunyi (j=1, ... , p) dilakukan inisialisasi dengan bobot secara random dengan besar antara – 0,5 sampai 0,5.
2. Penghitungan

$$\|v_j\| = \sqrt{\sum_{i=1}^{p} v_{ij}^2} \tag{2.74}$$

3. Nilai bobot kemudian diperbaharui

$$v_{ij} = \frac{\beta v_{ij}}{\|v_{ij}\|} \tag{2.75}$$

4. Set bias v_{0j} : bilangan acak antara - β sampai β

Proses pelatihan dengan Propagasi – Balik dapat dijelaskan dengan algoritma berikut :

1. Inisialisasi

Langkah pertama dilakukan dengan memilih laju pelatihan, pilih kesalahan global maksimum E_{max}, dilanjutkan dengan memberi nilai awal untuk bobot dan bias dengan nilai acak awal, jumlah neuron pada setiap lapisan dan juga jumlah lapisan tersembunyi.

2. Pelatihan : Propagasi Maju

Langkah berikutnya dilakukan dengan memasukan nilai masukan (x) ke lapisan masukan, menentukan nilai keluaran (y_k). Setiap masukan dan keluaran dapat digunakan berulang kali pada langkah pelatihan, sampai besar bobot tidak mengalami perubahan. Kemudian keluaran dihitung dengan menggunakan jumlah bobot yang sesuai dan fungsi aktivasi.

Setiap unit di lapisan tersembunyi (z_j, j = 1,…,m) menghitung semua sinyal input dengan bobotnya kemudian menghitung fungsi aktivasi setiap lapisan tersembunyi sebagai keluaran dari lapisan tersebut:

$$z_{\_inj} = v_{0j} + \sum x_i v_{ij} \quad (2.76)$$

$$z_j = f(z_{inj}) \quad (2.77)$$

Bobot antara neuron ke – i pada lapisan masukan dan neuron ke – j pada lapisan tersembunyi dituliskan sebagai v_{ij}. Sedangkan bias pada neuron di lapisan tersembunyi ke-j dituliskan sebagai v_{0j}. Kemudian hasil dari fungsi aktivasi tersebut dikirimkan ke lapisan keluaran. Pada lapisan keluaran sinyal dari lapisan tersembunyi kemudian diberikan bobot dan kemudian semuanya dijumlahkan, sesuai dengan persamaan berikut

$$y_{\_inj} = y_{0k} + \sum z_j y_{jk} \quad (2.78)$$

$$y_k = f(y_{\_inj}) \quad (2.79)$$

3. Pelatihan : Propagasi Balik

Langkah ini dilakukan dengan membandingkan nilai keluaran target dengan yang dihasilkan jaringan menggunakan fungsi kesalahan.

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2 \quad (2.80)$$

Hitung jumlah keluaran kesalahan dengan menjumlahkan semua kesalahan dari semua kelompok pelatihan. Kemudian dilakukan penghitungan sinyal kesalahan pada lapisan keluaran dalam hal ini jika digunakan fungsi sigmoid bipolar

$$\delta_k = (t_k - y_k) f'(y_{\_inj}) \quad (2.82)$$

$$\delta_k = \frac{1}{2}(t_k - y_k)(1 + y_k)(1 - y_k) \quad (2.84)$$

4. Penyusunan bobot untuk masukan yang berikut

Dengan menggunakan algoritma rekursif dari lapisan keluaran dan mundur untuk bobot di lapisan tersembunyi yang pertama.

Penghitungan perubahan bobot

$$w_{jk}(k+1) = w_{jk}(k) + \Delta w_{jk} = w_{jk} + \alpha\delta_k z_j$$

$$w_{0k}(k+1) = w_{0k}(k) + \Delta w_{0k} = w_{0k} + \alpha\delta_k \quad (2.85)$$

dimana $w_{jk}(k)$ adalah bobot pada keluaran dan waktu k sedangkan δ_k adalah sinyal kesalahan pada neuron ke-k di lapisan keluaran.

Selanjutnya pengaturan bobot pada lapisan tersembunyi akan dilakukan dimulai dengan mencari sinyal kesalahan pada lapisan tersembunyi.

$$\delta_j = (\delta_k w_{ij}) f'(z_{\_inj}) \quad (2.86)$$

Pengaturan nilai bobot dan bias pada lapisan tersembunyi

$$v_{ij}(k+1) = v_{ij}(k) + \Delta v_{ij} = v_{ij} + \alpha\delta_j x_i$$

$$v_{0j}(k+1) = v_{0j}(k) + \Delta v_{0j} = v_{0j} + \alpha\delta_j \quad (2.88)$$

5. Kemudian kelompok masukan yang baru dimasukan

Jika jumlah iterasi belum terpenuhi atau belum mencapai nilai Mean Square Error (MSE), maka kembali ke langkah kedua.

$$MSE = \frac{1}{n}\sum(y_k - t_k)^2 \quad (2.90)$$

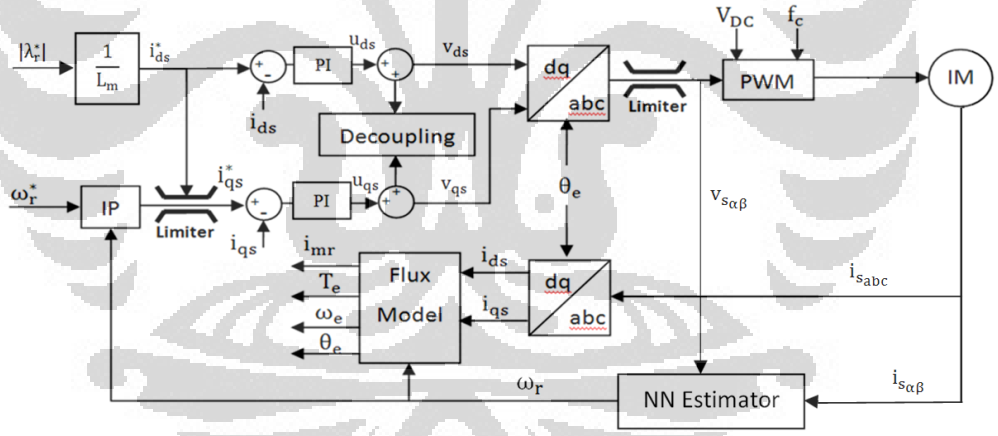
Proses pelatihan ini terjadi berulang sampai syarat yang terdapat di langkah lima terpenuhi.

Setelah pelatihan, jaringan syaraf kemudian diuji dengan propagasi maju menggunakan data yang berbeda dengan data pelatihan. Bobot dan bias yang telah diperoleh sebelumnya dari proses pelatihan dipertahankan dalam melaksanakan proses ini. Kemudian hasilnya dibandingkan dengan keluaran yang sebenarnya untuk mengetahui apakah jaringan syaraf telah dilatih dengan benar. Jika hasil pengujian memuaskan, yaitu memberikan nilai kesalahan yang rendah, maka jaringan tersebut siap untuk digunakan.

BAB 3
PERANCANGAN ESTIMATOR KECEPATAN BERBASIS JARINGAN SYARAF TIRUAN DAN SISTEM KENDALI MOTOR INDUKSI TIGA FASA

Pada bab sebelumnya telah dijelaskan mengenai pemodelan matematika motor induksi, pengendalian vektor, dasar dari jaringan syaraf tiruan dan metode pelatihan backpropagation. Sedangkan pada bab ini akan dijelaskan perancangan jaringan syaraf tiruan sebagai estimator kecepatan, perancangan pengendali arus dan kecepatan dari sistem kendali vektor, juga metode simulasi yang digunakan untuk mengetahui kinerjanya.

Pada skripsi ini jaringan syaraf tiruan akan digunakan sebagai estimator kecepatan rotor. Nilai kecepatan rotor tidak diperoleh dari umpan balik sensor kecepatan, tetapi diperkirakan. Diagram motor induksi tanpa sensor kecepatan yang digunakan adalah seperti pada gambar 3.1.



Gambar 3. 1 Diagram blok penggerak motor induksi dengan estimator kecepatan berbasis jaringan syaraf tiruan

Pengendalian kecepatan membutuhkan sensor untuk mengukur kecepatan atau posisi aktual pada saat sekarang. Kemudian informasi tersebut diumpan balik untuk dibandingkan dengan besar variabel referensi dan kesalahan digunakan sebagai sinyal kesalahan oleh pengendali. Sehingga menjadi perbaikan bagi sistem untuk mencapai kondisi yang dikehendaki. Tetapi penggunaan sensor memiliki keterbatasan dan biaya yang lebih besar. Karena itu untuk mengatasi permasalahan tersebut digunakan estimator kecepatan untuk memperkirakan

kecepatan. Hal itu dilakukan dengan memodelkan sistem dan menggunakan variabel lain yang lebih mudah untuk diukur sebagai masukan sehingga diperoleh kecepatan sebagai keluarannya.

Motor induksi merupakan sistem yang tidak linear. Pada skripsi ini akan digunakan jaringan syaraf tiruan untuk mengestimasi kecepatan dari motor induksi karena kemampuannya dalam memodelkan sistem yang tidak linear. Kinerja dari estimator tersebut akan dipelajari dan diuji melalui percobaan parameter jaringan syaraf tiruan dan kondisi kerja motor induksi yang berbeda. Parameter yang memberikan kesalahan terkecil atau kinerja yang baik yang akan digunakan untuk percobaan selanjutnya.

3.1 Perancangan Estimator Kecepatan Berbasis Jaringan Syaraf Tiruan

Jaringan syaraf tiruan memiliki fungsi identifikasi sistem. Pada saat ini kemampuan tersebut akan digunakan untuk mengidentifikasi sistem motor induksi untuk memperoleh perkiraan kecepatan. Jaringan syaraf tiruan dirancang untuk menjadi estimator kecepatan dari motor induksi. Sehingga yang menjadi tujuan dari pelatihan jaringan ini adalah dengan masukan yang diberikan dapat memberikan keluaran yang sesuai dengan jika digunakan pada motor induksi. Proses pelatihan dan penyesuaian bobot juga bias dilakukan untuk mengoptimasi jaringan sehingga kecepatan rotor perkiraan mendekati kecepatan rotor aktual dalam berbagai wilayah kerja kecepatan dan penggunaan.

Dalam melakukan perancangan jaringan syaraf tiruan diperlukan pemahaman yang cukup terhadap sistem tersebut. Sehingga dapat diketahui data masukan dan keluaran atau variabel yang digunakan sudah tepat untuk memodelkan sistem tersebut. Sebagai pemodelan digunakan persamaan yang dipakai untuk penggunaan Extended Kalman Filter sebagai estimator kecepatan, yaitu sebuah estimator yang melakukan perhitungan optimum secara rekursif untuk memperoleh kondisi atau parameter dari sistem yang tidak linear. Nilai dari variabel kondisi pada saat ini diprediksikan dengan model matematika dan nilai variabel perkiraan pada waktu sebelumnya. Persamaan motor induksi dua fasa yang digunakan untuk metode ini adalah persamaan motor induksi dua fasa pada

kerangka acuan stator dengan arus dan fluks sebagai variabel kondisinya dan kecepatan rotor yang diestimasi sebagai variabel tambahannya [3].

$$\frac{d}{dt}\begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ \lambda_{dr} \\ \lambda_{qr} \\ \omega_r \end{bmatrix} = \begin{bmatrix} -1/T_s'^* & 0 & L_m/(\sigma L_s L_r T_r) & \omega_r L_m/(\sigma L_s L_r) & 0 \\ 0 & -1/T_s'^* & -\omega_r L_m/(\sigma L_s L_r) & L_m/(\sigma L_s L_r T_r) & 0 \\ L_m/T_r & 0 & -1/T_r & \omega_r & 0 \\ 0 & L_m/T_r & \omega_r & -1/T_r & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ \lambda_{dr} \\ \lambda_{qr} \\ \omega_r \end{bmatrix} + \begin{bmatrix} 1/\sigma L_s & 0 \\ 0 & 1/\sigma L_s \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{\alpha s} \\ v_{\beta s} \end{bmatrix} \quad (3.1)$$

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

$$\begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_{\alpha s} \\ i_{\beta s} \\ \lambda_{dr} \\ \lambda_{qr} \\ \omega_r \end{bmatrix} \quad (3.2)$$

$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

dengan

$$\frac{1}{T_s'^*} = \frac{R_s + R_r\left(\frac{L_m}{L_r}\right)^2}{\sigma L_s} \quad (3.3)$$

$$T_r = \frac{R_r}{L_r} \quad (3.4)$$

Persamaan (3.1) merupakan persamaan kondisi untuk motor induksi. Dari persamaan tersebut dapat dilihat bahwa kecepatan rotor sebagai variabel tambahan berada pada matriks keadaan juga dan dapat disimpulkan bahwa sistem motor induksi merupakan sistem yang tidak linear. Sehingga diperlukan teknik pengendalian yang mampu untuk digunakan pada sistem yang tidak linear. Salah satu teknik yang dapat digunakan adalah jaringan syaraf tiruan. Sebagai vektor masukan adalah tegangan pada kerangka acuan stator $v_{s\alpha}$ dan $v_{s\beta}$. Persamaan (3.2) merupakan persamaan vektor keluaran dalam persamaan tersebut adalah arus stator pada kerangka acuan stator $i_{s\alpha}$ dan $i_{s\beta}$. Dengan mengetahui variabel tegangan dan arus pada persamaan kondisi dapat diperoleh variabel lain yang merupakan variabel perkiraan. Penggunaan jaringan syaraf tiruan sebagai estimator kecepatan itu akan dijelaskan pada bagian berikut.

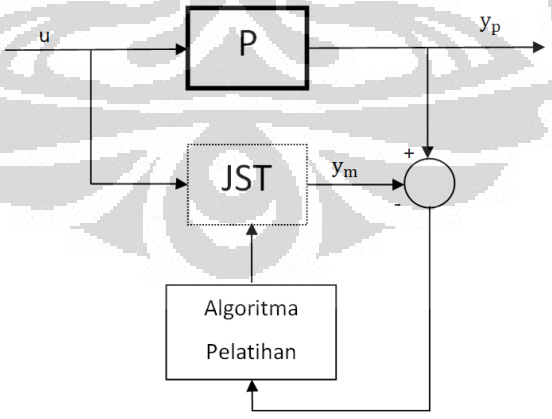
Struktur dari Jaringan Syaraf Tiruan yang digunakan dalam pemodelan ini adalah struktur Multi-Layer Feedforward Network yang terdiri dari delapan masukan, lapisan tersembunyi, dan satu vektor keluaran. Jumlah lapisan tersembunyi atau neuron yang digunakan ditentukan dengan percobaan parameter. Sebagai inputan adalah tegangan dan arus stator pada kondisi kerangka stator

untuk kondisi sekarang dan satu siklus sebelumnya ($v_{\alpha s}(n)$, $v_{\alpha s}(n-1)$, $v_{\beta s}(n)$, $v_{\beta s}(n-1)$, $i_{\alpha s}(n)$, $i_{\alpha s}(n-1)$, $i_{\beta s}(n)$, $i_{\beta s}(n-1)$) dan sebagai keluarannya kecepatan rotor $\omega_r(n)$. Variabel masukan tegangan dan arus dua fasa pada kerangka acuan stator untuk kondisi sekarang dan waktu sebelumnya dipilih karena seperti pada penggunaan estimator EKF yang menggunakan variabel tersebut sebagai variabel yang diketahui untuk perkiraan variabel lain dan karena penggunaan variabel tersebut pada jaringan syaraf tiruan memberikan keakuratan perkiraan yang baik [5]. Metode pelatihan yang digunakan adalah *supervised learning* dengan algoritma pelatihan backpropagation.

Proses yang diidentifikasi adalah fungsi dengan pola masukan dan keluaran bipolar sehingga akan digunakan sigmoid bipolar sebagai fungsi aktivasi pada bagian lapisan tersembunyi dan keluaran [5].

$$y_j = f(-net_j) = \frac{1-e^{(-net_j)}}{1+e^{(-net_j)}} \qquad (3.5)$$

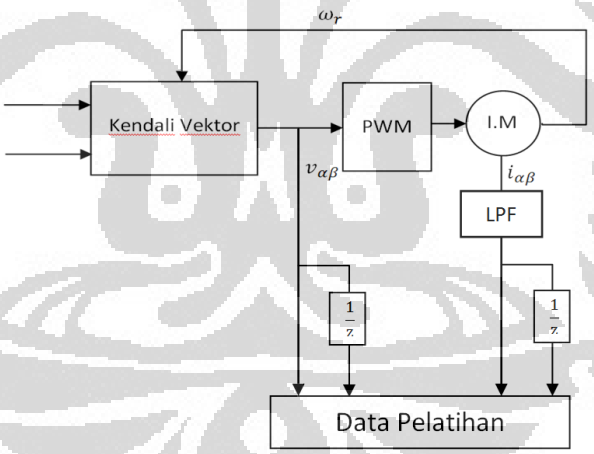
Proses pelatihan dari jaringan syaraf tiruan yang digunakan pada saat ini adalah pemodelan maju (*forward modelling*). Seperti pada gambar 3.2 jaringan syaraf tiruan dipasang paralel dengan sistem P dan kesalahan yang dihasilkan sistem y_p dan keluaran jaringan y_m digunakan sebagai sinyal perbaikan. Struktur pelatihan ini merupakan bentuk klasik, dimana sistem memberikan nilai target secara langsung kepada jaringan [6].



Gambar 3.2 Identifikasi pemodelan maju

Data pelatihan diperoleh dari motor induksi dengan sensor kecepatan yang dijalankan dalam berbagai kondisi kecepatan dan pembebanan yang beragam. Diagram blok dari cara memperoleh data pelatihan ditunjukan dengan gambar 3.3. Data tegangan keluaran dari kendali vektor dan arus dari motor induksi disimpan, kemudian digunakan untuk pelatihan secara tidak langsung sampai diperoleh jumlah kesalahan yang rendah atau epoh tercapai. Arus keluaran dari motor induksi terlebih dahulu difilter untuk menghilangkan pengaruh pencacahan yang dilakukan PWM. Untuk pembobotan awal digunakan metode Nguyen Widrow. Semua nilai masukan dinormalisasi sesuai persamaan (3.6) sehingga menjadi nilai per unit, kemudian setelah diproses nilai keluaran yang diperoleh dikembalikan ke nilai sebenarnya. Nilai masukan sebenarnya x_{act} dikurangi dengan nilai minimum x_{min} dan dibagi dengan selisih nilai maksimum x_{max} dengan nilai minimumnya sehingga diperoleh nilai per unit x_{pu}.

$$x_{pu} = \frac{x_{act} - x_{min}}{x_{max} - x_{min}} \tag{3.6}$$



Gambar 3.3 Skema Data Pelatihan

Setelah proses pelatihan selesai maka dilakukan proses pengujian untuk membuktikan pelatihan berhasil dengan baik. Pengujian dilakukan dengan memasukan set data yang berbeda dengan yang digunakan pada saat pelatihan. Jika hasil dari pengujian memberikan hasil yang memuaskan, maka jaringan sudah siap untuk digunakan.

3.2 Model Simulasi Dinamis Motor Induksi Tiga Fasa

Persamaan motor induksi tiga fasa dan model fluks yang akan digunakan untuk simulasi sudah dijelaskan di bab dua. Tetapi untuk kemudahan dituliskan kembali di bagian ini.

Model motor induksi

$$\frac{d}{dt}i_{\alpha s} = \frac{\left(R_s L_r i_{\alpha s} - N_p \omega_r L_m^2 i_{\beta s} - R_r L_m i_{\alpha r} - N_p \omega_r L_r L_m i_{\beta r} - L_r v_{\alpha s}\right)}{\left(L_m^2 - L_r L_s\right)} \quad (3.7)$$

$$\frac{d}{dt}i_{\beta s} = \frac{\left(N_p \omega_r L_m^2 i_{\alpha s} + R_s L_r i_{\beta s} + N_p \omega_r L_r L_m i_{\alpha r} - R_r L_m i_{\beta r} - L_r v_{\beta s}\right)}{\left(L_m^2 - L_r L_s\right)} \quad (3.8)$$

$$\frac{d}{dt}i_{\alpha r} = -\frac{\left(R_s L_r i_{\alpha s} - N_p \omega_r L_m L_s i_{\beta s} - R_r L_s i_{\alpha r} - N_p \omega_r L_r L_s i_{\beta r} - L_m v_{\alpha s}\right)}{\left(L_m^2 - L_r L_s\right)} \quad (3.9)$$

$$\frac{d}{dt}i_{\beta r} = -\frac{\left(N_p \omega_r L_m L_s i_{\alpha s} + R_s L_m i_{\beta s} + N_p \omega_r L_r L_s i_{\alpha r} - R_r L_r i_{\beta r} - L_m v_{\beta s}\right)}{\left(L_m^2 - L_r L_s\right)} \quad (3.10)$$

$$T_{e_{act}} = N_p L_m \left(i_{\beta s} i_{\alpha r} - i_{\alpha s} i_{\beta r}\right) \quad (3.11)$$

$$\lambda_\alpha = L_r i_{\alpha r} + L_m i_{\alpha s} \quad (3.12)$$

$$\lambda_\beta = L_r i_{\beta r} + L_m i_{\beta s} \quad (3.13)$$

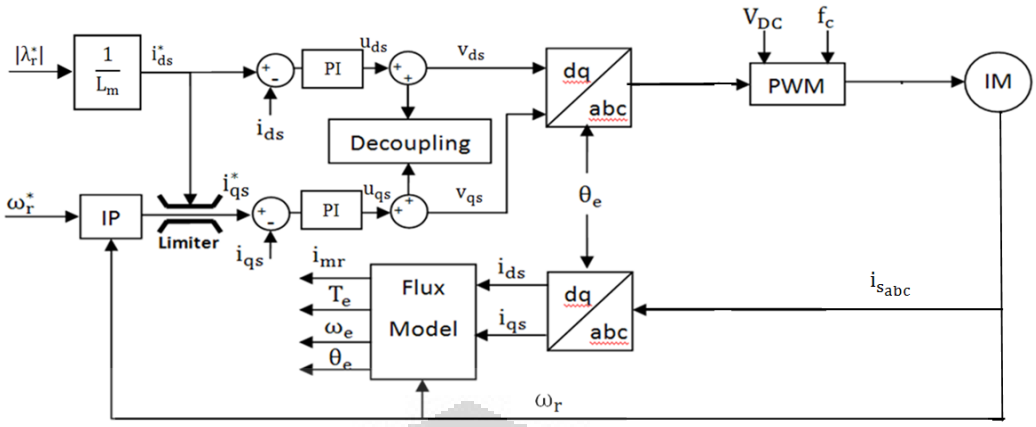
$$i_{mr_{act}} = \frac{\sqrt{{\lambda_\alpha}^2 + {\lambda_\beta}^2}}{L_m} \quad (3.14)$$

$$\theta_{e_{act}} = \tan^{-1}\left(\frac{\lambda_\alpha}{\lambda_\beta}\right) \quad (3.15)$$

$$\frac{d}{dt}\omega_r = \frac{T_{e_{act}} - T_L - B\omega_r}{J} \quad (3.16)$$

3.3 Model Simulasi Kendali Vektor

Skema sistem dan pengendali kecepatan yang digunakan pada skripsi ini sesuai dengan Gambar 3.4. Fluks rotor referensi $|\lambda_r^*|$, bernilai konstan, jika dibagi dengan induktansi magnetisasi menghasilkan arus magnetisasi yang sama dengan arus stator direct referensi. Besar arus-d referensi kemudian menjadi pembatas bagi besar arus-q referensi supaya tidak melebihi besar arus maksimal dari motor induksi. Pada skema tersebut terdapat blok pengendali kecepatan, Rotor Flux Oriented Control (RFOC), Pulse Width Modulation (PWM), dan Motor induksi.



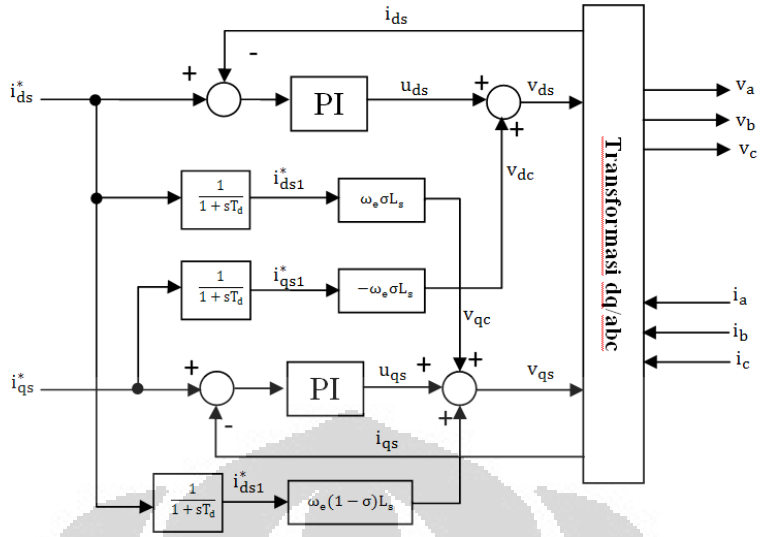
Gambar 3.4 Diagram blok penggerak motor induksi sumber tegangan arah fluks rotor

Pengendali kecepatan digunakan pengendali IP karena memiliki kinerja yang lebih baik dibanding dengan pengendali PI tetapi dengan kemudahan yang sama [7]. Fungsi dari blok ini memberikan arus referensi quadrature i_{qs}^* dengan kecepatan rotor referensi ω_r* dan kecepatan ω_r rotor sebenarnya sebagai masukan. Keluaran dari pengendali kecepatan dibatasi sehingga ketika mencapai nilai batasnya, nilai inisial dari iterasi menggunakan nilai iterasi sebelumnya untuk menghindari *windup phenomenon*. [8]

$$T_e^* = K_{spi} \int (\omega_r^* - \omega_r) dt - K_{spp} \omega_r \qquad (3.17)$$

Persamaan (3.17) adalah persamaan pengendali kecepatan IP. Keluaran dari pengendali kecepatan ini berupa torsi referensi T_e* yang digunakan untuk memperoleh arus quadrature, dengan K_{spp} dan K_{spi} sebagai gain proporsional dan integral.

Di dalam blok RFOC terdapat blok decoupling, pengendali arus, dan model fluks. Blok ini memiliki fungsi sebagai sistem penggerak motor, untuk memberikan vektor tegangan Vabc yang sesuai untuk membuat motor induksi berputar dengan kecepatan yang dikehendaki dengan masukannya adalah torsi referensi dari pengendali kecepatan.



Gambar 3.5 Rangkaian Decoupling

Di dalam blok RFOC terdapat bagian decoupling dan pengendali arus. Rangkaian decoupling berfungsi menjadikan tegangan-dq memiliki hubungan linear dengan arus-dq. Gambar 3.5 merupakan rangkaian decoupling yang digunakan sesuai dengan persamaan (2.37) – (2.42). Sebagai masukan dari rangkaian ini adalah arus – dq referensi i_{ds}*, i_{qs}* dan arus – dq sebenarnya i_{ds}, i_{qs}. Selisih arus referensi dari kedua sumbu menjadi masukan untuk pengendali arus PI yang menghasilkan tegangan u_{ds}, u_{qs} yang kemudian dijumlahkan dengan tegangan decouplingnya v_{dc}, v_{qc}. Hasilnya penjumlahnya ditransformasi v_{ds}, v_{qs} ke sumbu abc v_a, v_b, v_c untuk menjadi masukan bagi PWM.

Arus-dq referensi dengan menggunakan time konstan dipakai untuk menggantikan nilai arus-dq pada persamaan decoupling. [9]

$$i_{ds} = \frac{1}{T_d s+1} i^*_{ds}$$

$$i_{qs} = \frac{1}{T_d s+1} i^*_{qs} \qquad (3.18)$$

Berikut adalah persamaan pengendali arus PI, dengan K_{idp} dan K_{iqp} gain proporsional sedangkan K_{idi} dan K_{iqi} gain integral untuk arus – dq.

$$u_{ds} = K_{idp}(i^*_{ds} - i_{ds}) + K_{idi}\int(i^*_{ds} - i_{ds})dt$$

$$u_{qs} = K_{iqp}(i^*_{qs} - i_{qs}) + K_{iqi}\int(i^*_{qs} - i_{qs})dt \qquad (3.19)$$

Model fluks berfungsi untuk menentukan posisi rotor θ_e yang digunakan untuk mentransformasikan komponen sumbu dq menjadi komponen sumbu αβ atau sebaliknya. Arus ids dan iqs digunakan untuk mencari torsi motor dan arus

magnetisasi. Arus magnetisasi, arus-q referensi dan kecepatan rotor digunakan untuk memperoleh kecepatan sinkron dari motor. Posisi rotor diperoleh dengan mengintegralkan kecepatan sinkron motor. Untuk menghitung parameter model fluks ini digunakan persamaan (2.47) – (2.50).

Blok PWM berfungsi menghasilkan sinyal tegangan PWM V_{PWM} untuk digunakan oleh motor induksi sebagai masukannya adalah tegangan dari penggerak V_{abc}, tegangan V_{DC} dan frekuensi sampling f_c. Besar tegangan masukan dibatasi untuk tidak melebihi $m.V_{dc}/2$, dengan m adalah modulasi indeks rasio yang digunakan.

Pada perancangan dan simulasi ini blok IM digunakan sebagai motor induksi tiga fasa sebenarnya. Masukan dari blok ini adalah tegangan tiga fasa yang diperoleh blok PWM. Sedangkan keluarannya adalah arus tiga fasa stator yang diumpan balik ke blok RFOC untuk digunakan oleh model fluks dan kecepatan rotor yang diumpan balik ke pengendali kecepatan untuk simulasi dengan sensor kecepatan. Kedua umpan balik ini dipasangkan blok memory untuk menghindari terjadinya *algebraic loop*, yang biasanya terjadi karena masukan suatu blok dipengaruhi oleh keluaran pada blok yang sama maupun blok yang tidak secara langsung mempengaruhi.

Proses ini terjadi berulang dibatasi oleh waktu simulasi yang diberikan. Arus stator dan kecepatan rotor yang diumpan balik kemudian digunakan untuk proses waktu setelahnya. Pada kondisi awal semua variabel dan kondisi yang digunakan diset untuk bernilai nol.

3.4 Perancangan Pengendalian

Pada bagian ini akan dijelaskan mengenai desain pengendali pada skema penggerak motor induksi yang bekerja di wilayah kecepatan normal. Pengendali yang digunakan adalah pengendali arus yang merupakan pengendali PI dan pengendali kecepatan berupa pengendali IP.

3.4.1 Perancangan Pengendali Arus

Sebagai pengendali arus digunakan pengendali PI. Pengendali ini tidak dapat bekerja dengan baik untuk sistem yang tidak linear. Sehingga V_{ds} dan V_{qs} harus dilinierisasi dengan teknik dekopling, seperti yang sudah dibahas pada bab kedua. Dengan menggunakan persamaan dekopling tegangan, persamaan umum untuk pengendali PI dan persamaan (3.18) maka arus pada sumbu dq tidak saling mempengaruhi dan dapat diperoleh persamaan sebagai berikut [9]:

$$\left(R_s i_{ds} + L_s \sigma \frac{d}{dt} i_{ds}\right) = \left(k_{idp} + \frac{k_{idi}}{s}\right)(i^*_{ds} - i_{ds}) \tag{3.20}$$

$$(R_s + L_s \sigma s) i_{ds} = \left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{ds} - \left(k_{idp} + \frac{k_{idi}}{s}\right) i_{ds}$$

$$\left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{ds} = i_{ds}\left(R_s + L_s \sigma s + k_{idp} + \frac{k_{idi}}{s}\right)$$

$$\left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{ds} = \left(R_s + L_s \sigma s + k_{idp} + \frac{k_{idi}}{s}\right)\frac{1}{T_d s+1} i^*_{ds}$$

$$\left(k_{idp} + \frac{k_{idi}}{s}\right) = \frac{R_s}{T_d s+1} + \frac{L_s \sigma s}{T_d s+1} + \frac{k_{idp}}{T_d s+1} + \frac{k_{idi}}{T_d s+1} + \frac{k_{idi}}{s(T_d s+1)}$$

$$k_{idp} T_d s + k_{idi} T_d + k_{idp} + \frac{k_{idi}}{s} = R_s + L_s \sigma s + k_{idp} + \frac{k_{idi}}{s}$$

$$k_{idp} T_d s + k_{idi} T_d = R_s + L_s \sigma s \tag{3.21}$$

Persamaan di atas dipisahkan untuk memperoleh persamaan untuk k_{idp} dan k_{idi} sebagai berikut

$$k_{idp} T_d s = L_s \sigma s$$

$$k_{idp} = \frac{L_s \sigma}{T_d} \tag{3.22}$$

$$k_{idi} T_d = R_s$$

$$k_{idi} = \frac{R_s}{T_d} \tag{3.23}$$

Dengan pendekatan yang sama untuk tegangan pada sumbu-q

$$\left(R_s i_{qs} + L_s \sigma \frac{d}{dt} i_{qs}\right) = \left(k_{idp} + \frac{k_{idi}}{s}\right)(i^*_{qs} - i_{qs}) \tag{3.24}$$

$$(R_s + L_s \sigma s) i_{qs} = \left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{qs} - \left(k_{idp} + \frac{k_{idi}}{s}\right) i_{qs}$$

$$\left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{qs} = i_{qs}\left(R_s + L_s \sigma s + k_{idp} + \frac{k_{idi}}{s}\right)$$

$$\left(k_{idp} + \frac{k_{idi}}{s}\right) i^*_{qs} = \left(R_s + L_s \sigma s + k_{idp} + \frac{k_{idi}}{s}\right)\frac{1}{T_d s+1} i^*_{qs}$$

$$\left(k_{iqp} + \frac{k_{iqi}}{s}\right) = \frac{R_s}{T_d s+1} + \frac{L_s \sigma s}{T_d s+1} + \frac{k_{iqp}}{T_d s+1} + \frac{k_{iqi}}{T_d s+1} + \frac{k_{iqi}}{s(T_d s+1)}$$

$$k_{iqp}T_d s + k_{iqi}T_d + k_{iqp} + \frac{k_{iqi}}{s} = R_s + L_s\sigma s + k_{iqp} + \frac{k_{iqi}}{s}$$

$$k_{iqp}T_d s + k_{iqi}T_d = R_s + L_s\sigma s \quad (3.25)$$

Sehingga diperoleh persamaa k_{iqp} dan k_{iqd}

$$k_{iqp}T_d s = L_s\sigma s$$

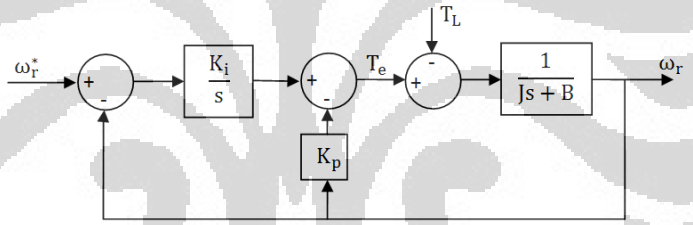
$$k_{iqp} = \frac{L_s\sigma}{T_d} \quad (3.26)$$

$$k_{iqi}T_d = R_s$$

$$k_{iqi} = \frac{R_s}{T_d} \quad (3.27)$$

3.4.2 Perancangan Pengendali Kecepatan

Pengendali kecepatan yang digunakan adalah pengendali IP. Pengendali jenis ini memiliki kelebihan step respons yang lebih baik, pengaturan yang sederhana seperti pengendali PI dan menghilangkan kesalahan steady –state [7].



Gambar 3.6 Diagram Blok Sistem Pengendali Kecepatan

Gambar 3.6 menunjukan model penggerak nominal yang akan digunakan untuk memperoleh gain pengendali kecepatan. Persamaan fungsi alih pengendali respons kecepatan rotor terhadap masukan sebagai berikut [10].

$$\frac{\omega_r(s)}{\omega_r^*(s)} = \frac{K_i}{Js^2+(K_p+B)s+K_i}$$

$$= \frac{\frac{K_i}{J}}{s^2+\frac{K_p+B}{J}s+\frac{K_i}{J}}$$

$$\triangleq \frac{\omega_n^2}{s^2+2\varsigma\omega_n+\omega_n^2} \quad (3.28)$$

Dengan mengidentifikasi fungsi alih yang didapat dapat diketahui damping ratio ς dan frekuensi naturalnya ω_n.

$$\varsigma = \frac{K_p+B}{2(K_iJ)^{\frac{1}{2}}} \quad (3.29)$$

$$\omega_n = \left(\frac{K_i}{J}\right)^{\frac{1}{2}} \quad (3.30)$$

Karena tidak ada zero, maka overshoot dari step response dapat dihindari dengan mengatur nilai damping ratio menjadi $\varsigma = 1$. Kemudian dengan menggunakan pembagian parsial dapat diperoleh persamaan kecepatan terhadap waktu.

$$\omega_r(s) = \frac{\omega_n^2}{s(s^2+2\omega_n+\omega_n^2)}$$

$$\omega_r(s) = \frac{\omega_n^2}{s(s+\omega_n)^2}$$

$$\omega_r(s) = \frac{1}{s} - \frac{1}{s+\omega_n} - \frac{\omega_n^2}{(s+\omega_n)^2} \quad (3.31)$$

$$\omega_r(t) = 1 - e^{-\omega_n t}(1 + \omega_n t) \quad (3.32)$$

Dengan menggunakan waktu bangkit, yaitu waktu yang dibutuhkan kecepatan rotor untuk naik dari 0% - 90% dapat diperoleh frekuensi natural.

$$0.9 = 1 - e^{-\omega_n t}(1 + \omega_n t) \quad (3.33)$$

Frekuensi natural yang diperoleh kemudian digunakan bersama dengan persamaan (3.26) – (3.27) untuk menghitung gain pengendali kecepatan.

$$K_p = 2J\omega_n - B \quad (3.34)$$

$$K_i = J\omega_n^2 \quad (3.35)$$

3.5 Metode Simulasi Estimator Kecepatan

Pada bagian ini akan dijelaskan mengenai simulasi yang akan dilakukan pada jaringan yang telah dilatih dan diuji pada bagian sebelumnya. Simulasi yang akan dilakukan dibagi menjadi dua bagian Simulasi pertama adalah simulasi variasi parameter untuk menentukan jumlah neuron pada lapisan tersembunyi juga laju pembelajaran yang kemudian akan digunakan sebagai kondisi default pada simulasi selanjutnya. Simulasi kedua merupakan simulasi kinerja estimator untuk membandingkan penggunaan estimator jaringan syaraf tiruan dari pada umpan balik kecepatan.

3.5.1 Simulasi Variasi Parameter

Parameter yang akan dibuat variasinya adalah jumlah unit pada lapisan tersembunyi dan laju pembelajaran. Hasil yang terbaik dari percobaan akan digunakan untuk simulasi kinerja estimator kecepatan.

3.5.1.1 Jumlah Unit pada Lapisan Tersembunyi

Pada simulasi dilakukan variasi jumlah unit lapisan tersembunyi dari nilai 1 unit sampai 50 unit. Laju pembelajaran yang digunakan sebesar 0,4. Jaringan tersebut akan dilatih dan diuji kemudian dibandingkan waktu yang dibutuhkan dan MSE dari setiap variasi struktur jaringan.

3.5.1.2 Besar Laju Pembelajaran

Pada simulasi ini akan digunakan variasi laju pembelajaran dengan nilai 0.1, 0.2, 0.3, dan 0.4. Jumlah unit pada lapisan tersembunyi yang digunakan adalah yang memberikan kinerja terbaik dari simulasi sebelumnya. Jaringan tersebut akan dilakukan dilatih dan diuji kemudian dibandingkan waktu yang dibutuhkan untuk setiap proses dan MSE dari setiap besar laju pembelajaran.

3.5.2 Simulasi Kinerja Estimator

Simulasi kinerja estimator dilakukan untuk membandingkan kinerja jika digunakan sensor kecepatan atau dilakukan umpan balik kecepatan dan digunakan estimator jaringan syaraf tiruan pada beberapa kondisi.

3.5.2.1 Simulasi Perubahan Kecepatan dengan Beban Nol

Dalam simulasi ini diberikan masukan kecepatan awal pada saat t = 0 sebesar 80 rad/s kemudian pada t = 6 s dinaikan kecepatannya hingga mencapai 100 rad/s mendekati kecepatan nominal motor. Torsi beban tetap dipertahankan bernilai nol.

3.5.2.2 Simulasi Perubahan Kecepatan dan Perubahan Beban 50%

Pada simulasi ini diberikan masukan kecepatan awal pada saat t = 0 sebesar 80 rad/s dengan beban torsi 50% beban torsi maksimum, pada t = 6 detik diturunkan kecepatannya hingga mencapai 60 rad/s.

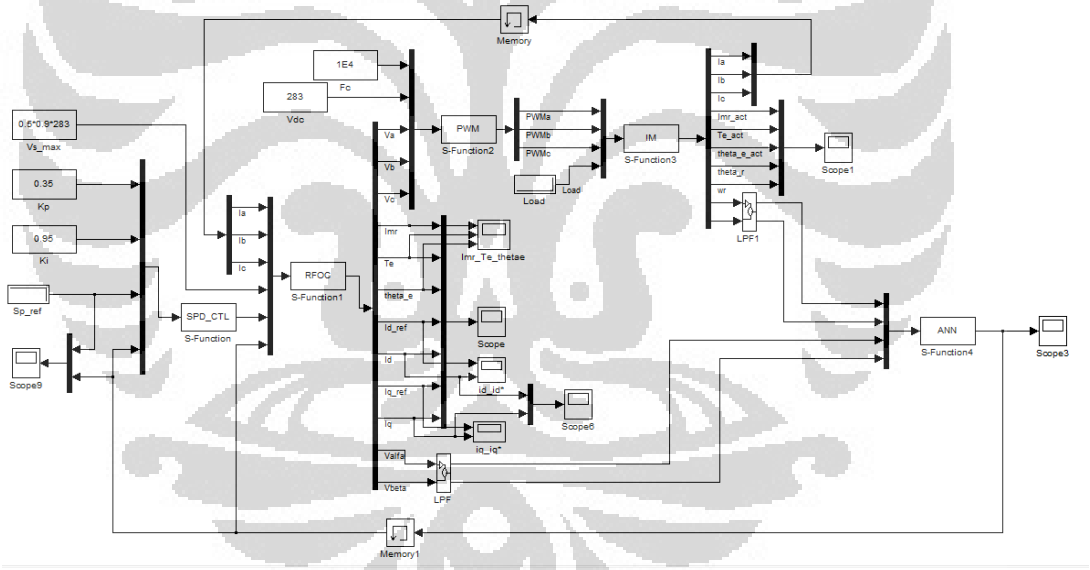
3.5.2.3 Simulasi Daya Tahan Beban

Pada simulasi ini motor induksi akan diberikan beban sebesar 20% dari beban torsi maksimum pada t = 5 detik ketika kecepatan 5 rad/s. Tes ini bertujuan untuk mengetahui kemampuan menahan beban dari penggerak tanpa sensor ini pada kecepatan rendah.

BAB 4
SIMULASI DAN ANALISA

4.1 Rangkaian Simulasi

Simulasi dilakukan dengan C–MEX S–Function di MATLAB/SIMULINK berdasarkan model gambar 3.4. Rangkaian simulasi yang digunakan terdapat pada gambar 4.1. Model dibagi ke dalam lima blok, yaitu "SPD_CTL" pengendali kecepatan dengan waktu sampling 1 ms digunakan struktur B, "RFOC" sistem pengendali fluks rotor dengan waktu sampling 0.1 ms struktur B, "PWM" menggunakan struktur A, "IM" sistem untuk motor induksi dengan struktur C, dan "ANN" sistem untuk estimator kecepatan jaringan syaraf tiruan digunakan struktur B dengan waktu sampling 1 ms.



Gambar 4.1 Rangkaian Simulasi

Blok "ANN" memiliki empat masukan yaitu tegangan dan arus αβ pada waktu sekarang, sedangkan nilai siklus sebelumnya diperoleh dari kondisi diskrit yang disimpan pada struktur B. Selain rangkaian simulasi ini terdapat juga rangkaian simulasi kendali vektor dengan sensor kecepatan dan rangkaian yang digunakan untuk memperoleh data pelatihan, keduanya dapat dilihat pada bagian LAMPIRAN. Parameter motor induksi yang digunakan pada simulasi ini terdapat pada tabel 4.1 [11].

Daya	1 HP
Jumlah pasang kutub	2
Hambatan Stator (Rs)	2,76 Ω
Hambatan Rotor (Rr)	2,90 Ω
Induktansi Stator (Ls)	234,9 mH
Induktansi Rotor (Lr)	234,9 mH
Induktansi Bersama (Lm)	227,9 mH
Inersia (J)	0,0436 kg m^2
Koefisien Gesek (B)	0,0005

Tabel 4.1 Parameter Motor Induksi

Data pelatihan sebanyak 50000 pola masukan dan keluaran akan digunakan untuk melatih jaringan. Data diperoleh dengan menggunakan rangkaian untuk memperoleh data. Rangkaian tersebut dijalankan dengan berbagai kecepatan dan besar beban untuk memberikan informasi yang cukup bagi jaringan. Nilai tegangan dan arus saat ini dan sebelumnya, juga kecepatan rotor direkam selama simulasi. Pelatihan dilakukan dengan cara menormalisasi data tersebut dan selanjutnya diproses menggunakan algoritma pelatihan secara tidak langsung dengan program yang ditulis dalam bentuk M – File.

4.2 Percobaan Variasi Parameter

Pada percobaan ini, parameter jumlah unit pada lapisan tersembunyi dan laju pembelajaran divariasikan sementara parameter yang lain diberikan nilai standar. Besar nilai parameter standar tersebut:

- Laju pembelajaran $\alpha = 0,4$
- Momentum $\mu = 0,4$
- Jumlah unit lapisan tersembunyi 30 unit
- Jumlah epoh sebagai batas berhenti 50.000
- Inisialisasi menggunakan metode Nguyen – Widrow

4.2.1 Jumlah Neuron pada Lapisan Tersembunyi

Jumlah Neuron	Waktu (detik)		MSE
	Training	Testing	
1	78,03	1,36	0.042242
2	64,05	1,45	0,041882
3	58,98	1,75	0.041783
4	60,85	1,59	0.04166
5	61,23	1,95	0.0415
6	62,82	1,95	0.041566
7	64,27	1,95	0,041183
8	64,89	2,24	0.04077
9	67,05	2,32	0.041319
20	91,82	3,55	0,00483
30	109,18	4,29	0,004285
40	129,06	5,10	0,005018
50	150,67	6,58	0,004604

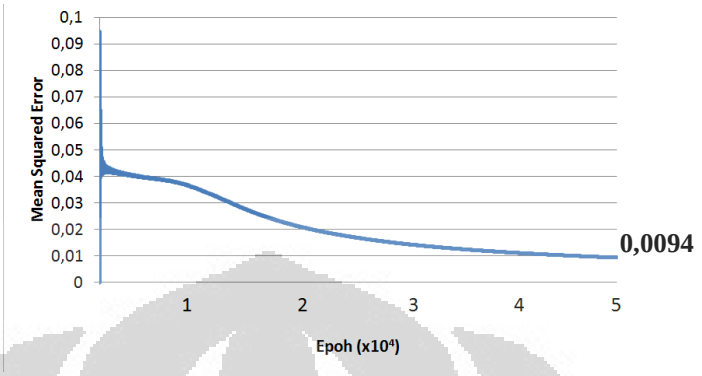
Tabel 4.2 MSE, Waktu Training dan Testing dengan Variasi Unit Lapisan Tersembunyi

Percobaan ini memvariasikan jumlah unit lapisan tersembunyi dari 1 – 50 unit dengan paramater lain bernilai standar. Pada setiap struktur dengan jumlah unit bervariasi dilakukan pelatihan menggunakan data yang dimiliki sebelumnya melalui pengukuran. Dari Tabel 4.2 dapat dilihat nilai kesalahan yang kecil terjadi ketika unit pada lapisan tersembunyi berjumlah tiga puluh. Jumlah unit pada lapisan tersembunyi mempengaruhi kemampuan jaringan dalam menyelesaikan permasalahan yang rumit. Tetapi jumlah yang banyak tidak menjamin kinerja dari jaringan menjadi lebih baik, karena mungkin saja terjadi over training [12]. Selain itu dapat dipahami juga semakin banyak unit pada lapisan tersembunyi menyebabkan semakin lama waktu yang dibutuhkan untuk pelatihan dan pengujian.

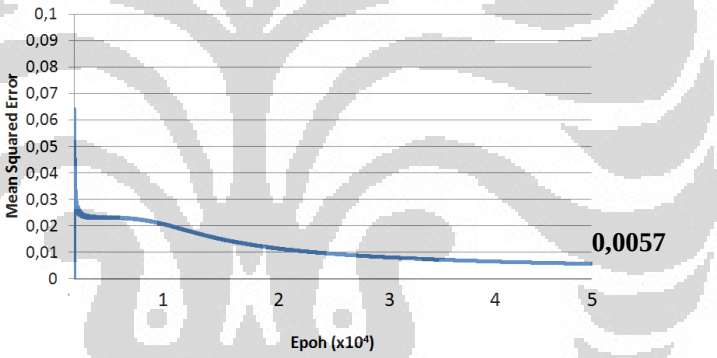
4.2.2 Besar Laju Pembelajaran

Pada percobaan ini dilakukan variasi besar laju pembelajaran sedangkan parameter yang lain bernilai standar. Laju pembelajaran merupakan parameter

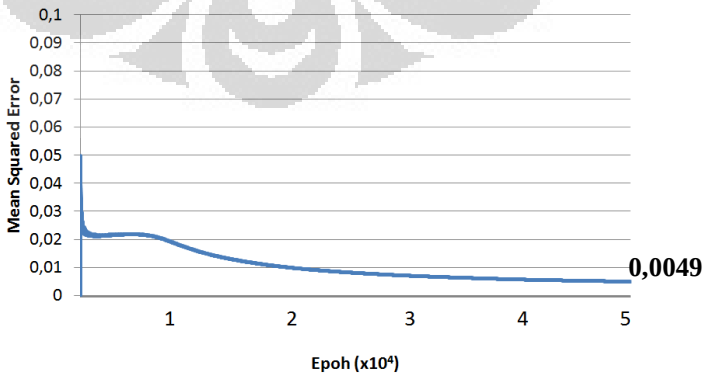
yang mempengaruhi proses pembelajaran. Jika laju pembelajaran bernilai terlalu kecil maka konvergensi akan berlangsung lambat, sedangkan jika terlalu besar akan menjadikan jaringan tidak stabil.



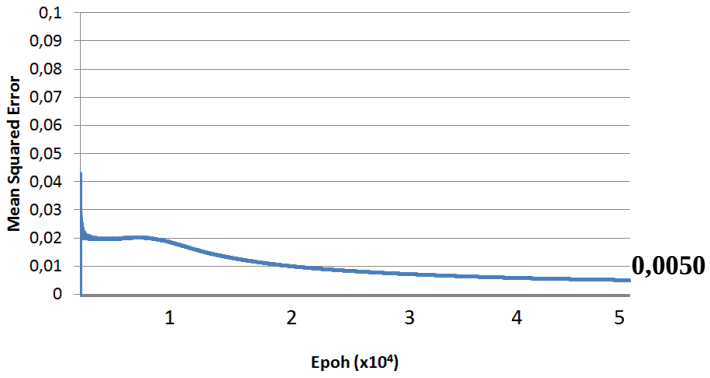
(a)



(b)



(c)



(d)

Gambar 4.2 Grafik error terhadap epoch jaringan dengan variasi laju pembelajaran. (a) α = 0,1. (b) α = 0,2. (c) α = 0,3. (d) α = 0,4

Hasil percobaan variasi laju pembelajaran ditunjukan dengan gambar 4.2. Nilai *Mean Squared Error* terendah diperoleh jika digunakan laju pembelajaran 0,3. Dari grafik tersebut dapat dilihat semakin besar laju pembelajaran maka fluktuasi dari grafik semakin besar juga. Hal ini terjadi karena jika laju pembelajaran bernilai besar maka dalam setiap iterasi bobot dan bias berubah dengan cepat. Sedangkan jika laju pembelajaran kecil, maka akan dibutuhkan jumlah iterasi yang banyak dan waktu yang lama sampai jaringan memiliki nilai kesalahan yang kecil.

4.3 Percobaan Kinerja Estimator

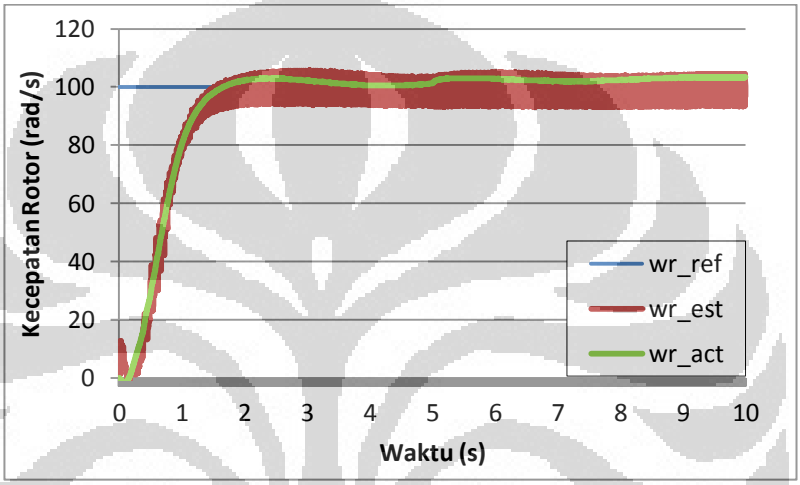
Pada percobaan ini, kinerja dari estimator kecepatan yang dirancang diuji dengan menggunakan beberapa kondisi kerja. Yaitu perubahan kecepatan dengan beban nol, perubahan kecepatan dengan beban, dan daya tahan perubahan beban.

1.3.1 Perubahan Kecepatan dengan Beban Nol

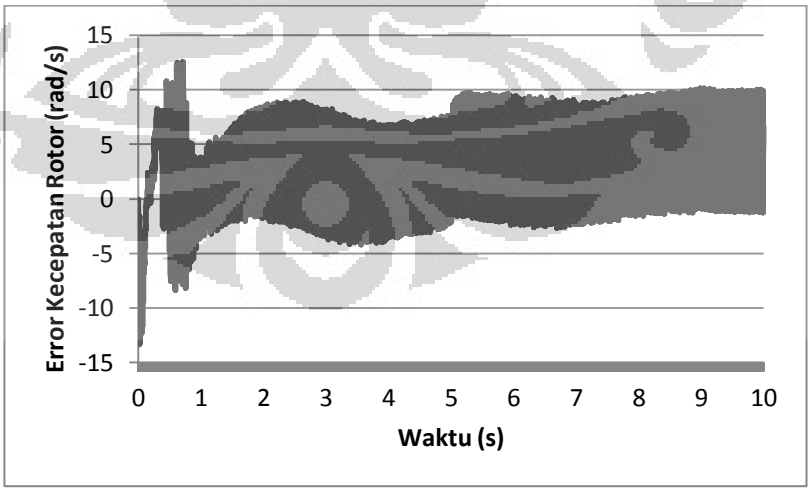
Simulasi ini bertujuan untuk mengetahui kinerja dari estimator kecepatan pada kondisi normal. Simulasi dilakukan selama 10 detik, rangkaian diberikan masukan step sebagai kecepatan referensi sebesar 100 rad/s. Respons dari masukan step tersebut diperlihatkan pada Gambar 4.2. Terhadap masukan ini

estimator memberikan respons yang cepat. Kecepatan rotor dapat sampai dan bertahan di kondisi tunak.

Pada Gambar 4.3 terdapat kurva yang menggambarkan selisih kecepatan perkiraan wr_est dan kecapatan actual wr_act. Dapat dilihat kecepatan estimasi terus berosilasi pada kecepatan tunak menyebabkan kesalahan estimasi. Pada kinerja ini estimator memberikan riak yang besar. Persentase kesalahan estimator kecepatan motor induksi pada percobaan ini 2,75%.



Gambar 4.3 Kurva kecepatan terhadap waktu pada percobaan beban nol

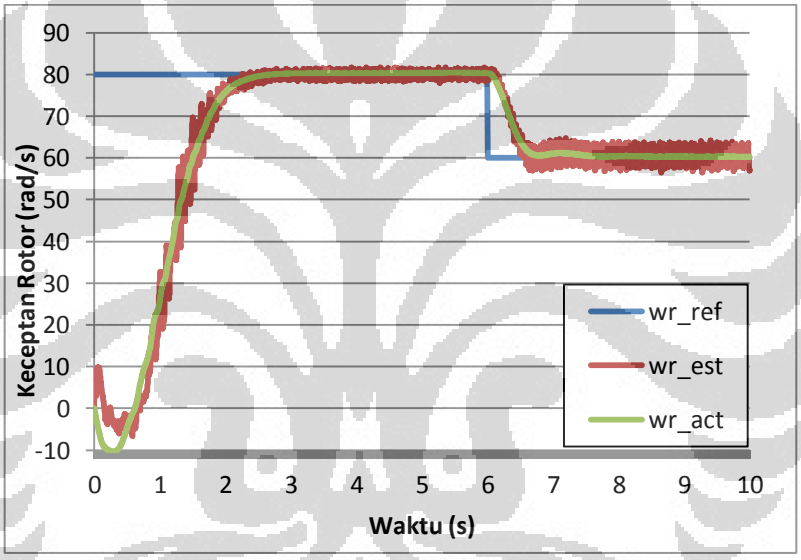


Gambar 4.4 Kesalahan estimasi kecepatan rotor pada percobaan beban nol

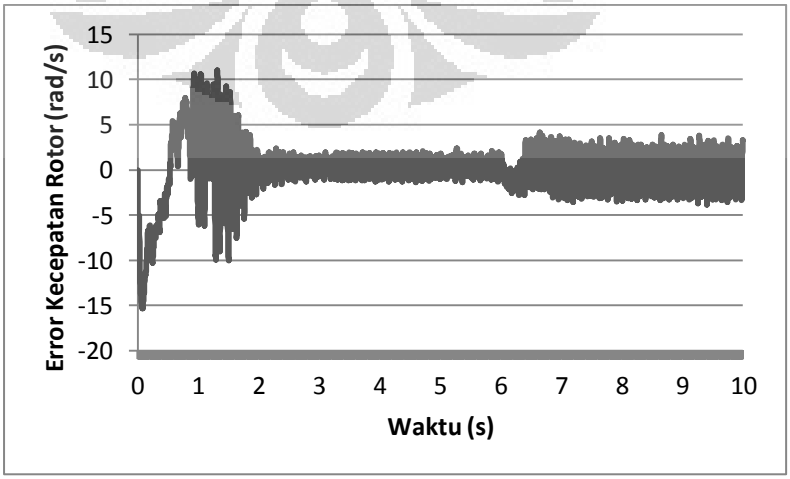
1.3.2 Perubahan Kecepatan dengan Beban 50%

Motor induksi dengan beban sebesar 50% dari torsi maksimum diberikan masukan input sebesar 80 rad/s kemudian kecepatannya diubah menjadi 60 rad/s pada waktu t = 6 detik. Tujuan dari simulasi ini untuk mengetahui kinerja estimator ketika digunakan.

Pada detik ke nol kecepatan rotor actual turun sampai negatif karena beban yang dipasangkan ke motor induksi. Pada kondisi tersebut estimator jaringan yang dirancang mengikuti juga perubahan tersebut. Persentase kesalahan estimator kecepatan motor induksi pada percobaan ini 1,81%



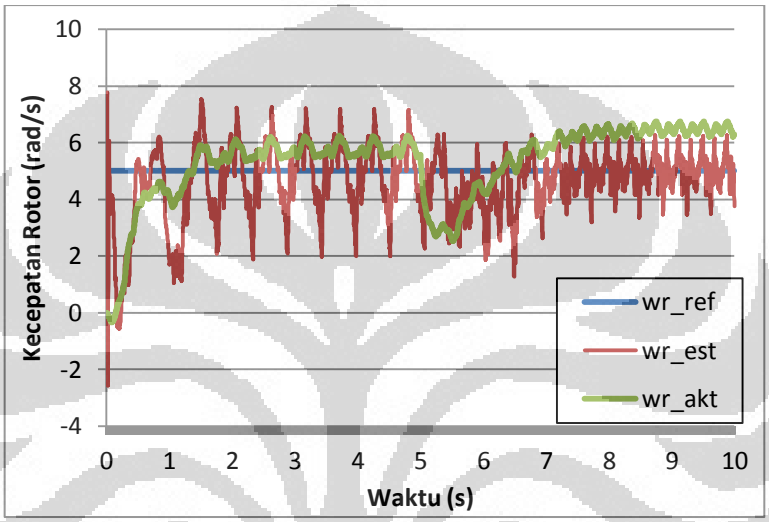
Gambar 4.5 Kurva kecepatan terhadap waktu pada percobaan beban 50%



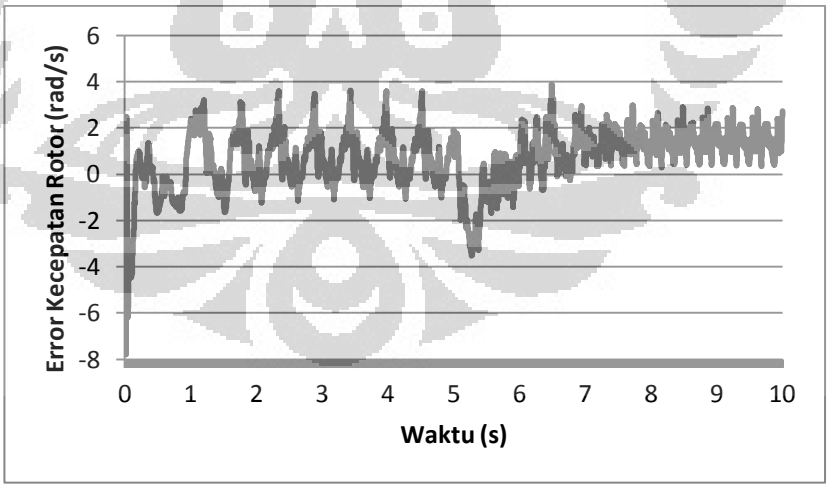
Gambar 4.6 Kesalahan estimasi kecepatan rotor pada percobaan beban 50%

4.3.3 Daya Tahan Perubahan Beban

Simulasi ini bertujuan untuk mengetahui kemampuan estimator ketika motor induksi dioperasikan pada kecepatan rendah yaitu 5 rad/s. Tujuan yang laiinya adalah untuk menguji kemampuan penggerak tanpa sensor ini untuk menahan beban gangguan. Disimulasikan dengan beban sebesar 20% beban maksimum ketika t = 5 detik.



Gambar 4.7 Kurva kecepatan terhadap waktu pada percobaan daya tahan beban



Gambar 4.8 Kesalahan estimasi kecepatan rotor pada percobaan daya tahan beban

Pada Gambar 4.7 ketika detik ke-5 kecepatan rotor aktual menurun karena pada saat itu diberikan beban. Estimator kemudian mengikuti pergerakan

dari kurva rotor aktual tersebut. Persentase kesalahan estimator kecepatan motor induksi pada percobaan ini 21,94%

Secara keseluruhan kinerja dari estimator kecepatan sudah cukup baik, karena sudah dapat mengikuti berbagai kondisi kecepatan ketika peralihan, tunak ataupun ketika diberikan beban. Padahal data pelatihan yang digunakan berbeda dengan yang diujikan pada saat ini. Hal tersebut menunjukan kemampuan generalisasi dari suatu sistem jaringan syaraf tiruan. Dari hasil simulasi yang diperoleh dapat diketahui kecepatan rotor pada percobaan kinerja dapat mencapai kecepatan referensi. Skema jaringan ini memberikan persen kesalahan antara kecepatan rotor perkiraan dan aktual yang kecil kecuali pada percobaan kinerja estimator daya tahan perubahan beban yang dilakukan pada kecepatan rendah.

BAB 5
KESIMPULAN

Berdasarkan simulasi dan analisa yang telah dilakukan, maka dapat diperoleh beberapa kesimpulan, sebagai berikut :

1. Proses identifikasi kecepatan motor induksi tiga fasa yang rumit dapat dilakukan dengan jaringan syaraf tiruan menggunakan arus dan tegangan stator sebagai masukan.
2. Dengan menggunakan metode pembelajaran dengan pengawas, algoritma belajar propagasi balik, laju pembelajaran = 0,3, struktur jaringan lapis banyak 8 – 30 – 1, dapat diperoleh MSE = 0,0049.
3. Percobaan variasi parameter menunjukan penambahan unit pada lapisan tersembunyi menyebabkan semakin lamanya waktu yang dibutuhkan untuk melakukan pelatihan dan pengujian. Sedangkan laju pembelajaran mempengaruhi kecepatan perubahan bobot – bias juga kefluktuatifan dari kurva error.
4. Estimator kecepatan motor induksi tiga fasa berbasis jaringan syaraf tiruan telah menunjukan kinerja yang cukup baik. Persen kesalahan antara kecepatan rotor estimasi dengan aktual untuk percobaan perubahan kecepatan dengan beban nol = 2,75%, percobaan perubahan kecepatan dengan beban 50% = 1,81%, percobaan daya tahan perubahan beban = 21,94%.

DAFTAR REFERENSI

[1] Stephen J. Chapman. Electric *Machinery and Power System Fundamentals.* New York : Mc Graw Hill, 2002.

[2] Texas Instruments Europe. *Field Oriented Control of 3-Phase AC-Motors.* Texas: Texas Instruments Europe, 1998.

[3] Peter Vas. *Sensorless Vector and Direct Torque Control*. New York : Oxford University Press, 1998.

[4] S. Wade, M. W. Dunnigan, and B. W. Williams. "Modeling and Simulation of Induction Machine Vector Control with Rotor Resistance Identification." *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 495-506,1997.

[5] Shady M. Gadoue, Damian Giaouris, and John W. Finch. "Sensorless Control of Induction Motor Drives at Very Low and Zero Speeds Using Neural Network Flux Observers." *IEEE Trans. Industrial Electron.*, vol. 56, no. 8, August 2009.

[6] The Institution of Electrical Engineers. *Neural Network Applications in Control.* London : The Institution of Electrical Engineers, 1995.

[7] F. Yusivar, H. Haratsu, T. Kitahara, S. Wakao, and T. Onuki. "Performance Comparison of the Controller Configurations for the Sensorless IM Drive Using the Modified Speed Adaptive Observer." *IEE-PEVD 2000,* Conference Publication no. 475, pp. 194-200, 2000.

[8] Feri Yusivar, S. Wakao. "Minimum Requirements of Motor Vector Control Modeling and Simulation Utilizing C MEX S-Function in MATLAB/SIMULINK."

[9] Fery. *Pengendali Vektor Arus dan Perbaikan Kesalahan Estimasi pada Motor Induksi Tanpa Sensor Kecepatan.* S.T thesis, Departemen Teknik Elektro, Universitas Indonesia, Depok, July 2004.

[10] Ghang-Ming Liaw and Faa-Jeng Lin. "A Robust Speed Controller for Induction Motor Drives." *IEEE Trans. Industrial Electron.*, vol. 41, no. 3, June 1994.

[11] Anne Prasetyowati. *Pengendalian Adaptif Fuzzy untuk Self Tuning PI Kontrol Kecepatan Motor Induksi Tiga Fasa tanpa Sensor Kecepatan dengan Observer MRAS.* M.T thesis, Departemen Teknik Elektro, Universitas Indonesia, Depok, November 2008.

[12] Fahmi Ardi, Jatmiko Endro Suseno, dan K Sofyan Firdausi. Pembuatan Perangkat Lunak untuk Menentukan Panjang Gelombang Berdasarkan Spektrum Cahaya Tampak dengan Metode Jaringan Syaraf Tiruan Menggunakan Matlab 7.0. Jurnal Universitas, Departemen Fisika, Universitas Diponegoro, Semarang.

DAFTAR PUSTAKA

Chapman, Stephen J., *Electric Machinery and Power System Fundamentals.* New York : Mc Graw Hill, 2002.

Chee-Mun, O. *Dynamic Simulation of Electric Machinery using MATLAB/Simulink.* New Jersey : Prentice Hall, 1998.

Fauzi, M. Ilham. *Aplikasi Jaringan Syaraf Tiruan untuk Identifikasi Sistem Kiln Semen.* M.T thesis, Departemen Teknik Elektro, Universitas Indonesia, Depok, Desember 1999.

Fery. *Pengendali Vektor Arus dan Perbaikan Kesalahan Estimasi pada Motor Induksi Tanpa Sensor Kecepatan.* S.T thesis, Departemen Teknik Elektro, Universitas Indonesia, Depok, July 2004.

Gadoue, Shady M., Damian Giaouris, and John W. Finch. "Sensorless Control of Induction Motor Drives at Very Low and Zero Speeds Using Neural Network Flux Observers." *IEEE Trans. Industrial Electron.*, vol. 56, no. 8, August 2009.

Krishnan, R. *Electric Motor Drives: Modelling, Analysis and Control*. New York : Prentice Hall, 2001.

Prasetyowati, Anne. *Pengendalian Adaptif Fuzzy untuk Self Tuning PI Kontrol Kecepatan Motor Induksi Tiga Fasa tanpa Sensor Kecepatan dengan Observer MRAS.* M.T thesis, Departemen Teknik Elektro, Universitas Indonesia, Depok, November 2008.

The Institution of Electrical Engineers. *Neural Network Applications in Control.* London : The Institution of Electrical Engineers, 1995.

The Mathworks Inc., SIMULINK, Dynamic System Simulation for Matlab, Using SIMULINK Version 3, 1998.

The Mathworks Inc., SIMULINK, Dynamic System Simulation for Matlab, Writing S-Function Version 3, 1998.

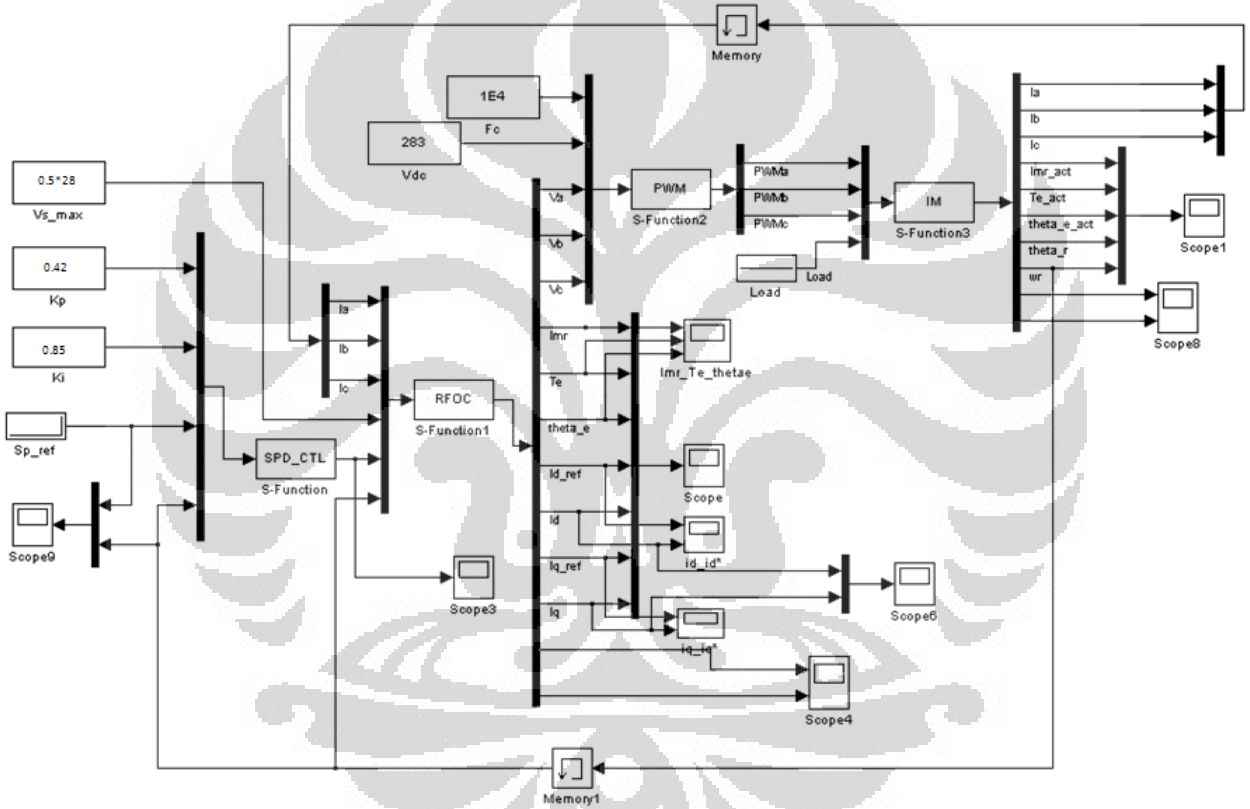
Tripathi, Awneesh Kumar. *Position Sensorless Vector Control of Induction Motor including Field Weakening Mode of operation.* M. Eng. Thesis, EE Dept, IISc, Bangalore, June 2008.

Yusivar, F., S. Wakao. "Minimum Requirements of Motor Vector Control Modeling and Simulation Utilizing C MEX S-Function in MATLAB/SIMULINK."

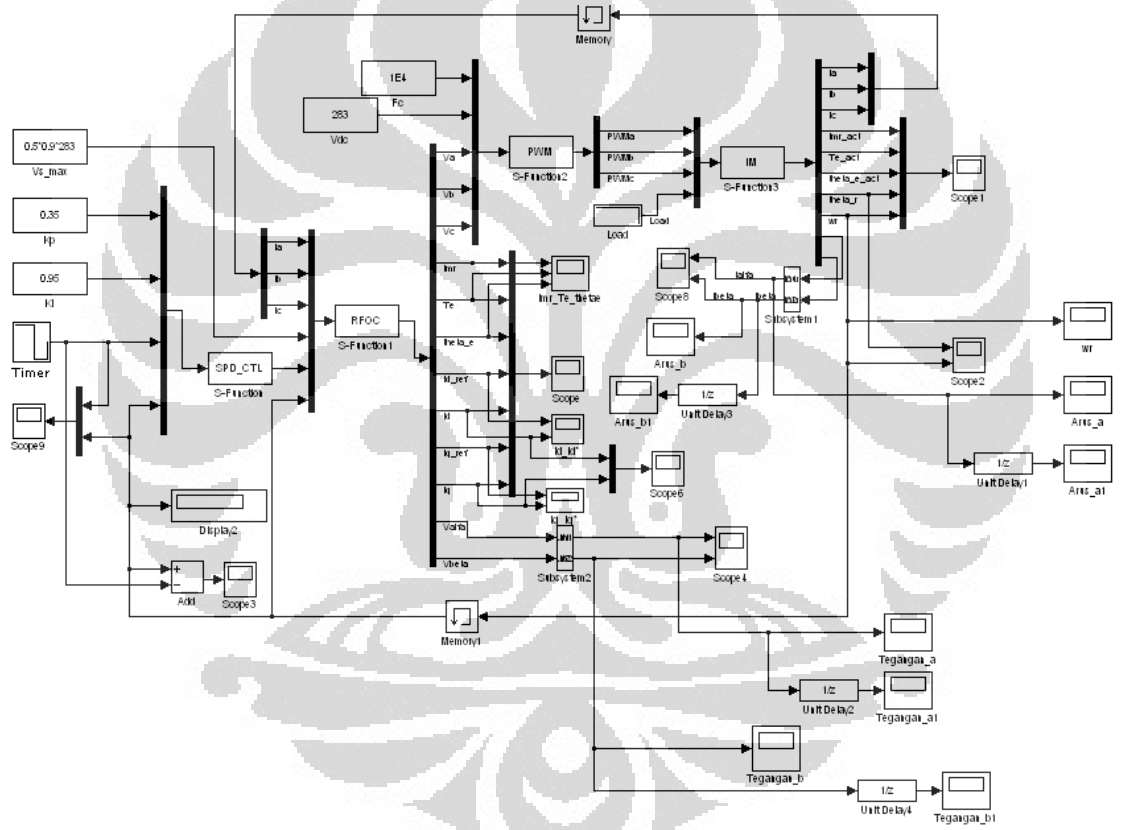
Vas, Peter. *Sensorless Vector and Direct Torque Control*. New York : Oxford University Press, 1998.

LAMPIRAN 1

Gambar Blok Diagram Motor Induksi Dengan Sensor Kecepatan



LAMPIRAN 2
Gambar Blok Diagram Motor Induksi untuk Data Pelatihan



LAMPIRAN 3
Kode Program C-MEX Blok RFOC

```
/* == SOURCE file list of "RFOC.c" with Structure B == */
#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME RFOC
#include "simstruc.h"
#include <math.h>

#define U(element) (*uPtrs[element]) /*Pointer to Input Port0*/

static void mdlInitializeSizes(SimStruct *S){
ssSetNumDiscStates(S, 16);
if (!ssSetNumInputPorts(S, 1)) return;
ssSetInputPortWidth(S, 0, 6);
ssSetInputPortDirectFeedThrough(S, 0, 1);
ssSetInputPortOverWritable(S, 0, 1);
if (!ssSetNumOutputPorts(S, 1)) return;
ssSetOutputPortWidth(S, 0, 12);
ssSetNumSampleTimes(S, 1);

ssSetOptions(S, (SS_OPTION_EXCEPTION_FREE_CODE
| SS_OPTION_DISCRETE_VALUED_OUTPUT));}

static void mdlInitializeSampleTimes(SimStruct *S){
ssSetSampleTime(S, 0, 1e-4);
ssSetOffsetTime(S, 0, 0.0);}

#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S){
real_T *X0 = ssGetRealDiscStates(S);
int_T nXStates = ssGetNumDiscStates(S);
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
int_T i;

/* initialize the states to 0.0 */
for (i=0; i < nXStates; i++) {
X0[i] = 0.0; }
} /* end mdlInitializeConditions*/

real_T K = 0.816497, L = 0.866025;
real_T Lm = 0.2279, Lr = 0.2349, Ls = 0.2349;
real_T Np = 2.0, Sig = 0.058711794789264;

static void mdlOutputs(SimStruct *S, int_T tid){
real_T *Y = ssGetOutputPortRealSignal(S,0);
real_T *X = ssGetRealDiscStates(S);

Y[0] = X[0];
Y[1] = X[1];
Y[2] = X[14];
Y[3] = X[2];
Y[4] = Np*(1-Sig)*Ls*X[7]*X[2];
Y[5] = X[3];
Y[6] = X[4];
Y[7] = X[5];
```

```
Y[8] = X[6];
Y[9] = X[7];
Y[10]= X[10];
Y[11]= X[11];
}/* end mdlOutputs*/

#define MDL_UPDATE /* Change to #undef to remove function. */
#if defined(MDL_UPDATE)
static void mdlUpdate(SimStruct *S, int_T tid){
real_T *X = ssGetRealDiscStates(S);
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

real_T ialph, ibet, imr, dimr,imr_new, Ids1, Iqs1, Ia, Ib, Ic;
real_T dt = 1e-4, Kidp = 0.17026420488886 , Kiqp =
0.17026420488886, Kidi = 34.074074074074074, Kiqi =
34.074074074074074, Rr = 2.9;
real_T we, theta_e_new, iqs_ref, Uds, Uqs, Vcd, Vcq, pi2 =
6.28318530718;
real_T Vds, Vqs, iqs_lim, Uds_temp_new, Uqs_temp_new;
real_T is_max = 6.069665441, Valph, Vbeth, alpha = 0.987804878,
betha = 0.385749;
real_T Kvp = 0.0006;


/* imr */
imr = X[2];
imr_new = (Rr*dt*X[12]+Lr*imr)/(Rr*dt+Lr);
dimr = (imr_new - imr)/dt;
if (imr_new < 0.001){imr_new = 0.001;}
X[2] = imr_new;

/* ids_ref iqs_ref */
X[4] = 3;
iqs_lim = sqrt(is_max*is_max-X[4]*X[4]);
iqs_ref = U(4)/(Np*Ls*(1-Sig)*X[2]);
if (iqs_lim < iqs_ref){iqs_ref = iqs_lim;}
if (-iqs_lim > iqs_ref) { iqs_ref = -iqs_lim;}
X[6]=iqs_ref;

/* Ids1 dan Iqs1*/
Ids1 = X[4]+alpha*(X[12]-X[4]);
X[12] = Ids1;
Iqs1 = X[6]+alpha*(X[13]-X[6]);
X[13] = Iqs1;

/* theta_e */
we = Np*U(5)+(Rr/Lr)*(X[13]/X[2]);
theta_e_new = X[3]+dt*we;
if (theta_e_new > pi2){theta_e_new = 0;}
if (theta_e_new < 0){theta_e_new = pi2;}
X[3]=theta_e_new;

ialph = K*(U(0)-0.5*U(1)-0.5*U(2));
ibet = K*L*(U(1)-U(2));
X[5] = ialph* cos(X[3])+ibet* sin(X[3]);
X[7] = -ialph*sin(X[3])+ibet*cos(X[3]);
```

```
/* Valph dan Vbeth, pengendali arus */
Uds_temp_new = X[8] + dt*Kidi*(X[4]-X[5]);
X[8] = Uds_temp_new;
Uds = Kidp*(X[4]-X[5])+X[8];

Uqs_temp_new = X[9] + dt*Kiqi*(X[6]-X[7]);
X[9] = Uqs_temp_new;
Uqs = Kiqp*(X[6]-X[7])+X[9];

Vcd = -we*Ls*Sig*X[13];
Vcq = we*Ls*Sig*X[12]+ we*Ls*(1-Sig)*X[2];

Vds = Uds+Vcd;
Vqs = Uqs+Vcq;

Valph = Vds*cos(X[3])-Vqs*sin(X[3]);
Vbeth = Vds*sin(X[3])+Vqs*cos(X[3]);

X[0] = K*Valph;
X[1] = (-0.5*Valph+L*Vbeth)*K;
X[14] = (-0.5*Valph-L*Vbeth)*K;

if (127.35 < X[0]){X[0] = 127.35;}
if (-127.35 > X[0]) {X[0] = -127.35;}
if (127.35 < X[1]){X[1] = 127.35;}
if (-127.35 > X[1]) {X[1] = -127.35;}
if (127.35 < X[14]){X[14] = 127.35;}
if (-127.35 > X[14]) {X[14] = -127.35;}

X[10]= Valph;
X[11]= Vbeth;
}
#endif /* MDL_UPDATE */


static void mdlTerminate(SimStruct *S)
{ } /*Keep this function empty since no memory is allocated*/

#ifdef MATLAB_MEX_FILE
/* Is this file being compiled as a MEX-file? */
#include "simulink.c" /*MEX-file interface mechanism*/
#else
#include "cg_sfun.h" /*Code generation registration function*/
#endif
```

LAMPIRAN 4
Kode Program C-MEX Blok ANN

```
/* == SOURCE file list of "ANN.c" with Structure B == */
#define S_FUNCTION_LEVEL 2
#define S_FUNCTION_NAME ANN
#include "simstruc.h"
#include <math.h>

#define U(element) (*uPtrs[element]) /*Pointer to Input Port0*/
#define A(S) ssGetSFcnParam(S, 0)
#define B(S) ssGetSFcnParam(S, 1)
#define C(S) ssGetSFcnParam(S, 2)
#define D(S) ssGetSFcnParam(S, 3)

static void mdlInitializeSizes(SimStruct *S){
ssSetNumDiscStates(S, 5);
if (!ssSetNumInputPorts(S, 1)) return;
ssSetInputPortWidth(S, 0, 4);
ssSetInputPortDirectFeedThrough(S, 0, 1);
ssSetInputPortOverWritable(S, 0, 1);
if (!ssSetNumOutputPorts(S, 1)) return;
ssSetOutputPortWidth(S, 0, 1);
ssSetNumSampleTimes(S, 1);
ssSetNumSFcnParams(S, 4);
ssSetSFcnParamNotTunable (S,0);
ssSetSFcnParamNotTunable (S,1);
ssSetSFcnParamNotTunable (S,2);
ssSetSFcnParamNotTunable (S,3);

ssSetOptions(S, (SS_OPTION_EXCEPTION_FREE_CODE
| SS_OPTION_DISCRETE_VALUED_OUTPUT));}

static void mdlInitializeSampleTimes(SimStruct *S){
ssSetSampleTime(S, 0, 1e-3);
ssSetOffsetTime(S, 0, 0.0);}

#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S){
real_T *X0 = ssGetRealDiscStates(S);
int_T nXStates = ssGetNumDiscStates(S);
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
int_T i;

/* initialize the states to 0.0 */
for (i=0; i < nXStates; i++) {
X0[i] = 0.0; } }

static void mdlOutputs(SimStruct *S, int_T tid){
real_T *Y = ssGetOutputPortRealSignal(S,0); /* memberikan blok
keluaran */
real_T *X = ssGetRealDiscStates(S);

Y[0] = X[4];
}
```

```
#define MDL_UPDATE
static void mdlUpdate(SimStruct *S, int_T tid) {
real_T *X = ssGetRealDiscStates(S);
InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

/*Inisialisasi Variabel*/
real_T Ia, Ia1, Ib, Ib1, Va, Va1, Vb, Vb1;
real_T Zin[30], Z[30];
real_T Yin = mxGetPr(C(S))[0];
real_T i, Yk;

/*Normalisasi dan delay input*/
Ia = U(0)/6.5;
Ib = U(1)/6.5;
Va = U(2)/160;
Vb = U(3)/160;

Ia1 = X[0];
Ib1 = X[1];
Va1 = X[2];
Vb1 = X[3];

X[0]= Ia;
X[1]= Ib;
X[2]= Va;
X[3]= Vb;

/*Penghitungan Lapisan Tersembunyi*/

Zin[0] =
mxGetPr(A(S))[0]+mxGetPr(B(S))[0]*Ia+mxGetPr(B(S))[1]*Ia1+mxGetPr(
B(S))[2]*Ib+mxGetPr(B(S))[3]*Ib1

+mxGetPr(B(S))[4]*Va+mxGetPr(B(S))[5]*Va1+mxGetPr(B(S))[6]*Vb+mxGe
tPr(B(S))[7]*Vb1;

Zin[1] =
mxGetPr(A(S))[1]+mxGetPr(B(S))[8]*Ia+mxGetPr(B(S))[9]*Ia1+mxGetPr(
B(S))[10]*Ib+mxGetPr(B(S))[11]*Ib1

+mxGetPr(B(S))[12]*Va+mxGetPr(B(S))[13]*Va1+mxGetPr(B(S))[14]*Vb+m
xGetPr(B(S))[15]*Vb1;

Zin[2] =
mxGetPr(A(S))[2]+mxGetPr(B(S))[16]*Ia+mxGetPr(B(S))[17]*Ia1+mxGetP
r(B(S))[18]*Ib+mxGetPr(B(S))[19]*Ib1

+mxGetPr(B(S))[20]*Va+mxGetPr(B(S))[21]*Va1+mxGetPr(B(S))[22]*Vb+m
xGetPr(B(S))[23]*Vb1;

Zin[3] =
mxGetPr(A(S))[3]+mxGetPr(B(S))[24]*Ia+mxGetPr(B(S))[25]*Ia1+mxGetP
r(B(S))[26]*Ib+mxGetPr(B(S))[27]*Ib1

+mxGetPr(B(S))[28]*Va+mxGetPr(B(S))[29]*Va1+mxGetPr(B(S))[30]*Vb+m
xGetPr(B(S))[31]*Vb1;
```

```
Zin[4] =
mxGetPr(A(S))[4]+mxGetPr(B(S))[32]*Ia+mxGetPr(B(S))[33]*Ia1+mxGetP
r(B(S))[34]*Ib+mxGetPr(B(S))[35]*Ib1

+mxGetPr(B(S))[36]*Va+mxGetPr(B(S))[37]*Va1+mxGetPr(B(S))[38]*Vb+m
xGetPr(B(S))[39]*Vb1;

Zin[5] =
mxGetPr(A(S))[5]+mxGetPr(B(S))[40]*Ia+mxGetPr(B(S))[41]*Ia1+mxGetP
r(B(S))[42]*Ib+mxGetPr(B(S))[43]*Ib1

+mxGetPr(B(S))[44]*Va+mxGetPr(B(S))[45]*Va1+mxGetPr(B(S))[46]*Vb+m
xGetPr(B(S))[47]*Vb1;

Zin[6] =
mxGetPr(A(S))[6]+mxGetPr(B(S))[48]*Ia+mxGetPr(B(S))[49]*Ia1+mxGetP
r(B(S))[50]*Ib+mxGetPr(B(S))[51]*Ib1

+mxGetPr(B(S))[52]*Va+mxGetPr(B(S))[53]*Va1+mxGetPr(B(S))[54]*Vb+m
xGetPr(B(S))[55]*Vb1;

Zin[7] =
mxGetPr(A(S))[7]+mxGetPr(B(S))[56]*Ia+mxGetPr(B(S))[57]*Ia1+mxGetP
r(B(S))[58]*Ib+mxGetPr(B(S))[59]*Ib1

+mxGetPr(B(S))[60]*Va+mxGetPr(B(S))[61]*Va1+mxGetPr(B(S))[62]*Vb+m
xGetPr(B(S))[63]*Vb1;

Zin[8] =
mxGetPr(A(S))[8]+mxGetPr(B(S))[64]*Ia+mxGetPr(B(S))[65]*Ia1+mxGetP
r(B(S))[66]*Ib+mxGetPr(B(S))[67]*Ib1

+mxGetPr(B(S))[68]*Va+mxGetPr(B(S))[69]*Va1+mxGetPr(B(S))[70]*Vb+m
xGetPr(B(S))[71]*Vb1;

Zin[9] =
mxGetPr(A(S))[9]+mxGetPr(B(S))[72]*Ia+mxGetPr(B(S))[73]*Ia1+mxGetP
r(B(S))[74]*Ib+mxGetPr(B(S))[75]*Ib1

+mxGetPr(B(S))[76]*Va+mxGetPr(B(S))[77]*Va1+mxGetPr(B(S))[78]*Vb+m
xGetPr(B(S))[79]*Vb1;

Zin[10] =
mxGetPr(A(S))[10]+mxGetPr(B(S))[80]*Ia+mxGetPr(B(S))[81]*Ia1+mxGet
Pr(B(S))[82]*Ib+mxGetPr(B(S))[83]*Ib1

+mxGetPr(B(S))[84]*Va+mxGetPr(B(S))[85]*Va1+mxGetPr(B(S))[86]*Vb+m
xGetPr(B(S))[87]*Vb1;

Zin[11] =
mxGetPr(A(S))[11]+mxGetPr(B(S))[88]*Ia+mxGetPr(B(S))[89]*Ia1+mxGet
Pr(B(S))[90]*Ib+mxGetPr(B(S))[91]*Ib1

+mxGetPr(B(S))[92]*Va+mxGetPr(B(S))[93]*Va1+mxGetPr(B(S))[94]*Vb+m
xGetPr(B(S))[95]*Vb1;
```

```
Zin[12] =
mxGetPr(A(S))[12]+mxGetPr(B(S))[96]*Ia+mxGetPr(B(S))[97]*Ia1+mxGet
Pr(B(S))[98]*Ib+mxGetPr(B(S))[99]*Ib1

+mxGetPr(B(S))[100]*Va+mxGetPr(B(S))[101]*Va1+mxGetPr(B(S))[102]*V
b+mxGetPr(B(S))[103]*Vb1;

Zin[13] =
mxGetPr(A(S))[13]+mxGetPr(B(S))[104]*Ia+mxGetPr(B(S))[105]*Ia1+mxG
etPr(B(S))[106]*Ib+mxGetPr(B(S))[107]*Ib1

+mxGetPr(B(S))[108]*Va+mxGetPr(B(S))[109]*Va1+mxGetPr(B(S))[110]*V
b+mxGetPr(B(S))[111]*Vb1;

Zin[14] =
mxGetPr(A(S))[14]+mxGetPr(B(S))[112]*Ia+mxGetPr(B(S))[113]*Ia1+mxG
etPr(B(S))[114]*Ib+mxGetPr(B(S))[115]*Ib1

+mxGetPr(B(S))[116]*Va+mxGetPr(B(S))[117]*Va1+mxGetPr(B(S))[118]*V
b+mxGetPr(B(S))[119]*Vb1;

Zin[15] =
mxGetPr(A(S))[15]+mxGetPr(B(S))[120]*Ia+mxGetPr(B(S))[121]*Ia1+mxG
etPr(B(S))[122]*Ib+mxGetPr(B(S))[123]*Ib1

+mxGetPr(B(S))[124]*Va+mxGetPr(B(S))[125]*Va1+mxGetPr(B(S))[126]*V
b+mxGetPr(B(S))[127]*Vb1;

Zin[16] =
mxGetPr(A(S))[16]+mxGetPr(B(S))[128]*Ia+mxGetPr(B(S))[129]*Ia1+mxG
etPr(B(S))[130]*Ib+mxGetPr(B(S))[131]*Ib1

+mxGetPr(B(S))[132]*Va+mxGetPr(B(S))[133]*Va1+mxGetPr(B(S))[134]*V
b+mxGetPr(B(S))[135]*Vb1;

Zin[17] =
mxGetPr(A(S))[17]+mxGetPr(B(S))[136]*Ia+mxGetPr(B(S))[137]*Ia1+mxG
etPr(B(S))[138]*Ib+mxGetPr(B(S))[139]*Ib1

+mxGetPr(B(S))[140]*Va+mxGetPr(B(S))[141]*Va1+mxGetPr(B(S))[142]*V
b+mxGetPr(B(S))[143]*Vb1;

Zin[18] =
mxGetPr(A(S))[18]+mxGetPr(B(S))[144]*Ia+mxGetPr(B(S))[145]*Ia1+mxG
etPr(B(S))[146]*Ib+mxGetPr(B(S))[147]*Ib1

+mxGetPr(B(S))[148]*Va+mxGetPr(B(S))[149]*Va1+mxGetPr(B(S))[150]*V
b+mxGetPr(B(S))[151]*Vb1;

Zin[19] =
mxGetPr(A(S))[19]+mxGetPr(B(S))[152]*Ia+mxGetPr(B(S))[153]*Ia1+mxG
etPr(B(S))[154]*Ib+mxGetPr(B(S))[155]*Ib1

+mxGetPr(B(S))[156]*Va+mxGetPr(B(S))[157]*Va1+mxGetPr(B(S))[158]*V
b+mxGetPr(B(S))[159]*Vb1;
```

```
Zin[20] =
mxGetPr(A(S))[20]+mxGetPr(B(S))[160]*Ia+mxGetPr(B(S))[161]*Ia1+mxG
etPr(B(S))[162]*Ib+mxGetPr(B(S))[163]*Ib1

+mxGetPr(B(S))[164]*Va+mxGetPr(B(S))[165]*Va1+mxGetPr(B(S))[166]*V
b+mxGetPr(B(S))[167]*Vb1;

Zin[21] =
mxGetPr(A(S))[21]+mxGetPr(B(S))[168]*Ia+mxGetPr(B(S))[169]*Ia1+mxG
etPr(B(S))[170]*Ib+mxGetPr(B(S))[171]*Ib1

+mxGetPr(B(S))[172]*Va+mxGetPr(B(S))[173]*Va1+mxGetPr(B(S))[174]*V
b+mxGetPr(B(S))[175]*Vb1;

Zin[22] =
mxGetPr(A(S))[22]+mxGetPr(B(S))[176]*Ia+mxGetPr(B(S))[177]*Ia1+mxG
etPr(B(S))[178]*Ib+mxGetPr(B(S))[179]*Ib1

+mxGetPr(B(S))[180]*Va+mxGetPr(B(S))[181]*Va1+mxGetPr(B(S))[182]*V
b+mxGetPr(B(S))[183]*Vb1;

Zin[23] =
mxGetPr(A(S))[23]+mxGetPr(B(S))[184]*Ia+mxGetPr(B(S))[185]*Ia1+mxG
etPr(B(S))[186]*Ib+mxGetPr(B(S))[187]*Ib1

+mxGetPr(B(S))[188]*Va+mxGetPr(B(S))[189]*Va1+mxGetPr(B(S))[190]*V
b+mxGetPr(B(S))[191]*Vb1;

Zin[24] =
mxGetPr(A(S))[24]+mxGetPr(B(S))[192]*Ia+mxGetPr(B(S))[193]*Ia1+mxG
etPr(B(S))[194]*Ib+mxGetPr(B(S))[195]*Ib1

+mxGetPr(B(S))[196]*Va+mxGetPr(B(S))[197]*Va1+mxGetPr(B(S))[198]*V
b+mxGetPr(B(S))[199]*Vb1;

Zin[25] =
mxGetPr(A(S))[25]+mxGetPr(B(S))[200]*Ia+mxGetPr(B(S))[201]*Ia1+mxG
etPr(B(S))[202]*Ib+mxGetPr(B(S))[203]*Ib1

+mxGetPr(B(S))[204]*Va+mxGetPr(B(S))[205]*Va1+mxGetPr(B(S))[206]*V
b+mxGetPr(B(S))[207]*Vb1;

Zin[26] =
mxGetPr(A(S))[26]+mxGetPr(B(S))[208]*Ia+mxGetPr(B(S))[209]*Ia1+mxG
etPr(B(S))[210]*Ib+mxGetPr(B(S))[211]*Ib1

+mxGetPr(B(S))[212]*Va+mxGetPr(B(S))[213]*Va1+mxGetPr(B(S))[214]*V
b+mxGetPr(B(S))[215]*Vb1;

Zin[27] =
mxGetPr(A(S))[27]+mxGetPr(B(S))[216]*Ia+mxGetPr(B(S))[217]*Ia1+mxG
etPr(B(S))[218]*Ib+mxGetPr(B(S))[219]*Ib1

+mxGetPr(B(S))[220]*Va+mxGetPr(B(S))[221]*Va1+mxGetPr(B(S))[222]*V
b+mxGetPr(B(S))[223]*Vb1;
```

```
Zin[28] =
mxGetPr(A(S))[28]+mxGetPr(B(S))[224]*Ia+mxGetPr(B(S))[225]*Ia1+mxG
etPr(B(S))[226]*Ib+mxGetPr(B(S))[227]*Ib1

+mxGetPr(B(S))[228]*Va+mxGetPr(B(S))[229]*Va1+mxGetPr(B(S))[230]*V
b+mxGetPr(B(S))[231]*Vb1;

Zin[29] =
mxGetPr(A(S))[29]+mxGetPr(B(S))[232]*Ia+mxGetPr(B(S))[233]*Ia1+mxG
etPr(B(S))[234]*Ib+mxGetPr(B(S))[235]*Ib1

+mxGetPr(B(S))[236]*Va+mxGetPr(B(S))[237]*Va1+mxGetPr(B(S))[238]*V
b+mxGetPr(B(S))[239]*Vb1;


Z[0] = (1-exp(-Zin[0]))/(1+exp(-Zin[0]));
Z[1] = (1-exp(-Zin[1]))/(1+exp(-Zin[1]));
Z[2] = (1-exp(-Zin[2]))/(1+exp(-Zin[2]));
Z[3] = (1-exp(-Zin[3]))/(1+exp(-Zin[3]));
Z[4] = (1-exp(-Zin[4]))/(1+exp(-Zin[4]));
Z[5] = (1-exp(-Zin[5]))/(1+exp(-Zin[5]));
Z[6] = (1-exp(-Zin[6]))/(1+exp(-Zin[6]));
Z[7] = (1-exp(-Zin[7]))/(1+exp(-Zin[7]));
Z[8] = (1-exp(-Zin[8]))/(1+exp(-Zin[8]));
Z[9] = (1-exp(-Zin[9]))/(1+exp(-Zin[9]));
Z[10] = (1-exp(-Zin[10]))/(1+exp(-Zin[10]));
Z[11] = (1-exp(-Zin[11]))/(1+exp(-Zin[11]));
Z[12] = (1-exp(-Zin[12]))/(1+exp(-Zin[12]));
Z[13] = (1-exp(-Zin[13]))/(1+exp(-Zin[13]));
Z[14] = (1-exp(-Zin[14]))/(1+exp(-Zin[14]));
Z[15] = (1-exp(-Zin[15]))/(1+exp(-Zin[15]));
Z[16] = (1-exp(-Zin[16]))/(1+exp(-Zin[16]));
Z[17] = (1-exp(-Zin[17]))/(1+exp(-Zin[17]));
Z[18] = (1-exp(-Zin[18]))/(1+exp(-Zin[18]));
Z[19] = (1-exp(-Zin[19]))/(1+exp(-Zin[19]));
Z[20] = (1-exp(-Zin[20]))/(1+exp(-Zin[20]));
Z[21] = (1-exp(-Zin[21]))/(1+exp(-Zin[21]));
Z[22] = (1-exp(-Zin[22]))/(1+exp(-Zin[22]));
Z[23] = (1-exp(-Zin[23]))/(1+exp(-Zin[23]));
Z[24] = (1-exp(-Zin[24]))/(1+exp(-Zin[24]));
Z[25] = (1-exp(-Zin[25]))/(1+exp(-Zin[25]));
Z[26] = (1-exp(-Zin[26]))/(1+exp(-Zin[26]));
Z[27] = (1-exp(-Zin[27]))/(1+exp(-Zin[27]));
Z[28] = (1-exp(-Zin[28]))/(1+exp(-Zin[28]));
Z[29] = (1-exp(-Zin[29]))/(1+exp(-Zin[29]));


/*Penghitungan Lapisan Output*/
Yin = Yin + Z[0]*mxGetPr(D(S))[0] + Z[1]*mxGetPr(D(S))[1] +
Z[2]*mxGetPr(D(S))[2] + Z[3]*mxGetPr(D(S))[3] +
Z[4]*mxGetPr(D(S))[4] +
         Z[5]*mxGetPr(D(S))[5] + Z[6]*mxGetPr(D(S))[6] +
Z[7]*mxGetPr(D(S))[7] + Z[8]*mxGetPr(D(S))[8] +
Z[9]*mxGetPr(D(S))[9] +
         Z[10]*mxGetPr(D(S))[10] + Z[11]*mxGetPr(D(S))[11] +
Z[12]*mxGetPr(D(S))[12] + Z[13]*mxGetPr(D(S))[13] +
Z[14]*mxGetPr(D(S))[14] +
```

```
        Z[15]*mxGetPr(D(S))[15] + Z[16]*mxGetPr(D(S))[16] +
Z[17]*mxGetPr(D(S))[17] + Z[18]*mxGetPr(D(S))[18] +
Z[19]*mxGetPr(D(S))[19] +
        Z[20]*mxGetPr(D(S))[20] + Z[21]*mxGetPr(D(S))[21] +
Z[22]*mxGetPr(D(S))[22] + Z[23]*mxGetPr(D(S))[23] +
Z[24]*mxGetPr(D(S))[24] +
        Z[25]*mxGetPr(D(S))[25] + Z[26]*mxGetPr(D(S))[26] +
Z[27]*mxGetPr(D(S))[27] + Z[28]*mxGetPr(D(S))[28] +
Z[29]*mxGetPr(D(S))[29];

Yk = 110*(1-exp(-Yin))/(1+exp(-Yin));

X[4] = Yk;

}

static void mdlTerminate(SimStruct *S)
{ } /*Keep this function empty since no memory is allocated*/

#ifdef MATLAB_MEX_FILE
/* Is this file being compiled as a MEX-file? */
#include "simulink.c" /*MEX-file interface mechanism*/
#else
#include "cg_sfun.h" /*Code generation registration function*/
#endif
```