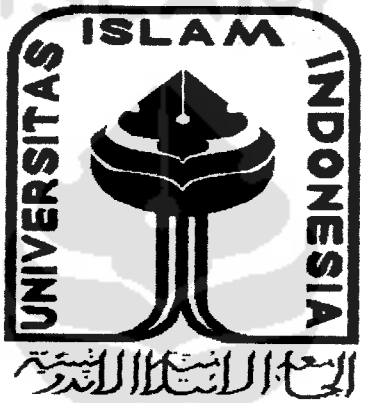
SIMULASI JARINGAN SYARAF TIRUAN DAN INTEGRATOR BERBASIS METODE *BACKPROPAGATION* SEBAGAI PENGENDALI KECEPATAN MOTOR DC BERBEBAN

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat
untuk Memperoleh Gelar Sarjana
Teknik Elektro



Disusun oleh :

Nama : Fitri Ayu Sarie
No. Mahasiswa : 01 524 143

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PENGESAHAN PEMBIMBING

SIMULASI JARINGAN SYARAF TIRUAN DAN INTEGRATOR BERBASIS METODE *BACKPROPAGATION* SEBAGAI PENGENDALI KECEPATAN MOTOR DC BERBEBAN

TUGAS AKHIR

Disusun oleh :

Nama : Fitri Ayu Sarie

No. Mahasiswa : 01 524 143

Jogjakarta, Juni 2007

Pembimbing I

(Wahyudi Budi Pramono, ST)

Pembimbing II

(Dwi Ana Ratna Wati, ST)

LEMBAR PENGESAHAN PENGUJI

SIMULASI JARINGAN SYARAF TIRUAN DAN INTEGRATOR BERBASIS METODE *BACKPROPAGATION* SEBAGAI PENGENDALI KECEPATAN MOTOR DC BERBEBAN

TUGAS AKHIR

oleh :

Nama : Fitri Ayu Sarie

No. Mahasiswa : 01 524 143

Telah dipertahankan di depan Sidang Penguji sebagai Salah Satu Syarat untuk
Memperoleh Gelar Sarjana Teknik Elektro
Fakultas Teknologi Industri Universitas Islam Indonesia
Jogjakarta, Juli 2007

Tim Penguji

Drs. Abdul Halim
Ketua

Wahyudi Budi Pramono, ST
Anggota I

Dwi Ana Ratna Wati, ST
Anggota II

Mengetahui,
Ketua Jurusan Teknik Elektro
Universitas Islam Indonesia

Ir. Yuwono, ST, M.sc

HALAMAN PERSEMBAHAN

Tugas akhir ini kupersembahkan kepada kedua orang tuaku papa dan mama serta kedua adekku Rian dan Arie mari kita bergandeng tangan untuk membahagiakan orang tua dan keluarga besar kita.

Serta semua keluarga besarku dari keluarga mama dan papa yang telah memberikan seluruh yang mereka miliki, dan khusus kupersembahkan untuk kakekku tercinta yang baru saja meninggalkan kami semua, sebelum tugas akhir selesai. Maaf ya kek sebelum kakek pergi cucu pertamamu belum bisa membahagiakan dengan gelar sarjana.

I love U all my Family....

MOTTO

"Dan barang siapa yang bertaqwa kepada Allah niscaya dia akan menjadikan baginya kemudahan dalam urusannya" (Qs.Ath Thalaq : 4)

"...Orang yang rugi adalah mereka yang merugikan diri sendiri dan keluarganya pada hari kiamat..." (Qs.Asy suuraa : 45)

"Sesungguhnya doa adalah ibadah." (HR. Imam empat)

"Hidup yang indah adalah hidup dalam kejujuran." (my word)

"Semakin keras kita berjuang maka semakin dekat kita dengan keberhasilan."(my word)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb.

Puji syukur kehadirat Allah SWT, atas nikmat iman, rahmat, hidayah dan pikiran yang diberikan. Sehingga penulis dapat menyelesaikan tugas akhir dengan judul "**Simulasi jaringan syaraf tiruan berbasis metode *backpropagation* sebagai pengendali kecepatan motor DC Berbeban**". Tidak lupa shalawat serta salam selalu tercurah kepada Nabi Muhammad. SAW beserta keluarga dan para sahabatnya.

Adapun maksud dari penyusunan tugas akhir ini adalah untuk memenuhi kurikulum S-1 Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia. Disamping itu untuk menambah pengetahuan terhadap ilmu yang telah dipelajari di bangku perkuliahan untuk diterapkan ke masyarakat.

Dalam penyusunan ini, penulis banyak mendapat bantuan dari berbagai pihak, sehingga penulis ingin menyampaikan ucapan terima kasih kepada :

1. Kedua orang tuaku papa Riza dan mama Halimah yang senantiasa memberikan dukungan moril, materi dan doa setiap saat.
2. Bpk Fathul Wahid, ST, M.Sc, selaku Dekan Fakultas Teknologi Industri (FTI) Universitas Islam Indonesia (UII)
3. Bpk Tito Yuwono, ST, M.Sc, selaku Kajur Teknik Elektro.
4. Bpk Wahyudi Budi Pramono, ST. selaku dosen pembimbing I, terima kasih atas bimbingan dan waktunya.
5. Ibu Dwi Ana Ratna Wati, ST. selaku dosen pembimbing II dan Ka.Lab pemrograman Matlab atas waktu, kebaikan, kesabaran dan ilmunya.
6. Dosen dan karyawan Fakultas Teknologi Industri UII, Ka.Lab dan laboran jurusan Teknik Elektro atas waktu, tempat dan ilmu yang diberikan.
7. Adikku, Rian dan Arie serta adik-adik sepupuku Siska, Ijek, Ayu, Melda, Putri, Putra, Vivi.

8. Keluarga besarku, Gede,Cik, Mangcik, Ujok, Tante Ana dan keluarga besar Tj. Gelam.
9. Keluarga besarku dalam kenangan kakakku Median Heriansyah (Alm), yaiku Amin Amit (Alm) dan kakekku Abu Hasan (Alm) yang baru saja meninggalkan kami.
10. Mas Him-himQ yang Always bantuin dari awal sampai akhir TA.
11. Sahabat-sahabat terbaikku Wina, Dewi, Amin, Adi 'kondo', Amel, Berna, thanks berat atas dukungannya selama ini.
12. Teman-teman kos di Ngganggrung Indah, mb Sari, Sonya aku kangen suasana 2001 awal tinggal di jogja.
13. Seluruh mahasiswa jurusan Teknik Elektro UII '01 dan seluruh pihak yang tidak dapat di sebutkan satu-persatu, yang telah memberikan dukungan dan doa.
14. Seluruh teman-teman STTKD '04, bu Gati, Mb Panca, aku bahagia bisa mengenal kalian.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat kesalahan dan kekurangan. Oleh karena itu, kritik dan saran yang membangun akan senantiasa penulis terima dengan senang hati. Akhirnya, harapan penulis semoga Tugas Akhir ini dapat bermanfaat bagi kita semua. Amiin…

Wassalamu`alaikum Wr.Wb

Jogjakarta, Juni 2007

Fitri Ayu Sarie

ABSTRAKS

Motor DC dan komputer banyak digunakan dalam kehidupan sehari-sehari, baik di rumah tangga, industri maupun lingkungan pendidikan yang sangat membutuhkan ketelitian dan penggunan yang serba otomatis. Jaringan Syaraf Tiruan dapat digunakan sebagai kendali motor DC yang dapat disimulasikan menggunakan *neural network toolbox* pada *software* Matlab 7.0.

Dengan menggunakan metode *Backpropagation* dan fungsi *Gradient Descent with Momentum* diperoleh struktur jaringan yang terbaik dengan beban 9e-5, terdiri dari 5 sel neuron lapisan *input*, 3 sel neuron lapisan tersembunyi dan 1 sel neuron lapisan *output*. Fungsi aktivasi pada setiap lapisan menggunakan fungsi identitas, dengan *learning rate* 0.1 dan *momentum coefisient* 0.9 menghasilkan *Mean Square Error* (MSE) 0.016789. Rata-rata perubahan kecepatan dengan penambahan beban sebesar 36.27 rpm.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Pada awalnya komputer diciptakan hanya sebagai alat hitung saja, namun dengan berkembangnya teknologi, peran komputer semakin mendominasi sebagai alat hitung. Lebih dari alat hitung, komputer diharapkan dapat digunakan untuk mengerjakan segala sesuatu oleh manusia baik di rumah tangga, industri bahkan di lingkungan pendidikan. Untuk memecahkan masalah dengan komputer, program harus dibuat terlebih dahulu kemudian akan diproses dan diolah.

Motor DC sudah tidak asing lagi karena banyak digunakan dalam kehidupan sehari-hari, baik dalam industri maupun rumah tangga. Motor DC yang beredar di perusahaan sebenarnya masih digerakkan secara manual oleh operator (manusia), sebagian sudah ada yang menggunakan *mikrokontroller*, logika *fuzzy*, jaringan syaraf tiruan maupun kendali-kendali lainnya sebagai pengendali motor DC tersebut. Motor DC yang digunakan di dunia industri akan lebih menghasilkan produk yang lebih bagus dan memiliki kualitas yang tinggi apabila kesalahan dari faktor manusia dapat diperkecil.

Karena penelitian ini melanjutkan tugas akhir dari saudara Romi yang berjudul "Simulasi Jaringan Syaraf Tiruan Metode *Backpropagation* Sebagai Pengendali Motor DC". Oleh karena itu dalam tugas akhir penulis, hanya ditambahkan beban (generator) dan pengendali kecepatan integrator. Sehingga dapat dipaparkan suatu percobaan simulasi sistem kontrol kendali kecepatan

motor DC berbeban dengan jaringan syaraf tiruan dan integrator yang diharapkan dapat memberikan kontribusi dalam dunia industri. Hasil aplikasi dari percobaan simulasi sistem kontrol ini biasanya dipakai pada tangga eskalator, lift, dan beberapa perusahaan yang memproduksi barang menggunakan mesin conveyor.

1.2 Identifikasi Masalah

Karena luasnya permasalahan dalam latar belakang masalah mengenai motor DC dan jaringan syaraf tiruan maka perlunya penelitian lebih lanjut lagi mengenai hal-hal tersebut.

1.3 Batasan Masalah

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

1. Motor DC dimodelkan dengan persamaan matematis.
2. Generator (beban) dimodelkan dengan persamaan matematis.
3. Mencari arsitektur JST terbaik dari beberapa pelatihan.
4. Pengendali kecepatan dengan JST dan integrator.
5. Pembuatan sistem disimulasikan menggunakan perangkat lunak Matlab 7.0.
6. Pelatihan dan pengujian JST menggunakan fungsi yang terdapat dalam Matlab.

1.4 Rumusan Masalah

Berdasarkan batasan masalah yang dijelaskan di atas, dapat ditarik kesimpulan rumusan masalah sebagai berikut:

1. Bagaimana membuat pemodelan motor DC dan beban (generator) pada *neural toolbox* matlab 7.0?
2. Bagaimana membuat sistem simulasi dengan menggunakan perangkat lunak matlab 7.0?
3. Bagaimana mengendalikan kecepatan motor DC berbeban dengan algoritma JST dan integrator metode *backpropagation*?

1.5 Tujuan Penelitian

Penelitian dan perancangan sistem memiliki beberapa tujuan, yaitu :

1. Merancang dan mensimulasikan sebuah sistem penggerak cerdas dengan algoritma jaringan syaraf tiruan dan kendali integrator metode *backpropagation.*
2. Mempelajari dan memanfaatkan *toolbox neural network* dan *simulink* pada Matlab sebagai media pelatihan dan simulasinya.
3. Membuat suatu pelatihan untuk menghilangkan atau memperkecil galat yang terjadi agar sistem JST dapat dikatakan sempurna.

1.6 Metodelogi Penelitian

1. Pengumpulan Data

   Data diperoleh dari studi pustaka berupa buku, artikel, makalah dan tutorial yang tersedia pada *website* di internet.

2. Studi Pustaka

   Pengumpulan data ini digunakan untuk mendapatkan informasi-informasi yang berkaitan dengan proses penyusunan tugas akhir, sehingga dapat digunakan sebagai acuan dalam proses pembuatan simulasi.

3. Pemecahan Masalah

   Setelah semua data terkumpul, maka dilakukan perancangan sistem, pembutan simulasi sistem dan pengujian sistem.

1.7 Sistematika Penulisan

Sistematika penulisan tugas akhir ini terdiri dari 5 bab bagian isi laporan, dengan penjelasan bab sebagai berikut :

BAB I : PENDAHULUAN

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

BAB II : LANDASAN TEORI

Memuat dasar-dasar teori yang berhubungan dengan penelitian dan juga dasar teori yang berhubungan dengan jaringan syaraf dan

pemodelan motor DC yang akan digunakan serta pengendali dengan JST dan *integrator*.

BAB III : PERANCANGAN SISTEM

Menjelaskan tentang pemrograman jaringan syaraf, mensimulasikan rancangan dan pengujian sistem yang telah dibuat, pembagian fungsi kerja dalam diagram blok dan diagram alir serta berisi lebih terperinci tentang apa yang telah disampaikan pada proposal Tugas Akhir.

BAB IV : PENGUJIAN, ANALISIS DAN PEMBAHASAN

Membahas tentang hasil pengujian dan analisis dari sistem yang dibuat dibandingkan dengan dasar teori sistem atau uraian alasan ilmiah yang lain.

BAB V : PENUTUP

Berisi kesimpulan dan saran-saran dari proses perancangan, simulasi sistem, serta keterbatasan-keterbatasan yang ditemukan dan juga asumsi-asumsi yang dibuat selama melakukan penelitian.

BAB II

LANDASAN TEORI

2.1 Sistem Jaringan Syaraf Manusia

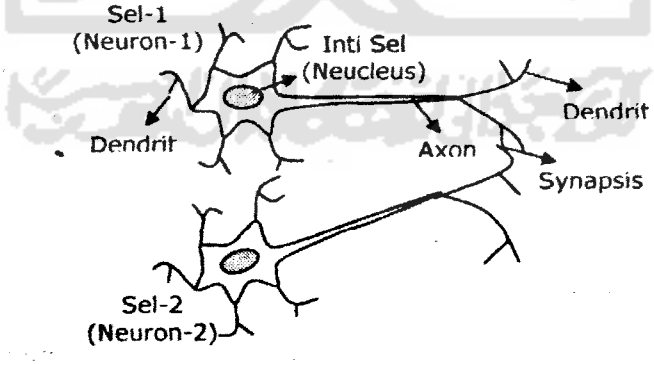
Dalam tubuh manusia dilengkapi dengan dua perangkat pengatur seluruh kegiatan tuduh. Kedua perangkat itu adalah sistem koordinasi yang terdiri dari sistem syaraf dan sistem hormon. Perbedaan antara kedua sistem tersebut adalah sistem syaraf bekerja berasal dari dalam tubuh (rasa lapar, kenyang, nyeri, kelelahan dan lain-lain) maupun yang berasal dari luar tubuh (bau, pahit, manis, sentuhan cahaya, suhu, tekanan, gaya berat dan lain-lain) dan pengaturannya dilakukan oleh benang-benang syaraf. Sedangkan sistem hormon bekerja lebih lambat tetapi lebih teratur dan berurutan dalam jangka waktu yang lama. Pengangkutan hormon dilakukan melalui pembuluh darah. Fungsi jaringan syaraf adalah sebagai berikut :

1. Sensor (reseptor yang mampu memonitor atau memantau dan mendeteksi besar serta kecepatan energi lingkungan).
2. Saluran komunikasi dari sensor ke pusat, yaitu serabut-serabut syaraf yang membawa impuls-impuls syaraf dengan pola *spasiotemporal* yang berfungsi untuk menterjemahkan impuls dari samping otak besar ke pusat.
3. Pusat komputasi dan pembuat keputusan berupa pola geometris kontak-kontak sinapsis antar neurondi pusat susunan syaraf.
4. Saluran komunikasi dari pusat ke efektor, berupa serabut-serabut syaraf yang membawa impuls dari pusat ke tepi.

5. Efektor berupa sel-sel otot maupun kelenjar.

2.1.1 Cara Kerja Sel Syaraf Biologis

Neuron terdiri atas tiga bagian utama, yaitu soma (badan induk neuron), aksom (jalur keluaran dari soma), dan dendrit (menerima impuls/rangsang dari ujung akson lainnya). Diperkirakan ada 10^{14} sinapsis setiap neuron dalam otak manusia. Karena sinapsis merupakan tempat hubungan satu neuron dengan neuron berikutnya, maka pada tempat itu sangat menguntungkan untuk mengatur penghantar sinyal. Beberapa sinapsis menghantarkan sinyal dari satu neuron ke neuron lainnya dengan mudah, sedangkan sinapsis yang lain sulit menghantarkan sinyal. Selain itu beberapa neuron pasca sinapsis bereaksi dengan sejumlah besar impuls sedangkan lainnya hanya bereaksi terhadap beberapa impuls saja. Jadi sinapsis melakukan beberapa tindakan selektif, misalnya dengan menghalangi sinyal lemah tetapi meneruskan sinyal kuat, menyeleksi, menguatkan sinyal lemah tertentu, dan tidak jarang menyalurkan sinyal ke berbagai arah, bukan hanya ke satu arah saja.



Gambar 2.1 Sistem jaringan syaraf manusia

Pada gambar 2.1 dapat diperhatikan bahwa terdapat ratusan sampai ribuan bongkol kecil yang disebut bongkol sinaptik (*sinaptik knob*) yang terdapat pada permukaan dendrit dan *neucleus*, kira-kira 80-90% diantaranya terdapat pada dendrit. Bongkol ini merupakan ujung terminal fibril syaraf yang berasal dari banyak neuron lain dan beberapa bongkol biasanya berasal dari satu neuron. Beberapa bongkol sinaptik ini bersifat eksitatif yang mensekresikan zat (seperti asetilkolin, norepinefrin, dopamin, serotonin) yang merangsang neuron, sedangkan lainnya bersifat inhibit yang mensekresikan suatu zat (seperti asam gamma, aminobutirat, dan glisin) yang menghibisi neuron.

2.2 Sistem Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan digunakan karena jaringan syaraf diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama pembelajaran.

JST dikenal juga sebagai model *free-estimator*, karena dibandingkan dengan cara perhitungan konvensional, JST tidak memerlukan atau menggunakan suatu model metematis dari permasalahan yang dihadapi. Kemudian dikenal juga sebagai *black box technology* (kotak hitam) atau *opaque* (tidak transparan), karena JST tidak dapat menerangkan bagaimana suatu hasil didapatkan. Hal inilah yang membuat JST mampu digunakan untuk menyelesaikan persoalan yang tidak terstruktur dan sulit didefinisikan, dan penerapannya yang telah meluas dipakai

sebagai alat bantu memecahkan masalah pada berbagai bidang dan disiplin ilmu. Beberapa aplikasi jaringan syaraf tiruan :

1. Aplikasi : Memilih suatu input data ke dalam satu kategori tertentu yang ditetapkan
2. Asosiatif : Menggambarkan suatu obyek secara keseluruhan hanya dengan sebuah bagian dari obyek lain.
3. Optimasi : Menemukan suatu jawaban atau solusi yang paling baik sehingga seringkali dengan meminimasikan suatu fungsi biaya (*optimizer*).
4. *Self Organization* : Kemampuan untuk mengolah data input tanpa harus memiliki data sebagai target.

Dengan kelebihan-kelebihan yang dimiliki, jaringan syaraf tiruan tetap mempunyai sejumlah keterbatasan, diantaranya jaringan syaraf kurang mampu dalam melakukan operasi-operasi numerik dengan presisi tinggi dan terkadang membutuhkan waktu berhari-hari untuk jumlah data yang besar karena sulitnya mengukur performansi sebenarnya dari jaringan syaraf tiruan.

2.2.1 Jaringan Syaraf Manusia Sebagai Dasar Jaringan Syaraf Tiruan

Pada dasarnya ada beberapa hal yang mendasari kerja jaringan syaraf manusia ini, diantaranya mengenai penyimpanan informasi dan daya ingat, akson dan dendrit yang bercabang-cabang sedemikian banyaknya, dan proses pengolahan informasi yang terdapat pada jaringan syaraf manusia. Dalam jaringan syaraf manusia, bila suatu sinyal tertentu melalui sinapsis secara berulang-ulang,

maka sinapsis tersebut menjadi lebih mampu manghantarkan sinyal pada kesempatan berikutnya, hal ini biasa dikenal dengan istilah penyimpanan informasi dan daya ingat. Hal ini mendasari adanya proses belajar atau pelatihan *(learning)*, jadi jaringan syaraf tiruan yang akan digunakan pasti melalui proses pelatihan secara berulang-ulang terlebuh dahulu.

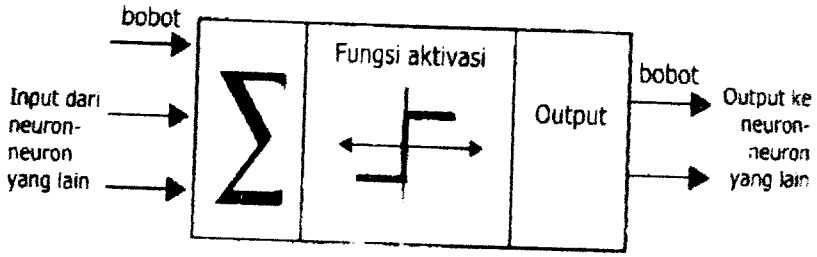
Dalam jaringan syaraf manusia, akson dan dendrit bercabang-cabang sedemikian banyaknya. Hal ini menunjukan bahwa adanya sistem paralel dan terdistribusi pada jaringan syaraf tiruan. Perbedaannya, akson dan dendrit pada jaringan syaraf manusia bercabang-cabang dengan pola yang tidak teratur.

Jaringan Syaraf Tiruan merupakan bagian dari *Artificial Intelligence* atau kecerdasan buatan yang berbasis hubungan, karena cara kerjanya melihat pada jaringan syaraf manusia. Secara garis besar dapat dijelaskan sebagai berikut : beberapa bongkol (baik *eksitasi* maupun *inhibisi*) masuk ke suatu neuron, oleh neuron masukan tersebut dijumlahkan, kemudian dibandingkan dengan nilai ambangnya. Hasil penjumlahan baru bias berarti jika besar kecilnya bobot hubungan telah teratur.

2.2.2 Cara Kerja Komponen Jaringan Syaraf Tiruan

Ada beberapa tipe jaringan syaraf tiruan, tetapi hampir semuanya memiliki komponen yang sama. Sama halnya seperti otak manusia, jaringan syaraf juga terdiri dari beberapa neuron dan ada hubungan antara neuron tersebut. Neuron-neuron tersebut akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke neuron-neuron yang lain. Pada JST hubungan ini

dikenal dengan nama bobot. Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tersebut.

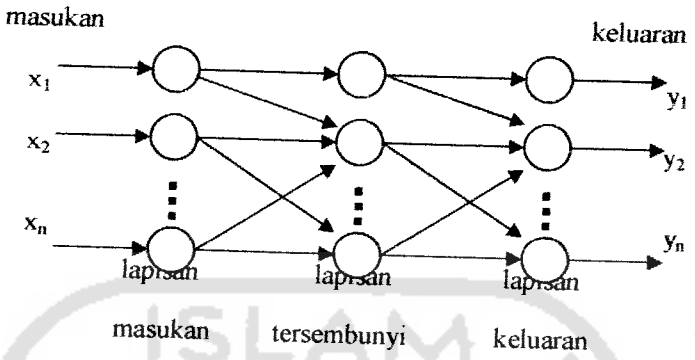


Gambar 2.2 Struktur neuron jaringan syaraf tiruan

Informasi *(input)* akan dikirim ke neuron dengan bobot kedatangan tertentu. *Input* ini akan diproses oleh suatu fungsi perambatan yang akan menjumlahkan nilai-nilai semua bobot yang datang. Hasil penjumlahan ini kemudian akan dibandingkan dengan suatu nilai ambang *(threshold)* tertentu melalui fungsi aktivasi setiap neuron. Apabila *input* tersebut melewati suatu nilai ambang tertentu, maka neuron tersebut akan diaktifkan, tetapi jika tidak maka neuron tidak akan diaktifkan. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan *output* melalui bobot-bobot *output*-nya ke semua neuron yang berhubungan dengannya. Hal ini dilakukan secara terus-menerus.

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan-lapisan *(layer)* yang disebut dengan lapisan neuron (*neuron layer*). Biasanya neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya (kecuali lapisan *input* dan lapisan *output*). Informasi yang diberikan pada jaringan syaraf akan dirambatkan dari lapisan ke lapisan, mulai dari lapisan *input* sampai ke lapisan *output* melalui lapisan yang lainnya, yang sering dikenal dengan nama lapisan tersembunyi (*hidden layer*), tergantung

pada algoritma pembelajarannya, bisa jadi informasi tersebut akan dirambatkan secara mundur pada jaringan.



Gambar 2.3 Jaringan syaraf dengan tiga lapisan

Beberapa jaringan syaraf ada juga yang tidak memiliki lapisan tersembunyi, dan ada juga yang neuron-neuronnya disusun dalam bentuk matriks.

2.2.3 Konsep Belajar Jaringan Syaraf Tiruan

Ciri utama yang dimiliki oleh sistem jaringan syaraf tiruan adalah kemampuan untuk belajar. Agar berfungsi seperti yang diinginkan, jaringan tidak diprogram seperti yang dilakukan pada sistem komputer sekarang ini, melainkan harus diajari.

Berdasarkan fungsi masukan keluarannya, fungsi jaringan syaraf tiruan ditentukan oleh parameternya (bobot-bobot koneksi). Untuk kasus yang diketahui fungsi pemetaannya, bobot-bobot tersebut dapat berharga tetap dan ditentukan pada waktu perancangan. Tetapi pada kebanyakan kasus, parameter jaringan yang

cocok belum diketahui, dan jaringan harus mencari sendiri besarnya bobot tersebut atau secara acak.

Suatu proses penyesuaian parameter secara berurutan dilakukan, dengan tujuan mendekati fungsi yang diinginkan. Proses penyesuaian parameter inilah yang disebut dengan proses belajar dalam sistem jaringan syaraf tiruan. Proses belajar dikategorikan dalam dua jenis :

1. Dengan pengawasan (*supervised learning*)
2. Tanpa pengawasan *(unsupervised learning*)

Proses belajar dengan pengawasan memerlukan keluaran target atau jawaban yang diperlukan dalam proses belajar sebagai dasar penghubung bobot. Jaringan diajar untuk menyelesaikan persoalan-persoalan yang terdapat dalam paket belajarnya.

Selama belajar apabila jaringan mengeluarkan jawaban yang salah, maka besar kesalahan dapat dicari, yaitu beda keluaran aktual dan acuannya. Sedangkan dalam belajar tanpa pengawasan, jaringan akan merubah bobot-bobotnya, sebagai tanggapan terhadap masukan, tanpa memerlukan keluaran acuan.

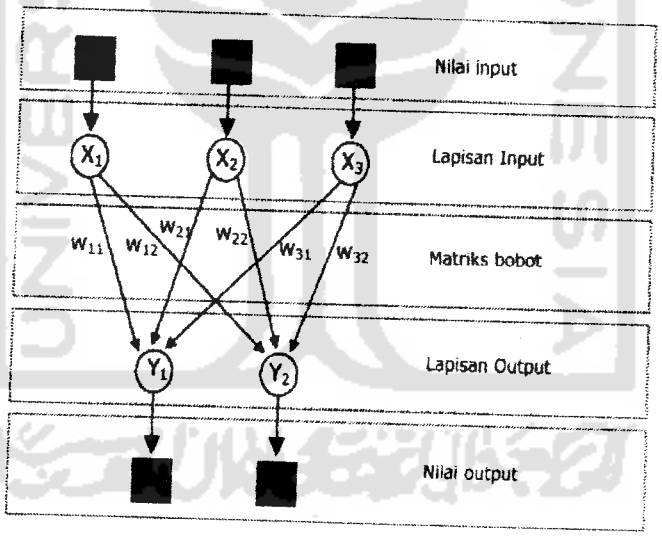
2.2.4 Arsitektur Jaringan Syaraf Tiruan

Neuron-neuron dalam jaringan syaraf dikelompokkan dalam lapisan-lapisan yang umumnya setiap lapisan yang sama akan memiliki keadaan yang sama juga. Apabila neuron-neuron dalam suatu lapisan (misalkan lapisan

tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misalnya lapisan *output*), maka setiap neuron dalam lapisan tersebut (misalkan lapisan tersembunyi) juga akan dihubungkan pada setiap neuron pada lapisan yang lainnya (misalkan lapisan *output*). Ada beberapa arsitektur jaringan syaraf, antara lain :

1. Jaringan dengan lapisan tunggal (*single layer net*)

Hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan hanya menerima masukan kemudian secara langsung mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Suatu unit input akan langsung berhubungan dengan unit *output*.

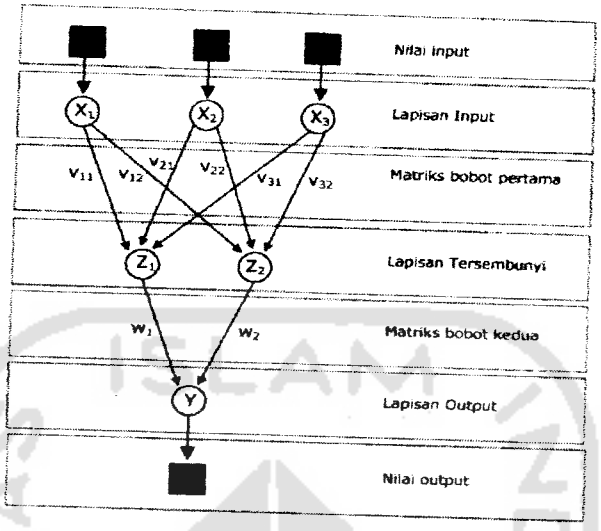


Gambar 2.4 Jaringan syaraf dengan lapisan tunggal

2. Jaringan dengan banyak lapisan (*multilayer net*)

Memiliki satu atau lebih lapisan tersembunyi yang terletak diantara lapisan *input* dan *output*. Dengan pembelajaran yang lebuh rumit, jaringan dapat

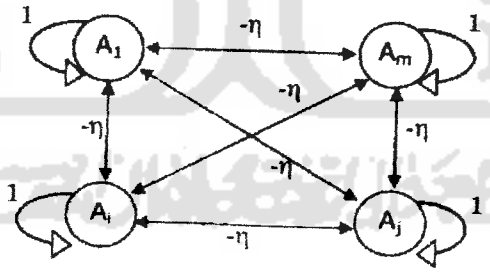
menyelesaikan permasalahan yang lebih sulit dan hasilnya pun lebih sukses dari pada lapisan tunggal.



Gambar 2.5 Jaringan syaraf dengan banyak lapisan

3. Jaringan dengan lapisan kompetitif (*competitive layer net*)

Umumnya hubungan antar neuron tidak diperlihatkan pada diagram arsitektur jaringan dengan lapisan kompetitif. Dari gambar berikut bobot-bobotnys ditunjukan oleh $-\eta$.



Gambar 2.6 Jaringan syaraf dengan lapisan kompetitif

2.2.5 Fungsi Aktifasi

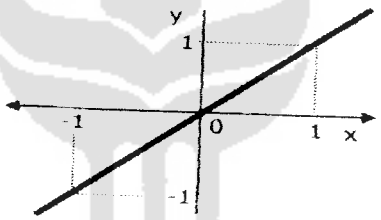
Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Pada setiap lapisan yang sama, neuron-neuron akan

memiliki fungsi aktifasi Yang sama. Suatu fungsi aktivasi untuk setiap sel sebuah jaringan syaraf tiruan *backpropagation* mempunyai beberapa karakteristik penting sebagai berikut :

1. Harus *continue*, dapat diturunkan.
2. Tidak naik atau turun secara monoton

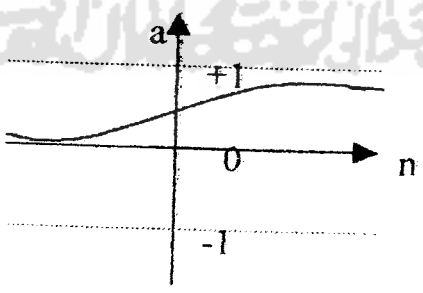
Untuk mendapatkan efisiensi perhitungan, turunannya mudah dihitung. Pada umumnya nilai turunan fungsi aktivasi dapat diperoleh dari nilai rentang tertentu. Ada beberapa fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan, tetapi yang biasa digunakan untuk metode *backpropagation* adalah :

1. *Purelin* atau fungsi linier (identitas), memiliki nilai *output* yang sama dengan nilai *input*



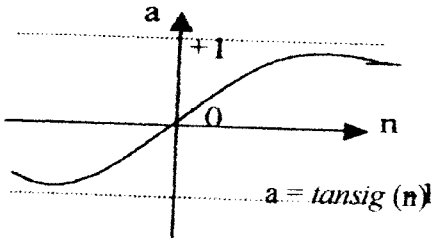
Gambar 2.7 Fungsi aktivasi *linear*/identitas

2. *Logsig* (*sigmoid* biner), memiliki nilai *range* antara 0s/d 1.



Gambar 2.8 Fungsi aktivasi *sigmoid* biner

3. *Tansig* (*sigmoid* bipolar), hampir sama dengan fungsi *sigmoid* biner, hanya saja output dari fungsi memiliki range antara 1 s/d -1



Gambar 2.9 Fungsi aktivasi *sigmoid* bipolar

2.2.6 Metode Backpropagation

Jaringan Syaraf Tiruan rambat balik atau yang biasa di kenal dengan nama *backpropagation* merupakan jaringan syaraf tiruan yang paling banyak diterapkan dalam berbagai bidang, seperti pengenalan pola, diagnosa kedokteran, klasifikasi gambar, menerjemahkan kode, dan bermacam-macam analisa pengenalan pola lainnya. Jaringan ini merupakan salah satu jenis yang mudah dipahami dan konsep belajarnya relatif sederhana, yaitu :

1. Belajar dari kesalahan.
2. Memasukkan secara umpan maju (*feed forward*) pola-pola masukan.
3. Menghitung dan propagasi balik kesalahan yang bersangkutan.
4. Mengatur bobot-bobot koneksi.

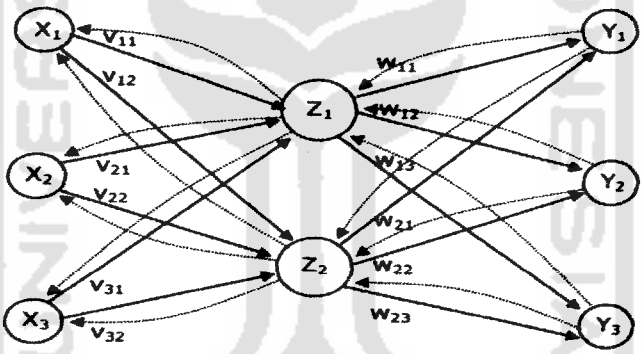
Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyi.

Algoritma *backpropagation* menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan *error*, tahap perambatan maju (*forward*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan fungsi aktivasi.

Setelah pelatihan selesai, koputasi jaringan hanya pada fase umpan maju yaitu memberikan pola-pola masukan. Sekalipun pelatihan lambat, cara kerja jaringan terlatih dapat menghasilkan keluaran sangat cepat.

2.2.6.1 Arsitektur *Backpropagation*

Arsitektur *backpropagation* dengan jaringan lapis banyak dan memiliki satu lapisan dalam (unit-unit Z) ditunjukkan dalam gambar berikut :



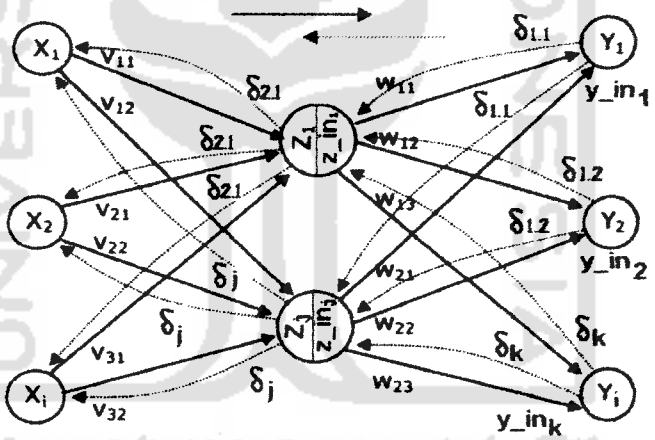
Gambar 2.10 Arsitektur jaringan sederhana

Pada Gambar 2.10 menunjukkan arah sinyal pada fase umpan maju. Selama operasi fase pelatihan perambatan balik, sinyal-sinyal *error* dikirim dengan arah yang sebaliknya. Unit-unit keluaran (y) dan unit-unit lapisan dalam (z) mempunyai bias. Bias pada unit keluaran dinotasikan dengan $w0_k$ bias pada unit lapisan dalam dinotasikan dengan $v0_j$, tindakan bias bekerja seperti pada

bobot-bobot koneksi, dimana unit-unit bias selalu mengeluarkan nilai 1. Unit-unit ini bersifat *optimal* pada penyajian lain mungkin dihilangkan.

2.2.6.2 Algoritma *Backpropagation*

Jaringan syaraf rambat balik (*backpropagation*) dilatih dengan pembelajaran. Jaringan dilatih dengan contoh *input* dan target *output*. Dalam masing-masing presentasi, bobot diubah untuk mengurangi perbedaan antara *output* jaringan dan target *output*. Setelah pelatihan dilakukan pengujian terhadap jaringan yang telah dilatih. Pembelajaran algoritma jaringan membutuhkan perambatan maju dan diikuti dengan perambatan mundur. Keduanya dilakukan untuk semua pola pelatihan.



Gambar 2.11 Arsitektur algoritma *backpropagation*

Selama perambatan maju, tiap masukan (x_i) menerima sebuah sinyal masukan dan mengirimkan sinyal ke tiap-tiap unit lapisan tersembunyi $Z_{1,2,\ldots,p}$. Tiap unit lapisan tersembunyi ini kemudian menghitung aktivasinya (z_in_j) dan mengirimkan sinyalnya ke tiap-tiap unit lapisan keluaran (z_j). Tiap unit keluaran

(y_in_k) menghitung aktivasinya (y_k) untuk membentuk respon pada jaringan terhadap pola masukan yang diberikan.

Selama pelatihan, tiap unit keluaran membandingkan perhitungan aktivasinya (y_k) dengan nilai targetnya (t_k) untuk menentukan kesalahan pola tersebut dengan nilai unit itu. Berdasarkan kesalahan ini, faktor δ_k ($k = 1,...m$) dihitung. δ_k digunakan untuk menyebarkan kesalahan pada unit *output* (y_k) ke semua unit pada lapisan sebelumnya (unit-unit tersembunyi yang dihubungkan ke y_k) dan digunakan nantinya untuk mengubah bobot - bobot antara *output* dan lapisan tersembunyi. Dengan cara yang sama, δ_j ($j = 1,...,p$) dihitung untuk tiap unit tersembunyi (z_j). Untuk (δ_j) tidak perlu menyebarkan kesalahan kembali ke lapisan *input*, tetapi digunakan nantinya untuk mengubah bobot-bobot antara lapisan tersembunyi dan lapisan *input*-nya.

Setelah seluruh faktor δ telah ditentukan, bobot untuk semua lapisan diatur secara serentak. Pengaturan bobot w_{jk} (dari tiap-tiap unit lapisan tersembunyi ke tiap-tiap unit *output*) didasarkan pada faktor δ_k dan aktivasi z_in_j dari unit tersembunyi z_j. Pengaturan bobot v_{ij} (dari vektor unit *input* ke tiap-tiap unit lapisan tersembunyi) didasarkan pada faktor δ_k dan aktivasi unit *input* x_i.

Suatu jangka waktu (*epoch*) adalah satu set putaran vektor-vektor pelatihan. Beberapa *epoch* diperlukan untuk pelatihan sebuah jaringan syaraf tiruan *backpropagation*. Dalam algoritma ini dilakukan perbaikan bobot setelah masing-masing pola pelatihan disajikan. Setelah pelatihan selesai bobot-bobot yang telah diperbaiki disimpan.

2.3 Motor DC

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

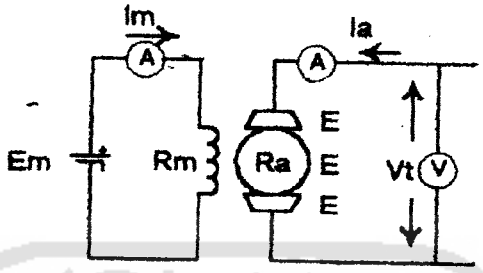
Antara motor arus searah dengan generator arus searah tidak ada perbedaan konstruksi. Pada prinsipnya motor arus searah dapat digunakan sabagai generator arus searah, begitu juga sebaliknya.

2.3.1 Prinsip Motor DC

Meskipun penerapannya sangat luas, semua motor DC bekerja menurut prinsip yang sama. Sebuah motor DC adalah segulung kawat yang dialiri arus listrik dan ditempatkan di dalam suatu medan magnet. Akibatnya, gulungan kawat ini akan mengalami suatu gaya yang sebanding dengan arus dan kekuatan medan magnetnya. Arah gaya membentuk sudut siku-siku terhadap arus dan arah medan magnet. Arah gaya ini akan terbalik jika arus atau arah medan magnetnya dibalik. Jika arah medan magnet dan arus keduanya dinaikkan, maka arah gayanya tidak akan berubah. Sifat ini memungkinkan motor-motor tertentu dapat berputar dengan arus searah (DC) maupun bolak-balik (AC).

2.3.2 Karakteristik Motor DC

Di bawah ini adalah gambar rangkaian ekivalen motor DC shunt dengan eksitasi terpisah beserta persamaan yang berlaku :



Gambar 2.12 Rangkaian ekivalen motor DC

$$E_a = V_t - I_a R_a , \quad (2.1)$$

$$E_a = C\, n\, \phi , \quad (2.2)$$

$$n = {}^{Vt - Ia\,Ra}/_{C\,\phi} , \quad (2.3)$$

keterangan :

E_a = tegangan jangkar (volt)

V_t = tegangan (volt)

I_a = arus jangkar (ampere)

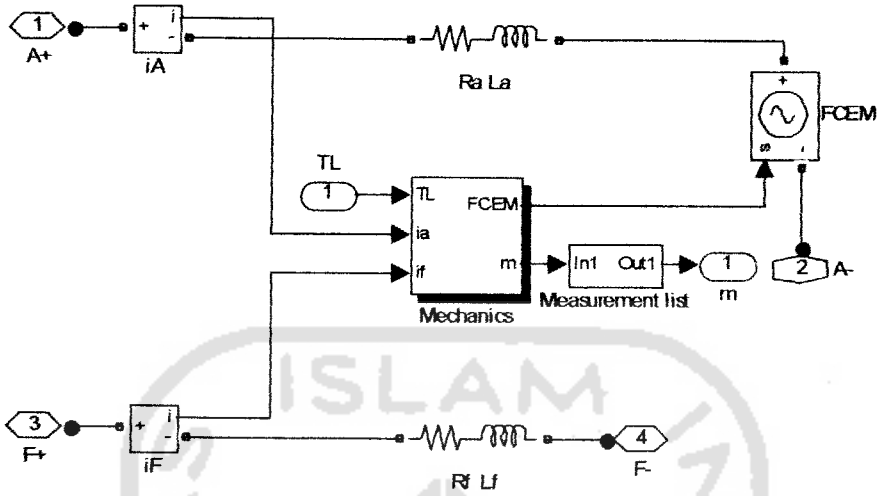
R_a = tahanan jangkar (ohm)

C = konstanta

n = kecepatan putar jangkar

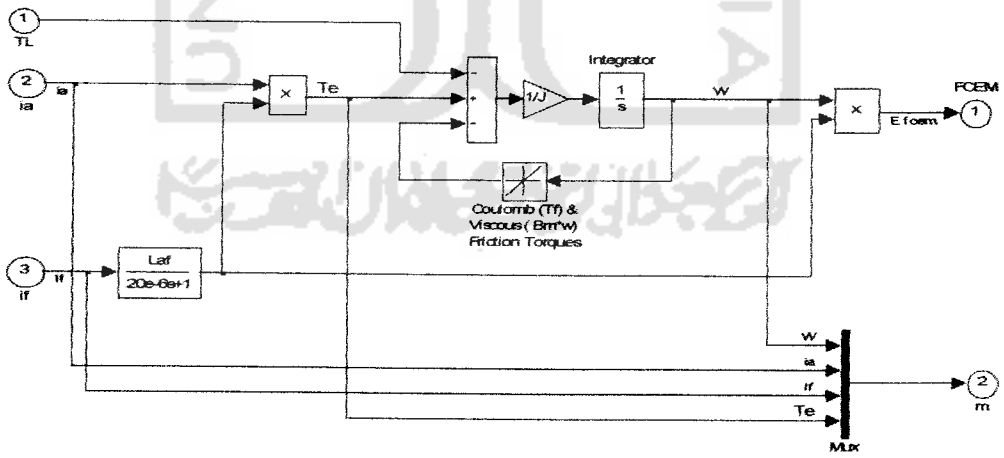
ϕ = fluks medan

Berikut adalah gambar pemodelan motor DC dari simulasi :

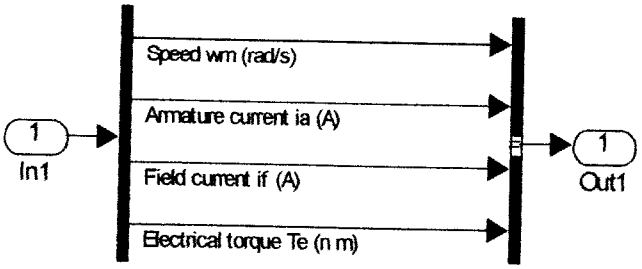


Gambar 2.13 Pemodelan motor DC

Dari gambar pemodelan motor DC pada simulasi terdapat 2 blok subsystem yaitu *mechanics dan measurement list*. Gambar blok-blok yang terdapat dalam 2 blok subsystem ini adalah :



Gambar 2.14 Blok-blok *mechanics* (mesin DC)



Gambar 2.15 Blok-blok *measurement list.*

Keterangan :

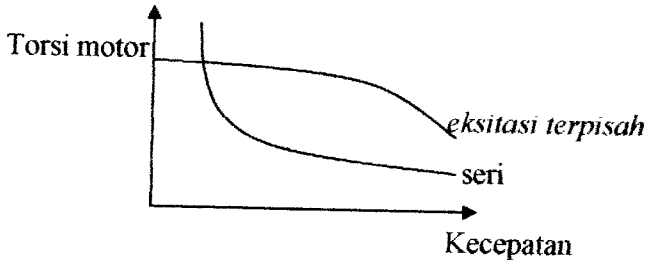
ω m = kecepatan (rpm)

ia = arus jangkar (A)

if = arus medan (A)

Te = torsi elektrik (N.m)

Dari persamaan 2.3 diketahui bahwa pada motor *eksitasi terpisah*, bartambahnya kopel (arus jangkar bertambah) mengakibatkan kecepatan (n) menurun. Pada motor seri, bertambahnya kopel (arus) akan menyebabkan bertambahnya harga fluks (ϕ), karena fluks pada motor seri merupakan fungsi arus jangkar. Untuk harga arus jangkar sama dengan nol, harga fluks juga nol sehingga dari persamaan 2.3, diperoleh harga n menuju tak terhingga. Sedangkan untuk harga I_a yang cukup besar, harga n akan mendekati nol. Dengan demikian karakteristik kecepatan-kopel untuk motor eksitasi terpisah dari seri dapat digambarkan sebagai berikut :



Gambar 2.16 karakteristik kecepatan-kopel motor *eksitasi terpisah* dan seri

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

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

Arus dialirkan ke kumparan rotor melalui dua buah sikat yang sekaligus merupakan kontak dengan cincin penghantar pada rotor (komutator). Komutator dibagi menjadi tiga bagian yang disusun berdekatan. Kedua sikat akan memindahkan arus dari satu kumparan ke kumparan lainnya sesuai dengan putaran motor. Salah satu kekurangan dari motor sederhana ini adalah adanya

perubahan torsi pada setiap putaran motor. Pada kecepatan tinggi, perubahan torsi tidak masalah dan masih ada beban dan kelembaman atau *inertia* yang memperhalus gerakan motor. Pada kecepatan rendah, perubahan torsi membuat putaran motor cenderung meloncat dari satu posisi ke posisi lainnya.

2.3.3 Pengaturan Kecepatan Motor DC

Pengaturan kecepatan memegang peranan penting dalam motor arus searah, karena motor arus searah mempunyai karakteristik kopel-kecepatan yang menguntungkan dibandingkan dengan motor lainnya. Dari persamaan 2.1 sampai 2.3 di atas, dapat dilihat bahwa kecepatan (n) dapat diatur dengan mengubah besaran ϕ, V_t dan R_a.

1. Pengaturan kecepatan dengan mengatur medan *eksitasi terpisah* (ϕ), dengan menyisipkan tahanan variabel yang dipasang seri terhadap kumparan medan (motor *eksitasi terpisah*), maka dapat diatur arus medan dan fluksnya. Rugi panas yang ditimbulkan sangat kecil pengaruhnya. Karena besarnya fluks yang dicapai oleh kumparan medan terbatas, kecepatan yang diaturpun akan terbatas.
2. Pengaturan kecepatan dengan mengatur tegangan (V_t), dikenal dengan metode *Ward Leonard*. Menghasilkan suatu pengaturan kecepatan yang sangat halus dan banyak dipakai untuk lift, mesin bubut dan lain-lain. Satu-sanya kerugian dalam sistem ini adalah biaya untuk penambahan generator dan penggerak awal.

3. Pengaturan kecepatan dengan mengatur tahanan (R_a), dengan menyisipkan tahanan variabel terhadap tahanan jangkar. Cara ini jarang dipakai, karena penambahan tahanan seri terhadap tahanan jangkar menimbulkan rugi panas yang cukup besar.

2.3.4 Aplikasi Jaringan Syaraf Tiruan Pada Motor

Jaringan Syaraf Tiruan (JST) dan integrator metode *backpropagation* dalam tugas akhir ini adalah sebagai kontrol pengendali kecepatan motor DC berbeban. Masukan dari JST adalah set point kecepatan, sehingga JST mengatur tegangan (keluaran JST) sebagai masukan dari motor agar mendapatkan kecepatan motor yang diinginkan. Di dalam JST terdapat pelatihan-pelatihan yang tingkat *error*nya sekecil mungkin agar kecepatannya tetap konstan.

2.4 integrator

Kecepatan yang masuk dari set point (input) yang telah dikurangi dengan kecepatan output maka keluarannya akan masuk ke blok *inregrator*. Sesuai dengan namanya pada blok ini *input* akan diintegralkan terhadap waktu simulasi. Persamaannya :

$$y(t) = \int_{to}^{t} u(t)dt + yo, \qquad (2.4)$$

keterangan :

u = keluaran Fcn

yo = *initial condition* = xi

Hasil dari integral ini merupakan kondisi kecepatan yang dikendalikan agar kecepatan motor akan kembali ke set point(input) walaupun ditambah beban yang bervariasi. Untuk menghindari tegangan yang berlebih maka keluaran dari blok *integrator* akan masuk ke blok *saturation.*

2.5 Saturation

Nilai maksimum tegangan yang terdapat dalam motor DC eksitasi terpisah adalah 150 volt. Oleh karena itu digunakan fungsi dari blok saturation untuk membatasi tegangan sebesar 150 volt, agar saat kecepatan maksimum motor 1750 rpm kecepatan akan tetap.

Prompt	Nilai(volt)
upper limit	150
lower limit	0

Tabel 2.1 Parameter *Saturation*

Dengan adanya blok ini maka tegangan motor yang telah ditentukan tidak akan terlewati.

2.6 Beban

Dalam hal ini beban yang dimaksud adalah sebagai pengurang kecepatan motor agar tujuan dari simulasi tugas akhir ini tercapai. Beban tersebut direalisasikan dari rumus di bawah ini:

$$Tm - T_L - bm\omega m = dm\frac{d\omega m}{dt} \qquad (2.5)$$

Keterangan :

T_m = torsi motor (N.m)

T_L = torsi load (N.m)

b_m = koef gesek (N.m.s)

ω_m = kecepatan (rpm)

$dm\frac{d\omega m}{dt}$ = derivative

atau rumus yang terdapat dalam *help* matlab sebagai berikut:

$$j\frac{dm}{dt} = T_E - \text{sgn}(w)T_L - B_m\omega - T_f \qquad (2.6)$$

Keterangan :

$j\frac{dm}{dt}$ = total inersia (kg.m^2)

T_E = torsi electrik (N,m)

sgn = konstanta

ω = kecepatan (rpm)

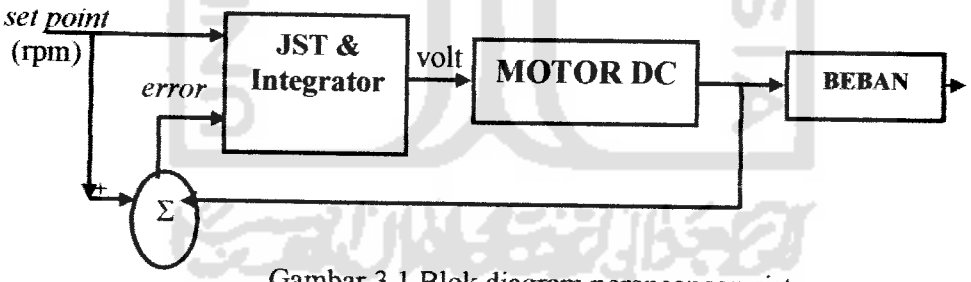
T_L = torsi load (N.m)

T_f = Torsi gesek konstan (N.m)

BAB III

PERANCANGAN SISTEM

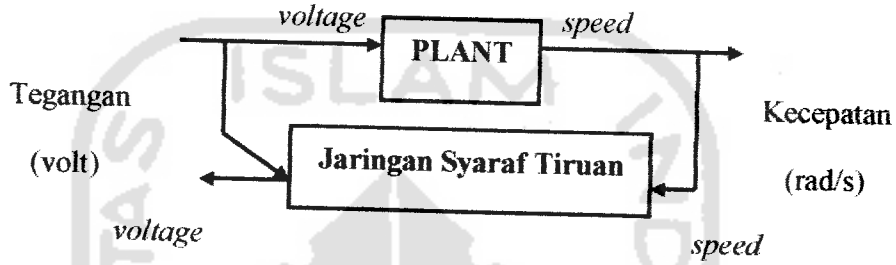
Pada proses belajar Jaringan Syaraf Tiruan (JST) dilakukan secara *on-line/continue* (terus-menerus), sehingga JST memerlukan hasil pengendaliannya (kecepatan yang dihasilkan motor) untuk memperbaiki tanggapan motor. Dalam perancangan sistem, masukan dalam jaringan syaraf adalah berupa kecepatan yang diinginkan, sedangkan keluaran jaringan syaraf yang juga berfungsi sebagai masukan motor adalah tegangan DC, keluaran dari motor DC adalah kecepatan yang menjadi masukan beban. Sebagai keluaran motor dan juga sebagai hasil akhir dari sistem adalah kecepatan motor yang dibebani dapat diamati dari besarnya putaran motor. Untuk mengetahui lebih jelas perancangan sistem kendali ini dapat dilihat pada diagram blok berikut :



Gambar 3.1 Blok diagram perancangan sistem

Pelatihan dari sistem pengendalian dirancang dengan menggunakan metode *inverse*, dimana masukan dari *plant*/model motor adalah sebagai target atau keluaran dari jaringan syaraf, sehingga skenario keluarannya akan digunakan

kembali sebagai masukan. Karena pada pelatihan menggunakan metode *inverse*, maka masukan dan keluaran dari sistem kendali yang sebenarnya akan dibalik pada saat pelatihan. Pada saat pelatihan, masukan dari jaringan syaraf tiruan adalah keluaran dari motor yaitu kecepatan atau putaran dari motor, sedangkan keluaran atau target dari jaringan syaraf adalah merupakan masukan motor yaitu tegangan.

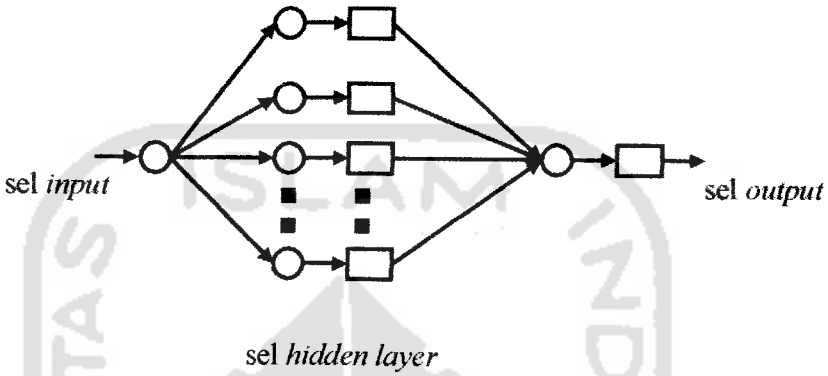


Gambar 3.2 Model *inverse* pelatihan

Kemampuan dari jaringan syaraf tiruan akan dipergunakan untuk mengidentifikasikan kecepatan motor. Selanjutnya hasil proses identifikasi dipergunakan pada proses pengendalian kecepatan motor. Perangkat lunak yang dipergunakan dalam perancangan sistem adalah Matlab versi 7.0, karena perangkat lunak ini memiliki bahasa yang bisa digunakan untuk komputasi teknik dan dapat digunakan untuk perhitungan, visualisasi dan pemrograman. Selain itu perangkat lunak Matlab 7.0 juga memiliki neural network toolbox, sehingga memudahkan dalam perancangan program jaringan syaraf maupun pensimulasian dari sistem yang telah dilatih. Beberapa kegunaan lain dari Matlab diantaranya adalah pengembangan algoritma, pemodelan, simulasi dan pembuatan antarmuka GUI (*Graphical User Interface*).

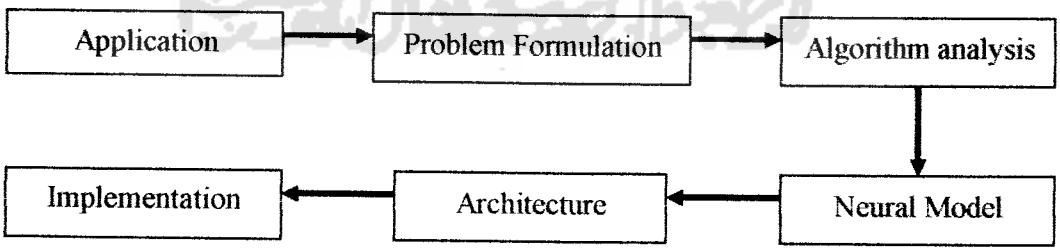
3.1 Perancangan Jaringan Syaraf Tiruan

Jaringan yang digunakan memiliki satu sel neuron pada masukan, satu sel neuron pada lapisan keluaran. Sedangkan untuk jumlah lapisan tersembunyi akan dijadikan sebagai pengamatan.



Gambar 3.3 Arsitektur Jaringan Syaraf Tiruan

Jaringan syaraf tiruan yang digunakan sebagai pengendali kecepatan motor DC ini menggunakan metode *backpropagation*. Metode *backpropagation* termasuk jenis jaringan yang *autoassociative*, yaitu *range* masukan yang diproses ke dalam jaringan sama dengan *range* hasil yang dikeluarkannya. Proses pembangunan jaringan syaraf tiruan secara umum dapat dilihat dari diagram blok berikut ini :



Gambar 3.4 Proses pembangunan jaringan syaraf tiruan

3.1.1 Prosedur Pelatihan

Galat pada jaringan akan dipropagasi-balikan selama pelatihan. Galat pada lapisan keluaran akan menentukan galat pada lapisan tersembunyi (*hidden layer*), yang akan digunakan nantinya untuk pengaturan bobot dan bias pada lapisan tersembunyi tersebut. Proses iterasi akan dilakukan berulang kali sampai galat telah sampai pada suatu *level* toleransinya atau mencapai putaran iterasi (*epoch*) maksimum. Langkah pelatihan untuk jaringan syaraf tiruan yang menggunakan metode *backpropagation* untuk lebih jelasnya dapat dilihat melalui langkah-langkah seperti berikut ini :

Langkah 0 : Inisialisasi bobot (v) dan bias (v0) pada lapisan *input* ke lapisan tersembunyi dan serta bobot (w) dan bias (w0) pada lapisan tersembunyi ke lapisan keluaran,

Langkah 1 : Menentukan jumlah neuron pada lapisan tersembunyi (hl) ketinggian pelatihan (lr/α), maksimum *epoch* (*goal*/me) dan toleransi kesalahan (te), Jika kondisi tidak tercapai, di lakukan langkah 2 – 9,

Langkah 2 : Untuk setiap pasangan pelatihan, di lakukan langkah 3 – 8,

Perambatan maju / *Feedforward* :

Langkah 3 : Tiap unit masukan (x_i ; i = 1, ... , n) menerima sinyal masukan x_i dan menghantarkan sinyal ke semua unit lapisan di atasnya (lapisan tersembunyi),

Langkah 4 : Setiap unit tersembunyi (z_j ; j = 1, ... , p) menjumlahkan bobot sinyal masukan,

$$z\_in_j = \upsilon_{0j} + \sum_{i=1}^{n} x_i \upsilon_{ij} , \qquad (3.1)$$

Digunakan fungsi aktivasi untuk meghasilkan keluaran

$$z_j = f(z\_in_j) , \qquad (3.2)$$

Mengirimkan sinyal ini ke seluruh unit pada lapisan di atasnya (unit keluaran),

Langkah 5 : Tiap unit keluaran (y_k ; k = 1, ... , m) menjumlahkan bobot sinyal masukannya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} , \qquad (3.3)$$

Digunakan fungsi aktivasi untuk menghasilkan keluaran

$$y_k = f(y\_in_k) , \qquad (3.4)$$

Perambatan balik / *Backward* :

Langkah 6 : Tiap unit keluaran (y_k , k = 1, ... , m) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya,

$$\delta_k = (t_k - y_k) f'(y\_in_k) , \qquad (3.5)$$

Menghitung koreksi bobotnya (digunakan untuk memperbaharui w_{jk} nantinya)

$$\Delta w_{jk} = \alpha \delta_k z_j , \quad (3.6)$$

Menghitung koreksi biasnya (digunakan untuk memperbaharui $v0_j$ nantinya)

$$\Delta w_{0k} = \alpha \delta_k , \quad (3.7)$$

Mengirimkan δ_k ke unit-unit pada lapisan dibawahnya.

Langkah 7 : Tiap unit lapisan tersembunyi (z_j ; j = 1, ... , hl) menjumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} , \quad (3.8)$$

Mengalikan dengan turunan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta_j = \delta\_in_j f'(z\_in_j) , \quad (3.9)$$

Menghitung koreksi bobotnya (digunakan untuk memperbaharui v_{ij} nantinya)

$$\Delta v_{ij} = \alpha \delta_j x_i , \quad (3.10)$$

Menghitung koreksi biasnya (digunakan untuk memperbaharui $v0_j$ nantinya)

$$\Delta v_{0j} = \alpha \delta_j , \quad (3.11)$$

Perbaikan bobot dan bias :

Langkah 8 : Tiap unit keluaran (y_k ; k = 1, ... , m) diperbaiki bobot dan biasnya (j = 1, ... ,p)

$$v_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} , \qquad (3.12)$$

$$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k} , \qquad (3.13)$$

pada tiap unit lapisan tersembunyi (z_j ; j = 1, ... , p) diperbaiki bias dan bobotnya (i = 1, ... , n)

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} . \qquad (3.14)$$

$$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j} , \qquad (3.15)$$

Langkah 9 : Tes kondisi berhenti.

3.1.2 Prosedur Pengujian

Bobot dan bias yang telah diperbaiki selama pelatihan, akan digunakan kembali sebagai prosedur pengujian. Pada prosedur pengujian, jaringan syaraf tiruan yang digunakan hanya berupa perambatan maju saja (*feedforward*). Prosedur aplikasinya dapat dilihat pada beberapa langkah berikut :

Langkah 0 : Penentuan bobot awal (hasil dari pelatihan),

Langkah 1 : Untuk setiap vektor masukan , lakukan langkah 2 - 4,

Langkah 2 : Untuk setiap masukan, distribusikan masukan x_i ke setiap unit diatasnya (unit dalam/lapisan tersembunyi),

Langkah 3 : Tiap unit masukan (x_i ; i = 1, ... , n) menerima sinyal masukan x_i dan menghantarkan sinyal ini ke semua unit lapisan diatasnya (lapisan tersembunyi),

Langkah 4 : Setiap unit tersembunyi (z_j ; j = 1, ... , p) menjumlahkan sinyal masukan terbobot,

$$z\_in_j = \upsilon_{0j} + \sum_{i=1}^{n} x_i \upsilon_{ij} \ , \qquad (3.16)$$

Menggunakan fungsi aktivasi untuk menghasilkan keluaran

$$z_j = f(z\_in_j) \ , \qquad (3.17)$$

Mengirimkan sinyal ini ke seluruh unit pada lapisan di atasnya (unit keluaran),

Langkah 5 : Tiap unit keluaran (y_k ; k = 1, ... , m) menjumlahkan bobot sinyal masukannya,

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \ , \qquad (3.18)$$

Menggunakan fungsi aktivasi untuk menghasilkan keluaran

$$y_k = f(y\_in_k) \ , \qquad (3.19)$$

3.1.3 Pemrograman Jaringan dengan *Procedure* dan *Function*

Penentuan notasi atau tata nama yang digunakan dalam pelatihan untuk jaringan syaraf tiruan metode *backpropagation* dalam perancangan sistem adalah sebagai berikut :

x	Vektor masukan ke dalam jaringan.
t	Vektor keluaran target dari jaringan.
hl	Jumlah neuron pada lapisan tersembunyi
me	Maksimum *epoch*
te	Toleransi kesalahan
v	Bobot pada lapisan masukan ke lapisan tersembunyi.
v0	Bias pada lapisan masukan ke lapisan tersembunyi.
w	Bobot pada lapisan lapisan tersembunyi ke lapisan keluaran.
w0	Bias pada lapisan tersembunyi ke lapisan keluaran.
δ_k	Koreksi kesalahan untuk pengaturan bobot dan bias pada unit lapisan keluaran ke lapisan tersembunyi.
δ_j	Koreksi kesalahan untuk pengaturan bobot dan bias pada unit lapisan tersembunyi ke lapisan masukan.
α	*Learning rate* atau laju belajar.
z	Sel pada lapisan tersembunyi.
z_in	Fungsi aktivasi untuk sel lapisan tersembunyi.
y	Sel pada lapisan keluaran.
y_in	Fungsi aktivasi untuk sel lapisan keluaran.
Δ	Perubahan

Σ Penjumlahan

Pada sistem kendali kecepatan motor DC, karena data yang digunakan dalam pelatihan terlalu banyak, maka jaringan syaraf tiruan dibangun dengan jaringan *feedforward* menggunakan fungsi *newff* yang disediakan oleh *toolbox neural network* pada Matlab. Struktur jaringan syaraf tiruan *backpropagation* dengan *newff* adalah sebagai berikut :

```
net = newff (PR,[S1 S2 ...SN1],{TF1 TF2...TFN1},BTF,BLF,PF)
```

PR Matriks berukuran Rx2 yang berisi nilai minimum dan maksimum, dengan R adalah jumlah variabel *input*.

Si Jumlah neuron pada lapisan ke-i, dengan i =1,2,...,N1.

Tfi Fungsi aktivasi pada lapisan ke-i, dengan i =1,2,...,N1, *default* = *tansig* (*sigmoid* bipolar).

BTF Fungsi pelatihan jaringan, *default* = *trainlm* (*Levenberg-Marquardt*).

BLF Fungsi pelatihan untuk bobot, *default* = *learngdm* (*Gradient descent with momentum*).

PF Fungsi kinerja kesalahan atau *error*, *default* = MSE (*Mean Square Error*).

Pelatihan jaringan (BTF) untuk sistem kendali motor DC berbeban menggunakan fungsi *learngdm*. Fungsi ini tidak hanya merespon *gradient* lokal saja, namun juga mempertimbangkan kecenderungan yang baru saja terjadi pada permukaan *error*. Besarnya perubahan bobot ini dipengaruhi oleh suatu konstanta mc (*momentum coefisient*) yang bernilai antara 0 sampai 1. Apabila nilai mc = 0, maka perubahan bobot hanya akan dipengaruhi oleh *gradient*-nya. Namun apabila

nilai mc = 1, maka perubahan bobot akan sama dengan perubahan bobot sebelumnya.

Besarnya perubahan bobot adalah :

Kondisi awal $dW = lr * gW$, (3.20)

Untuk selanjutnya $dW = mc * dW + (1 - mc) * lr * gW$, (3.21)

dW Perubahan bobot

lr *Learning rate*

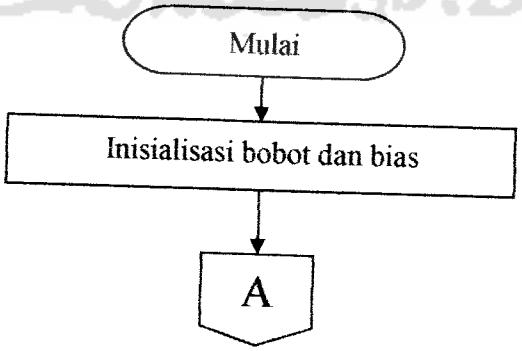
gW *Gradient* kinerja bobot

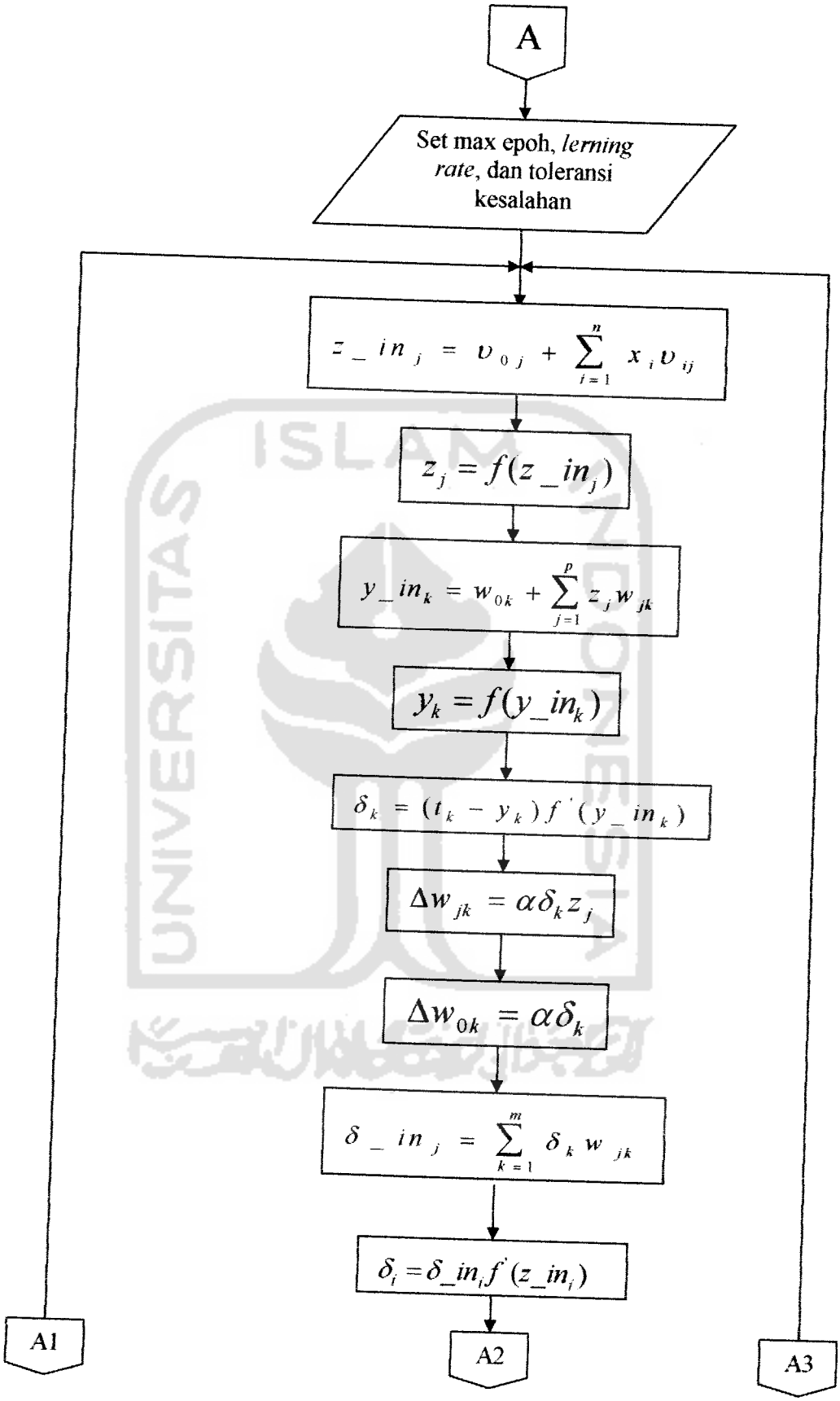
mc *Momentum* (*default* : 1e-10)

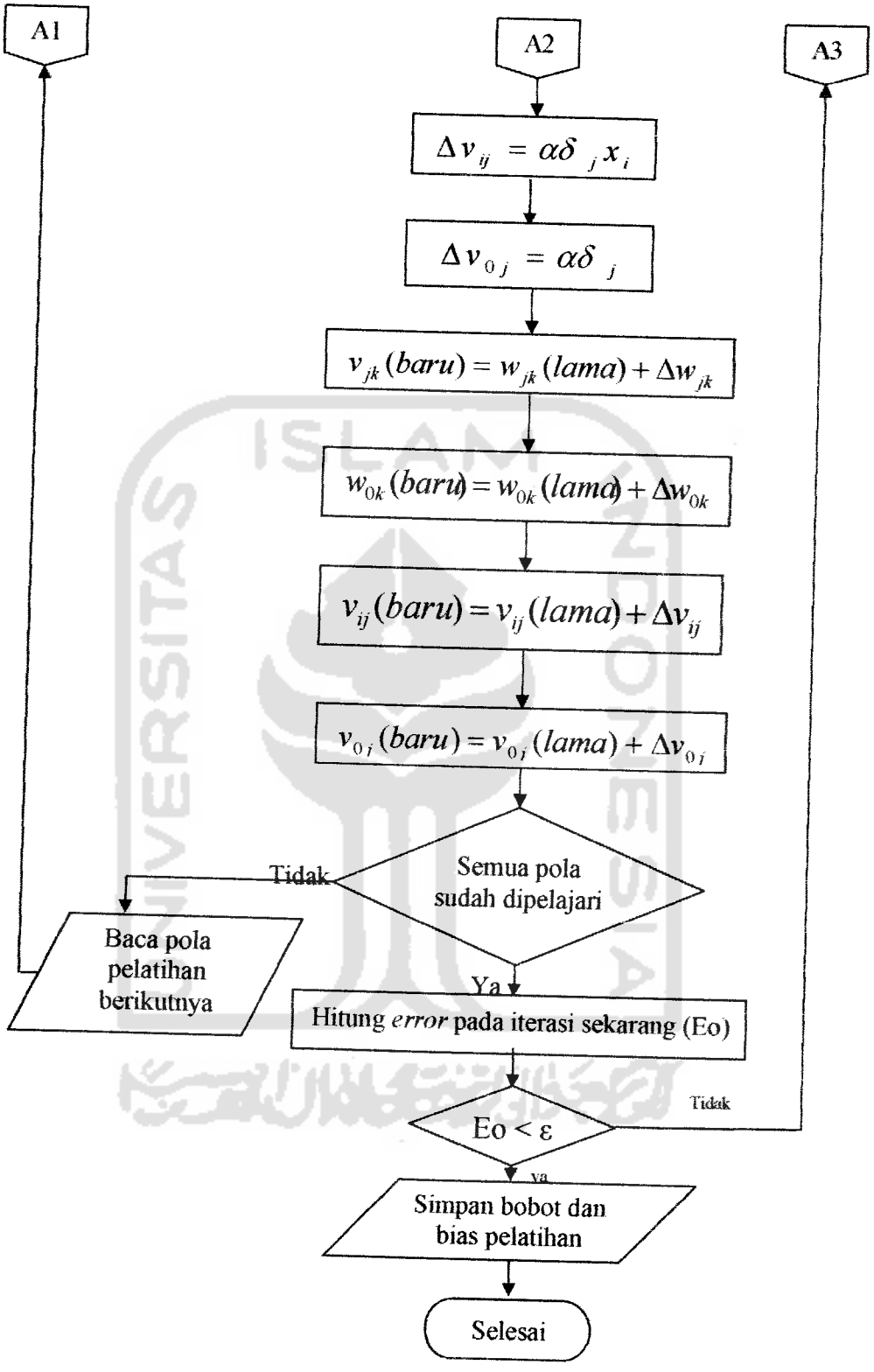
Beberapa parameter yang ditentukan sebelum melatih jaringan antara lain yaitu jumlah iterasi (*epoch*), target *error* (*goal*), *momentum coefisient* (mc), *learning rate* (lr) dan tampilan nilai iterasi (*show*).

3.2 Diagram Alir Jaringan Syaraf Tiruan Metode *Backpropagation*

Dengan mengacu pada langakah-langkah prosedur pelatihan 3.1.1 di atas (langkah 0 sampai dengan langkah 9), maka diagram alir untuk prosedur pelatihannya adalah sebagai berikut :



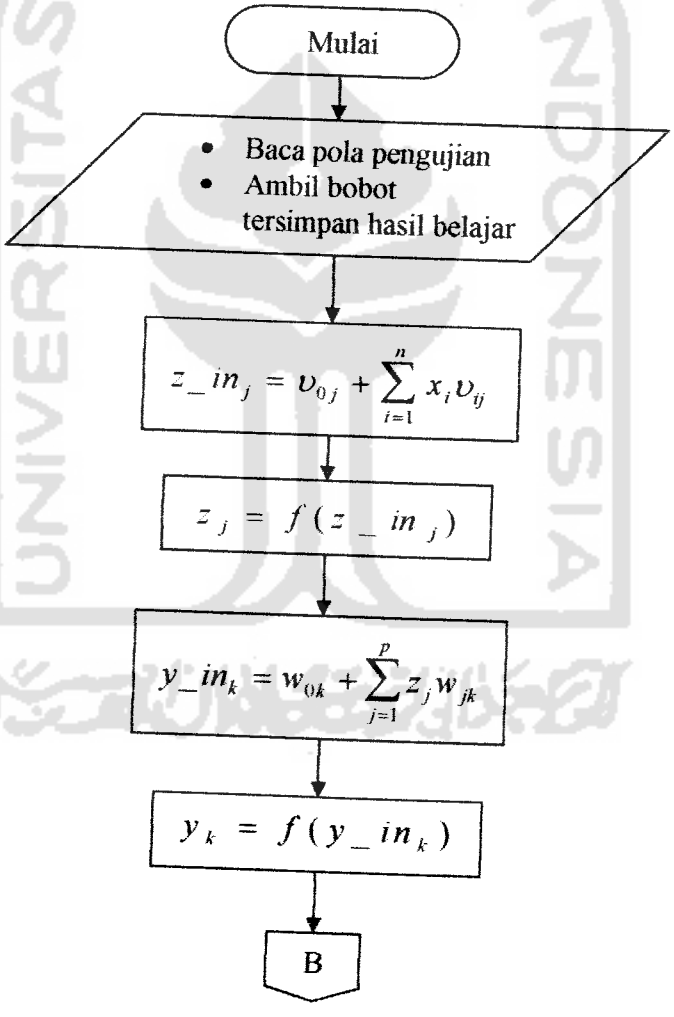

A
Set max epoh, lerning rate, dan toleransi kesalahan
$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$
$z_j = f(z\_in_j)$
$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$
$y_k = f(y\_in_k)$
$\delta_k = (t_k - y_k) f'(y\_in_k)$
$\Delta w_{jk} = \alpha \delta_k z_j$
$\Delta w_{0k} = \alpha \delta_k$
$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$
$\delta_j = \delta\_in_j f'(z\_in_j)$
A1
A2
A3

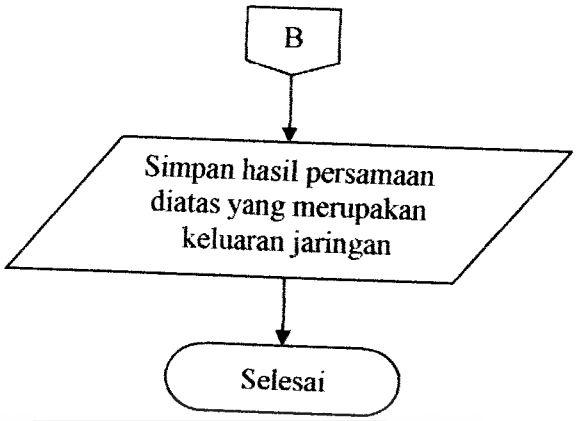


Gambar 3.5 Diagram alir/*flowchart* prosedur pelatihan

Setelah jaringan belajar dan menyimpan bobot-bobot dan biasnya, baru dapat dilakukan pengujian jaringan. Pada pengujian inilah jaringan yang telah dilatih digunakan untuk menyelesaikan masalah. Sebelum jaringan dilatih, maka jaringan belum bisa digunakan untuk menyelesaikan suatu masalah.

Begitu juga dengan mengacu pada langkah-langkah prosedur pengujian 3.1.2 diatas (langkah 0 sampai dengan langkah 5), maka diagram alir prosedur pengujiannya adalah sebagai berikut :





Gambar 3.6 Diagram alir/*flowchart* prosedur pengujian

3.3 Parameter Motor DC dan Beban

Motor yang disimulasikan diambil dari data motor sebenarnya yang ada pada laboratorium instalasi dan mesin listrik dasar. Berikut ini adalah data *board* yang ada pada motor sebenarnya :

Type : GSDT

Exciting shunt

Rpm = 1750

Armature control = ~ Rpm

Field control = ~ Rpm

Bearing : DE 6203ZZ NDE 6204ZZ

Serial number R2A308012

Weight : 19 Kg

Date : 1992

Design : -

Output = 0.5 H

Rating cont : - *Class ins* F

- *Class ins* f

Armature : - *Voltage* = 150 V

- *Current* = 3.2 A

- *Resistance* = 46.875 Ohm

- *Inductance* = 0.01 H

Field : - *Voltage* = 100 V

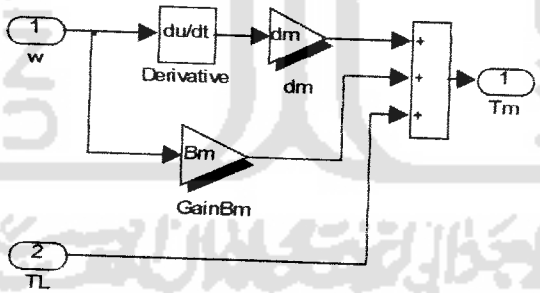
- *Current* = 0.48 A

- *Resistance* = 208.33 Ohm

- *Inductance* = 0,03 H

Pemodelan beban yang dibuat dari rumus :

$Tm - T_L - bm\omega m = dm\frac{d\omega m}{dt}$, adalah :



Gambar 3.7 Pemodelan Beban

BAB IV

ANALISA DAN PEMBAHASAN

Sebelum melakukan pelatihan jaringan syaraf tiruan, terlebih dahulu dilakukan perbandingan terhadap masukan yang berupa tegangan dan keluaran yang berupa kecepatan dari motor sebenarnya. Berikut ini adalah tabel perbandingan yang didapat dari keadaan motor sebenarnya dengan tegangan medan tetap sebesar 100 volt.

Tabel 4.1 Perbandingan kecepatan dan tegangan pada motor sebenarnya tanpa beban

Tegangan jangkar (volt)	**Kecepatan (rpm)**
150	1913.3
125	1657
100	1311.9
75	980.2
50	664
25	311.8

Dari tabel diatas, dapat diketahui bahwa pada saat tegangan jangkar 150 volt kecepatan maksimum dapat mencapai 1913.3 rpm, tetapi pada *data board* yang terdapat pada motor sebenarnya, kecepatan maksimum saat tegangan jangkar 150 volt adalah 1750 rpm. Hal seperti ini banyak terjadi pada keadaan motor sebenarnya, yang disebabkan karena usia motor yang cukup lama dan penggunaan yang sering dilakukan, sehingga menyebabkan perubahan pada beberapa piranti pada motor yang sudah tidak sesuai lagi dengan standarisasi dari pabrik seperti

pada saat awal motor diproduksi. Data yang terdapat pada motor akan sesuai jika usia dan penggunaan motor masih dalam usia dan penggunaan yang wajar.

Dengan mengunakan data yang sama pada motor sebenarnya, data *input* dan *output* dari hasil simulasi disimpan kedalam *workspace* Matlab untuk dijadikan sebagai masukan dan target pada pelatihan jaringan syaraf tiruan sebagai pengendali motor DC. Pelatihan dengan menggunakan *for-while loops* kurang mendapatkan hasil yang lebih maksimum, disebabkan karena data *input* dan target jaringan syaraf terlalu banyak, kurang lebih sebanyak 150676 data *input* dan dengan jumlah yang sama untuk data targetnya. Sebagai perbandingan, untuk melakukan 1000 iterasi dengan 1 lapisan tersembunyi dan 7 sel neuron pada jaringan syaraf yang menggunakan *for-while loops*, membutuhkan waktu kurang lebih selama 18 jam. Berbeda dengan pelatihan yang menggunakan fungsi *newff* yang disediakan oleh Matlab. Dengan menggunakan struktur jaringan syaraf, data *input* dan target yang sama seperti diatas, 1000 iterasi dapat dilakukan hanya dalam hitungan menit. Pelatihan dan pengujian jaringan syaraf menggunakan Matlab akan lebih cepat jika semua data *input*, *output* dan bobot-bias dijadikan kedalam bentuk perhitungan matrik seperti yang terdapat pada fungsi *newff*.

4.1 Pelatihan Jaringan Saraf Tiruan *Backpropagation*

Agar jaringan syaraf tiruan dapat menghasilkan pelatihan yang lebih cepat, maka perlu dilakukan normalisasi pada nilai-nilai data *input* dan pada nilai-nilai setiap bobot dan bias awal pelatihan. Pada data *input* normalisasi dilakukan sedemikian rupa sehingga *range input* dan *output* bernilai antara -0,5 sampai 0,5

pada data *input* pelatihan (kecepatan), normalisasi dilakukan dengan cara membagi data kecepatan dengan nilai maksimum dari data kecepatan itu sendiri. Sedangkan pada bobot dan bias awal pelatihan, fungsi *newff* pada Matlab nilai bobot dan bias telah di normalisasi dengan metode *nguyen-widrow*. Dibawah ini adalah persamaan untuk normalisasi bobot awal pelatihan dengan metode *nguyen-widrow*

$$\beta = 0.7(p)^{1/n} = 0.7\sqrt[n]{p} \tag{4.1}$$

$$v_{ij} = \frac{\beta v_{ij}(old)}{\left\| v_{ij}(old) \right\|} \tag{4.2}$$

$$v_{oj} = -\beta \ and \ \beta \tag{4.3}$$

Keterangan :

β : Faktor skala

n : Banyaknya sel masukan

p : Banyaknya sel lapisan tersembunyi

v_{ij} : Bobot yang sudah ternormalisasi

v_{oj} : Bias yang sudah ternormalisasi

$v_{ij}(old)$: Bobot awal, nilai acak antara -0.5 sampai 0.5

Pada pelatihan jaringan saraf tiruan sebagai pengendali motor DC, dilakukan beberapa pengamatan yaitu jumlah fungsi aktivasi, nilai *learning rate*, nilai *momentum*, jumlah lapisan tersembunyi dan jumlah sel neuron pada setiap lapisan tersembunyi yang digunakan.

4.1.1 Pelatihan Menggunakan 1 Lapisan Tersembunyi

Berikut ini adalah data hasil pengamatan dari pelatihan dengan menggunakan 1 lapisan tersembuyi dan dikelompokan kedalam tabel-tabel sesuai fungsi aktivasi yang digunakan.

Penjelasan tabel hasil pelatihan dan tabel hasil pengujian :

No : Nomor.

HL : Jumlah lapisan tersembunyi pada jaringan syaraf (*hidden layer*) .

Neuron : Angka pertama adalah jumlah neuron lapisan masukan, angka berikutmya adalah jumlah neuron lapisan tersembunyi. (angka 1 dibelakang menunjukan jaringan terdiri dari 1 target)

LR : Nilai kecepatan pelatihan (*learning rate*).

MC : Nilai koefisien momentum (*momentum coefisient*).

F. Aktivasi : Fungsi aktivasi pada setiap neuron *hidden layer* dan *output layer*.

Iterasi : Banyaknya iterasi yang dicapai (*epoch*).

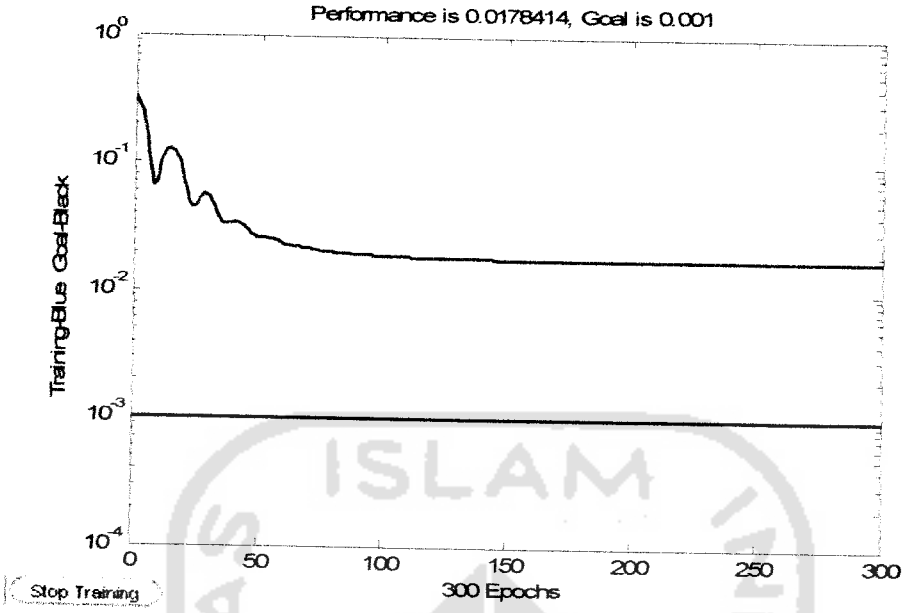
MSE : Nilai rata-rata *error* kuadrat (*mean square error*).

Ket : Keterangan dari berhentinya pelatihan.

Cetak tebal : Hasil pelatihan terbaik dalam satu tabel perbandingan.

Tabel 4.2 Hasil pelatihan dengan fungsi aktivasi *identitas - sigmoid* biner beban 0

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	1	7 – 1	0.3	0.8	*purelin – logsig*	300	0.0179374	*iterasi*
2	1	7 – 1	0.4	0.7	*purelin – logsig*	300	0.017853	*iterasi*
3	**1**	**7 – 1**	**0.5**	**0.9**	***purelin – logsig***	**300**	**0.0178414**	***iterasi***
4	1	7 – 1	0.2	0.6	*purelin – logsig*	300	0.018075	*iterasi*
5	1	7 – 1	0.6	0.9	*purelin – logsig*	300	0.0178446	*iterasi*

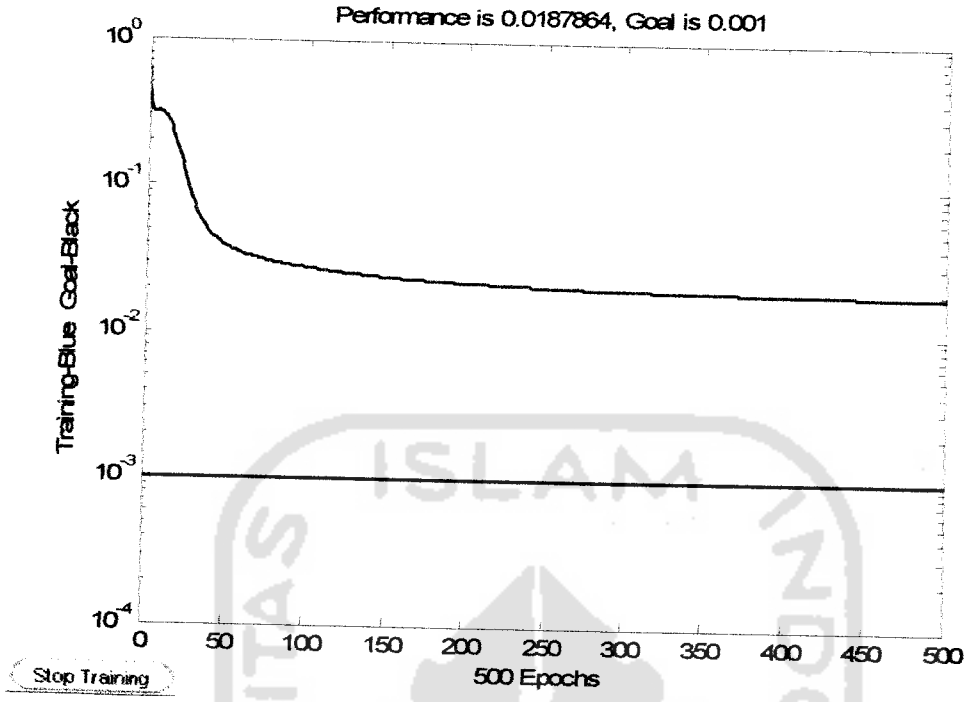


Gambar 4.2 Hasil pelatihan dengan 1 lapisan tersembunyi, 7 sel neuron, *learning rate* 0.5dan *momentum* 0.9

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.3 Hasil pelatihan dengan fungsi aktivasi *sigmoid* biner – *sigmoid* bipolar beban 0.00003 Nm.

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	**1**	**7 – 1**	**0.2**	**0.7**	***logsig-tansig***	**500**	**0.0187864**	***iterasi***
2	1	7 – 1	0.4	0.9	*logsig-tansig*	500	0.389343	*iterasi*
3	1	7 – 1	0.6	0.8	*logsig-tansig*	500	0.019562	*iterasi*

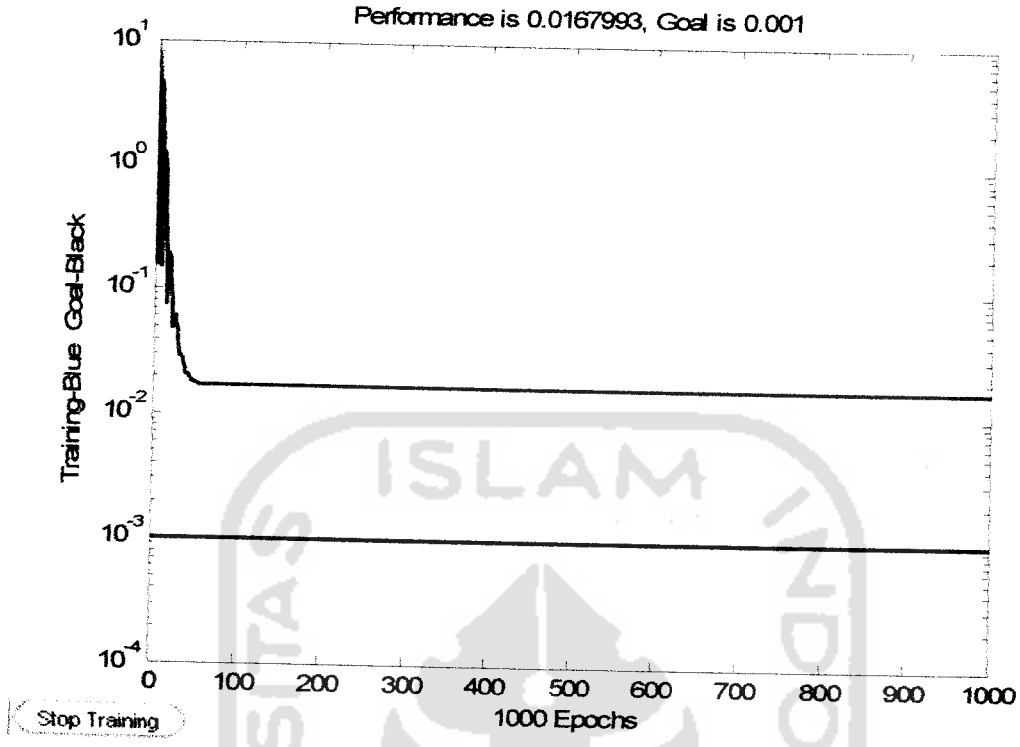


Gambar 4.3 Hasil pelatihan dengan 1 lapisan tersembunyi, 7 sel neuron, *learning rate* 0.2dan *momentum* 0.7

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.4 Hasil pelatihan dengan fungsi aktivasi *sigmoid* biner – *sigmoid* bipolar beban 0.00007 Nm.

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	1	7 - 1	0.1	0.9	*purelin -purelin*	200	0.0168101	*iterasi*
2	1	7 - 1	0.3	0.7	*purelin -purelin*	200	0.0167995	*iterasi*
3	1	7 - 1	0.5	0.7	*purelin -purelin*	1000	0.016813	*iterasi*
4	**1**	**7 - 1**	**0.2**	**0.8**	***purelin -purelin***	**1000**	**0.0167993**	***iterasi***

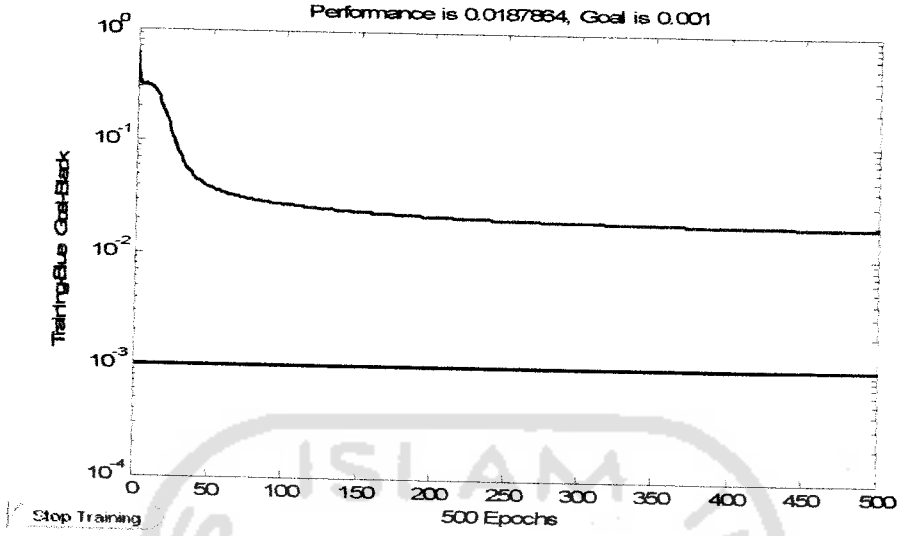


Gambar 4.4 Hasil pelatihan dengan 1 lapisan tersembunyi, 7 sel neuron, learning rate 0.2dan momentum 0.8

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

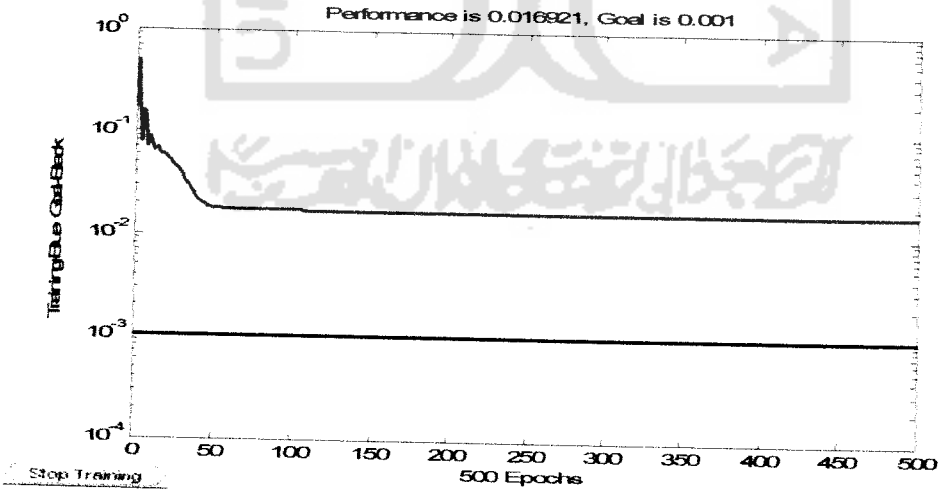
4.1.2 Pelatihan Menggunakan Lebih dari 1 Lapisan Tersembunyi

Berikut ini adalah data hasil pengamatan dari pelatihan dengan menggunakan lebih dari 1 lapisan tersembuyi dan dikelompokan kedalam tabel-tabel sesuai fungsi aktivasi yang digunakan dan jumlah lapisan tersembunyi.



Tabel 4.5 Hasil pelatihan dengan fungsi aktivasi *identitas* – *sigmoid* biner – *identitas* beban 0.0001 Nm.

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	2	4 - 2 - 1	0.1	0.7	*purelin-logsig-purelin*	300	0.0620271	*iterasi*
2	2	4 - 2 - 1	0.4	0.9	*purelin-logsig-purelin*	500	0.0170838	*iterasi*
3	2	4 - 2 - 1	0.5	0.8	*purelin-logsig-purelin*	500	0.0170327	*iterasi*
4	**2**	**4 - 2 - 1**	**0.7**	**0.7**	***purelin-logsig-purelin***	**500**	**0.016921**	***iterasi***

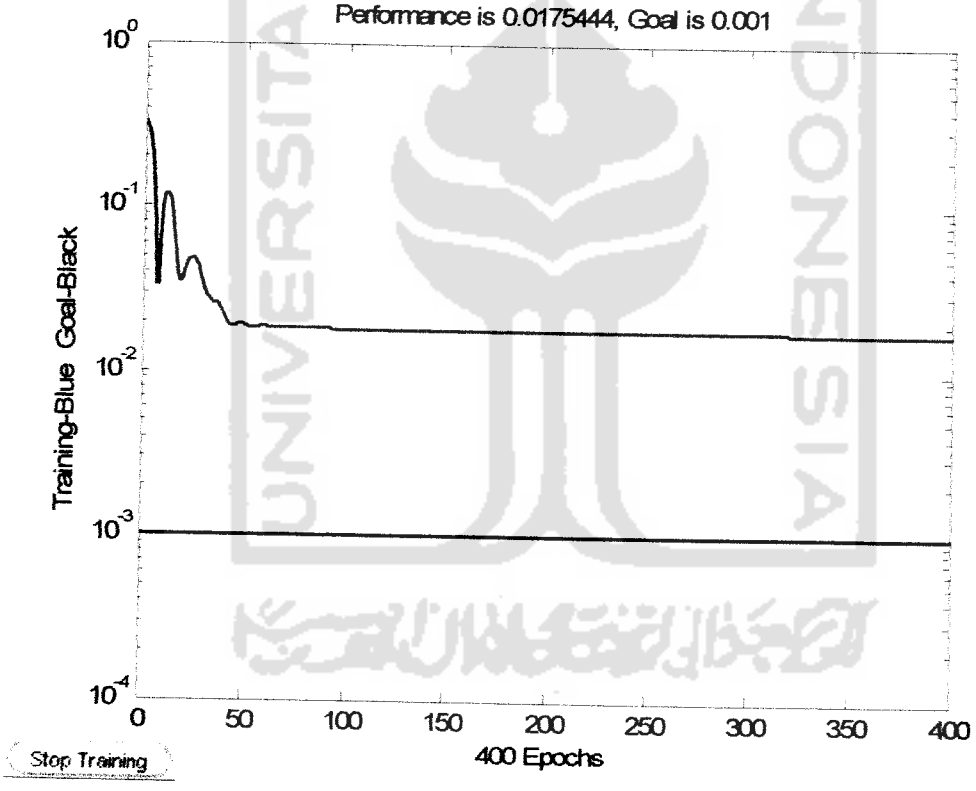


Gambar 4.5 Hasil pelatihan dengan 2 lapisan tersembunyi, 4-2 sel neuron, *learning rate* 0.7dan *momentum* 0.7

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.6 Hasil pelatihan dengan fungsi aktivasi *identitas* – *sigmoid* biner – *sigmoid* biner beban 0.0003 Nm.

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	2	7 - 5 - 1	0.4	0.7	*purelin-logsig-logsig*	500	0.0179935	*iterasi*
2	2	7 - 5 - 1	0.6	0.8	*purelin-logsig-logsig*	400	0.0180511	*iterasi*
3	**2**	**7 - 5 - 1**	**0.7**	**0.9**	***purelin-logsig-logsig***	**400**	**0.0175444**	***iterasi***
4	2	7 - 5 - 1	0.7	0.5	*purelin-logsig-logsig*	400	0.0177779	*iterasi*
5	2	7 - 5 - 1	0.9	0.2	*purelin-logsig-logsig*	400	0.0177046	*iterasi*

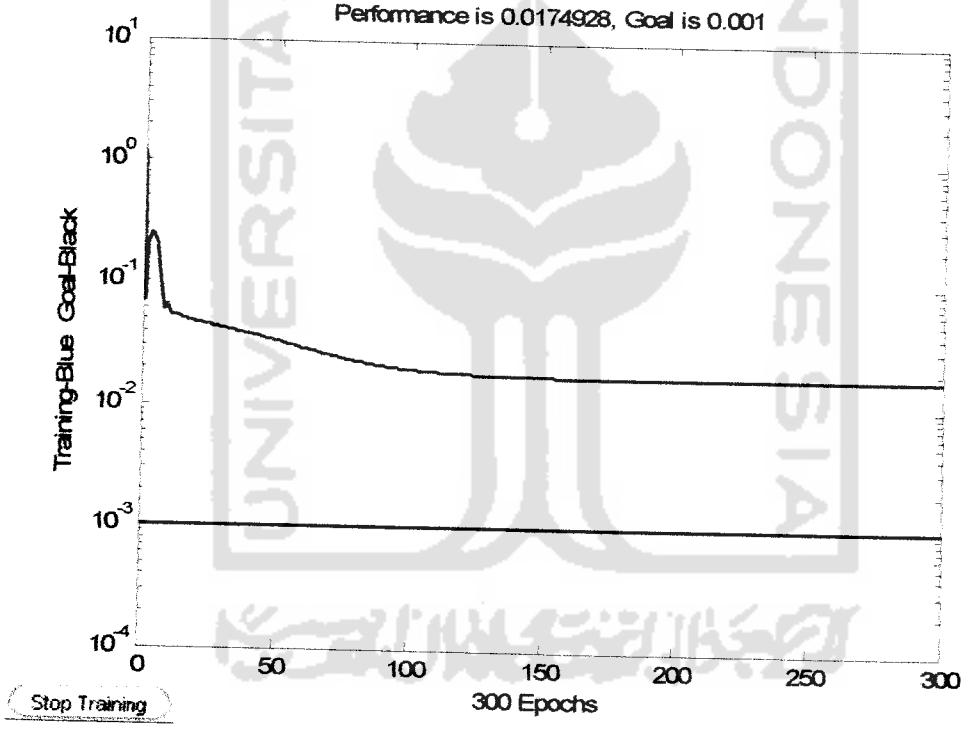


Gambar 4.6 Hasil pelatihan dengan 2 lapisan tersembunyi, 7-5 sel neuron, learning rate 0.7dan momentum 0.9

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.7 Hasil pelatihan dengan fungsi aktivasi *sigmoid* biner - *sigmoid* biner – *sigmoid* bipolar beban 0.0005 Nm.

No.	HL	Neuron	LR	MC	F. aktivasi	Iterasi	MSE	Ket.
1	2	7 - 5 - 1	0.1	0.9	*logsig-logsig-tansig*	400	0.0306853	*iterasi*
2	2	7 - 5 - 1	0.2	0.8	*logsig-logsig-tansig*	400	0.385003	*iterasi*
3	2	7 - 5 - 1	0.6	0.7	*logsig-logsig-tansig*	300	0.0179101	*iterasi*
4	2	7 - 5 - 1	0.8	0.7	*logsig-logsig-tansig*	300	0.0178616	*iterasi*
5	**2**	**7 - 5 - 1**	**0.5**	**0.5**	***logsig-logsig-tansig***	**300**	**0.0174928**	***iterasi***

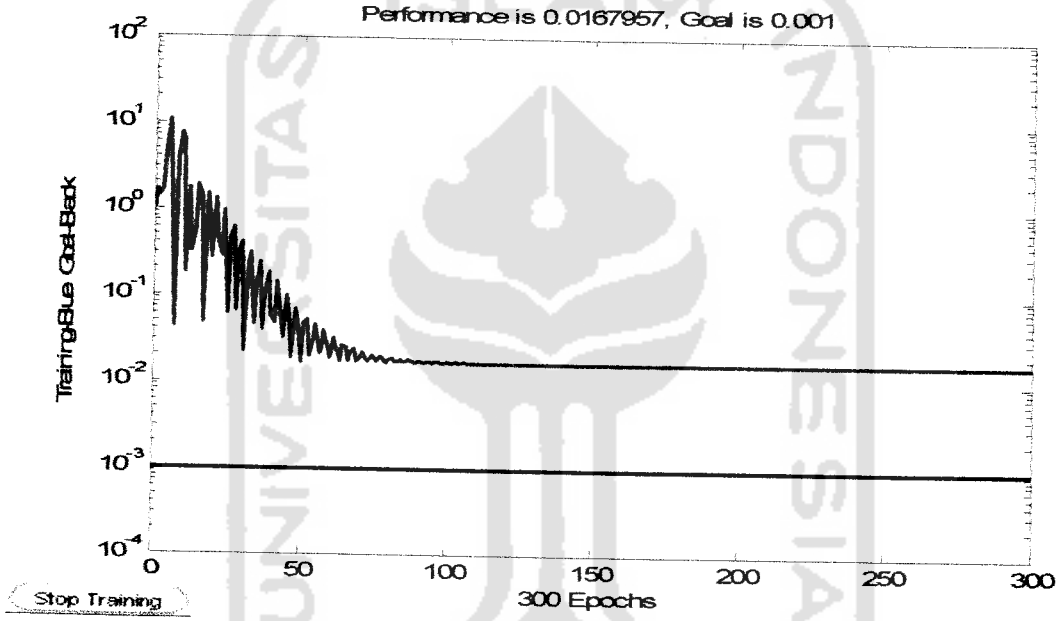


Gambar 4.7 Hasil pelatihan dengan 2 lapisan tersembunyi, 7-5 sel neuron, learning rate 0.5dan momentum 0.5

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.8 Hasil pelatihan dengan fungsi aktivasi *identitas – identitas – identitas* beban 0.00005 Nm.

No	H	Neuron	LR	MC	F. aktivasi	Itera	MSE	Ket.
1	2	5 - 3 - 1	0.2	0.7	*purelin-purelin-purelin*	300	0.016805	*iterasi*
2	**2**	**5 - 3- 1**	**0.3**	**0.9**	***purelin-purelin-purelin***	**300**	**0.0167957**	***iterasi***
3	2	5 - 3 - 1	0.5	0.8	*purelin-purelin-purelin*	300	0.0167989	*iterasi*
4	2	5 -3 - 1	0.6	0.6	*purelin-purelin-purelin*	300	0.0168014	*iterasi*

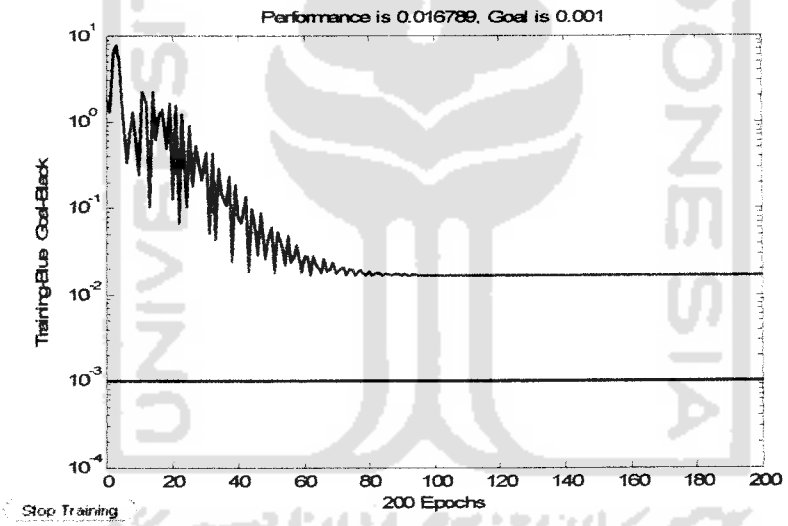


Gambar 4.8 Hasil pelatihan dengan 2 lapisan tersembunyi, 5-3 sel neuron, *learning rate* 0.3dan *momentum* 0.9

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Tabel 4.9 Hasil pelatihan dengan fungsi aktivasi *identitas – identitas – identitas* beban 0.00009 Nm.

N	HL	Neuron	LR	M	F. aktivasi	Iterasi	MSE	Ket.
1	**2**	**5 - 3 - 1**	**0.1**	**0.9**	***purelin-purelin-purelin***	**200**	**0.016789**	***iterasi***
2	2	5 - 3- 1	0.3	0.7	*purelin-purelin-purelin*	200	0.0168053	*iterasi*
3	2	5 - 3 - 1	0.5	0.8	*purelin-purelin-purelin*	200	0.0167982	*iterasi*
4	2	5 -3 - 1	0.4	0.6	*purelin-purelin-purelin*	200	0.0167993	*iterasi*
5	2	5 -3 - 1	0.2	0.4	*purelin-purelin-purelin*	200	0.0168169	*iterasi*



Gambar 4.9 Hasil pelatihan dengan 2 lapisan tersembunyi, 5-3 sel neuron, *learning rate* 0.1dan *momentum* 0.9

Target MSE tidak tercapai, pelatihan berhenti karena target iterasi sudah tercapai.

Berikut ini adalah hasil pengamatan dari pelatihan dengan menggunakan berbagai lapisan tersembuyi dan dikelompokan kedalam tabel-tabel sesuai pelatihan terbaik dari tiap-tiap tabel diatas .

Tabel 4.10 Hasil pelatihan terbaik dari masing – masing pengelompokan

No.	HL	Neuron	LR	MC	F. aktivasi	Beban	Iterasi	MSE	Ket.
1	1	7 – 1	0.5	0.9	*purelin-logsig*	0	300	0.0178414	iterasi
2	1	7 – 1	0.2	0.7	*logsig-tansig*	0.00003	500	0.0187864	iterasi
3	1	7 – 1	0.2	0.8	*purelin-purelin*	0.00007	1000	0.0167993	iterasi
4	2	4 - 2- 1	0.7	0.7	*purelin-logsig-purelin*	0.0001	500	0.016921	iterasi
5	2	7- 5 - 1	0.7	0.9	*purelin-logsig-logsig*	0.0003	400	0.0175444	iterasi
6	2	7 - 5 - 1	0.5	0.5	*logsig-logsig-tansig*	0.0005	300	0.0174928	iterasi
7	2	5 - 3 - 1	0.3	0.9	*purelin-purelin-purelin*	0.00005	300	0.0167957	iterasi
8	**2**	**5 - 3 - 1**	**0.1**	**0.9**	***purelin-purelin-purelin***	**0.00009**	**200**	**0.016789**	**iterasi**

Dari tabel 4.10 diatas, hasil pelatihan terbaik adalah adalah pelatihan pada tabel nomor 8. Pelatihan tersebut berhenti ketika iterasi yang telah ditentukan sudah tercapai dengan MSE 0.016789. Sehingga untuk pengujian jaringan akan digunakan struktur yang sama dengan hasil pelatihan pada tabel nomor 8 tersebut.

Pada tabel semua pelatihan tidak ada pelatihan yang berhenti karena *gradient* sudah mencapai target, jika terdapat pelatihan yang berhenti karena

gradient sudah tercapai artinya MSE yang dihasilkan sudah mencapai nilai yang paling minimum untuk arsitektur jaringan syaraf sebagai pengendali level air pada tangki dengan 1 lapisan tersembunyi. Nilai *gradient* menggunakan nilai *default* yang di tentukan fungsi *newff* yaitu 1e-10. Nilai *gradient* yang dihasilkan dan ditampilkan akan selalu dipengaruhi oleh perubahan dari permukaan MSE. Penentuan nilai *momentum* akan berpengaruh langsung kepada perubahan bobot sesuai dengan persamaan 3.20 dan persamaan 3.21.

Jika diamati dari hasil pelatihan diatas maka akan menemukan pelatihan dengan fungsi aktivasi *sigmoid* bipolar (*tansig*) menghasilkan nilai MSE yang lebih kecil dibandingkan dengan menggunakan fungsi aktifasi yang lain. Fungsi aktivasi *sigmoid* bipolar memiliki nilai *range output* antara 1 sampai -1, dengan range *output* seperti itu maka nilai keluarannya dapat digunakan sebagai pengendali dua katup yaitu katup aliran masuk dan katup aliran keluar. Saat *output* bernilai positif maka akan berfungsi sebagai pengendali katup aliran masuk, dan saat bernilai negatif sebagai pengendali katup aliran air keluar dari tangki. Nilai positif akan didapat saat *set point* bernilai lebih besar dari pada kondisi di *plant*, dan sebaliknya nilai negatif akan didapat saat *set point* bernilai lebih kecil dari kondisi di *plant*. Oleh karena itu sangat diutamakan saat pelatihan menggunakan fungsi aktivasi *sigmoid* bipolar, dan dari hasil pelatihanya terlihat bahwa dengan menggunakan fungsi aktivasi *sigmoid* bipolar akan menghasilkan nilai MSE yang lebih kecil. Jumlah lapisan dan sel neuron pada masing-masing lapisan tersembunyi berpengaruh besar terhadap nilai MSE yang dihasilkan, dan

juga sangat dipengaruhi oleh *learning rate* dan *momentum coefisient* yang digunakan sesuai dengan arsitektur jaringan syaraf

Nilai *learning rate* dan *momentum coefisient* akan berpengaruh terhadap perubahan MSE pada setiap iterasi. Semakin besar nilai *learning rate*, akan semakin cepat pelatihan mendekati nilai *error* minimum, tetapi menghasilkan perubahan MSE yang tidak stabil. Jika nilai *learning rate* digunakan terlalu kecil, maka akan menyebabkan pelatihan lebih lama mendekati nilai error minimum yang ditentukan dan iterasi semakin besar. Berbeda dengan nilai *momentum coefisient*, semakin kecil nilai *momentum* yang digunakan maka semakin banyak iterasi yang dibutuhkan untuk mencapai nilai *error* minimum. Sehingga nilai yang digunakan untuk pelatihan tidak terlalu besar dan tidak terlalu kecil, sesuai dengan variasi nilai antara *learning rate* dan *momentum*.

Penentuan jumlah target iterasi (*epoch*) dilihat dari struktur pelatihan jaringan. Jika jaringan memiliki lapisan tersembunyi dan jumlah neuron yang banyak, maka target iterasi di set tidak terlalu besar agar pelatihan tidak terlalu menggunakan memori pada PC (*personal computer*) terlalu banyak. Semakin banyak jumlah lapisan dan jumlah sel neuron pada masing-masing lapisan, komputasi-pun akan semakin banyak. Semakin banyak komputasi, semakin besar memori PC akan digunakan dan akan semakin lama waktu yang ditempuh untuk mencapai *error* minimum, tetapi menghasilkan perubahan MSE yang tidak stabil. Jika nilai *learning rate* digunakan terlalu kecil, maka akan menyebabkan pelatihan lebih lama mendekati nilai error minimum yang ditentukan dan iterasi semakin besar. Berbeda dengan nilai *momentum coefisient*, semakin kecil nilai

momentum yang digunakan maka semakin banyak iterasi yang dibutuhkan untuk mencapai nilai *error* minimum. Sehingga nilai yang digunakan untuk pelatihan tidak terlalu besar dan tidak terlalu kecil, sesuai dengan variasi nilai antara *learning rate* dan *momentum*.

Penentuan jumlah target iterasi dilihat dari struktur pelatihan jaringan. Jika jaringan memiliki lapisan tersembunyi dan jumlah neuron yang banyak, maka target iterasi di set tidak terlalu besar agar pelatihan tidak terlalu menggunakan memori pada PC (*personal computer*) terlalu banyak. Semakin banyak jumlah lapisan dan jumlah sel neuron pada masing-masing lapisan, komputasi-pun akan semakin banyak. Semakin banyak komputasi, semakin besar memori PC akan digunakan dan akan semakin lama waktu yang ditempuh untuk mencapai *error* minimum.

Dari tabel pelatihan 4.10, maka struktur jaringan yang akan digunakan adalah strukur jaringan pada tabel nomor 8. Dimana struktur jaringan terdiri dari 2 sel neuron *input*. *Input* pertama adalah *set point*, sel neuron kedua adalah perubahan dari kecepatan yang dihasilkan motor. Lapisan *input* (v) terdiri dari 5 sel neuron, sedangkan lapisan tersembunyi terdiri dari 2 lapisan. Lapisan tersembunyi pertama (w) terdiri dari 3 sel neuron dan sesuai dengan target sistem jaringan syaraf, maka lapisan tersembunyi kedua atau lapisan output (w_out) terdiri dari 1 sel neuron. Fungsi aktivasi yang digunakan pada setiap lapisannya adalah fungsi identitas.

4.2 Pengujian Jaringan Syaraf Tiruan *Backpropagation*

Setelah mendapat struktur jaringan terbaik maka hal pertama yang akan dilakukan adalah menguji jaringan tersebut dengan tujuan untuk mengetahui apakah jaringan yang telah didapatkan dengan target *error* yang telah ditentukan sudah dapat menyamai target yang telah ditentukan juga. Pengujian menggunakan *input* dan target yang sama seperti saat melakukan pelatihan mencari jaringan.

Pada simulasi menggunakan *Graphical User Interface* (GUI), *set point* kecepatan di masukan pada kotak *edit set point* atau dengan cara mengatur masukannya melalui *slider* dibawah kotak *edit set point*. Untuk menjalankan simulasi dapat dilakukan dengan cara menekan *button START*. Hasil dari pengujian akan ditampilkan pada kotak *edit motor input*. Gambar dari tampilan GUI dapat dilihat pada gambar dibawah ini :

Gambar 4.11 Tampilan pengujian menggunakan GUI

Set point pada GUI adalah masukan pada jaringan syaraf yaitu berupa kecepatan motor yang di inginkan. Sedangkan keluaran jaringan syaraf adalah

masukan pada motor yaitu berupa tegangan. *Button RESET* akan berfungsi mengembalikan keadaan *set point, motor input* dan *slider* pada nilai awal menjadi sama dengan nol. *Button CLOSE* berfungsi untuk menutup tampilan GUI, sedangkan *Button EXIT* berfungsi menutup semua tampilan *window* yang ada pada Matlab atau keluar dari program Matlab. Karena tampilan GUI mengendalikan simulasi dengan *simulink*, maka saat simulasi dijalankan, tampilan perbandingan antara kecepatan *set point* dan kecepatan keluaran dari motor akan ditampilkan berupa grafik yang dapat di lihat dengan cara men-*double* klik blok *scope* paling kanan pada *simulink*.

Berbeda dengan pengujian menggunakan *simulink*, masukan nilai *set point* di masukan melalaui blok *step* dan jaringan diolah dengan blok-blok terpisah yang dijadikan satu kedalam blok *subsystem*. Berikut ini adalah tabel perbandingan hasil pengujian menggunakan *simulink*, dimana nilai beban, *input* kecepatan dipilih atau ditentukan secara acak.

Tabel 4.12 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.00007 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1-P_2) (rpm)
1	100	99.58	0.42
2	400	399.66	0.34
3	750	749.75	0.25
4	1400	1399.92	0.08
5	1750	1750	0

Dari tabel 4.12 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 0.22 rpm

$$selisih\ rata-rata = \frac{\sum selisih}{banyaknya\ data} \tag{4.4}$$

Hasil penelitian ini membuktikan bahwa simulasi jaringan syaraf tiruan sebagai kendali kecepatan motor DC sudah cukup baik, dengan persentase *mean square error* kecepatan pengujian sebesar 0.00412 %.

$$MSE\ selisih = \frac{\sum(selisih)^2}{banyaknya\ data} \times \frac{100}{nilai\ data\ \max} \% \tag{4.5}$$

Tabel 4.13 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.00009 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih (P_1-P_2) (rpm)
1	400	399.5	0.5
2	900	899.66	0.34
3	1300	1299.75	0.25
4	1500	1499.8	0.2
5	1750	1749.88	0.12

Dari tabel 4.13 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 0.28 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.00551 %.

Tabel 4.14 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.00015 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih (P_1-P_2) (rpm)
1	400	399.15	0.85
2	750	749.24	0.76
3	1050	1049.32	0.68
4	1450	1449.41	0.59
5	1750	1749.5	0.5

Dari tabel 4.14 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 0.68 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.0269 %.

Tabel 4.15 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.0001 Nm

No	Masukan Set Point(P_1) (rpm)	Keluaran Kecepatan(P_2) (rpm)	Selisih(P_1-P_2) (rpm)
1	500	499. 5	0. 5
2	1000	999.65	0.35
3	1300	1299.7	0.3
4	1500	1499.75	0.25
5	1750	1749.8	0.2

Dari tabel 4.15 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 0.32 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.00644 %.

Tabel 4.16 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.0003 Nm

No	Masukan Set Point(P_1) (rpm)	Keluaran Kecepatan(P_2) (rpm)	Selisih(P_1-P_2) (rpm)
1	300	298	2
2	750	748	2
3	1250	1248.2	1.8
4	1400	1398.5	1.5
5	1750	1748.55	1.45

Dari tabel 4.16 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 1.75 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.178 %.

Tabel 4.17 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.0005 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1- P_2) (rpm)
1	500	496.8	3.2
2	750	747	3
3	1100	1097.1	2.9
4	1400	1397.2	2.8
5	1750	1747.27	2.73

Dari tabel 4.17 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 2.93 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.49 %.

Tabel 4.18 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.00006 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1- P_2) (rpm)
1	500	499.75	0.25
2	900	899.85	0.15
3	1250	1249.95	0.05
4	1500	1500.01	0.01
5	1750	1750.07	0.07

Dari tabel 4.18 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 0.11 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.00105 %.

Tabel 4.19 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.001 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1- P_2) (rpm)
1	500	493.75	6.25
2	800	749	6
3	1100	1094	6
4	1400	1394	6
5	1750	1744	6

Dari tabel 4.19 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 6.05 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 2.086 %.

Tabel 4.20 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.002 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1- P_2) (rpm)
1	1000	987.5	12.5
2	1100	1087.5	12.5
3	1250	1237.6	12.4
4	1550	1537.7	12.3
5	1750	1737.57	12.43

Dari tabel 4.20 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 12.43 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 8.8 %.

Tabel 4.21 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban 0.0005 Nm

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Selisih(P_1- P_2) (rpm)
1	400	397	3
2	750	747	3
3	1100	1097	3
4	1300	1297.16	2.84
5	1750	1747.35	2.65

Dari tabel 4.21 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, diperoleh rata-rata selisih kecepatan sebesar 2.9 rpm. dengan persentase *mean square error* kecepatan pengujian sebesar 0.48 %.

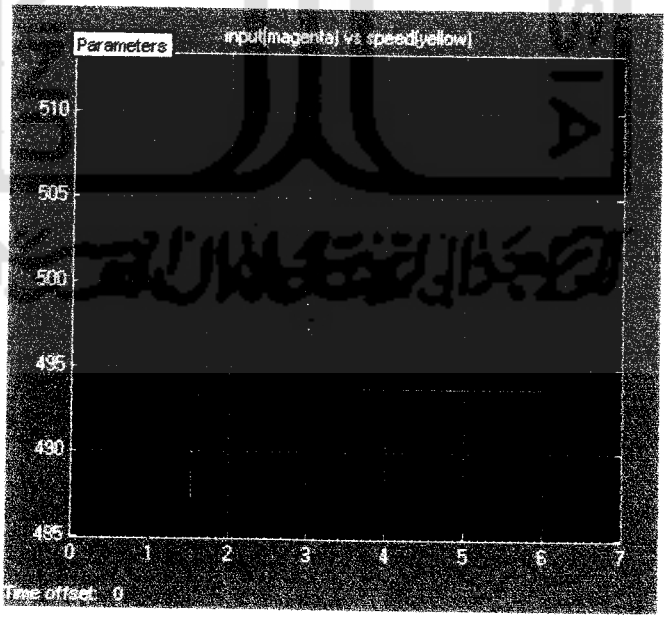
4.2.1 Pengujian Jaringan Syaraf Tiruan dengan beban yang berubah

Tabel 4.22 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban berubah dari 0.00001 Nm ke 0.001 Nm, belum dikendalikan JST.

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Perubahan Kecepatan(rpm)
1	500	500.068 → 494	6.068
2	900	900.17 → 894	6.17
3	1200	1200.25 → 1194	6.25
4	1500	1500.33 → 1494	6.33
5	1750	1750.4 → 1744	6

Dari tabel 4.22 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, dengan beban berubah saat *run* dari 0.00001 Nm bertambah menjadi 0.001 Nm. Diperoleh rata-rata selisih kecepatan dengan perubahan beban sebesar 6.164 rpm.

Gambar perubahan beban tersebut adalah sebagai berikut.



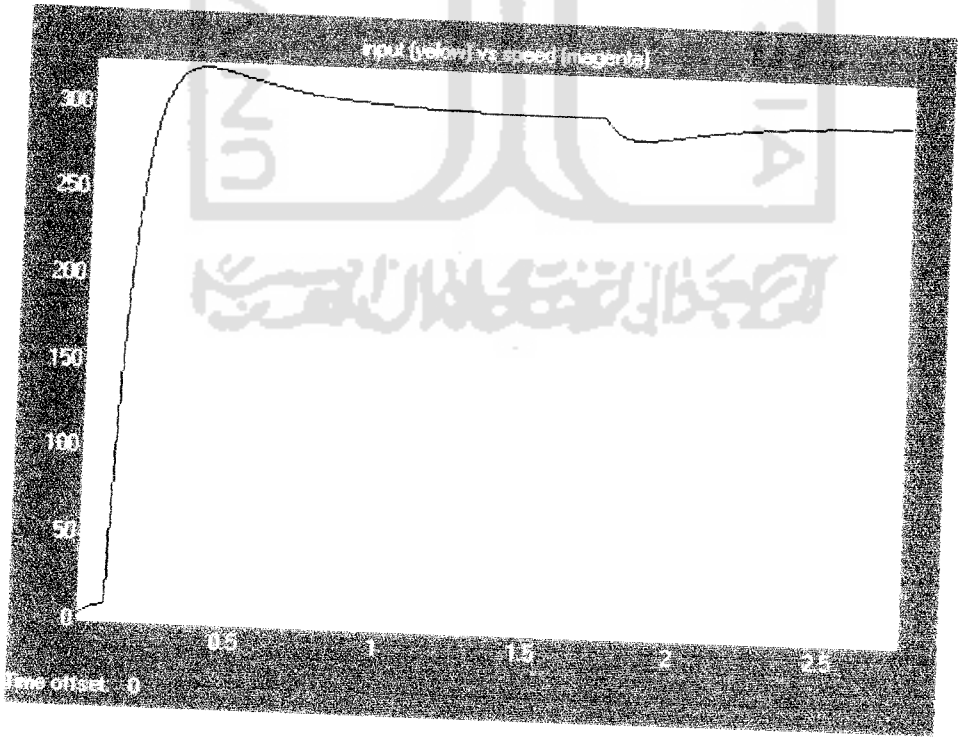
Gambar 4.22 Hasil perubahan beban 0.00001Nm jadi 0.001Nm tanpa kendali.

Tabel 4.23 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban berubah dari 0.00001 Nm ke 0.006 Nm dengan pengendalian JST dan integrator.

No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Perubahan Kecepatan(rpm)
1	300	322 → 287.7	43.3
2	600	620.2 → 591.7	28.5
3	1000	1018 → 988.2	29.8
4	1300	1316 → 1288	28
5	1700	1726 → 1688	38

Dari tabel 4.23 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, dengan beban berubah saat *run* dari 0.00001 Nm bertambah menjadi 0.006 Nm. Diperoleh rata-rata perubahan kecepatan dengan penambahan beban sebesar 33.54 rpm.

Gambar perubahan beban tersebut adalah sebagai berikut :

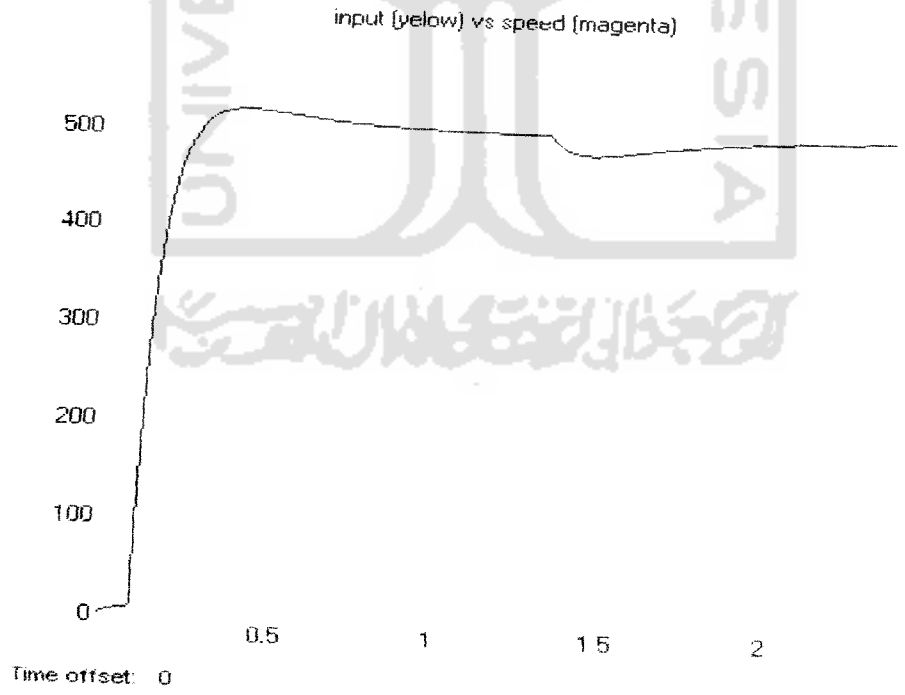


Gambar 4.23 Hasil perubahan beban dari 0.00001 Nm menjadi 0.006 Nm.

Tabel 4.24 Perbandingan kecepatan *set point* dan kecepatan motor dengan beban berubah dari 0.00001 Nm ke 0.01 Nm dengan pengendalian JST dan integrator.

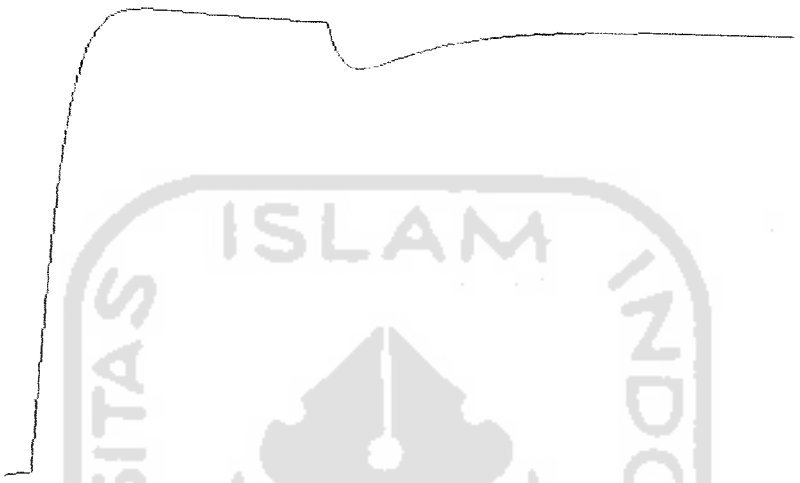
No	Masukan Set Point (P_1) (rpm)	Keluaran Kecepatan (P_2) (rpm)	Perubahan Kecepatan(rpm)
1	500	520.8 → 482.3	38.5
2	900	918.6 → 882.75	35.85
3	1200	1217 → 1179	38
4	1500	1515 → 1482	33
5	1600	1616 → 1580	36

Dari tabel 4.24 diatas, dengan 5 data *masukan* kecepatan yang diambil secara acak, dengan beban berubah saat *run* dari 0.00001 Nm bertambah menjadi 0.01 Nm. Diperoleh rata-rata perubahan kecepatan dengan penambahan beban sebesar 36.27 rpm.

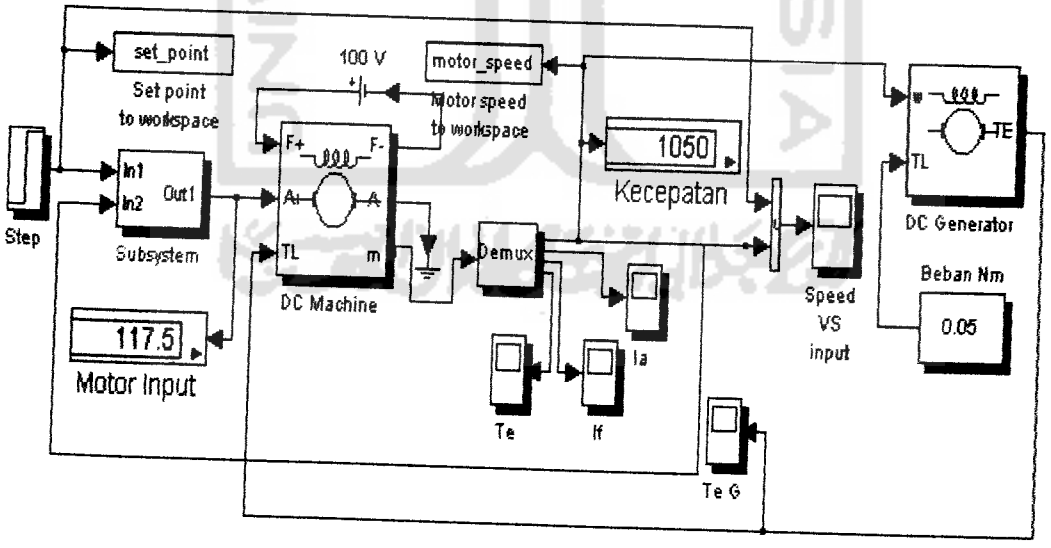


Gambar 4.24 Hasil perubahan beban dari 0.00001 Nm menjadi 0.01 Nm

Berikut gambar (4.25) yaitu perubahan kecepatan dengan nilai set point (input) 1050 rpm dan diberi beban dari 0 menjadi 0.05 Nm (beban maksimum).



Gambar 4.25 Hasil perubahan beban dari 0 jadi 0.05 Nm dengan input 1050 rpm.

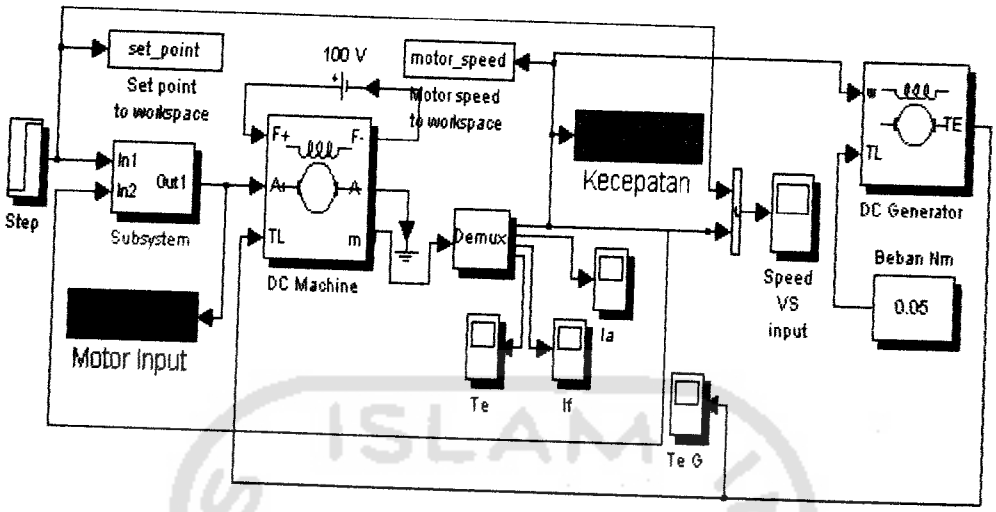


Gambar 4.26 Hasil simulasi dengan input 1050 dan beban 0.05 Nm.

Dari tabel 4.25 diatas, dengan masukan kecepatan (set point) 1050 rpm dan beban berubah saat *run* dari 0 bertambah menjadi 0.05 Nm. Melalui gambar tampak terlihat bahwa saat beban maksimum (0.05 Nm), jst dan integrator dapat mengendalikan kecepatan motor yang turun saat ditambah beban kembali ke set point.

Gambar 4.26 adalah perubahan kecepatan dengan nilai set point (input) maksimum sebesar 1750 rpm dan diberi beban dari 0 menjadi 0.05 Nm (beban maksimum).

Gambar 4.26 Hasil perubahan beban dari 0 jadi 0.05 Nm dengan input 1750 rpm.



Gambar 4.27 Hasil simulasi dengan input 1750 dan beban 0.05 Nm.

Dari gambar 4.26 diatas, masukan kecepatan (set point) 1750 rpm dan beban berubah saat *run* dari 0 bertambah menjadi 0.05 Nm. Melalui gambar tampak terlihat bahwa saat beban maksimum (0.05 Nm), jst dan integrator tidak dapat mengendalikan kecepatan motor yang turun saat ditambah beban kembali ke set point karena tegangan motor sudah di set maksimum (150 volt) dalam blok saturation sesuai dengan parameter motor DC..

Untuk beban besar (maksimum) kecepatan tidak dapat mengikuti perubahan kembali ke set point, hal ini disebabkan koefisien atau gain integrator yang tidak tepat. Sehingga untuk selisih yang terlalu besar pengendali tidak mampu mengembalikan kecepatannya. Penelitian ini belum mencapai unjuk kerja yang diharapkan karena struktur yang digunakan saat pengujian adalah hasil pelatihan yang berhenti disebabkan karena MSE minimum pelatihan belum mencapai target MSE tetapi berhenti karena target iterasi sudah tercapai. Hal ini

juga disebabkan keterbatasan komputer yang digunakan saat pelatihan dan simulasi, sehingga dengan data masukan yang banyak, perlu dicoba menggunakan komputer dengan tingkat proses komputasi yang lebih tinggi atau dengan menggunakan metode pembelajaran yang lebih cepat agar dapat mencapai MSE target yang sekecil mungkin.

Pengaturan gain integrator pada penelitian ini adalah hanya dapat mengendalikan perubahan kecepatan dengan selisih target dan keluaran maksimal 74 rpm. Lebih dari itu pengendali tidak dapat mengembalikan keluaran ke set point semula. Hal ini dikarenakan gain integratornya yang tidak tepat.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah pelatihan dan pengujian dilakukan, maka dapat disimpulkan bahwa :

1. Struktur terbaik jaringan syaraf tiruan untuk sistem kendali kecepatan motor DC terdiri dari 5 sel neuron lapisan *input*. Lapisan tersembunyi terdiri dari 2 lapisan, dimana lapisan tersembunyi pertama memiliki 3 sel neuron, lapisan tersembunyi kedua terdiri dari 1 sel neuron. (*Mean Square Error*) MSE yang dihasilkan adalah 0.016789 dengan fungsi aktivasi setiap lapisan menggunakan fungsi *purelin* (fungsi identitas).
2. Perubahan beban dalam rentang 0 sampai 0.05 Nm masih bisa diperbaiki dan dikendalikan oleh jaringan syaraf tiruan dan integrator dengan input maksimal 1430 rpm.
3. Untuk beban maksimum 0.05 Nm kecepatan tidak dapat dikendalikan oleh JST dan integrator saat set point lebih besar dari 1430 rpm sampai 1750 rpm.

5.2 Saran

1. Jaringan dilakukan dengan metode yang berbeda, agar menghasilkan nilai *Mean Square Error* yang lebih kecil lagi.

2. Mengganti dengan model motor yang lain, tetapi dengan struktur jaringan syaraf yang sama untuk membuktikan apakah jaringan mampu beradaptasi dengan data motor yang berbeda.
3. Tambahkan gain integrator yang tepat agar dapat mengendalikan kecepatan motor DC yang selisih target dan keluarannya lebih dari 75 rpm.

LAMPIRAN

Program randperm

```
% ------------------------------------------------------
%            Simulasi Jaringan Saraf Tiruan            -
%       Berbasis Metode BackPropagation Sebagai        -
%       Pengendali Kecepatan Motor DC Berbeban         -
%   'Program set blok from workspace data pelatihan'   -
%       --=== Fitri Ayu Sarie (01 524 143) ==--        -
% ------------------------------------------------------
clear;
clc;
g = randperm(150)/150;
t = linspace(0,1,150);
teg = [t; g]';

% Jalankan simulasi untuk pengambilan data pelatihan
% Save workspace data hasil dengan nama data_in_fix.mat
```

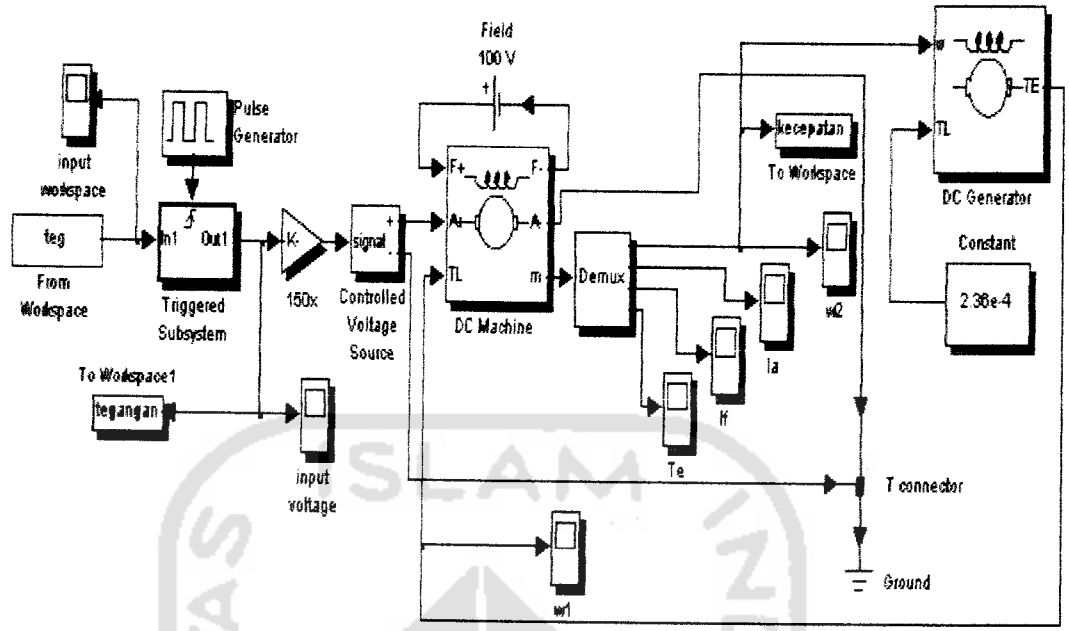
```
%Newff with Traingdm
% ----------------------------------------
%      Simulasi Jaringan Saraf Tiruan     -
% Berbasis Metode BackPropagation Sebagai -
% Pengendali Kecepatan Motor DC Berbeban  -
%           'Program Pelatihan'           -
% ----------------------------------------

clear;
clc;
load(input('namafile data pelatihan : ','s'));
% Data input & target
X  = (kecepatan)/1750;
x1 = [0 X'];
x2 = [X' 0]-x1;
t  = [0 tegangan'];
x  = [x1; x2];
% Membangun jaringan syaraf feedforward
net = newff(minmax(x),[7 1],{'purelin'
'purelin'},'traingdm');
% Melihat bobot-bobot awal input, lapisan, dan bias
v_awal      = net.IW{1}
v0_awal     = net.b{1}
w_awal      = net.LW{2,1}
```

```
w0_awal      = net.b{2}
w_out_awal   = net.LW{3}
w0_out_awal = net.b{1}

% Set max epochs, goal, lr, mc & show step
net.trainParam.epochs    = 1000;
net.trainParam.goal      = 1e-3;
net.trainParam.lr        = 0.2;%0.1
net.trainParam.mc        = 0.8;%0.9
net.trainParam.show      = 1;
% Melakukan pembelajaran
net = train(net,x,t);
% Melihat bobot-bobot awal input, lapisan, dan bias
v_akhir      = net.IW{1}
v0_akhir     = net.b{1}
w_akhir      = net.LW{2,1}
w0_akhir     = net .b{2}
w_out_akhir  = net.LW{2}
w0_out_akhir = net.b{1}
% Melakukan simulasi
y = sim(net,x);
% Menggambar grafik
pause;
r = 1:length(x);
```

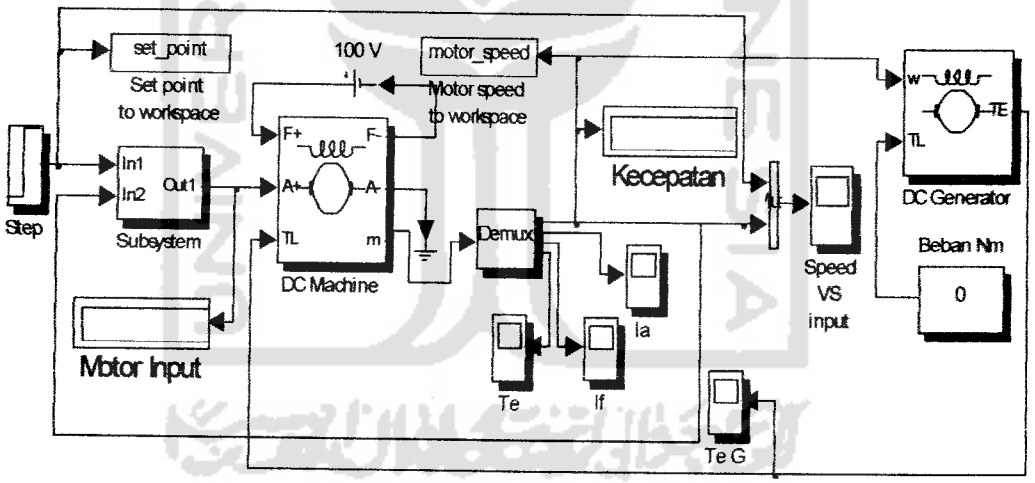
```
plot(r,t,'r-',r,y,'b-');
title('Learning Rate  Momentum ');
legend('Target','Hasil pengujian',0);
xlabel('Length data input');
ylabel('Tegangan');
grid;
```

```
% Fungsi pengujian 2 input 5-3-1
% ---------------------------------------
%      Simulasi Jaringan Saraf Tiruan     -
% Berbasis Metode BackPropagation Sebagai -
% Pengendali Kecepatan Motor DC Berbeban  -
%          'fungsi JST Pengujian'         -
% --=== Fitri Ayu Sarie (00 524 143) ==-- -
% ---------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function fy_out = neuralbp2(in_1,in_2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
x = [in_1/1750 in_2/1750];
%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
v  = [0.47008   -0.81074    -0.20179   -0.38432
1.0015
      0.52384    -0.087159   -0.96354    0.64264    -
0.11054];
v0 = [-0.76786  -0.54192    -0.73045   -0.25363   -
0.089317];
```

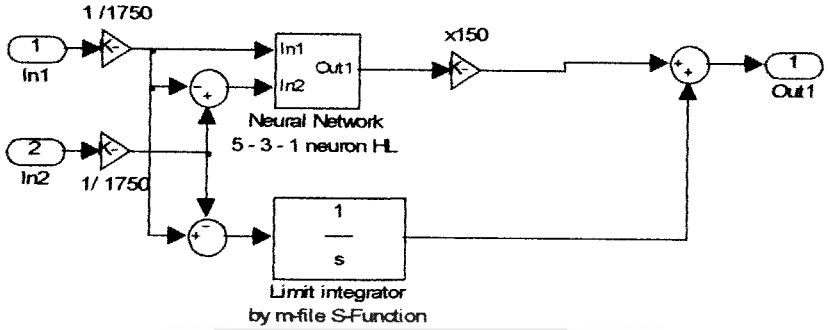
```
w  = [1.1328        0.64938        1.4516    0.067861
     -0.1943
      0.58644      0.093118       -0.16147       -0.65332
     -0.31559
      0.74758     -0.54839        -0.77743       -0.72149
     -0.63254];
w0 = [1.7165    -1.0705     0.33936];
w_out  = [-2.1318    1.8786     -0.69901];
w0_out = [1.7760];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z  = v0+(x*v);
fz = purelin(z);
y  = w0+(fz*w');
fy = purelin(y);
y_out = w0_out+(fy*w_out');
fy_out = purelin(y_out);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



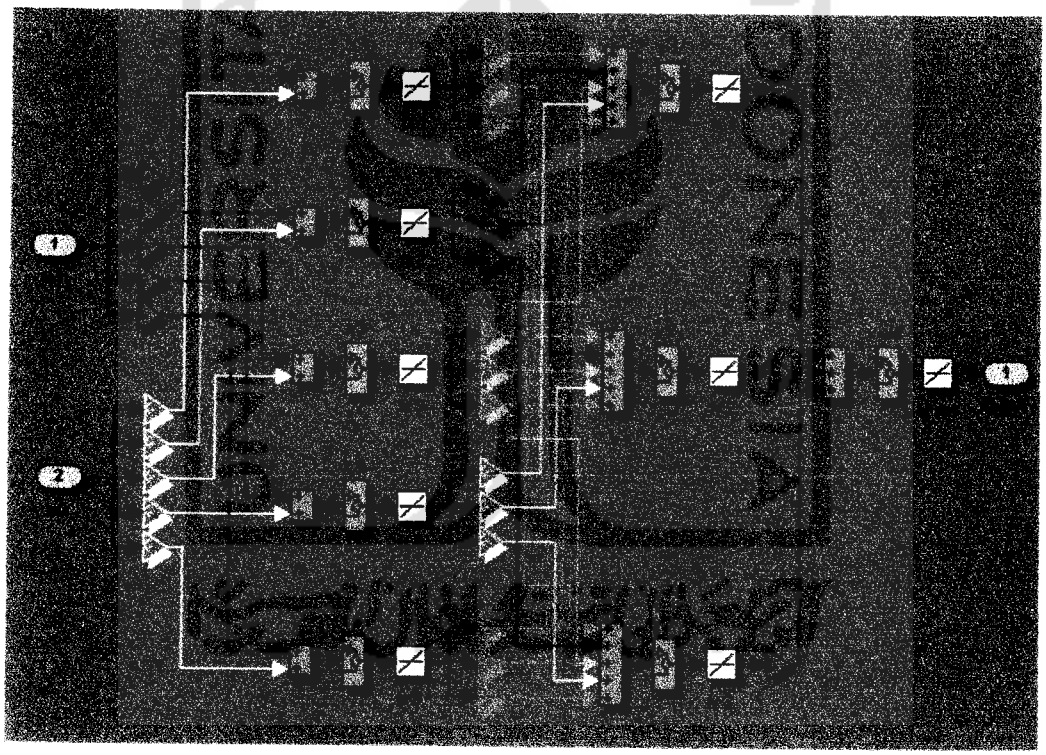
Rangkaian simulasi pengambilan data pelatihan



Rangkaian utama simualasi jaringan syaraf tiruan berbasis metode *backpropagation* sebagai pengendali kecepatan motor DC berbeban



Rangkaian simulasi pada blok *Subsystem* JST



Rangkaian simulasi dalam blok Neural Network 5-3-1 neuron HL